applied sciences

MDPI

Article

A Semantic Model for Enhancing Data-Driven Open Banking Services

Manuel Paneque *,†, María del Mar Roldán-García † and **José García-Nieto †**

ITIS Software, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, 29071 Málaga, Spain

* Correspondence: mpaneque@uma.es

† These authors contributed equally to this work.

Abstract: In current Open Banking services, the European Payment Services Directive (PSD2) allows the secure collection of bank customer information, on their behalf and with their consent, to analyze their financial status and needs. The PSD2 directive has lead to a massive number of daily transactions between Fintech entities which require the automatic management of the data involved, generally coming from multiple and heterogeneous sources and formats. In this context, one of the main challenges lies in defining and implementing common data integration schemes to easily merge them into knowledge-base repositories, hence allowing data reconciliation and sophisticated analysis. In this sense, Semantic Web technologies constitute a suitable framework for the semantic integration of data that makes linking with external sources possible and enhances systematic querying. With this motivation, an ontology approach is proposed in this work to operate as a semantic data mediator in real-world open banking operations. According to semantic reconciliation mechanisms, the underpinning knowledge graph is populated with data involved in PSD2 open banking transactions, which are aligned with information from invoices. A series of semantic rules is defined in this work to show how the financial solvency classification of client entities and transaction concept suggestions can be inferred from the proposed semantic model.

Keywords: data integration; knowledge graphs; ontology; open banking; PSD2

check for updates

Citation: Paneque, M.; Roldán-García, M.d.M.; García-Nieto, J. A Semantic Model for Enhancing Data-Driven Open Banking Services. *Appl. Sci.* **2023**, *13*, 1447. https://doi.org/10.3390/app13031447

Academic Editor: Lina Soualmia

Received: 30 November 2022
Revised: 13 January 2023
Accepted: 18 January 2023
Published: 21 January 2023

Copyright: © 2023 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 (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Open Banking services nowadays are powerful facilities in the Fintech sector for digitally sharing financial information with third parties, enabling customers to approve their transactions through secure open APIs. This approach promotes the rapid digital transformation of payment services, traditionally managed by bank entities, and the generation of digital platforms and companies that offer their customers payment and enhanced financial services. In this context, the European Payment Services Directive (PSD2) allows the secure collection of bank customer information, on their behalf and with their consent, with the purpose of analyzing their financial status and needs [1]. This new paradigm leads to massive daily transactions between Fintech entities, which require the automatic management of the data involved, which generally come from multiple and heterogeneous sources and formats.

In this regard, current and past initiatives like the Helix project (AEI-010500-2020-34 NextGeneration EU) and the Estrella European Initiative (IST-2004-027655) [2] are focused on developing the fundamental bases of a future online platform that democratizes the access of SMEs to financing and risk management tools. The objective of this platform is to dynamize economy and security in B2B transactions. Today, these tools are only available to large companies. The main goal in such projects is to comprehensively manage commercial risk operations with customers through the possibility of online contracting of credit insurance by invoice, credit insurance by a debtor, as well as direct access to financing

of circulating capital through Factoring, all automatically and instantaneously in a single online transaction, based on predictive algorithms and the historical payment behavior of the different debtors.

A key challenge in this scenario is the definition and implementation of common data integration schemes, and merging them into knowledge graphs easily, hence allowing data reconciliation and leading to sophisticated analysis, e.g., by cross-checking information between different account statements to verify the coincidence of the company's balances. This can serve as an additional security measure against attempted fraud. In this sense, Semantic Web technologies constitute a suitable framework for the semantic integration of data that enables linking with external sources and enhances systematic querying. The development of new ontologies and their use for data integration is widely documented in the existing literature for different domains of application [3–6]. In the scope of finance and banking, a series of approaches have been described [2,7–9] in which ontologies are designed for describing the main concepts and relationships involving data governance and management activities in this sector. A comprehensive survey of this direction was recently presented in [10].

A concrete example is the well-known Financial Industry Business Ontology (FIBO) [8], which is still evolving since its initial publication in 2014 and has increasing support for use cases related to security master data management, reporting, risk analysis, and management. Nevertheless, although all of these initiatives constitute a starting point for future work, specific practical aspects and standards (such as the PSD2) should be considered to facilitate modern open banking data integration and digital applications.

To cope with this issue, an ontology approach called OBO (Open Banking Ontology) is proposed in this work to operate as a semantic data mediator in real-world open banking operations. Following this ontological scheme, a new knowledge graph is generated and populated with data involved in actual PSD2 open banking transactions, which are aligned with information from invoices, according to semantic reconciliation mechanisms. To allow this, text processing techniques and semantic queries are adapted to extract two corpora of words. After this, they are compared with a fuzzy search method to establish a similarity score. This methodology follows similar approaches successfully applied in other fields such as biomedical informatics [11] or the recognition of company names featured in press news [12].

In addition, a series of semantic rules are established to show how financial solvency classification of entities and transaction concept suggestions can be inferred from the proposed semantic model.

Concretely, the main contributions of this work can be summarized as follows:

- An ontology is defined using Web Ontology Language (OWL 2) [13], for the first time for the semantic annotation of bank statement and invoice management operations in the context of PSD2 transactions. The proposal, called Open Banking Ontology (OBO), is used as a semantic integration data scheme for different information sources that enables data ingestion, querying, and reasoning in a structured and homogeneous way.
- According to the OBO definition, a semantic model is also developed for knowledge graph generation, which is populated by means of specific mapping functions for each different data source. The resulting graph comprising all the data instances is integrated into a repository using Resource Description Framework (RDF) [14], which allows querying with the same language (SPARQL). A series of reconciliation mechanisms were tested with data from more than 70,000 invoices and 33,000 open banking customers through multiple PSD2 operations.
- The proposal is validated through several real-world use cases, including intelligent matching bank statements with their corresponding invoices. Moreover, a set of SWRL reasoning rules is also specified for classifying the financial solvency of clients, including an automatic generation of concepts that would facilitate conciliation.

The remainder of this paper is organized as follows. Background concepts are explained in the next section, which also includes a review of related work. In Section 3,

the ontology design is detailed, including the software methods implemented to build the knowledge graph. Semantic reconciliation mechanisms and reasoning use cases are explained in Section 4. Finally, Section 5 contains the main conclusions and lines of future research. In addition, annexes with complementary information are included.

2. Background Concepts and Related Work

Before defining the semantic approach, this section is devoted to explicating the basic concepts of semantic web technologies for the representation and structure of a knowledge graph. A review of related works in the literature is carried out in order to situate our proposal within the current state of the art.

2.1. Preliminary Concepts

This subsection is devoted to describing a set of relevant concepts to make the paper more self-contained.

- Ontology [15]. This is a description framework that provides a formal representation of the real world, shared by many users, by defining concepts and relationships between them. It defines a set of axioms and primitives to represent a specific domain of knowledge or discourse. In computer science, ontology refers to an artifact designed to allow the formal modeling of knowledge about a domain. Ontologies are part of the standards developed by the W3C consortium for the semantic web. In the semantic web, they are used to standardize vocabularies that facilitate the exchange of information between systems. This standard provides services to solve queries and to populate and publish knowledge graphs [16] so that they can be reusable and facilitate interoperability between heterogeneous systems and databases. Instances of ontology classes and properties annotate data or resources semantically. Furthermore, the reasoning capabilities of ontologies allow semantic web applications to infer implicit knowledge from explicitly expressed knowledge, enabling the creation of intelligent applications.
- RDF [14]. To define an ontology in the context of web semantics, we used the standard language RDF, according to which resources are described in terms of properties and values. RDF statements are represented as triples, consisting of a *subject*, a *predicate*, and an *object*. RDFS (RDF Schema) [17] semantically extends RDF to define resource classes and the properties that those classes describe. RDFS provides the mechanisms for explaining the specific RDF vocabularies for each application. Although the expressiveness of RDFS is still insufficient for all applications, it is enough for data exchange and allows semantic languages to be built on top of it. However, for many other applications, it is necessary to have other characteristics, such as cardinality constraints, transitive properties, inverse properties, and range. When used for a specific class, it is able to express classes as unions or intersections of other classes. In addition, ontology languages should meet other requirements, such as having a well-defined syntax, being based on formal semantics necessary for reasoning, and supporting efficient reasoning mechanisms while maintaining expressiveness.
- OWL [17]. Proposed by the W3C Ontology Working Group, this is a semantic markup language for publishing and sharing ontologies on the web. OWL is equivalent to a very expressive description logic, where an ontology corresponds to the Tbox [18]. This equivalence allows language to exploit the results of researchers in the logic of descriptions. OWL extends RDF and RDF Schema, with the main goal of bringing the expressiveness and reasoning power of description logic to the semantic web. OWL provides two sublanguages: OWL Lite for simple applications and OWL-DL, which represents the subset of the language equivalent to the logic of descriptions, the reasoning mechanisms of which are complex. A new version of this language is OWL 2 [13], which includes new functionalities, e.g., keys, property strings, data ranges, and qualified cardinality constraints. These new features are oriented towards increasing expressiveness.

2.2. Related Work

In the past and current literature, not many studies can be found that propose ontologies dealing with the economic and/or finance domains of knowledge. However, these constitute important contributions that are worth highlighting.

A first attempt was introduced by Castells et al. in [7], where an interesting ontology-based platform was developed to integrate finance base contents and semantics in a knowledge base that provides a conceptual view of low-level contents, including navigation semantic search facilities. However, it still constitutes a preliminary proposal for which the ontology developed is unavailable, and the semantic search is limited to values of specific properties only.

In 2008, as a result of the European Estrella protect (ESTRELLA Project http://www.estrellaproject.org/ (accessed on 27 November 2022)), the Legal Knowledge Interchange Format (LKIF) [2] appeared as an XML Schema for representing theories and arguments (proofs). An approach in LKIF consists of a set of axioms and defeasible inference rules. The individuals and predicates can be imported from an ontology represented in the Web Ontology Language (OWL). Importing an ontology also imports the axioms of the ontology. Other LKIF files may also be imported, enabling complex theories to be modularized.

Also in the domain of financial products, but with a different orientation, an ontology was proposed in [9] (2011) for annotating financial products and customers. The ultimate aim of this work was to model a recommender system for linking products with client's profiles, and so the proposal was restricted to this specific application.

As remarked in the introduction, an important past proposal is the Financial Industry Business Ontology (FIBO) (EDM Council FIBO https://spec.edmcouncil.org/fibo (accessed on 20 November 2022)) [8] (2013), which was conceived as an ontological model composed of modules or sub-ontologies, each one for a specific purpose in the regulatory domain of finance. This ontology is aligned with LKIF, as they both focus on the legal regulation of the different actors in financial transactions. Although they do not cover PSD2 bank statement directives, they could be easily aligned to OBO with regard to classes such as *service provider*, *legal person* or *regulatory agency*.

More recently, in 2017, the OntoREA [19] model was proposed to cover the REA accounting UML model that conceptualizes the economic logic of accounting in terms of resources that are exchanged in economic events between economic agents. It is a mapping translation using OntoUML [20], although no OWL specification is provided and it is oriented to top-level economic concepts.

Similarly, the COFRIS [21] ontology was proposed, also in 2017, to consider financial information systems that model accounting reports. It is also designed with the OntoUML language but with a particular orientation through financial reporting according to regulations, hence covering scopes other than open banking statements and reconciliation methods.

In 2021, the authors of this paper [22] argued that there was a need for a semantized banking schema. They integrated existing COVID-19 ontologies in order to analyze how the COVID-19 pandemic affected the banking sector.

Therefore, although these ontologies describe many of the top-level concepts in the domain of economic regulation, they represent initial steps in the semantic specification of specific data-driven mechanisms for allowing advanced analysis in financial statements. The OBO ontology proposed here aims to describe one step beyond this direction, with a special focus on open banking statements and transactions.

3. Semantic Model

This section describes the methodology followed for designing the proposed ontology and gives details of its implementation. In this context, a series of components and mechanisms are also introduced to show how the underpinning knowledge graph is built and populated with real-world open banking data.

3.1. Methodology

When defining a new ontology, a good practice is to follow a well-grounded methodology that considers all the required aspects for knowledge domain description. Concretely, the "Ontology 101 Development Process" [23] was used, which consists of the following seven steps:

1. Determine the domain and scope of the ontology. The ontology domain represents the different actors and the information generated in open banking services. More specifically, bank statements and invoice management systems are contextualised to allow for data reconciliation tasks.
2. Consider reusing existing ontologies. After reviewing the literature, several ontologies were considered to describe accounting entries [8]. Still, none of them represents bank statements oriented to PSD2 format and other related aspects such as invoice contextual information. These ontologies usually contain a class *Invoice*, *Receipt* or other similar terms, but they do not ontologically describe an invoice, with relationships and components. In the proposed approach, the invoice concept plays an essential role since it allows considering all the meta-information, involving it to be further used in reconciliation mechanisms for linking with third-party bank statements. For example, those products or services the invoice refers to are usually unknown to the legal person or company that a given client has invoiced.
3. Enumerate important terms in the ontology. As mentioned in the previous step, important terms in OBO correspond to data in invoices and bank statements, involving open banking service transactions. Examples of these terms are: *companyId*, *customerId*, *issueDate*, *maturityDate*, *itemCurrencyAmt*, and *documentType*, which are related to invoicing, as well as those related to bank statements, such as: *Statement*, *Customer*, *Account number*, *Bank*, *Concept*, and *beneficiary*.
4. Define the classes and the class hierarchy. From the relevant terms, we extract which of them are classes of ontology. Figure 1 shows the set of classes that have been defined to describe an open bank service. Examples of these classes are: *Item*, *Statement*, *AccountObj*, and *Bank*. The class *Item* comprises two subclasses, *Opened* and *Cleared*, for specifying whether Items are pending to be paid or not.
5. Define the properties of classes. Object properties and data properties were created to connect instances of different classes and define their attributes. Examples of object properties are: *participating*, which connects a customer to the bank account; *statementHasOffice*, which relates a bank movement to the office in which it was registered; and *ItemHasCompany*, which connects an item with the company. Examples of data properties are: *statementConcept*, *beneficiary*, *accountNumber itemIssueDate*, *itemAmount*. Tables 2–6 describe details of these properties in description logic.
6. Define the facets of the slots. In this step, cardinality and value constraints are defined. In the proposed ontology, examples of constraints are: the range of the property *itemAmount* has to be a decimal number; the range of the property *itemIssueDate* has to be a date; the *concept* of a statement has to be of type *string*, and so on.
7. Create instances. The instances (or individuals) in the current approach are generated from the JSON documents containing actual PSD2 bank statements and the CSV files with data of the items (invoices). These data are transformed to RDF through mapping functions according to the classes and properties defined in OBO, and an RDF knowledge graph is then created.

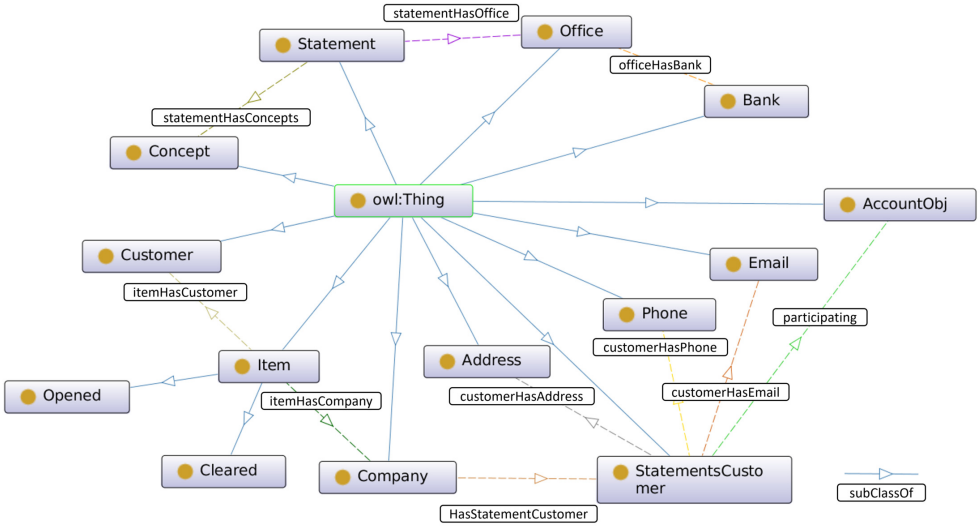

Figure 1. General overview of the Open Banking Ontology proposed in this study.

3.2. Ontology Implementation

The Open Banking Ontology (OBO) proposed here was developed with Protégé editor (Protégé Web Site https://protege.stanford.edu/ (accessed on 25 November 2022)) after several rounds of design with experts in the domain of knowledge (in the context of the Helix project). As a result, the proposed ontology is made up of 14 classes, 10 object properties, 35 data properties, and 250 axioms.

Figure 1 shows a general overview of the OBO proposal, and for the sake of simplicity, a selection of the main classes and properties are detailed below. The properties are formalized using the OWL-DL semantic logical description syntax. A summary of this syntax, which can be used to support the interpretation, is shown in Table 1. The remaining elements of OBO can be examined in the software repository, which is available at https://ontologies.khaos.uma.es/obo (accessed on 28 November 2022).

1. The class ***Company*** represents those companies that take part in a given PSD2 transaction. For this class, 5 data properties were defined: *businessName*, to indicate the company name, *companyId*, which models the company identifier, *corporationId* to identify the corporation (when existing) to which the company belongs, *currencyCode*, which represents the currency the company works with, and *fiscalId* to store the tax identification number. In addition, the *Company* class has an object property, *hasStatementCustomer*, which links the company with its PSD2 bank statements. Table 2 describes the properties of the Company class in the description logic definition.
2. The ***Customer*** class was included to annotate customers that participate in transactions with companies. A set of 3 data properties were defined for this class: *businessName*, which represents the customer's name, *customerId*, which stores the customer identifier, and *fiscalId* to set the number of the client's tax identification. Description logics of these properties can be found in Table 3.
3. The ***Item*** class represents the invoices that customers should pay. Its main properties are defined in description logic in Table 4, comprising 7 data properties and 2 object properties. As for data properties, it is worth mentioning textitemAmount, which stores the invoice amount in the currency associated with the corresponding company, *itemCurrencyAmt*, to annotate the amount of the invoice, *itemIssueDate*, which describes the invoice issue date, *currencyCode*, which indicates the currency associated with the invoice and in which it is expressed, and *indicator*, which represents a series of descriptors associated with the invoice. The two object properties are: *itemCompany*, which links each invoice with its company, and *ItemCustomer*, which connects the invoice to the customer.

4. The class ***Statement*** represents the bank statements carried out on a particular bank account. Among its main data properties, a set of 5 representatives ones is: *statementAmount*, to annotate the transaction amount, *accountNumber*, which indicates the bank account in which the statement was made, *statementCurrency*, which stores the currency associated with the statement, *statementBeneficiary*, representing the beneficiary of the bank statement, *concept* and *statementValueDate*, which indicate the concept and value date of the statement, respectively. The object property *statementHasOffice* was defined for this class to associate each transaction with the corresponding bank branch. Table 5 describes the properties of the *Statement* class.
5. The ***Bank*** class models the bank information. A total of 3 data properties were defined for this class: *bankGroupId*, which stores the identifier of the bank group to which the bank belongs, *bankId*, to represent the identifier of the bank, and *bankName*, which stores the name of the bank. This class is important for annotating not only traditional banks, but also open banking entities that take part in PSD2 transactions with customers.
6. The class ***StatementsCustomer*** represents the holders of the bank accounts in which the movements are made. For this class, a series of data properties were defined to store the customer's identity document, name, date of birth, address, email, and telephone number, as described in Table 6. The customer is related to his/her/their bank accounts through the object property *participating*, specifying the role in which he/she/they participate(s) (owner or authorized). This is modeled using the *AccountObj* class.

Table 1. Basic OWL-DL semantic syntax used to formally define the proposed ontology.

	Abstract Syntax	**DL Syntax**
Operators	$intersection(C_1, C_2, \cdots, C_n)$	$C_1 \sqcap C_2 \sqcap \cdots C_n$
	$union(C_1, C_2, \cdots, C_n)$	$C_1 \sqcup C_2 \sqcup \cdots C_n$
Restrictions	for at least 1 value V from C	$\exists V.C$
	for all values V from C	$\forall V.C$
	R is Symmetric	$R \equiv R^-$
Class Axioms	$A\ partial(C_1, C_2, \cdots, C_n)$	$A \sqsubseteq C_1 \sqcap C_2 \sqcap \cdots C_n$
	$A\ complete(C_1, C_2, \cdots, C_n)$	$A \equiv C_1 \sqcap C_2 \sqcap \cdots C_n$

Table 2. Data and object properties of class Company.

Object Properties	**Description Logics**
hasStastementCustomer	∃ hasStastementCustomer Thing ⊑ Company ⊤ ⊑ ∀ hasStastementCustomer StatementsCustomer
Data Properties	**Description logics**
businessName	∃ businessName Datatype Literal ⊑ Company ⊤ ⊑ ∀ businessName Datatype string
companyId	∃ companyId Datatype Literal ⊑ Company ⊤ ⊑ ∀ companyId Datatype int
corporationId	∃ corporationId Datatype Literal ⊑ Company ⊤ ⊑ ∀ corporationId Datatype int
currencyCode	∃ currencyCode Datatype Literal ⊑ Company ⊤ ⊑ ∀ currencyCode Datatype string
fiscalId	∃ fiscalId Datatype Literal ⊑ Company ⊤ ⊑ ∀ fiscalId Datatype string

Table 3. Data properties of class Customer.

Data Properties	Description Logics
businessName	$\exists$ businessName Datatype Literal $\sqsubseteq$ Customer $\top \sqsubseteq \forall$ businessName Datatype string
customerId	$\exists$ customerId Datatype Literal $\sqsubseteq$ Customer $\top \sqsubseteq \forall$ customerId Datatype int
fiscalId	$\exists$ fiscalId Datatype Literal $\sqsubseteq$ Company $\top \sqsubseteq \forall$ fiscalId Datatype string

Table 4. Data and object properties of class Item.

Object Properties	Description Logics
itemHasCompany	$\exists$ itemHasCompany Thing $\sqsubseteq$ Item $\top \sqsubseteq \forall$ itemHasCompany Company
itemHasCustomer	$\exists$ itemHasCustomer Thing $\sqsubseteq$ Item $\top \sqsubseteq \forall$ itemHasCustomer Customer
Data properties	**Description logics**
indicator	$\exists$ indicator Datatype Literal $\sqsubseteq$ Item $\top \sqsubseteq \forall$ indicator Datatype string
itemAmount	$\exists$ itemAmount Datatype Literal $\sqsubseteq$ Item $\top \sqsubseteq \forall$ itemAmount Datatype decimal
itemCurrencyAmt	$\exists$ itemCurrencyAmt Datatype Literal $\sqsubseteq$ Item $\top \sqsubseteq \forall$ itemCurrencyAmt Datatype decimal
itemIssueDate	$\exists$ itemIssueDate Datatype Literal $\sqsubseteq$ Item $\top \sqsubseteq \forall$ itemIssueDate Datatype dateTime
itemMaturityDate	$\exists$ itemMaturityDate Datatype Literal $\sqsubseteq$ Item $\top \sqsubseteq \forall$ itemMaturityDate Datatype dateTime
currencyCode	$\exists$ currencyCode Datatype Literal $\sqsubseteq$ Item $\top \sqsubseteq \forall$ currencyCode Datatype string
itemId	$\exists$ itemId Datatype Literal $\sqsubseteq$ Item $\top \sqsubseteq \forall$ itemId Datatype int

Table 5. Data and object properties of class Statement.

Object Properties	Description Logics
statementHasOffice	$\exists$ statementHasOffice Thing $\sqsubseteq$ Statement $\top \sqsubseteq \forall$ statementHasOffice Office
Data properties	**Description logics**
accountNumber	$\exists$ accountNumber Datatype Literal $\sqsubseteq$ Statement $\top \sqsubseteq \forall$ accountNumber Datatype string
statementAmount	$\exists$ statementAmount Datatype Literal $\sqsubseteq$ Statement $\top \sqsubseteq \forall$ statementAmount Datatype int
statementBeneficiary	$\exists$ statementBeneficiary Datatype Literal $\sqsubseteq$ Statement $\top \sqsubseteq \forall$ statementBeneficiary Datatype string
statementCurrency	$\exists$ statementCurrency Datatype Literal $\sqsubseteq$ Statement $\top \sqsubseteq \forall$ statementCurrency Datatype string
statementValueDate	$\exists$ statementValueDate Datatype Literal $\sqsubseteq$ Statement $\top \sqsubseteq \forall$ statementValueDate Datatype dateTime
concept	$\exists$ concept Datatype Literal $\sqsubseteq$ Statement $\top \sqsubseteq \forall$ concept Datatype string

Table 6. Data and object properties of class StatementsCustomer.

Object Properties	**Description Logics**
participating	∃ participating Thing ⊑ StatementsCustomer ⊤ ⊑ ∀ participating AccountObj
Data properties	**Description logics**
customerBirthDate	∃ customerBirthDate Datatype Literal ⊑ StatementsCustomer ⊤ ⊑ ∀ customerBirthDate Datatype dateTime
customerDocument	∃ customerDocument Datatype Literal ⊑ StatementsCustomer ⊤ ⊑ ∀ customerDocument Datatype string
customerNames	∃ customerNames Datatype Literal ⊑ StatementsCustomer ⊤ ⊑ ∀ customerNames Datatype string
address	∃ address Datatype Literal ⊑ StatementsCustomer ⊤ ⊑ ∀ address Datatype string
email	∃ email Datatype Literal ⊑ StatementsCustomer ⊤ ⊑ ∀ email Datatype string
phone	∃ phone Datatype Literal ⊑ StatementsCustomer ⊤ ⊑ ∀ phone Datatype string

3.3. Building the Knowledge Graph

Once the OBO ontology has been defined, the data involved can be integrated and consolidated by following the defined semantic model in an RDF repository. The information is then stored in the form of a knowledge graph with a homogeneous format, regardless of the source and avoiding semantic inconsistency. In order to clarify the technology that was used to deploy the model, Table 7 summarizes the package and languages that were involved in each step of the process. This process is illustrated in Figure 2, showing that a series of mapping functions were implemented using RDFLib (https://rdflib.dev/ (accessed on 24 November 2022)) to automatically translate the customer's data from the issued invoices and bank statements into RDF format, hence populating the knowledge graph derived from the OBO scheme.

- Mapping invoice data into RDF. These data are obtained from the ERP systems of the companies involved in CSV format, which comprise four different files: (1) for open invoices, (2) for closed invoices, (3) for companies that issue invoices to customers, and (4) for customer information, by means of the *Customer* class in OBO. In addition, contents regarding classes *Item*, *Company*, and *Customer* are also stored in the repository.
- Mapping PSD2 statements into RDF. Data concerning bank statements are collected from the corresponding APIs (of banking entities) in JSON format. The service client provides the user with an access token with which all bank movements can be accessed. Examples of JSON files in Code Listings A1 and A2 in Appendix A show information obtained from the PSD2-compliant API services of banks concerning the customers and their bank accounts. Classes *StatementCustomer* and *Statement*, among others, are populated in the form of semantic instances (individuals).

Once the RDF triplets corresponding to bank statements and invoices have been created, they are stored in the RDF repository, which is deployed by means of a Stardog (https://www.stardog.com/ (accessed on 22 November 2022)) instance service. Therefore, the integrated data can be efficiently queried through a SPARQL endpoint. In addition, the RDF repository is powered by an SWRL inference engine to explode different sets of semantic rules designed to conduct classification and validation tasks on financial subjects, which, together with the semantic reconciliation mechanisms, are explained in the next section.

Table 7. Methods and packages used at each step.

Step	Package	Language	Description	Reference
OBO Ontology	Protégé	OWL	Define Ontology	https://protege.stanford.edu/ (accessed on 25 November 2022)
JSON Mapping	RDFLib	Python	Translate JSON files and create RDF data	https://rdflib.dev/ (accessed on 24 November 2022)
CSV Mapping	RDFLib	Python	Translate CSV files and create RDF data	https://rdflib.dev/ (accessed on 24 November 2022)
RDF Repository	Stardog	SPARQL	Store and query the RDF data	https://www.stardog.com/ (accessed on 22 November 2022)
Fuzzy Similarity Search	RapidFuzz	Python	Calculate the strings similarity	https://maxbachmann.github.io/RapidFuzz/ (accessed on 26 November 2022)
Genetic Algorithm	jMetalPy	Python	Calculate the subset sum of the amounts	https://jmetalpy.readthedocs.io/ (accessed on 21 November 2022)
Inference	Pellet	Java	Reasoning with the elements and rules defined in the ontology	https://www.stardog.com/ (accessed on 22 November 2022)

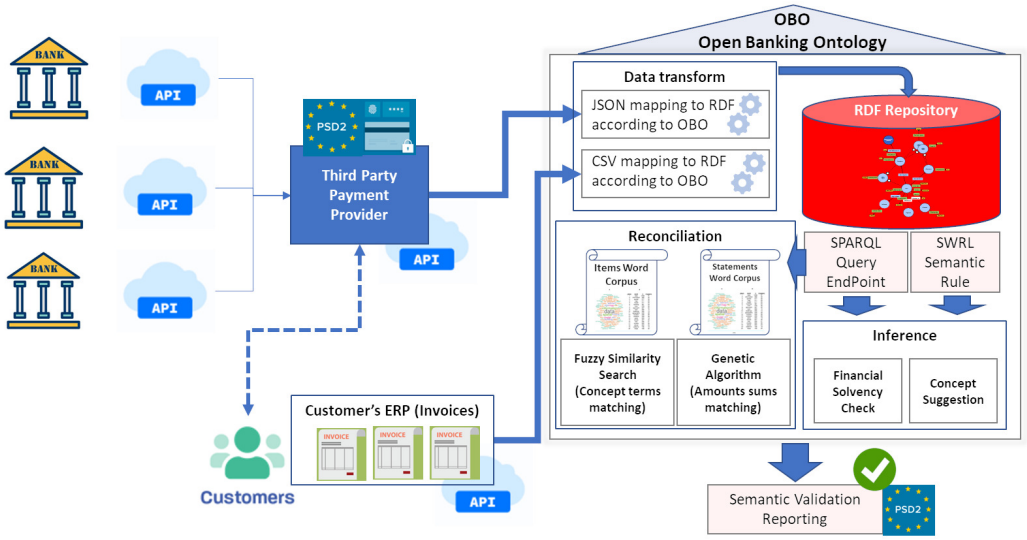

Figure 2. General overview of the semantic model employed in this study, with regards to the OBO ontology.

4. Reconciliation and Inference

This section aims to show the usability of the proposed semantic model by carrying out a series of real-world use cases oriented toward data reconciliation and semantic inference. The data involved, although previously anonymized, are captured in the context of the aforementioned Helix innovation initiative (The HELIX Innovation Project (https://www.ongranada.com/proyectos-2/ (accessed on 21 November 2022)) and are made up of a cluster of financial organizations that work to maximize the liquidity of small and medium-sized companies).

4.1. Semantic Reconciliation

One of the most common operations in current banking, which usually requires strenuous expert efforts, consists of (manual) movement inspections to match partial payment statements concerning different concepts in invoices. A way to facilitate these processes is to filter those operations whose concepts and payments can be semantically matched in an automatic way. This approach can now be conducted by means of the semantic system implemented here, according to the OBO TBox (Terminological Box) (Following the Description Logic nomenclature, ontology classes and properties are called "Tbox", while ontology individuals are called "Abox" (Assertional box)) for concept disambiguation.

For validation, a number of 74,000 real-world invoices and 150 bank statements were used, the contents of which were stored in the RDF repository with the mapping functions explained in the previous section. In this regard, as shown in Figure 2, a series of SPARQL queries are applied to building two-word corpora. One for each item concept involves the invoice data that companies usually store in their corresponding ERPs (customer's business and fiscal name to whom the invoice was financed, etc.), and another one for each *Statement*, with concepts of bank transactions (beneficiary, bank, etc.).

Table 8 shows the SPARQL queries defined to construct these corpora, with two examples as results. Concretely, the first set of queries (first row) retrieves the properties *ItemsConcepts*, *ItemsBusinessName*, and *ItemsFiscalId* to obtain the information in the *Items* for two invoices, while queries (second row) are used to select properties *PSD2Concepts* and *PSD2Beneficiary* of two sample bank *Statements*.

Table 8. SPARQL Queries for work corpus generation.

SPARQL	Result

`# Query to get corpus generated for items.`
`PREFIX`
`obo: <http://ontologies.khaos.uma.es/obo/>`
`SELECT ?i ?indicator_field`
`WHERE`
`  {?i obo:indicator ?indicator_field.}`
 	
`SELECT ?i ?indicator_field`
`WHERE`
`  { ?i obo:itemHasCustomer ?indicator.`
`  ?indicator obo:businessName`
`  ?indicator_field`
`  }`
 	
`SELECT ?i ?indicator_field`
`WHERE`
`  { ?i obo:itemHasCustomer ?indicator.`
`  ?indicator obo:fiscalId`
`  ?indicator_field`
`  }`
	`{`
	`  "obo_item00035ed7-8a02-e3dd":[`
	`  "1",`
	`  "1-1",`
	`  "1201****-I",`
	`  "EXAMPLE SL",`
	`  "B2756****"`
	`  ],`
	`  "obo_item0007874d-30a9-a4b0":[`
	`  "1",`
	`  "1902****-F",`
	`  "H-0345****",`
	`  "8771****",`
	`  "EXAMPLE2 SL",`
	`  "****5028G"`
	`  ],`
	`}`

`# Query to get corpus generated for invoices.`
`PREFIX`
`obo: <http://ontologies.khaos.uma.es/obo/>`
`SELECT ?s ?concept_field`
`WHERE`
`  { ?s rdf:type PSD2:Statement.`
`  ?s obo:concept ?concept_field.`
`  }`
 	
`SELECT ?s ?concept_field`
`WHERE`
`  { ?s rdf:type PSD2:Statement.`
`  ?s obo:statementBeneficiary`
`  ?concept_field`
`}`
	`{`
	`  "obo_statement06ae705c`
	`  -75f6-4dc0":[`
	`  "EXAMPLE COMPANY S.L.",`
	`  "TRANSFERENCIA INMEDIATA",`
	`  "ref 300XX",`
	`  "BBVAES",`
	`  "Example company"`
	`  ],`
	`  "obo_statement06fdf166`
	`  -8dc6-4cc9":[`
	`  "CON:PAGO FACTURAS",`
	`  "VT. XX-XX-2020",`
	`  "Example company 2, SL"`
	`  ]`
	`}`

Once the two-word corpora are built, a fuzzy search RapidFuzz algorithm (https://maxbachmann.github.io/RapidFuzz/ (accessed on 26 November 2022)) [24] for string similarity is applied to calculate semantic matching between invoices and bank statements. This algorithm is configured with a Leverstein distance metric and a confidence threshold of 80% similarity to filter unnecessary matches. This threshold was set after a series of preliminary tuning tests until an adequate level of similarity was reached, taking into account the expert knowledge domain. Table 9 shows a set of two (anonymized) highly scored results obtained from a total number of 225 matches (from >11 MM possible combinations).

The first column of the table indicates the semantic attributes, and the other columns show the semantic similarity values.

Table 9. Examples of highly scored semantic matching results obtained by the fuzzy search string similarity RapidFuzz algorithm.

Semantic Attribute	Semantic Matching 1	Semantic Matching 2
URI Item	itemacffa	item55214
URI Statement	statement55d2j	statementd27d4
Item Descriptor	EXAMPLE COMPANY, S.L.	SERVICE FRANCE S.L.U.
Statement Descriptor	A FAVOR DE EXAMPLE COMPANY S.L	SERVICE FRANCE, SL
Score	95	88

In addition, these filtered matches take into account the currency, as well as the difference between the maturity date of the invoice and the date of the movement, which should be less than one year since it is not expected that a client takes more than a year to pay an invoice. This is checked by means of SPARQL query shown in Code Listing 1, for which a sample of results is shown in Table 10, comprising two new results that had not been obtained previously.

Table 10. Results obtained after searching for the exact amount.

Semantic Attribute	Semantic Matching 1	Semantic Matching 2
URI Item	itemb0e44	item40f83
URI Statement	statement14984	statement41d07
Item Descriptor	1, EXP-JCR-100, CLIENT X	1, S07220G, CLIENT Y
Statement Descriptor	PETER, FACT 22000-543	CONTRACT 564, DOSSIER RISK
Amount	241.16	5596.00
Diff Days	−10	20

Listing 1. Match by an exact amount.

```
PREFIX obs: <http://ontologies.khaos.uma.es/obo/>
SELECT ?uri_item ?uri_statement ?amount ?date_statement ?date_item
WHERE{
FILTER(?amount = ?amount_item && ?date_item_diff >= ?date_statement
       && ?date_item <= ?date_statement)
   {SELECT ?uri_item ?uri_statement ?amount_item (?amount_statement/100 as ?amount)
         ?date_statement ?date_item ?date_item_diff
   WHERE{
       ?uri_item obs:itemCurrencyAmt ?amount_item.
       ?uri_statement obs:statementAmount ?amount_statement.
       ?uri_statement obs:statementValueDate ?date_statement.
       ?uri_item obs:itemMaturityDate ?date_item.
       ?uri_item obs:currencyCode ?currency.
       ?uri_statement obs:statementCurrency ?currency.
       values ?duration { "P1Y"^^xsd:duration }
       bind( (?date_item + ?duration) as ?date_item_diff)
     }
   }
}
```

Finally, further checking is addressed in this use case to find matches in the case of partial payments, e.g., in those situations where an invoice is paid with several movements and the opposite, when several invoices are paid with the same statement. In these cases, to calculate the possible sums of amounts, a Genetic Algorithm (GA)-based approach was used to find the optimal combinations of the sum of subsets. This algorithm was codified by means of the JMetalPy (https://jmetalpy.readthedocs.io/ (accessed on 21 November 2022)) [25] framework, implementing an Elitist GA variant with population size 100, SPX-Crossover and bit-flip mutation operators, selection and replacement by binary tournament, and stop condition with 25,000 fitness evaluation functions.

A clear sample result from this context is shown in Table 11, where the GA finds an invoice whose amount (1039.88 €) exactly matches the sum of the two corresponding bank statements (800 € and 239.88 €). Differently, Table 12 contains a sample matching in which a bank movement amount (21,913.10 €) is used to pay the corresponding two invoices (16,637.50 € and 5275.60 €). It is worth noting that, without the previous semantic reconciliation concerning items and statements descriptions, these numerical matchings would be highly complex because of the number of possible combinations in the search space.

Table 11. Results of multiple movements for an invoice.

Semantic Attribute	Subset Sum Result	
URI Item	item23dcb	item23dcb
URI Statement	statement335ad	statement5eb2b
Item Descriptor	HAPPY FRIENDS, S.A	HAPPY FRIENDS, S.A
Statement Descriptor	HAPPY FRIENDS S.A.	HAPPY FRIENDS
Score	98	95
Item Currency Amt	1039.88	1039.88
Statement Amount	800.00	239.88
Currency Code	EUR	EUR
Diff Days	1	1

Table 12. Results of multiple invoices for a statement.

Semantic Attribute	Subset Sum Result	
URI Item	item24b8b	item3069d
URI Statement	statement55d2d	statement55d2d
Item Descriptor	HAPPY FRIENDS S.A	HAPPY FRIENDS, S.A
Statement Descriptor	HAPPY FRIENDS	HAPPY FRIENDS, S.A.
Score	98	95
Item Currency Amt	16637.50	5275.60
Statement Amount	21913.10	21913.10
Currency Code	EUR	EUR
Diff Days	1	1

4.2. Semantic Inference

This last use case is devoted to illustrating two inference tasks oriented towards allowing customer classification and concept suggestion by means of semantic reasoning.

First, the values of the *obs:debtRatio* data property of the *Customer* class are calculated by means of several SPARQL queries for each customer (*obs:itemHasCustomer*) and company (*obs:itemHasCompany*). The debt ratio is calculated with Equation (1), which is applied to the aggregation of the amounts (*obs:itemCurrencyAmt*) of all the items of the class *Opened* (these amounts are the outstanding items), divided by the sum of the amount of all the customer's items. Then, in accordance with the debt values, a set of SWRL semantic rules were defined to establish a classification of customers, as follows: the *Low* label is assigned to those customers with a debt ratio lower than 30% (The rule is shown in Code Listing 2), the *Medium* label is assigned to those with a debt ratio between 30% and 70% (The rule is shown in Code Listing 3), and finally, the *High* label is assigned to customers with a debt ratio higher than 70% (The rule is shown in Code Listing 4).

$$debtRatio = \frac{\sum amount\ opened}{\sum amount\ cleared + \sum amount\ opened} * 100 \quad (1)$$

Listing 2. Low debt.

```
obs:Customer(?c) ^ obs:debtRatio(?c, ?dRatio) ^ swrlb:lessThan(?dRatio, 30)
-> obs:debtType(?c, "Low")
```

Listing 3. Average debt.

```
invoice:Customer(?c) ^ obs:debtRatio(?c, ?dRatio) ^ swrlb:lessThan(?dRatio, 70)
^ swrlb:greaterThanOrEqual(?dRatio, 30) -> obs:debtType(?c, "Average")
```

Listing 4. High debt.

```
ons:Customer(?c) ^ obs:debtRatio(?c, ?dRatio)
^ swrlb:greaterThanOrEqual(?dRatio, 70) -> obs:debtType(?c, "High")
```

Therefore, the value of data property *obs:debtType* is inferred by the reasoner in the proposed semantic model, hence allowing an additional population of this element in the RDF repository. In this way, to obtain a given customer's classification according to his/her debt with a particular company, the SPARQL Query shown in Code Listing 5 can be used (this refers to the specific case of customer ID 367 with company ID 226).

Listing 5. Debt Type.

```
PREFIX obs: <http://ontologies.khaos.uma.es/obo/>
SELECT ?debtType
WHERE{
    ?item obs:itemHasCompany ?c. FILTER(?c=company226)
    ?item obs:itemHasCustomer ?u. FILTER(?u=customer367)
    ?u obs:debtType ?debtType
}
```

Second, taking advantage of the information stored in the knowledge graph, it is possible to infer concepts suggested for different bank movements. To this end, the SWRL Rule shown in Code Listing 6 was defined to merge information from classes *Opened* and *Customer*, comprising the data properties that annotate the customer's fiscal identification number (*fiscalId*), the maturity date of the item (*itemMaturityDate*), and the amount of the closed item (*itemAmount*). The semantic reasoner is then used to concatenate these values and to form the *suggestedConcept* data property, thereby reconciling bank movements.

Listing 6. Suggested concept.

```
obs:Opened(?o) ^ obs:Customer(?c) ^ obs:fiscalId(?c, ?fId)
^ obs:itemMaturityDate(?o, ?mDate) ^ obs:itemHasCustomer(?o, ?c)
^ obs:itemAmount(?o, ?a) ^ swrlb:stringConcat(?con, ?fId, ?mDate, ?a)
-> obs:suggestedConcept(?o, ?con)
```

An example of a SPARQL query for this use case is shown in Query shown in Code Listing 7, where, given the URI of an individual of the *Opened class*, the reasoner is able to calculate the value for the *suggestedConcept* property.

Listing 7. Example of suggested concept query.

```
PREFIX obs: <http://ontologies.khaos.uma.es/obo/>
SELECT ?concept
WHERE{
    ?i rdf:type obs:Opened. FILTER(?i = obs:item843)
    ?i obs:suggestedConcept ?concept
}
```

5. Conclusions

In this work, the Open Bank Ontology (OBO) is proposed for the semantic annotation and consolidation of data involved in PSD2 transactions between customers, banks and third-party financial entities. A semantic model is also used for knowledge graph construction and data harmonization in the context of a real-world open banking project (Helix), enabling the definition of SPARQL queries and SWRL rules for reasoning. For validation purposes, a series of mapping mechanisms, queries, and intelligent algorithms were also

developed with a special focus on data reconciliation, which seeks to enable automatic correspondence between bank statements of customer payments and the corresponding invoices delivered by companies. In addition, a set of reasoning SWRL inference rules were defined to obtain new knowledge from the integrated data, with the specific aims of automatically classifying customers according to their debt patterns and standardizing the suggested concepts for invoices.

In terms of practical implications, as already mentioned, payment processes such as PSD2 are stimulating the emergence of applications capable of greatly simplifying the day-to-day management and procedures of SMEs and users. This translates into the creation of systems capable of integrating all of a company's payments within the company, as well as providing access to the different bank accounts held by the company, even if they are held at different banks. The key idea lies in integrating all corporate areas into the same interface, which allows for a much clearer perspective on the company's status. In this sense, the proposed ontology enables the constitution of a semantic model that can facilitate the integration of all these functionalities, as well as other external yet important factors (such as the impact of COVID-19 [22]), as it offers a common data framework that is also oriented towards being consistent with open banking standards.

In the same regard, it is worth noting that the OBO ontology constitutes the first attempt to model bank movements and activities with a special focus on the European PDS2 standard. Therefore, new extensions are expected to follow this work, including Linked Data federation with other related knowledge graphs in the domain of Fintech applications and standards.

In future work, we plan to integrate more data from different invoice management systems and update the OBO ontology to incorporate new relevant attributes from different perspectives, such as opinions on social networks, behavioral features of the company in its commercial relations with customers, etc. This new knowledge will allow further analyses to be carried out, taking into account further factors and actors.

Author Contributions: Conceptualization, M.P. and M.d.M.R.-G.; methodology, M.P. and M.d.M.R.-G.; software, M.P.; validation, M.P., M.d.M.R.-G. and J.G.-N.; investigation, M.P.; data curation, M.P.; writing—original draft preparation, M.P., M.d.M.R.-G. and J.G.-N.; writing—review and editing, M.P., M.d.M.R.-G. and J.G.-N.; supervision, M.d.M.R.-G. and J.G.-N.; project administration, M.d.M.R.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partially funded by the Spanish Ministry of Science and Innovation via the Aether Project with grant number PID2020-112540RB-C41 (AEI/FEDER, UE), the Ministry of Industry, Commerce and Tourism via the Helix initiative with grant number AEI-010500-2020-34, and the Andalusian PAIDI program with grant number P18-RT-2799.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Appendix A

This appendix section includes anonymized fragments of the data returned by the open banking API. The snippets show the structure and fields that define the documents in JSON format. The data includes information identifying the customers and the movements associated with the banking products contracted by the customer with the bank.

Listing A1. "Customer.json" sample (anonymized) file accessed from PSD2-compliant open banking API.


```
{
   "customers":[
      {
         "_id":"A18XXXXXX",
         "address":[
            "LUIS PASTEUR 29071,MALAGA, MALAGA,SPAIN"
         ],
         "document":"A18XXXXXX",
         "birth_date":null,
         "names":"EXAMPLE SA",
         "phones":[
            "9*****500",
            "9*****063",
            "9*****062",
            "9*****010"
         ],
         "accounts_obj":[
            {
               "id":"ESXXXXXXXXXXXXXXXXXXXX5",
               "participation":"Titular"
            },
            {
               "id":"ESXXXXXXXXXXXXXXXXXXXX0",
               "participation":"Titular"
            }
         ],
         "emails":[
            "c*****@******.es"
         ]
      }
   ]
}
```


Listing A2. "Statement.json" sample (anonymized) file accessed from PSD2-compliant open banking API.


```
{
   "statements":[
      {
         "_id":"476XXXX",
         "value_date":"2019-XX-XX",
         "deposit_date":"2019-XX-XX",
         "amount":264XXX,
         "beneficiary":"",
         "bank_reference":"0000XXX",
         "balance":419XXX,
         "currency":"EUR",
         "import_date":"2019-XX-XXT07:30:41+00:00",
         "concepts":[
            "AUTO, S.L.",
            "TRANSFER. EN DIV.",
            "TRANSF.DIVISAS",
            "Origen: CRD",
            "REFERENCIA 204XX"
         ],
         "bank_id":15,
         "bank_group_id":1,
         "account":"ESXXXXXXXXXXXXXXXXXXXX4",
         "bank_name":"Santander",
         "category":null,
         "office":"3XXX"
      }
   ]
}
```

References

1. Polasik, M.; Huterska, A.; Iftikhar, R.; Mikula, Š. The impact of Payment Services Directive 2 on the PayTech sector development in Europe. *J. Econ. Behav. Organ.* **2020**, *178*, 385–401. [CrossRef]
2. Gordon, T.F. An Overview of the Legal Knowledge Interchange Format. In Proceedings of the Business Information Systems Workshops, Berlin, Germany, 3–5 May 2010; Abramowicz, W., Tolksdorf, R., Węcel, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 240–242.
3. del Mar Roldán García, M.; García-Nieto, J.; Aldana-Montes, J.F. An ontology-based data integration approach for web analytics in e-commerce. *Expert Syst. Appl.* **2016**, *63*, 20–34. [CrossRef]
4. Singarayan, T.; Jeganathan, A.; Leema, G. Semantic Integration of Classical and Digital Libraries. In *Multimedia Information Extraction and Digital Heritage Preservation*; World Scientific: Singapore, 2011; pp. 51–65. [CrossRef]
5. Sobral, T.; Galvao, T.G.; Borges, J. An Ontology-based approach to Knowledge-assisted Integration and Visualization of Urban Mobility Data. *Expert Syst. Appl.* **2020**, *150*, 113260. [CrossRef]
6. del Mar Roldán-García, M.; Uskudarli, S.; Marvasti, N.B.; Acar, B.; Aldana-Montes, J.F. Towards an ontology-driven clinical experience sharing ecosystem: Demonstration with liver cases. *Expert Syst. Appl.* **2018**, *101*, 176–195. [CrossRef]
7. Castells, P.; Foncillas, B.; Lara, R.; Rico, M.; Alonso, J.L. Semantic Web Technologies for Economic and Financial Information Management. In Proceedings of the The Semantic Web: Research and Applications, Crete, Greece, 10–12 May 2004; Bussler, C.J., Davies, J., Fensel, D., Studer, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 473–487.
8. Bennett, M. The financial industry business ontology: Best practice for big data. *J. Bank. Regul.* **2013**, *14*, 255–268. [CrossRef]
9. Tang, H.; Song, L. Ontologies in financial services: Design and applications. In Proceedings of the International Conference on Business Management and Electronic Information, Guangzhou, China, 13–15 May 2011; Volume 5, pp. 364–367. [CrossRef]
10. Lara, R.; Cantador, I.; Castells, P., Semantic Web Technologies For The Financial Domain. In *The Semantic Web: Real-World Applications from Industry*; Cardoso, J., Hepp, M., Lytras, M.D., Eds.; Springer: Boston, MA, USA, 2008; pp. 41–74. [CrossRef]
11. Bhasuran, B.; Murugesan, G.; Abdulkadhar, S.; Natarajan, J. Stacked ensemble combined with fuzzy matching for biomedical named entity recognition of diseases. *J. Biomed. Inform.* **2016**, *64*, 1–9. [CrossRef] [PubMed]
12. Sehgal, N.; Crampton, A. Information Extraction for Additive Manufacturing Using News Data. In Proceedings of the International Conference on Advanced Information Systems Engineering, Cairo, Egypt, 23–25 August 2019; pp. 132–138. [CrossRef]
13. Hitzler, P.; Krötzsch, M.; Parsia, B.; Patel-Schneider, P.F.; Rudolph, S. *OWL 2 Web Ontology Language Primer*, 2nd ed.; Available online: https://www.w3.org/TR/owl2-primer/ (accessed on 19 November 2022).
14. Staab, S.; Studer, R. *Handbook on Ontologies (International Handbooks on Information Systems)*; Springer: Berlin/Heidelberg, Germany, 2004.
15. Gruber, T.R. A translation approach to portable ontology specifications. *Knowl. Acquis.* **1993**, *5*, 199–220. [CrossRef]
16. Ehrlinger, L.; Wöß, W. Towards a Definition of Knowledge Graphs. *Semantics* **2016**, *48*, 2.
17. Bechhofer, S.; Harmelen, F.V.; Hendler, J.A.; Horrocks, I.; McGuinness, D.L.; Patel-Schneider, P.F.; Stein, L.A. OWL Web Ontology Language Reference. Available online: https://www.w3.org/TR/owl-ref/ (accessed on 19 November 2022).
18. Haase, P.; Stojanovic, L. Consistent Evolution of OWL Ontologies. In Proceedings of the European Semantic Web Conference, Crete, Greece, 29 May–1 June 2005; Volume 3532, pp. 182–197. [CrossRef]
19. Fischer-Pauzenberger, C.; Schwaiger, W. The OntoREA Accounting Model: Ontology-based Modeling of the Accounting Domain. *Complex Syst. Inform. Model. Q.* **2017**, *11*, 20–37. [CrossRef]
20. Guizzardi, G.; Wagner, G.; Almeida, J.P.; Andradea Guizzardi, R.S. Towards Ontological Foundations for Conceptual Modeling: The Unified Foundational Ontology (UFO) Story. *Appl. Ontol.* **2015**, *10*, 259–271. [CrossRef]
21. Blums, I.; Weigand, H.H. Towards a Core Ontology for Financial Reporting Information Systems (COFRIS). In Proceedings of the OTM Workshops, Rhodes, Greece, 23–27 October 2017.
22. Patel, A.; Debnath, N.; Mishra, A.; Jain, S. Covid19-IBO: A Covid-19 Impact on Indian Banking Ontology Along with an Efficient Schema Matching Approach. *New Gener. Comput.* **2021**, *39*, 647–676. [CrossRef] [PubMed]
23. Noy, N. Ontology Development 101: A Guide to Creating Your First Ontology. Available online: https://protege.stanford.edu/publications/ontology_development/ontology101.pdf (accessed on 18 November 2022).
24. Hyyrö, H. Bit-Parallel LCS-length Computation Revisited. In Proceedings of the 15th Australasian Workshop on Combinatorial Algorithms (AWOCA), Ballina Beach Resort, NSW, Australia, 7–9 July 2004; pp. 16–27.
25. Benítez-Hidalgo, A.; Nebro, A.J.; García-Nieto, J.; Oregi, I.; Del Ser, J. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. *Swarm Evol. Comput.* **2019**, *51*, 100598. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.