



Article

Automatic Semantic Modeling for Structural Data Source with the Prior Knowledge from Knowledge Base [†]

Jiakang Xu ^{1,2,3,4}, Wolfgang Mayer ⁵, Hongyu Zhang ^{2,3,4,*} , Keqing He ⁶ and Zaiwen Feng ^{1,2,3,4,7,8,*} 

¹ National Key Laboratory of Crop Genetic Improvement, Huazhong Agricultural University, Wuhan 430070, China

² Hubei HongShan Laboratory, Huazhong Agricultural University, Wuhan 430070, China

³ College of Informatics, Huazhong Agricultural University, Wuhan 430070, China

⁴ Hubei Key Laboratory of Agricultural Bioinformatics, Huazhong Agricultural University, Wuhan 430070, China

⁵ Industrial AI Research Centre, University of South Australia, Mawson Lakes, SA 5095, Australia

⁶ School of Computer, Wuhan University, Wuhan 430072, China

⁷ State Key Laboratory of Hybrid Rice, Wuhan University, Wuhan 430072, China

⁸ Macro Agricultural Research Institute, Huazhong Agricultural University, Wuhan 430070, China

* Correspondence: zhy630@mail.hzau.edu.cn (H.Z.); zaiwen.feng@mail.hzau.edu.cn (Z.F.)

[†] This paper is an extended version of our paper published in 2021 IEEE International Conference on Data, Information, Knowledge and Wisdom (DIKW), Haikou, Hainan, China, 20–22 December 2021; pp. 2034–2041.

Abstract: A critical step in sharing semantic content online is to map the structural data source to a public domain ontology. This problem is denoted as the Relational-To-Ontology Mapping Problem (*Rel2Onto*). A huge effort and expertise are required for manually modeling the semantics of data. Therefore, an automatic approach for learning the semantics of a data source is desirable. Most of the existing work studies the semantic annotation of source attributes. However, although critical, the research for automatically inferring the relationships between attributes is very limited. In this paper, we propose a novel method for semantically annotating structured data sources using machine learning, graph matching and modified frequent subgraph mining to amend the candidate model. In our work, *Knowledge graph* is used as prior knowledge. Our evaluation shows that our approach outperforms two state-of-the-art solutions in tricky cases where only a few semantic models are known.

Keywords: semantic model; frequent subgraph mining; knowledge graph; ontology

MSC: 68P15



Citation: Xu, J.; Mayer, W.; Zhang, H.; He, K.; Feng, Z. Automatic Semantic Modeling for Structural Data Source with the Prior Knowledge from Knowledge Base. *Mathematics* **2022**, *10*, 4778. <https://doi.org/10.3390/math10244778>

Academic Editor: Weihua Xu

Received: 20 November 2022

Accepted: 13 December 2022

Published: 15 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 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

Structural data sources are still one of the most prevalent modes for the storage of enterprise or Web data. It is a long-standing and urgent issue in many real database research fields to automatically integrate heterogeneous data sources [1,2]. Though relational schemata are suitable for ensuring data integrity, they are short of semantic descriptions that support efficient semantic integration of different sources. The construction of ontology from relational databases is a basic problem for the development of the Semantic Web [3]. Common ontology provides a method to express the semantics of a relational schema and facilitate integrating heterogeneous data sources. It is appropriate to manually indicate semantic descriptions if the integration of only a few data sources is required. However, as the number of heterogeneous schemata increases, manually labeling becomes tedious. To settle this problem, a standard way is to design a common ontology and build a source description of the assigned mappings between the sources and the ontology automatically [4]. This problem is named as Relational-To-Ontology Mapping Problem (*Rel2Ont*) [5].

Formally, the problem of *Rel2Ont* can be expressed as below [5]. Assume a data source s is an n -ary relation which contains a series of attributes $\mathcal{A}_s = (a_1, \dots, a_n)$. The *attribute*

mapping function $\phi : \mathcal{A}_s \mapsto \mathcal{D}_m$ constructs the mapping between the attributes of the sources s and the node set \mathcal{D}_m of the semantic model m . Given a source s , a semantic model m , and an attribute mapping ϕ , a source description is defined as a triple $\delta = (s, m, \phi)$. Suppose we have a gold standard model for a new source s^* , given an ontology \mathcal{O} , and a set of source descriptions $\Delta = \{(s_1, m_1, \phi_1), \dots, (s_l, m_l, \phi_l)\}$, for a new source s^* . How do we derive the semantic model m^* and the attribute mapping function ϕ^* so that $\delta^* = (s^*, m^*, \phi^*)$ maximizes the *precision* and *recall* between the semantic model m^* and the *gold standard* semantic model m^\dagger of the data source s^* ?

Lately, machine learning techniques have been employed to address the Re120nt issue. For instance, Karma [6] can automatically learn semantic models for a new data source by leveraging the knowledge from domain ontology and historical semantic models of sources in the same domain. Binh Vu et al. [7] proposed a novel way to learn semantic models for data sources by using a probabilistic graphical model (PGM). Recently *knowledge graphs*, as one of the main trends driving the next wave of technologies [8], have become a novel form for representing knowledge and the basis of multiple applications from common applications to specific industrial use cases [9]. A knowledge graph is a structured representation of facts consisting of entities, relationships, and semantic descriptions [10]. Relationships among semantic models of a data source can be inferred by exploiting a knowledge graph as prior knowledge [11]. For example, Giuseppa et al. developed a semi-automatic tool SeMi for constructing large-scale knowledge graphs from structured data sources by building the semantic models of data sources [12].

These automatic semantic annotation methods based on machine learning enormously elevate the data matching efficiency of structural data sources. However, these strategies also have the following drawbacks: (a) Limited known semantic models of data sources: the performance of Karma is better when plenty of data sources are available for training a standard learning graph. For instance, when Karma uses 29 known museum data sources to train the learning graph, it can generate a candidate semantic model for the new museum data source with an accuracy of up to 80% [6], whereas in practical applications, the data sources might be limited, perhaps to 2 or 3, so the existing methods still need to be greatly improved. (b) Lack of linked data: when inferring semantic relations from knowledge graphs based on machine learning methods [11], it is assumed that there are adequate linked data usable in the same domain as the destination data source, which greatly depends on the amount of linked data. If there is little or no linked data available, the capacity to dig available patterns will be greatly reduced. (c) Finite ability to extract long patterns: the calculation of inferring patterns from the method of inferring semantic relationships from knowledge graphs is very complicated [11]. In a reasonable time, only the patterns of three or four nodes may be inferred. Therefore, it is challenging to use SPARQL queries to extract long patterns from a lot of triples.

In this article, we extend our previous work [13] and present a novel machine-learning-based procedure, which is helpful to figure out the Re120nto issue with the prior knowledge of the knowledge graph. For this reason, we attribute the Re120nto problem to a customized frequent subgraph mining problem. Primarily, it runs the Steiner tree generation algorithm to output reasonable semantic models by utilizing existing semantic models and domain ontology [6]. We select the first ranked candidate model cm as the seed model for further amending. Then, we remove incorrect relationships of cm by using a knowledge graph as prior knowledge and machine learning techniques. Since some relationships are removed from the initial candidate semantic model, the resulting model may be incomplete. Accordingly, to improve the completeness of the model, we use a *grow-and-store* strategy [14] to mine the top- σ frequent subgraphs in the knowledge graph as final candidate models. The underlying *heuristic* hypothesis is that the correct semantic model may have higher frequency in the knowledge graph than other substructures.

The contributions of our paper are as follows: (a) A new pipeline for automatically learning the semantic model of a new structural data source is arranged by utilizing several existing semantic models, domain ontologies, and domain specific knowledge graphs. (b) A novel approach is put forward for discovering and eliminating the incorrect relationships

in candidate semantic models by machine learning and graph matching techniques. (c) We use the *grow-and-store* approach and modify a frequent subgraph mining algorithm to calculate the subgraph frequency of a domain-specific knowledge graph and mine frequent subgraphs. The top- σ frequent subgraphs are acquired as the most rational semantic models of the structured data source.

The rest of this article is structured as below. In Section 2, we represent related work. We introduce an illustrative example in Section 3. We exploit a new way for inferring semantic relations of the destination data source in Section 4. In Section 5, we display our experimental evaluation results of our method and draw conclusions in Section 6.

2. Related Work

Since the manual creation and scheme of the mapping between relations to ontologies is a labor-intensive process, several *machine learning* technologies have been suggested to tackle the Re120nt problem. Taheriyani et al. presented a method to automatically learn a semantic model of a new source utilizing domain information and historical semantic models [6]. Considering lack of known available semantic model in many domains, Taheriyani et al. further put forward a method to automatically learning the semantic relationships within a given data source exploiting Linked Open Data (LOD) [11]. The limitation of [11] is that when the available LODs are sparse, the exactness of the outputted semantic models is severely affected and not enough useful patterns are obtained. Diego et al. [5] combined machine learning with constraint programming to infer mapping rules from previous mapping instances to deal with attributes that cannot be matched to the ontology. Binh et al. [7] presented a method for automatically learning semantic models using a probabilistic graphical model. Their approach is more robust to noise than previous methods. Giuseppe et al. proposed a semi-automatic approach for inferring semantic relations based on a graph neural network trained on a background knowledge graph [12]. In our article, we learned a semantic model for a new source by integrating the knowledge of known semantic mappings, knowledge graphs, and domain ontology. Our approach helps to study the exact semantic models without sufficient historical mappings. We studied this problem in our previous work [13]. Comparing to our previous work, the following four points are strengthened in this article. First, multiple candidate semantic types for each attribute of structural data source are considered, while in our previous work, we assumed that all the correct semantic types of attributes are known. Second, we eliminated the incorrect relationships in *seed model* by using two different methods in the pipeline to improve the accuracy of our method. Third, we optimized our previous algorithm for adding missing substructures to improve its efficiency. Finally, we conducted more experiments to evaluate our approach by using more datasets. Compared to our previous work with only one dataset and one semantic modeling approach being and evaluated, in this article, we compared our approach with two state-of-the-art semantic modeling methods by exploiting three datasets.

Search-based approaches were also exploited to tackle the Re120nt problem. Pintel et al. proposed a new semi-automatic matching method *IncMap* [15], which used the lexical and structural similarities between ontology and relational schemata to generate the mapping from relational schemata to ontology. Sequeda et al. defined a specific query mapping QODI [16] for an ontology-based data integration system (OBDI). QODI generates path correspondences, rather than entity correspondences, to facilitate the representation of queries. Moreover, Sequeda et al. exploited a semi-automatic software Ultrawrap Mapper [17] for the creation of a mapping from Relational Databases to RDF in the R2RML language. Ultrawrap Mapper aligns the original schemata and target ontology using QODI techniques and gives mapping suggestions [17]. The MIRROR system [18] yields an R2RML mapping file containing the mappings for a given relational database. Naglaa et al. proposed a ProGOMap (Property Graph to Ontology Mapper) system for the automatic generation of mappings from property graphs to a domain ontology [19]. The PG-to-Ontology mappings can be automatically generated by using the aligned axioms. Florian et al. developed a prototype for a semantic data lake for addressing the heterogeneous format of the activity logs

and the content data of cross-platform collaboration [20]. In their prototype, ontology-based data access is implemented based on a mapping between an ontology and the ingested data. All of the approaches in [15–20] focus on constructing mappings between relational scheme and domain ontology through a search-based algorithm. Different from these search-based approaches, our method attempts to learn mapping rules by leveraging the amount of knowledge bases, including not only historical relational schemes and domain ontology, but also knowledge graphs.

As mentioned above, *semantic labeling* is a significant step for solving the Re120nt problem. Several works for addressing the problem of semantic labeling exist. For instance, Krishnamurthy et al. [21] tried to leverage the distribution and characteristic properties of the data for learning semantic types of source attribute. Pham et al. [22] used machine learning technologies to infer the correct semantic type by calculating similarities between unlabeled and labeled attributes. Mulwad et al. [23] leveraged Linked Open Data (LOD) information to build up the known semantic message algorithm for the annotation of tables on the web.

3. Illustrative Example

In this section, a typical procedure for building a semantic model of a sample data source from the Crystal Bridges Museum in the United State (CB) <https://crystalbridges.org/> (accessed on 14 December 2022) is provided. The gold semantic model of the CB is exhibited in Figure 1. In the article, the *correct* model of a data source means the gold standard model of this source.

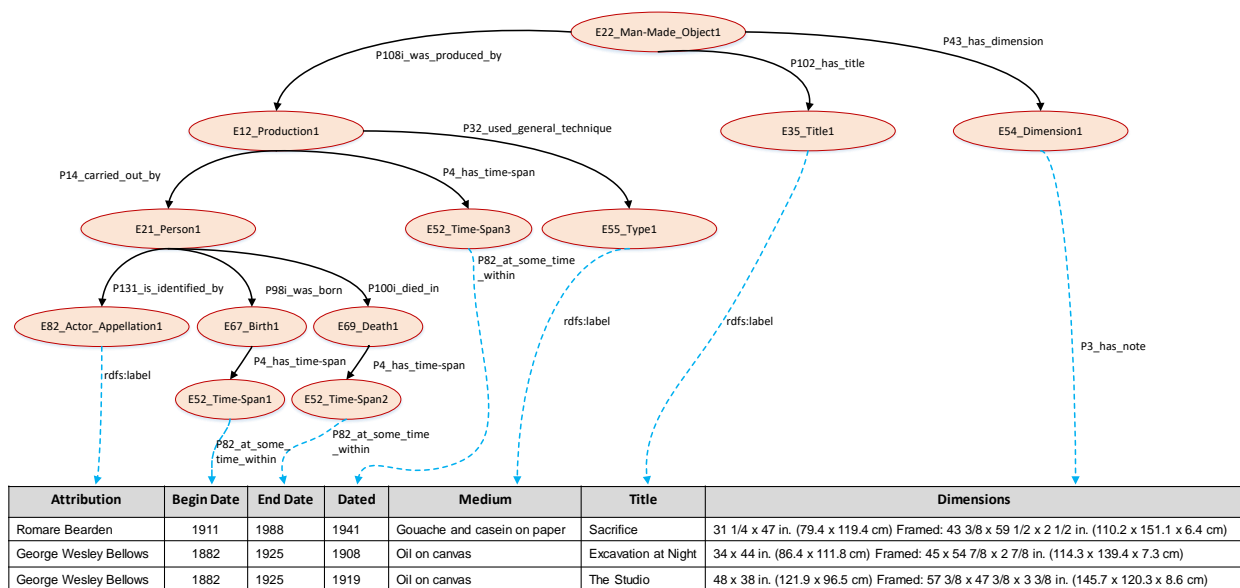


Figure 1. CB data source and its semantic model.

The semantic model m is a directed graph which includes two categories of nodes: *class nodes* (C_m) and *data nodes* (D_m). C_m is the set of classes in the ontology, and D_m corresponds to data properties. The term semantic model is synonymous with semantic mapping with a minor difference. Semantic mapping is a schema mapping from the data source to an ontology. This mapping can be represented as a semantic graph which is also called a semantic model. As Figure 1 shows, the class nodes are ontology classes, and the data nodes are source attributes. For example, in the seed model of CB, the entities E55_Type1 and E54_Dimension1 are class nodes, and data source attributes Medium and dimensions are data nodes. The edges of the semantics model can be classified into *object properties* and *data properties*. In Figure 1, object properties defined in the ontology are shown by the black-colored links between class nodes. Furthermore, data properties (shown in blue

color) are relationships between data nodes and class nodes. For instance, `P102_has_title` and `rdfs:label` are object properties and data links, respectively.

Suppose that we only have two data sources whose gold standard semantic models are given. The data sources are the tables describing the information about artworks in the National Portrait Gallery (NPG) and Getty: Resources for Visual Art and Cultural Heritage (GT), respectively. Our goal is to learn a semantic model for a new source such as CB by leveraging a small number of historical semantic mappings and the domain ontology and using a knowledge graph as background knowledge. In the next section, we present our method for automatically learning a semantic model for a new source with a few known semantic models and a knowledge graph.

4. Our Approach

In this section, we introduce our method for automatically inferring the semantic models of a structural data source. The whole pipeline of this method is illustrated in Figure 2. The domain ontology, several historical semantic mappings from data sources to domain ontology, a new data source, and domain knowledge graphs are the inputs of our approach. The overall pipeline consists of two phases. In the first phase, we find candidate semantic types for each attribute and find a candidate semantic model *cm* for a new data source by using a Steiner tree algorithm. A *cm* is usually a partially correct model. In the second phase, we amend the *cm* to eliminate incorrect entities and relationships. We first train a decision tree model to distinguish ambiguous relationships and then use a graph matching technique to remove incorrect relationships. Next, a modified frequent subgraph mining algorithm is used to add missing substructures. The output of our approach is a semantic model that describes how the specified semantic types are linked with the seed model.

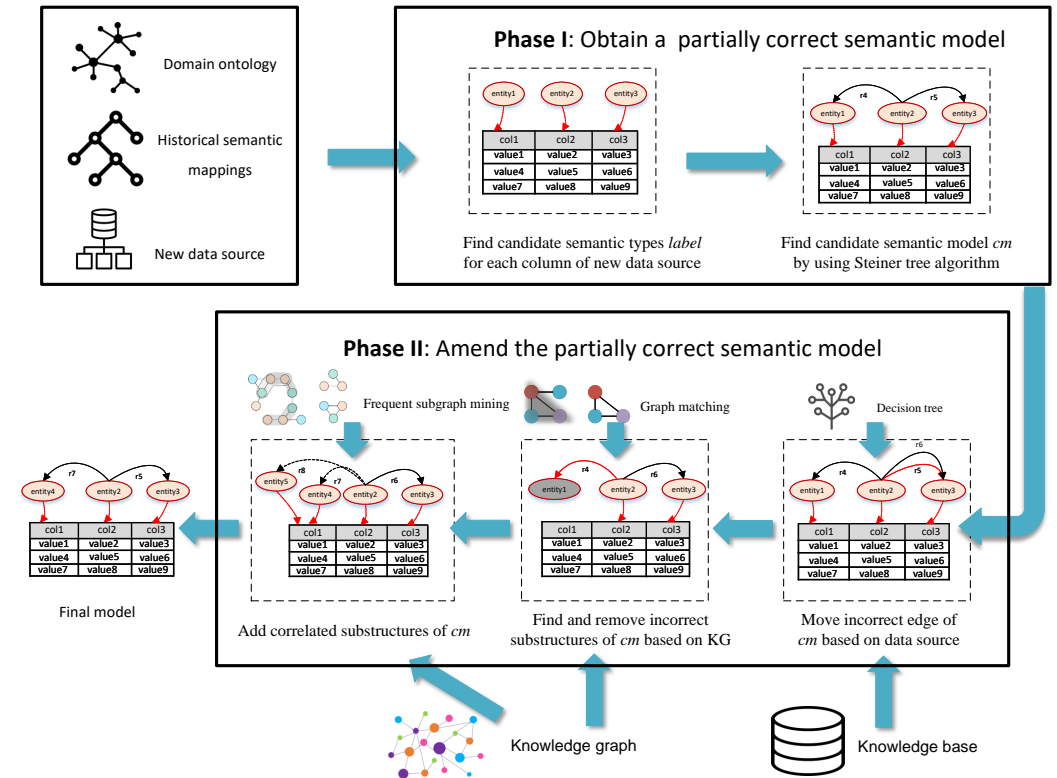


Figure 2. The overall pipeline.

4.1. Obtaining the Seed Semantic Model

The first phase abides by the classical solution [4] of the Re120nt problem which includes two sub-steps, i.e., *semantic labeling* and *relationship discovery*. Semantic labeling [24] is

the process of annotating the semantic type for an attribute by scoring a confidence value for the assignment of an attribute from s to a type $l \in L_o$ (L_o is the set of all possible candidate semantic types). In this work, we use the approach proposed by Krishnamurthy et al. [21] which is called *SemanticTyper* to automatically obtain the semantic type of source attributes. Semantic types for each attribute of a new data source can be labeled automatically by using a semantic labeling function trained from a set of manually annotated sources.

Relationship discovery is the process of linking all the semantic annotations with multiple relationships to formulate a semantic model. We find all the relationships and build a candidate model cm by using a Steiner tree algorithm. The relationships of the matched data nodes (attributes) are identified for the generation of the semantic models $T^* = \{T^1, T^2, \dots, T^k\}$ of data sources. First, a directed weighted graph $\mathcal{G}_O = (\mathcal{V}_O, \mathcal{E}_O)$, called *alignment graph*, is constructed on top of the historical semantic mappings and expanded using semantic types \mathcal{L}_O and the ontology \mathcal{O} . The alignment graph provides an integrated view on top of the historical semantic descriptions δ_T . Similar to a semantic model, both class and data nodes are contained in \mathcal{G}_O . Note that the alignment graph is weighted by a weighting function $w_O : \mathcal{E}_O \mapsto \mathcal{R}$ so that edges inferred from the ontology have higher weights than the edges shown in the known semantic models. The details of the algorithm and weighting function are illustrated in [6]. Then, the top- σ candidate semantic models are acquired from the learned semantic types and alignment graph by solving the Steiner Tree Problem (STP) [5,6]. Given a graph $G = (V, E)$ and a subset of its nodes $T \subseteq V$, a Steiner Tree $G_s = (V_s, E_s)$ ($T \subseteq V_s \subseteq V$ and $E_s \subseteq E$) is a subtree of G which contains all the nodes in T and may include extra nodes from V to guarantee the connectedness. The Steiner Tree Problem (STP) can be described as finding the Steiner Tree which has the minimum sum of the weights of the edges in E_s with given graph G and a weight function $w_f : E \mapsto \mathcal{R}$ [25]. For leveraging a STP to formulate the Re120nto schema mapping problem for a new source s^* , we apply the method proposed in [6] and build the alignment graph $\mathcal{I}_O^{S^*} = (\mathcal{V}_O^{S^*}, \mathcal{E}_O^{S^*})$, where $\mathcal{V}_O^{S^*} = (\mathcal{V}_O, \mathcal{A}_{S^*})$. The set of edges $\mathcal{E}_O^{S^*}$ consists of \mathcal{E}_O and $\mathcal{M}_O^{S^*}$ (i.e., $\mathcal{E}_O^{S^*} = \mathcal{E}_O \cup \mathcal{M}_O^{S^*}$), where \mathcal{E}_O is the set of all edges in the alignment graph, and $\mathcal{M}_O^{S^*}$ represents the edges which connect each attribute of s^* to the nodes in the alignment graph induced by the semantic types (i.e., the set of nodes in $\mathcal{D}_{\mathcal{G}_O}$). For example, in Figure 3, the blue-colored dashed lines are the set $\mathcal{M}_O^{S^*}$. Next, a weighting function $w_I : E \mapsto \mathcal{R}^+$ is associated with the alignment graph. Here, we weight the edges of the alignment graph by the weighting function in [6]. After weighting edges, we use an approximation algorithm of STP, such as the BANKS algorithm [26], to build a set of subgraphs $T^* = (V^*, E^*)$ of the alignment graph $\mathcal{I}_O^{S^*}$ for the new source s^* . The output of the Steiner tree algorithm is the top- σ candidate semantic models. Since our approach attempts to amend one partially correct semantic model, we only select the first candidate semantic model cm with the lowest weight as the seed model for further amending.

Figure 3 shows the seed model of the data source CB generated by the method in [6] using the known semantic model of source NPG and source GT. This seed model will be used as the start of amending in the next sub-section.

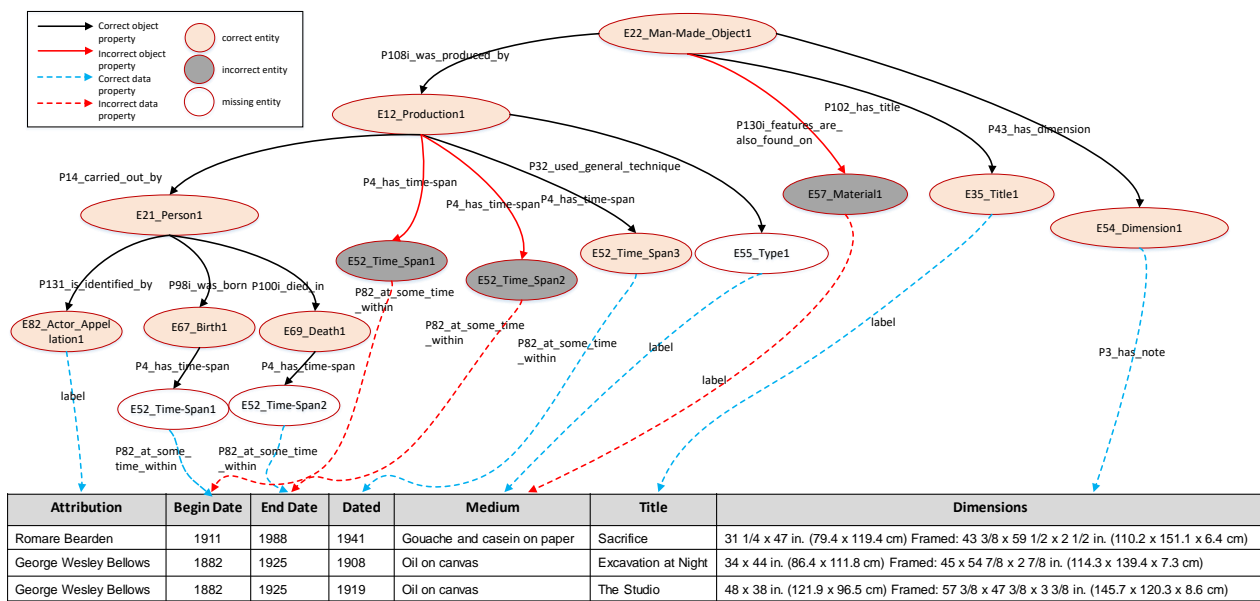


Figure 3. The seed model of CB. The incorrect entities are in the gray background, and incorrect relationships are colored red. The missing entities are on a white background

4.2. Amending the Seed Model

Compared with the gold semantic model, some substructures may be missing, and some wrong relationships may appear in the seed model. For example, as Figure 3 shows, in the seed model of CB, the red-colored relationships, i.e., (E22_Man-Made_Object1, P130i_features_are_also_found_on, E57_Material1), etc., are improper and should not appear in the semantic model. To improve the quality of the seed model, these incorrect relationships must be eliminated from seed models. In the meantime, compared to the gold standard model, the seed model lacks three entities (which are in the white background), i.e., E52_Time-Span1, E52_Time-Span2, E55_Type1 and their corresponding incoming links.

In this section, we present a way to modify the seed model to move or remove faulty relationships and add missing relationships to the model. Here, the meaning of moving an erroneous relationship is to attach the relationship to a node in the graph. We propose two approaches to remove or move potentially incorrect relationships. First, we use machine learning techniques to distinguish some ambiguous relationships based on the data source. Then, some incorrect substructures of cm can be detected through matching model fragments in a knowledge graph. After removing or moving incorrect relationships, we add potentially missing substructures of cm by using a modified frequent subgraph mining algorithm. As a result, a high-quality semantic model is obtained.

4.2.1. Move Incorrect Relationships

One entity may be linked to multiple other entities by various relationships. For example, the entity E52_Time-Span can be linked with E12_Production, E8_Acquisition, E67_Birth, and E69_Death by relationship P4_has_time-span in the gold semantic models of the museum-crm dataset. We define such an entity as an ambiguous entity and the attributes it labeled as ambiguous attributes. In the candidate model generated by the Steiner Tree algorithm, some elements may be wrong because of multiple possible relationships. For example, as Figure 3 shows, the relationships P4_has_time-span from the entity E12_Production to E52_Time-Span2 and E52_Time-Span3 are incorrect. In fact, these entities should be linked with E67_Birth and E69_Death by relationship P4_has_time-span respectively. In this section, we put forward a machine learning method to move such ambiguous relationships. We treat the problem of distinguishing ambiguous relationships as a multi-category classification problem. Similarity metrics are used as features of the

learning matched function to determine whether different ambiguous attributes have the same relationship and thereby infer the correct links.

Our method of removing incorrect relationships is summarized in the following: (1) For training data sources, we gather all ambiguous attributes (their corresponding ambiguous relationships are known), extract several features and train a decision tree model [27]; (2) For a new data source, we find all the ambiguous attributes and use the trained decision tree model to determine the correct linking position; and (3) We move the relationships according to the predicted result.

We used the following candidate features, including attribute name similarity, value similarity, distribution similarity, and histogram similarity to a decision tree model. Besides these, we also use an external *knowledge base* to generate additional features. We briefly describe these similarities in the following: (1) Attribute Name Similarity: Usually, there is a *title* for each column of a structural tabular data source such as a Web table or spreadsheet. We treat these headings as attribute names and use them to compare the similarities between attribute names and entity names. The similarity may infer the correct relationship to which the entity is linked. For example, if an attribute is named *birthDate*, its labeled entity should be E52_Time-Span link with entity E67_Birth rather than entity E69_Death. (2) Value Similarity: Value similarity is the most commonly used similarity measure, which has been used in various matching systems. Since the same semantic types usually contain similar values, value similarity plays a significant role in recognizing attributes labeled by the identical semantic types. In our method, two different value similarity metrics are applied, i.e., Jaccard similarity [28] and TF-IDF cosine similarity [28] for computing the value similarity of textual data. (3) Distribution Similarity: For numeric data, value similarity is always ineffective to distinguish semantic types because they always have the similar value range. However, their distribution of values may be different because they have different potential meanings. Therefore, we use statistical hypothesis testing as one of the similarity measures to analyze the distribution of values in attributes. We also used the Kolmogorov–Smirnov test (KS test) [29] as one of the similarity metrics. (4) Histogram Similarity: Histogram similarity calculates value histograms in textual attributes and compares their histograms. The statistical hypothesis test for the histograms is the Mann–Whitney test (MW test) [29]. In our method, the MW test is used for computing the histogram similarity, considering it calculates the distribution distance based on medians. (5) External Knowledge Base: To improve the accuracy of our approach further, we used an external knowledge base as a candidate feature. In the cultural heritage community, the Getty Union List of Artist Names (ULAN) <https://www.getty.edu/research/tools/vocabularies/ulan> (14 December 2022) is an authoritative reference dataset containing over 650,000 names of over 160,000 artists. For the museum datasets, some ambiguous attributes can be distinguished by retrieving the information from ULAN. For example, we can validate the information of biography of artists by comparing the information with an attribute labeled by E52_Time-Span with ULAN to determine if the information with E52_Time-Span really represents the birth date of artists.

Our approach to distinguishing ambiguous attributes is stated in detail as follows. Given an ambiguous entity, where the number of possible links is $k > 1$, we randomly select k attributes $\{a_{r1}, a_{r2}, \dots, a_{rn}\}$ from the training attributes as reference attributes. The reference attributes should contain all possible relationships. For other training attributes $\{a_1, a_2, a_3, \dots, a_n\}$, we compute multidimensional feature vectors f_i by comparing against all reference attributes and searching the external knowledge base. During training, we label each f_i as a number j (ranging from 1 to k), where j means the j th possible relationships. Given a new ambiguous attribute a_0 , we compute the feature f_0 and use the learned tree model to label f_0 as a number ranging from 1 to k . Unlike [22], for each attribute, we compute only one feature vector by comparing it with all the reference attributes. If the label of f_0 is m , it means that a_0 is linked by the m th possible relationships. If the predicted result is not consistent with the relationship in the seed model, we move the relationship according to the predicted result.

For the seed model of CB, as Figure 3 shows, the corresponding relationships of ambiguous attributes Begin Date, and Death Date are (E12_Production, P4_has_time-span, E52_Time-Span). These are wrong predictions by the Steiner Tree algorithm. Figure 4 shows the changes after moving the ambiguous relationships. Through our method, the correct relationships of these attributes can be predicted, and the incorrect relationships are moved into the correct linking position, i.e., (E67_Birth, P4_has_time-span , E52_Time-Span1) for Begin Date and (E69_Death, P4_has_time-span , E52_Time-Span2) for End Date.

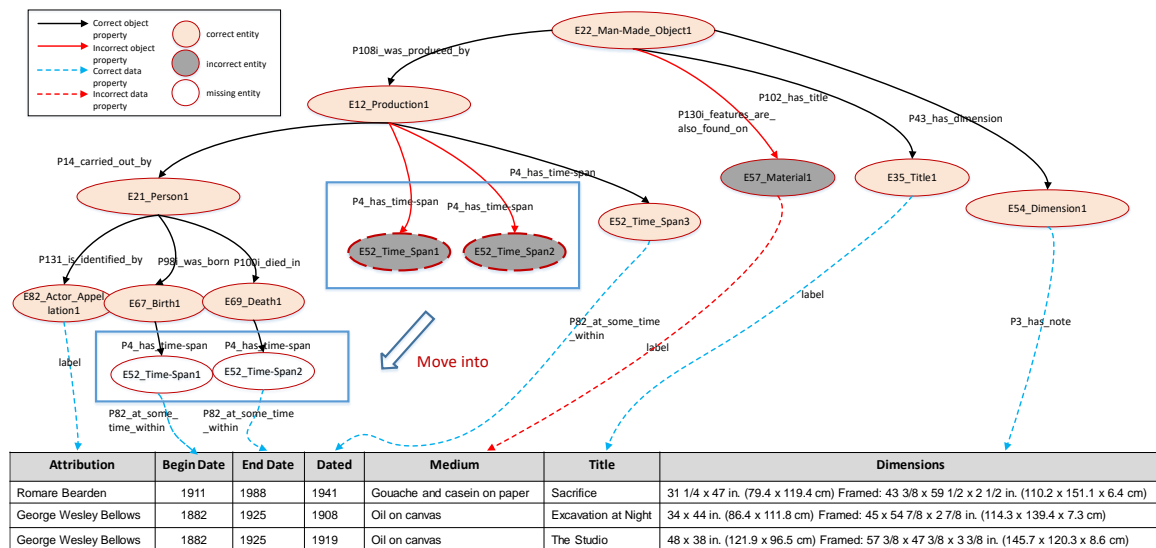


Figure 4. The change of seed model of CB after moving ambiguous relationships.

4.2.2. Remove Incorrect Relationships

There may still be incorrect substructures in the candidate model even though we move incorrect relationships. We can leverage the knowledge graph as *prior* knowledge to identify and remove them. The underlying idea of our method is that if incorrect relationships are included in *sd*, then *sd* has no isomorphic matches in the knowledge graph. If we remove incorrect substructures, the amended model must be a subgraph of the knowledge graph. Algorithm 1 shows how we remove incorrect relationships using a knowledge graph. The input is a seed model *sd* and a knowledge graph \mathcal{G} . First, we remove all the edges in *cm* that do not appear in the knowledge graph (Lines 4–8). (E22_Man-Made_Object, P130i_features_are_also_found_on, E57_Material). Obviously, with this relationship, *sd* cannot be matched to any occurrence in \mathcal{G} . This type of incorrect substructure is easily detected and removed. However, some incorrect substructures may appear in the knowledge graph but not occur in the gold model of *cm*. For these parts, we calculate the maximum common subgraph (MCS) [30] between \mathcal{G} and *sd* to find a subgraph of *sd* which is subgraph-isomorphic to \mathcal{G} with maximum nodes. The substructures which do not appear in the *mcs* are removed (Lines 9–11). At this point, all the incorrect substructures of *sd* are removed. If we remove an incorrect relationship, the corresponding attribute is temporarily unlabeled by any entities. We denote such an attribute as an isolated column *isoCols*. After removing incorrect relationships, the algorithm iterates over all the columns of *sd* and returns all the isolated columns that are not annotated by entities.

Figure 5 shows the change after running Algorithm 1. For the seed model of CB, the relationship (E22_Man-Made_Object, P130i_features_are_also_found_on, E57_Material) does not exist in the knowledge graph. Therefore, this incorrect relationship is removed during the process in Lines 4–8 of Algorithm 1. Medium is the isolated column after removing this incorrect relationships because it is not annotated by any semantic types. Since the annotation of the isolate column Medium is missing, the seed model is an incomplete model and needs to be complemented with the missing substructures. In the next step, we use a modified frequent subgraph mining algorithm for the completion of the seed model.

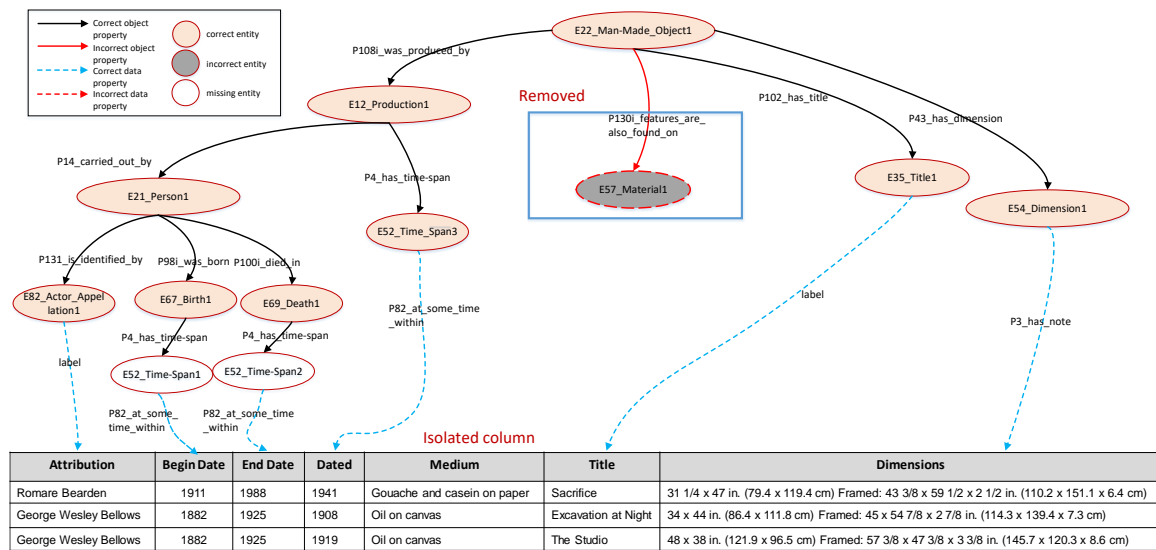


Figure 5. The change of seed model of CB after removing incorrect relationships.

Algorithm 1: Algorithm for removing incorrect relationships.

```

1 Algorithm removeIncorrectRel(Seed Model sd, Knowledge Graph  $\mathcal{G}$ )
2 begin
3   isoCols  $\leftarrow \phi$ ;
4   foreach  $e \in sd.edges()$  do
5     if  $e$  not appear in  $\mathcal{G}$  then
6       remove  $e$  in  $sd$ ;
7     end
8   end
9   if  $sd$  is not subgraph isomorphic to  $\mathcal{G}$  then
10     $sd \leftarrow MCS(\mathcal{G}, sd)$ 
11  end
12  foreach  $col \in sd.columns()$  do
13    if  $col$  not annotated by an entity then
14      isoCols  $\leftarrow result \cup col$ 
15    end
16  end
17  return isoCols;
18 end

```

4.2.3. Add Missing Substructures

In the seed model, some relationships may be missing. To enhance the integrity of the seed model, we need to add extra substructures. Note that there must exist data types in the data source that the added substructures can match. For instance, as shown in Figure 1, if some attributes in the data are date or time matching E67_Birth1, it may be reasonable to amend the model and choose entity E52_Time-Span1 to match these attributes.

Each column of the new data source may correspond to several candidate semantic types, and every semantic type has an ascribed confidence score computed during the initial semantic labeling step. However, the semantic type with the highest confidence score may not capture the correct semantics of the source attribute. For example, the correct semantic type of column Medium of data source CB is E55_Type, while its first learned semantic type is E57_Material. Accordingly, we consider multiple candidate semantic types for each column of the source when semantic models are constructed.

Considering multiple semantic types may make our method less efficient in constructing a semantic model of a source. Hence, we apply several heuristics to increase the effectiveness of our method in our algorithm. Algorithm 2 is used to check and reduce candidate semantic types of an isolated column based on their confidence score and a knowledge graph. First, we denote η as the ratio between the confidence score of the first

ranked candidate type and the second candidate type (Line 4). However, η might not be preset high enough to avoid the consideration of too many candidate semantic types. We empirically set three as the threshold value of η based on the experimental results in Table 1. If η is larger than three, we only select the first candidate type as the final candidate type (Lines 5–8). Second, we remove all the candidate types whose confidence scores are lower than 0.05 of a column (Lines 10–13). Finally, some candidate semantic types with high confidence scores can be quickly excluded by searching within the knowledge graph using the subgraph matching technique. For the candidate semantic type ct with a relatively high confidence score (larger than 0.05), we attempted to search all the paths $paths$ in \mathcal{G} that connect ct into candidate model sd after removing a presumed incorrect edge. If no such graph connecting a path in $paths$ into sd is subgraph-isomorphic to the knowledge graph, we remove this candidate type (Lines 14–25). A VF2 algorithm [31] is used for checking subgraph isomorphism. This process aims to find the entities in sd that do not co-occur with other substructures in the knowledge graph. If such entities are considered as candidate semantic types, the frequent subgraph mining algorithm may return none of the resulting models while costing much time. After running Algorithm 2, some candidate types are removed, which prunes large portions of the search space.

Algorithm 2: Algorithm of reducing candidate semantic types for an isolated column.

```

1 Algorithm reduceSemanticTypes(Seed Model  $sd$ , Knowledge Graph  $\mathcal{G}$ , isolated column  $isoCol$ )
2 begin
3   Let  $cTypes$  be the top-4 candidate types of  $isoCol$ ;
4    $\eta \leftarrow cTypes[0].score / cTypes[1].score$ ;
5   if  $\eta > 3$  then
6     | select the first candidate type;
7   end
8   foreach  $ct \in cTypes$  do
9     | if  $ct.score < 0.05$  then
10    | |  $cTypes.remove(ct)$ ;
11    | | continue;
12    | end
13    | Let  $paths$  be all the possible edge paths that connect  $ct$  into  $sd$ ;
14    |  $isIncorrectType = \mathbf{true}$ ;
15    | foreach  $path \in paths$  do
16    | |  $tmpModel \leftarrow sd.addPath(path)$ ;
17    | | if  $isSubgraphIsomorphic(\mathcal{G}, tmpModel)$  then
18    | | |  $isIncorrectType = \mathbf{false}$ ;
19    | | | break;
20    | | end
21    | end
22    | if  $isIncorrectType$  then
23    | |  $cTypes.remove(ct)$ ;
24    | end
25  end
26 end

```

Table 1. Counts of mislabeled attributes in three datasets in a different range of η .

Datasets	$\eta > 1$	$\eta > 2$	$\eta > 3$	$\eta > 4$	$\eta > 5$	$\eta > 6$
ds_{edm}	30	10	4	3	2	1
ds_{crm}	34	7	2	1	0	0
ds_{weapon}	20	9	7	7	6	6

In the seed model of CB, for the isolated column *Medium*, the confidence scores of the last two candidate semantic types are lower than 0.05, so we filter them out in candidate types. For the incorrect type *E57_Material* of attribute *Medium*, there is only one possible relationship connecting it into sd , i.e., (*E22_Man-Made_Object*, *P45_consists_of*, *E57_Material*). However, if we try to add this relationship in sd , we find the intermediate

model is not subgraph-isomorphic to \mathcal{G} . So the candidate semantic type `E57_Material` might be incorrect, and we can eliminate it from consideration. After reducing the candidate semantic types, there is only one candidate semantic type i.e., `E55_Type1` for the column `Medium`.

Algorithm `addMissingSubStructures` (Algorithm 3) is proposed to search and add deleted substructures of imperfect seed semantic model sd . First, we use Algorithm 1 to remove incorrect substructures and obtain $isoCols$ of sd (Line 4). If the attribute set $isoCols$ is empty, the algorithm returns sd as the result (Lines 5–7). In this case, none of the incorrect relationships is detected by Algorithm 1. The seed model might be the correct model. Then, we iterate over the set $isoCols$, and for each attribute $isoCol$, we use Algorithm 2 to reduce its candidate semantic types (Lines 8–10). After these steps, we enumerate all the possible combinations of candidate semantic types of $isoCols$ and run the modified frequent subgraph mining algorithm (Lines 13–21). Unlike the traditional algorithm, we abandon the parameter frequency threshold τ for the uncertainty of the frequency threshold of the correct model. Without the limitation on frequency, the efficiency of frequent subgraph mining algorithms may decrease. Accordingly, we propose a set of pruning strategies for speeding up our algorithm. These pruning strategies will be introduced below. The output of Algorithm 3 is the set $result$, which stores all frequent subgraphs as complementary semantic models.

Algorithm 3: Algorithm for repairing the seed model.

```

1 Algorithm addMissingSubStructures(Seed Model  $sd$ , Knowledge Graph  $\mathcal{G}$ , Constraint Map  $cm$ )
2 begin
3    $result \leftarrow \phi$ ;
4    $isoCols \leftarrow removeIncorrectRel(sd, \mathcal{G})$ ;
5   if  $isoCols == \phi$  then
6     | return  $sd$ ;
7   end
8   foreach  $isoCol \in isoCols$  do
9     |  $reduceSemanticTypes(sd, \mathcal{G}, isoCol)$ ;
10  end
11  Let  $types$  be the set of the combinations of candidate types of  $isoCols$ ;
12  Let  $Edges$  be the set of edges of in  $\mathcal{G}$ ;
13  foreach  $newNodes \in types$  do
14    | foreach  $e \in Edges$  do
15      | | if  $e$  is the edge of  $sd$  then
16        | | |  $result \leftarrow result \cup subgraphExtension(\mathcal{G}, e, newNodes, sd, cm)$ ;
17        | | | Remove  $e$  from  $Edges$ ;
18      | | end
19    | end
20  end
21  return  $result$ ;
22 end

```

As the sub-function of Algorithm 3, Algorithm `subgraphExtension` (Algorithm 4) is used to search and add missing substructures from a specific imperfect semantic model \mathcal{S} . The inputs of Algorithm 4 are a seed model sd , a knowledge graph \mathcal{G} , an incomplete semantic model \mathcal{S} , a constraint map cm , and the candidate semantic types $newNodes$ that the seed model may link with. The *grow-and-store* strategy is adopted in the algorithm `subgraphExtension` for recursively mining the top- σ frequent subgraphs of \mathcal{G} . In the meantime, Algorithm 3 can ensure the coverage between the seed model sd and all of the candidate semantic types $newNodes$ of source attributes in the mined top- σ frequent substructures. Here, our *heuristic* is that if the frequency of an amended semantic model is higher than a certain threshold in the knowledge graph, it indicates that the model could be correct. Further, the higher the frequency of an amended model in the knowledge graph, the more likely it is to be a reasonable semantic model. The output of the algorithm `subGraphExtension` is the top- σ semantic models with complemented missing substructures.

Algorithm 4: Algorithm of subgraph extension.

```

1 function subgraphExtension(Knowledge Graph  $\mathcal{G}$ , Incomplete semantic model  $\mathcal{S}$ , Semantic Labels
  newNodes, Seed Model  $sd$ , Constraint Map  $cm$ )
2 begin
3    $freqs \leftarrow [0]$ ;
4    $cm = \text{HashMap} \langle \text{node}, \text{value} \rangle$  //  $cm$  keys : entity types that appear in the semantic
  model to be repaired;  $cm$  values : maximum occurrence of the key in the
  semantic model to be repaired;
5   foreach  $e \in \text{Edges}$  and node  $n \in \mathcal{S}$  do
6     if  $e$  can be used to extend  $n$  and  $e$  is a valid edge then
7       Let  $ext$  be the extension of  $\mathcal{S}$  with  $e$ ;
8       if  $ext$  covers all nodes in  $sd$  and newNodes and  $sd$  is a subgraph of  $ext$  then
9          $result \leftarrow result \cup ext$ ;
10        return  $result$ ;
11      end
12      check every key  $et$  in Constraint Map  $cm$ ;
13      if  $count(et, ext) > cm[et]$  then
14        continue;
15      end
16      foreach  $e \in sd.edges$  do
17        if  $e.source \in ext.nodes$  and  $e.target \in ext.nodes$  and  $e \notin ext.edges$  then
18          continue;
19        end
20      end
21      if  $freq(ext, \mathcal{G}) > min(freqs)$  then
22         $freqs \leftarrow freqs \cup freq(ext, \mathcal{G})$ ;
23         $sort(freqs)$ ;
24        remove the  $min$  element in  $freqs$ ;
25        if  $ext$  has not been generated before then
26           $result \leftarrow result \cup \text{subgraphExtension}(\mathcal{G}, ext, newNodes, sd, cm)$ ;
27        end
28      end
29    end
30  end
31  return  $result$ 
32 end

```

The inputs of Algorithm 4 are illustrated as below. Semantic Labels is a candidate combination of candidate semantic types of the source attributes that does not occur in the seed model of a new data source. Constraint Map is a HashMap which constraints the maximum count of entities for each semantic type that may appear in a correct model. Generally, Constraint Map is prescribed through domain expertise. For instance, as Figure 1 shows, in the semantic model of the CB, the Constraint Map $\{ \langle E52_Time-Span, 3 \rangle, \langle E35_Title, 1 \rangle \}$ limits the count of entities for the semantic types E52_Time-Span, and E35_Title must be less than three and one, respectively. The intermediate variable $freqs$ is a sorted integer list used to record the top- σ frequencies of all subgraphs throughout the algorithm process.

For each relationship e in knowledge graph \mathcal{G} and a semantic type n in the incomplete model \mathcal{S} , first Algorithm 4 validated if e can be linked with the semantic type n and whether e is a valid edge in the knowledge graph (Line 6). For instance, assume that entities E22_Man-Made_Object1 and E35_Title1 and a relationship P102_has_title occur in an initial subgraph. Suppose n is semantic type E22_Man-Made_Object1 in \mathcal{S} and e is object property P43_has_dimension. Since \mathcal{G} contains the triple (E22_Man-Made_Object1, P43_has_dimension, E54_Dimension1), it implies that P43_has_dimension is effective and can be used to link with E22_Man-Made_Object1. Let ext be the extension of \mathcal{S} with e (Line 7). During the run of Algorithm 4, when all the nodes and links in the seed model sd and candidate semantic types $newNodes$ are covered in ext , the ext is the most possibly plausible semantic model which correctly captures the semantics of the data source. Next, ext is merged into the set $result$, and the algorithm ceases and returns $result$ (Lines 8–11). The set $result$ is used to store the top- σ semantic models as the output of the

algorithm `subGraphExtension`. As Algorithm 3 demonstrates, the set *newNodes* represents one possible combination of all candidate semantic types of *isoCols*, *newNodes* may be erroneous to annotate semantic types for *isoCols*. Algorithm 4 may return `none`, which indicates that there are some incorrect candidate types in the *newNodes*.

We apply some pruning strategies to improve the efficiency of the algorithm. In order to prevent the recurrence of a given entity type *et* in the model, one of the pruning strategies we employ is to leverage `Constraint Map` to limit the searching space. We check each entity *et* in `Constraint Map` *cm*. If there is an entity type *et* that appears in *ext* exceeding *cm(et)* times, the further search for current *ext* is ceased (Lines 13–15). The Function `count()` is used to calculate the number of appearances of *et* in the intermediate subgraph *ext*.

We reduce the search space by leveraging the structure of *sd*. We iterate over the edge of *sd*, and if there is an edge *e* whose source node and target node exist in *ext* but *e* does not exist in *ext*, we stop searching *ext* (Lines 16–20). For example, in the seed model of CB, entity `E22_Man-Made_Object` and entity `E12_Production` are connected by relationship `P108i_was_produced_by`. For an intermediate substructure *ext* containing these entities, if there is no link between the two entities, *ext* can not be extended to the model which contains the relationship `P108i_was_produced_by`. It is vain to further search substructure *ext*. In practice, a large amount of search space can be reduced by using this pruning strategy. For the seed model of CB, the running time is about 15 s, while without this optimization, the respective running time is about 5 min.

Next, we use a Minimum Image-based Metric based on a subgraph matching algorithm [32] (Line 21) in the function `freq(ext, G)` to calculate the frequencies of *ext* occurring in *G*. All the substructure *ext* in which the frequency is lower than the minimum frequency of *freqs* will be discarded. The intuition behind this pruning strategy is that the correct semantic model may be a subgraph with higher frequency in the knowledge graph. During the process of search, only the top- σ frequent subgraphs are retained as candidate semantic models in the frequency-based pruning strategy.

In Algorithm 4, during the whole search, top- σ frequencies of all subgraphs are stored in an integer list *freqs* (line 22) in which the length is σ . Here, *freqs* merges the frequency of *ext* (`freq(ext, G)`) (Line 22) if the frequency of *ext* is higher than the minimum value of the current *freqs*. Then, Algorithm 4 sorts *freqs* (Line 23) and removes the minimum value in the sorted *freqs* (Line 24).

Thereafter, for removing duplicate models, Algorithm 4 checks if *ext* has been searched in the previous procedure. We exploit the *canonical code* method proposed in [14] to detect the duplicate models (Line 25). The algorithm `subgraphExtension` is recursively executed (Line 26) for further search if the substructure *ext* has not been generated before.

The incomplete model of CB before adding missing substructures is shown in Figure 5 (wrong entity `E57_Material1` and its relationship were removed). After reducing the candidate semantic types, `E55_Type1` is the only candidate type for isolated column `Medium`. Through the process of our modified subgraph mining algorithm, the missing relations (`E12_Production1`, `P32_used_general_technique`, `E55_Type1`) can be linked into the incomplete model, and the gold standard model (Figure 1) can be output as a frequent subgraph. Hence, the seed model of CB is amended into a correct semantic model.

5. Evaluation

5.1. Experimental Setting

To assess our method, we conducted the experiments on three datasets, i.e., `museum_edm` (ds_{edm}), `museum_crm` (ds_{crm}), and `wepaon_1od` (ds_{wepaon}). The datasets ds_{edm} and ds_{crm} both contain 29 different data sources from different art museums in the USA and have different data formats (CSV, XML, and JSON). Nevertheless, two different famous data models are used as museum domain ontology: European Data Model (EDM) <http://pro.europeana.eu/page/edm-documentation> (accessed on 14 December 2022), and CIDOC Conceptual Reference Model (CIDOC-CRM) www.cidoc-crm.org (accessed on 14 December 2022); ds_{wepaon} includes 15 data sources about weapon ads. The ontology of ds_{wepaon} is an extension of the schema.org ontology. The background knowledge graphs were constructed by

capturing these data sources and mapping them to the corresponding domain ontology. For each specified data source of a dataset, we built a knowledge graph that integrated the data from all of the data sources excluding this one. For example, for the s_1 of ds_{crm} , the knowledge graph integrates all the data sources, i.e., s_2, s_3, \dots, s_{29} , except s_1 . Table 2 lists the details of these three datasets. For facilitating the construction of knowledge graphs, we reconstructed the semantic models of the ground-truth datasets and transformed all the data sources into CSV format. The datasets, experimental results, and our code are available on Github <https://github.com/Zaiwen/ModelCorrection> (accessed on 14 December 2022). Our objective is to assess the effectiveness of our method if only a few known semantic models of similar sources are available. So we use only two or three data sources of the datasets for training and the others for testing. We repeat the process three times and average the results. Our experiments were run on a single machine with an Intel i7 10500 CPU 3.40GHz and 16 GB RAM.

Table 2. The evaluation of datasets.

	ds_{edm}	ds_{crm}	ds_{weapon}
#data sources	29	29	15
#classes in domain ontologies	120	83	718
#properties in domain ontologies	351	270	295
#nodes in the gold-standard models	409	750	230
#data nodes in the gold-standard models	123	362	81
#class nodes in the gold-standard models	286	388	149
#links in the gold-standard models	380	724	215
#average entities in knowledge graphs	55,432	57,558	5403
#average relationships in knowledge graphs	73,722	82,654	6529

5.2. Empirical Preliminary Experiments

In learning the candidate semantic types of an attribute of a data source s_i , we use the known semantic models and their corresponding data sources as training data. For example, if we are learning the candidate types of data source s_i , the training data are all the data sources $\{s_k | k = 1, \dots, j \text{ and } k \neq i\}$. The semantic labeler we use is SemanticTyper [21]. In this work, we only consider the top four semantic types. Table 3 shows the mean reciprocal rank (MRR) [33] scores of semantic labeling in three datasets.

Table 3. MRR of semantic labeling.

Datasets	ds_{edm}	ds_{crm}	ds_{weapon}
MRR scores	0.907	0.937	0.879

For an attribute, if the confidence score of its first ranked candidate semantic type is much higher than the second candidate type, the first ranked candidate type is assumed to be correct. To reduce the number of possible candidate types, we consider only the first ranked candidate semantic type for such an attribute. Let η be the ratio between the confidence score of the first candidate type and the second candidate type. Table 1 shows the counts of the mislabeled attributes in different ranges of η . The counts of mislabeled attributes decrease with increasing η . However, there are a few mislabeled attributes whose η is very large. When η is larger than three, the count of mislabeled attributes in three datasets tends to be relatively stable. Therefore, in Algorithm 2, we only select the first candidate type for the attribute whose η is larger than three for the balance between the accuracy of candidate types and the efficiency of our algorithm.

5.3. Effectiveness of Our Approach

To evaluate our approach, standard mean reciprocal rank (MRR) [33] was used. We compared the obtained models with the gold semantic model to assess the correctness of them based on precision and recall as in Taheriyani et al. [6]:

$$precision = \frac{|rel(sm) \cap rel(f^*(sm'))|}{|rel(f^*(sm'))|} \quad (1)$$

$$recall = \frac{|rel(sm) \cap rel(f^*(sm'))|}{|rel(sm)|} \quad (2)$$

$$f^* = \underset{f}{argmax} |rel(sm) \cap rel(f(sm'))| \quad (3)$$

where $rel(sm)$ is the set of the triples (u, e, v) of a semantic model sm , and f is a mapping function, which maps nodes in sm' to nodes in sm .

Table 4 shows the results of our experiments. Two state-of-the-art semantic modeling systems were compared: Karma [6] and PGM-SM [7]. Among them, our method improves the two baseline methods by an average of 10.68%, 13.85%, and 9.08% on ds_{edm} , ds_{crm} , and ds_{weapon} , respectively. Our approach uses knowledge graphs for amending the incorrect substructures in learned semantic models and realizes noticeable modification, indicating that knowledge graphs are useful prior knowledge to improve the quality of learned semantic models.

Table 4. Performances of our method on ds_{edm} , ds_{crm} , and ds_{weapon} .

Datasets	Known Models	Precision			Recall			F1		
		Karma	PGM-SM	Ours	Karma	PGM-SM	Ours	Karma	PGM-SM	Ours
ds_{edm}	2	0.864	0.791	0.920	0.846	0.754	0.913	0.855	0.770	0.917
	3	0.858	0.778	0.920	0.840	0.757	0.912	0.849	0.765	0.916
ds_{crm}	2	0.738	0.824	0.897	0.703	0.721	0.878	0.721	0.773	0.888
	3	0.770	0.828	0.908	0.731	0.725	0.892	0.751	0.777	0.900
ds_{weapon}	2	0.795	0.809	0.870	0.734	0.758	0.834	0.765	0.784	0.852
	3	0.837	0.805	0.924	0.783	0.810	0.902	0.810	0.808	0.913

While our approach performs well in most cases, the accuracy of the semantic model generated by our method is inferior to the semantic model generated by KARMA in some specific cases. For example, the prediction accuracy and recall of the semantic model generated by KARMA of s_9 are 0.6 and 0.75, respectively, while s_2 and s_6 are used as the training set of ds_{edm} . However, after amending using our approach, the precision and recall of the final model are 0.4 and 0.5, respectively. Figure 6 shows the correct model, seed model, and final model of s_9 in ds_{edm} . Compared to the seed model, our approach moves the correct relationship (Person, biographicalInformation, biography) into the error relationship (CulturalHeritageObject1, description, biography). The incorrect entity CulturalHeritageObject1 appears in the seed model because of the wrong predicted semantic type of attribute birthDate. The relationship (Person1, biographicalInformation, biography) does not co-occur with the entity CulturalHeritageObject1 in the knowledge graph. During running Algorithm 1, the relationship (Person1, biographicalInformation, biography) is removed at line 10. This phenomenon indicates that our approach is sensitive to the incorrect semantic labeling result. Since our approach attempts to improve the seed model generated by the Steiner Tree algorithm, the performance of our approach is highly sensitive to the quality of the seed model. If the seed model differs greatly from the corresponding correct model, our approach may be unable to recover the correct model.

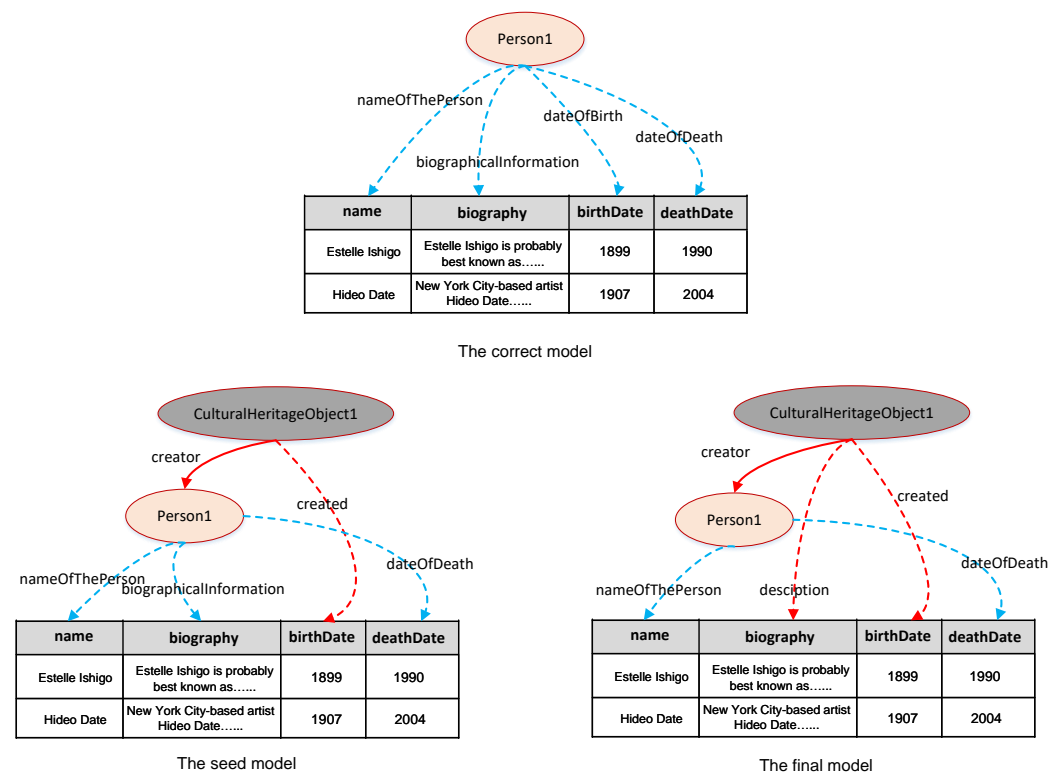


Figure 6. The predict result of s_9 in ds_{edm} , using s_2 and s_6 as training sets.

5.4. Efficiency of Our Approach

We measured the running time of our method. The results are listed in Table 5. Phase I refers to the process of moving and removing the incorrect substructures and phase II is the process of adding missing substructures. The running time of our method is positively correlated with the size of the knowledge graph. Since the size of knowledge graphs in ds_{weapon} is much smaller than that of ds_{edm} and ds_{crm} (Table 2), we can see that the running time is much less. In our approach, the running time mainly depends on graph algorithms whose running time is affected by the size of the knowledge graph and the size of the semantic models. While the size of the knowledge graphs created in different scenarios in our experiment is close, the size of semantic models in ds_{edm} is smaller than that in ds_{crm} . Therefore, the running time of phase II of ds_{edm} is less.

Table 5. Average running time of our method on ds_{edm} , ds_{crm} and ds_{weapon} .

Datasets	Phase I	Phase II
ds_{edm}	45.395 s	13.581 s
ds_{crm}	46.782 s	33.543 s
ds_{weapon}	4.232 s	2.730 s

6. Conclusions

In this article, we propose a novel approach for solving the Re120nt problem by leveraging a knowledge graph as background knowledge. First, we require a partially correct semantic model called seed model by running the Steiner tree algorithm [26]. We move or remove imperfect relationships in the seed model by using machine learning and a graph matching technique. After eliminating the incorrect substructures, a modified frequent subgraph mining algorithm is applied to search the top- σ frequent substructures covering the seed model and the source attributes candidate semantic types from the domain knowledge graph. Our experimental results indicate that we can generate high-quality semantic models even when the known semantic models are lacking and that we

can outperform two state-of-the-art semantic modeling systems in terms of the correctness of the resulting models. In the future, we would like to further develop our method in the following areas. Firstly, our decision tree model for distinguishing ambiguous relationships can be enhanced by a feature selection algorithm [34] to further improve the accuracy. Secondly, we will explore an automatic method for extracting a Constraint Map from historical data. Finally, we will try to extend our approach to those data sources containing a set of relations.

Author Contributions: Conceptualization, W.M. and Z.F.; Methodology, J.X.; Software, J.X.; Validation, J.X.; Formal analysis, J.X.; Investigation, J.X.; Resources, H.Z.; Data curation, J.X.; Writing(original draft), J.X.; Writing (review and editing), J.X., W.M. and Z.F.; Visualization, J.X.; Supervision, H.Z., K.H. and Z.F.; Project administration, Z.F.; Funding acquisition, Z.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research project was supported in part by the open funds of the National Key Laboratory of Crop Genetic Improvement under Grant ZK202203, Huzhong Agricultural University, and in part by the Major Project of Hubei Hongshan Laboratory under Grant 2022HSZD031, and in part by the Innovation fund of Chinese Marine Defense Technology Innovation Center under Grant JJ-2021-722-04, and in part by the Fundamental Research Funds for the Chinese Central Universities under Grant 2662020XXQD01, 2662022JC004, and in part by the open funds of State Key Laboratory of Hybrid Rice, Wuhan University.

Data Availability Statement: The data that support the findings of this study are openly available in Github at <https://github.com/Zaiwen/ModelCorrection> (accessed on 14 September 2022).

Acknowledgments: This research project was supported in part by the open funds of the National Key Laboratory of Crop Genetic Improvement under Grant ZK202203, Huzhong Agricultural University, and in part by the Major Project of Hubei Hongshan Laboratory under Grant 2022HSZD031, and in part by the Innovation fund of Chinese Marine Defense Technology Innovation Center under Grant JJ-2021-722-04, and in part by the Fundamental Research Funds for the Chinese Central Universities under Grant 2662020XXQD01, 2662022JC004, and in part by the open funds of State Key Laboratory of Hybrid Rice, Wuhan University. Some of the initial work in this article was done when the last author did his post-doctoral research in the ITMS, University of South Australia between 2016 and 2019. We appreciate the valuable suggestion from Markus Stumptner, Georg Grossmann, Wenhao Li, Selasi Kwashie, and Amir Kashefi from ITMS, University of South Australia, and Wangyu Huang from Data-to-Decision CRC. Numerical computations were performed on the Hefei Advanced Computing Center in China.

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

References

1. Rahm, E.; Bernstein, P.A. A survey of approaches to automatic schema matching. *Vldb J.* **2001**, *10*, 334–350. [[CrossRef](#)]
2. Dhamankar, R.; Lee, Y.; Doan, A.; Halevy, A.; Domingos, P. iMap: Discovering complex semantic matches between database schemas. In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, Paris, France, 13–18 June 2004.
3. Hazber, M.A.; Li, R.; Li, B.; Zhao, Y.; Alalayah, K.M. A survey: Transformation for integrating relational database with semantic Web. In Proceedings of the 2019 3rd International Conference on Management Engineering, Software Engineering and Service Sciences, Wuhan, China, 12–14 January 2019; pp. 66–73.
4. Doan, A.; Halevy, A.; Ives, Z. *Principles of Data Integration*; Elsevier: Amsterdam, The Netherlands, 2012; ISBN-13: 978-0124160446.
5. Una, D.D.; Rümmele, N.; Gange, G.; Schachte, P.; Stuckey, P.J. Machine Learning and Constraint Programming for Relational-To-Ontology Schema Mapping. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 13–19 July 2018.
6. Taheriyani, M.; Knoblock, C.A.; Szekely, P.; Ambite, J.L. Learning the semantics of structured data sources. *J. Web Semant.* **2016**, *37*, 152–169. [[CrossRef](#)]
7. Vu, B.; Knoblock, C.; Pujara, J. Learning semantic models of data sources using probabilistic graphical models. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019.
8. Bellomarini, L.; Sallinger, E.; Vahdati, S. Knowledge graphs: The layered perspective. In *Knowledge Graphs and Big Data Processing*; Springer: Cham, Switzerland, 2020; pp. 20–34.
9. Hubauer, T.; Lamparter, S.; Hasse, P.; Herzig, D. Use cases of the industrial knowledge graph at siemens. *International Semantic Web Conference*. 2018. Available online: <https://www.semanticscholar.org/paper/Use-Cases-of-the-Industrial-Knowledge-Graph-at-Hubauer-Lamparter/ecc8a846aee63be0a571ece752e87d7d266bbe9a> (accessed on 14 December 2022).

10. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Philip, S.Y. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 494–514. [[CrossRef](#)] [[PubMed](#)]
11. Taheriyani, M.; Knoblock, C.A.; Szekely, P.; Ambite, J.L. Leveraging linked data to discover semantic relations within data sources. In *International Semantic Web Conference*; Springer: Cham, Switzerland, 2016; pp. 549–565.
12. Futia, G.; Vetrò, A.; De Martin, J.C. SeMi: A SEMantic Modeling machLine to build Knowledge Graphs with graph neural networks. *SoftwareX* **2020**, *12*, 100516. [[CrossRef](#)]
13. Feng, Z.W.; Xu, J.K.; Mayer, W.; Huang, W.Y.; He, K.Q.; Stumptner, M.; Grossmann, G.; Zhang, H.; Ling, L. Automatic Semantic Modeling for Structural Data Source with the Prior Knowledge From Knowledge Graph. In Proceedings of the 2021 IEEE International Conference on Data, Information, Knowledge and Wisdom (DIKW), Haikou, China, 20–22 December 2021; pp. 2034–2041.
14. Yan, X.; Han, J. GSPAN: Graph-based substructure pattern mining. In Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, 9–12 December 2002; pp. 721–724.
15. Pinkel, C.; Binnig, C.; Kharlamov, E.; Haase, P. IncMap: Pay as you go matching of relational schemata to OWL ontologies. In Proceedings of the 8th International Workshop on Ontology Matching co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, 21 October 2013; Volume 1111, pp. 37–48.
16. Tian, A.; Sequeda, J.; Miranker, D.P. QODI: Query as context in automatic data integration. In Proceedings of the 12th International Semantic Web Conference (ISWC 2013), Sydney, NSW, Australia, 21–25 October 2013; pp. 624–639.
17. Sequeda, J.F.; Miranker, D.P. Ultrawrap Mapper: A semi-automatic relational database to RDF (RDB2RDF) mapping tool. In Proceedings of the ISWC 2015 Posters and Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, 11 October 2015; Volume 1486.
18. de Medeiros, L.F.; Priyatna, F.; Corcho, Ó. MIRROR: Automatic R2RML mapping generation from relational databases. In Proceedings of the ICWE 2015, Rotterdam, The Netherlands, 23–26 June 2015; Volume 9114, pp. 326–343.
19. Fathy, N.; Gad, W.; Badr, N.; Hashem, M. ProGOMap: Automatic Generation of Mappings From Property Graphs to Ontologies. *IEEE Access* **2021**, *9*, 113100–113116. [[CrossRef](#)]
20. Schwade, F.; Schubert, P. A Semantic Data Lake for Harmonizing Data from Cross-Platform Digital Workspaces using Ontology-Based Data Access. *AMCIS 2020 Proceedings*. 2020. Available online: https://aisel.aisnet.org/amcis2020/ai_semantic_for_intelligent_info_systems/ai_semantic_for_intelligent_info_systems/2/ (accessed on 14 December 2022).
21. Ramnandan, S.K.; Mittal, A.; Knoblock, C.A.; Szekely, P. Assigning Semantic Labels to Data Sources. In Proceedings of the 12th ESWC, 2015. Available online: https://link.springer.com/chapter/10.1007/978-3-319-18818-8_25 (accessed on 14 December 2022).
22. Pham, M.; Alse, S.; Knoblock, C.A.; Szekely, P. Semantic labeling: A domain-independent approach. In *International Semantic Web Conference*; Springer: Cham, Switzerland, 2016; pp. 446–62.
23. Mulwad, V.; Finin, T.; Joshi, A. Semantic message passing for generating linked data from tables. In *International Semantic Web Conference*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 363–378.
24. Rümmele, N.; Tyshetskiy, Y.; Collins, A. Evaluating approaches for supervised semantic labeling. *arXiv* **2018**, arXiv:1801.09788.
25. Winter, P. Steiner Problem in Networks—A Survey. *Networks* **1987**, *17*, 129–167. [[CrossRef](#)]
26. Bhalotia, G.; Hulgeri, A.; Nakhe, C.; Chakrabarti, S.; Sudarshan, S. Keyword Searching and Browsing in Databases Using BANKS. In Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, 26 February–1 March 2002; pp. 431–440.
27. Safavian, S.R.; Landgrebe, D. A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man, Cybern.* **1991**, *21*, 660–674. [[CrossRef](#)]
28. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: New York, NY, USA, 2008.
29. Lehmann, E.L.; Romano, J.P. *Testing Statistical Hypotheses*; Springer: Berlin/Heidelberg, Germany, 2005.
30. McGregor, J.J. Backtrack Search Algorithms and the Maximal Common Subgraph Problem. *Softw. Pract. Exp.* **1982**, *12*, 23–34. [[CrossRef](#)]
31. Cordella, L.P.; Foggia, P.; Sansone, C.; Vento, M. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE PAMI* **2004**, *26*, 1367–1372. [[CrossRef](#)] [[PubMed](#)]
32. Elseidy, M.; Abdelhamid, E.; Skiadopoulou, S.; Kalnis, P. GRAMI: Frequent Subgraph and Pattern Mining in a Single Large Graph. *Proc. VLDB Endow.* **2014**, *7*, 517–528. [[CrossRef](#)]
33. Craswell, N. Mean reciprocal rank. In *Encyclopedia of Database Systems*; Springer: Berlin/Heidelberg, Germany, 2009; p. 1703.
34. Zhang, S.; Cheng, D.; Hu, R.; Deng, Z. Supervised feature selection algorithm via discriminative ridge regression. *World Wide Web* **2018**, *21.6*, 1545–1562. [[CrossRef](#)]