Detecting Hidden Relations in Geographic Data

Ngoc-Thanh Le
Faculty of Information Technology
University of Science
Ho Chi Minh City, Vietnam
lnthanh@fit.hcmus.edu.vn

Ryutaro Ichise
Principles of Informatics Research Division
National Institute of Informatics
Tokyo, Japan
ichise@nii.ac.jp

Hoai-Bac Le
Faculty of Information Technology
University of Science
Ho Chi Minh City, Vietnam
lhbac@fit.hcmus.edu.vn

***Abstract*—The amount of linked data is growing rapidly, and so finding suitable entities to link together requires greater effort. For small data sets, it is easy enough to find entities in the data sources and link these together manually; however, doing so for large data sets is impractical. For large sets, a way is needed to discover entities and connect them automatically. In this paper, we present an algorithm to detect hidden *owl:sameAs* links or hidden relations in data sets. Since geographic names are often highly ambiguous, we used data sets comprising geographic names to implement and evaluate our algorithm. We experimentally compare our algorithm with a naïve algorithm that only uses a URI's name feature. We found that it is more accurate than the naïve algorithm in most cases, especially for resources in which there is little matching information about features.**

***Keywords*-Linked Data; Knowledge Discovery; Link Prediction;**

I. INTRODUCTION

Linked data refers to data published on the Web in such a way that it is machine-readable. It is linked to other external data sets and can in turn be linked to from external data sets [1]. Linked data uses the Resource Description Framework (RDF) to make typed statements that link arbitrary things in the world, and things are named by Uniform Resource Identifiers (URIs) and linked together by predicates.

In this paper, we mainly focus on *owl:sameAs* links. These links indicate that two URIs refer to the same thing, implying that the subject and object must be the same resource. When users create an entity to describe a thing using their own information features, if they know of other data sources on the Web that also provide information about this thing, then they can link these sources together. In this manner, the information about the thing becomes richer.

We should recognize that a linked data structure is very similar to a graph in which URIs are nodes and links are edges. Various graph algorithms exist, and the literature on them is well developed; in fact, many approaches for analyzing graphs have been extended to linked data structures [2], [3], [4]. On the basis of these observations, we decided to turn linked data into a graph upon which we can use graph mining techniques to solve the following problems.

As of 19 January 2010, the Linked Data Community estimates that the number of triples on Linking Open Data [5] is about 13 billion and the number of links is about 143 million. The amount of linked data has been growing steadily. Therefore, it may soon be difficult to find suitable entities to connect with *owl:sameAs* links. In some cases, mistakes may be made, such as linking entities that refer to different things. This means that *owl:sameAs* may be inappropriately used. In addition, a single data source may have redundant descriptions, creating confusion as to which items should be linked. Moreover, even if one manages to make an appropriate choice in some way, there is no guarantee that others will make the same choice. Finally, incorrect data affect new data in many ways. The overall effect of these problems is that information on the Web will become more and more ambiguous.

Certain data are often ambiguous; in particular, geographic names, e.g., the name of rivers, mountains, and place names of population concentrations, tend to be very ambiguous. For example, the name "Isosaai" refers to 491 places in Finland [6]. Also, there are 1724 different coordinates sharing the name "San Jose" [7] in the GeoNet and GNIS geographic name databases. Raphael Volz et al. list three types of ambiguity [7]:

1) Different geographic locations share the same name
2) One location has different names
3) A location name also stands for some other word

In our work, we are interested in geographic information and its problems. Our data set has over 2.5 million geographic names. If the above problems affect it, this would be very difficult for us to detect or resolve.

For small data sets, it is easy enough to find entities referring to the same thing in data sources and link them together manually; however, doing so for large data sets is impractical. For large sets, a way is needed to discover entities and connect them with *owl:sameAs* links automatically. The task of discovering entities can be viewed as detecting hidden relations in linked data. In other words, hidden relations are possible links that have not yet been created. The main idea behind our solution is to extract useful features by applying supervised learning on frequent graphs. We then use these extracted features to discover entities in data sources.

In brief, the contribution of this study is developing an

 ISBN: 978-1-61208-104-5

algorithm to detect hidden relations in geographic data. The remainder of the paper is organized as follows. Section 2 briefly describes related work. The problem of detecting hidden relations and related concepts are introduced in Section 3. Section 4 describes our approach to detect hidden relations. Section 5 presents our evaluation corpus and comparatively discusses our approach's performance. We present conclusions and directions for further work in Section 6.

II. Related Work

LinkedMDB [8] demonstrates a novel way of link discovery and publishing linkage metadata to facilitate high volume and dense interlinking of RDF data sets. Because the data sources in LinkedMDB are about movies, it chooses movie titles as the feature to discover *owl:sameAs* links. Furthermore, users of LinkedMDB can give feedback on the quality of links. Because its stored attributes are information about titles and feedbacks, LinkedMDB can achieve high accuracy. However, it is not easy to apply the ideas behind LinkedMDB to other Web data sources that often mix terms of different attributes.

Silk [9] discovers *owl:sameAs* links that are used by DBpedia and by GeoNames to identify cities. Silk uses a declarative language for specifying which types of RDF links between data sources should be discovered as well as which conditions entities must fulfill in order to be linked. Depending on which data sources are linked, Silk has different thresholds ("accept" and "verify") for identifying similarity heuristics and qualifying the amounts of discovered links. This approach, however, only focuses on links of pairs of data sources: there is no guarantee that the information extracted from two data sources will enough to find suitable entities in remain data sources. In contrast to this approach, the solution we are advocating allows us to gather more information (by using data as keywords) in order to discover links.

III. Problem of Hidden Relations

What happens if data is published on the Web without *owl:sameAs* links? In such cases, each thing exists as a unique entity in a specific domain in which no two entities mention the same thing. This prevents people from contributing their own views and opinions about a thing. For example, someone talking about Mt. Fuji might describe its geographic location and climate at its peak whereas someone else might describe it as a scenic attraction. If entities such as these were not connected by an *owl:sameAs* link, a search might not return results on both of them. As a result, when users add more information about this thing, data might be duplicated. On the other hand, connecting these two descriptions by using an *owl:sameAs* link would help users to track down different information about the same resource. This means that the more *owl:sameAs* links there are, the richer the information will be.

Let us consider another scenario. When users create a new entity and want to link it to other entities with an *owl:sameAs* link, they have to find entities referring to the same resource from a mass of linked data. We call this task hidden entity detection or hidden relation detection, where the relations are *owl:sameAs* links. Hidden relations are possible links which have not yet been created. A possible link between b_y and c_y of the instance graph y in Figure 1 is an example of a hidden relation whereby c_y is found in data set C such that can be appropriately linked to b_y with *owl:sameAs*. Because there is a huge amount of linked data on the Web and it is steadily growing, it is not simple to detect such relations manually even if the entity's *domain*[1] is known.

Hidden entities can be linked to others, so we would have more sufficiently linked data after connecting these entities together. The problem is that an entity does not always link to all other entities in each domain, and the task of finding links among all domains would be extremely time consuming. Moreover, the URI identity often depends on the context in which it is used [10]; this means it is important to think about trustworthiness when creating relations among resources. That is, we need to check information describing resources in order to determine whether they are things we want to link together.

IV. Detecting Hidden Relations

A. Frequent Linked Data Graph

Linked data entities are either URIs or literals, and these are connected together by links. We can model such data as a graph. Many graph-related algorithms have been developed, and they have proven advantageous for solving a variety of problems in chemical informatics, computer vision, video indexing, and text retrieval [11]. We can consider URIs as the nodes of a graph and that all of them refer to the same resource through an *owl:sameAs* link. Because each URI is used only once per graph, URIs are represented abstractly by their *domain* name. For example, www.geonames.org is an abstraction of the URI www.geonames.org/964596. As a result, URIs having the same domain form a data set. Another reason for using *domain* name to represent URIs abstractly is that a resource in linked data often describes a type of information. The number of fields and their meaning for describing entities are treated similarly. Links among URIs are also represented as abstract entities. Abstract URIs and their links are made into an abstract graph.

Furthermore, each node represents a unique entity, and an edge describes a relationship between entities. For example, GeoNames store many name-feature relations as relational graphs. Particularly interesting among relational graphs are patterns that appear with high frequency [12] called frequent

[1]URIs have the same domain name

Algorithm 1 DHR_cSpan($g, D, local\_sups, S$)

Input: An abstract graph g, an instant graph dataset D, a set of support thresholds between any two domains *local_sups*.
Output: The closed frequent graph set S.

1) **if** $\exists g' \in S, g \subset g'$ and $support(g) = support(g')$ **then**
2) **return**;
3) extend g to g' as much as possible s.t. $support(g) = support(g')$;
4) **if** $\exists g'$ **then** add g' to S;
5) scan D, find every edge e such that:
6) $support(g' \cup \{e\}) \geq$ minimum{*local_sups* of domains in graph $g' \cup \{e\}$}.
7) **for each** satisfied $g' \cup \{e\}$ **do**
8) DHR_cSpan($g' \cup \{e\}, D, local\_sups, S$);
9) **return;**

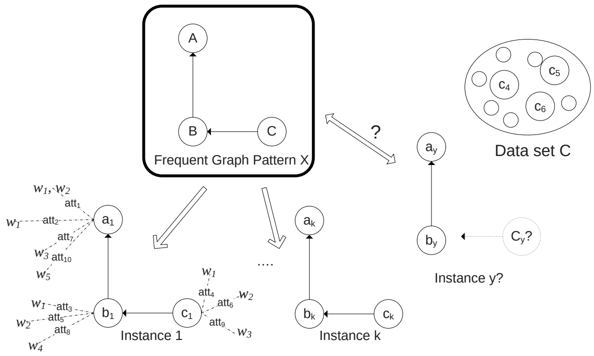

Figure 1. A hidden entity of an instance graph

patterns or graphs. Frequent graphs tend to have common relations among entities. We can extract features from the entities of such graphs and use them to identify hidden entities.

In linked data, however, the number of relation graphs is large and links are diverse. Often, there are too many frequent graphs. Because of this, it is better to mine only closed frequent graphs [12]. A frequent graph is closed if and only if there does not exist an extended graph that has the same support. The field of closed frequent graph mining has developed many algorithms, including cSpan [12], A-Close [13], CLOSET [14], CloSpan [15], and CHARM [16]. For our research, we chose to use cSpan [12] for its simplicity and efficiency in finding frequent graphs in real data. The cSpan algorithm requires choosing a support threshold for the frequency. However, we faced a problem in choosing a fixed threshold for data sets having different numbers of links. When huge data sets are connected to small data sets, it can lead to the following situation: With a fix threshold, graphs created from huge data sets tend to be very frequent because there are likely to be many links among the data. Graphs created from small data sets become relatively infrequent in comparison and hence may get dropped. For that reason, we had to modify cSpan slightly so that it could support variable thresholds. This means that, depending on which data sets are to be connected, the threshold is determined by the percentage of links between the two smallest data sets. Setting the threshold in this way enabled us to mine frequent graphs better. From here on, we shall use frequent graphs as a framework to solve our problems. Algorithm 1 (DHR_cSpan) specifies the process by which the frequent graphs are extracted. Line 6 shows the modification from algorithm cSpan of including variable thresholds.

Figure 1 illustrates a frequent graph X that has been extracted from a geographic data set on the basis of *owl:sameAs* links. There are many instances of this frequent graph $(1, \ldots, k)$. In each instance the frequent graph represents specific things. For example, supposing that the frequent graph X' includes two entities, such as Census and GeoNames, and Census links to GeoNames with an *owl:sameAs* link. Then a link from http://www.rdfabout.com/rdf/usgov/geo/us/sd/counties/perkins_county to http://sws.geonames.org/5763584/ is an instance of the frequent graph X'. Another instance is the link from http://www.rdfabout.com/rdf/usgov/geo/us/ma/counties/middlesex_county/framingham to http://sws.geonames.org/4937230/. Besides the instances of complete frequent graphs, there are graphs that lack one or more entities, such as instance y in the figure. Instance y is missing a node c_y from data set C. The reason is that c_y does not exist in this data set or there is no link to it. The way to find such missing entities is a problem that we address.

B. Attributes of the Entity

In the process of forming linked data, an RDF triple, consisting of a subject, predicate, and object, is used to represent information about resources. The subject is the URI of the described resource. The object is a literal value describing the properties of the resource or the URI of other resources. The predicate refers to links between the subject and object. Because relations in our frequent linked data graph are *owl:sameAs* links, we will consider all links except *owl:sameAs* to be attributes of the entity and the objects that are linked to as attributes' content. For example, in Figure 2, links such as *name* (link to literal value), *alternateName* (link to literal value), *inCountry* (link to URI) and even its URI name are attributes of the entity http://sws.geonames.org/283862/, whereas the *owl:sameAs* link connecting to http://dbpedia.org/resource/Gilo is not an attribute of the entity.

For the frequent graph X in Figure 1, there are three sets of attributes corresponding to three abstracted entities. The attributes' content not only describes the entity but also provides some information about the surrounding entities. Accordingly, using attributes and their content to find hidden entities is feasible. We can use useful data from

 ISBN: 978-1-61208-104-5

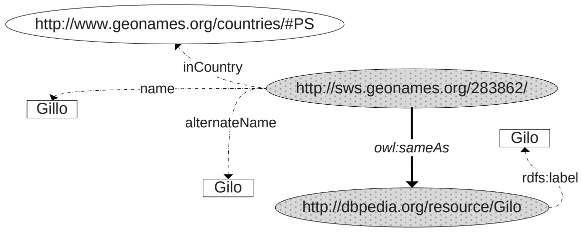

Figure 2. Attributes of an instance entity

the attributes' content as keywords for discovering entities that can be linked to it. However, the attributes of each entity in different domains vary in quantity and quality; even entities in the same domain will have such differences. Moreover, not all attributes are useful for finding hidden entities. Therefore, choosing only the most useful attributes is a prerequisite for creating a hidden relations detection algorithm.

C. Choosing Useful Attributes

The data set has information related to geographic names. As a result, we chose the feature "word" (lexical) for identifying useful attributes. The feature "word" in our paper is a sequence of characters separated by spaces. Our assumption is that entities are linked when the contents of their respective attributes have at least one word in common. This means that they mention the same concept. In Figure 2, for example, the entities http://sws.geonames.org/283862/ and http://dbpedia.org/resource/Gilo have common word "Gilo" in the attributes *alternateName*, *URI name* of the DBpedia entity and *rdfs:label*. Hence, word "Gilo" seems to be useful information for identifying the described resource. By collecting such words, we should be able to find related entities more easily. The question is, into which attributes are these words often distributed? If this question can be answered, it means that we have useful attributes. To achieve this, we should collect the words and the attributes containing those words in each instance graph. Words that do not appear in all of the entities of a graph will be removed from further consideration.

Table I shows the words extracted from attributes of the first instance graph in Figure 1, where att_i for i = $\{1, 2, \ldots\}$ are attributes of the entities of the graph, and w_j for j = $\{1, 2, \ldots\}$ are words extracted from the attributes. Since each entity belongs to a specific domain, we consider its attributes to be domain attributes. Words w_3, w_4, and w_5 do not appear in all entities of the graph. Therefore, they are removed from further consideration. Other instance graphs are similarly processed. The result is a large table of words and attributes. Our goal is to seek feature attributes that can be used to extract content for predicting hidden relations. Accordingly, we rank attributes by increasing their weight one unit whenever they appear on the word table of the instance graphs. For example, in Table I, the first attribute

Table I
WORD IN ATTRIBUTES OF AN INSTANCE GRAPHS

Instance Graph	Words	Store Attribute (in domain)
1	w_1	$att_1(A)$ $att_2(A)$ $att_3(B)$ $att_4(C)$
	w_2	$att_1(A)$ $att_5(B)$ $att_6(C)$
	w_3	$att_7(A)$ $att_9(C)$
	w_4	$att_8(B)$
	w_5	$att_{10}(A)$

appear two times, so its rank is 2. If the first attribute appears three times in the second instance graph, its rank becomes 5, and so on. Attributes that exist in many instance graphs will certainly have higher ranks than ones that only exist in a few instance graphs. Such high ranking attributes play a major role in detecting hidden relations. However, we need to consider that some attributes might be useful in some graphs but useless in other graphs. In some cases, attributes can even cause noise. Therefore, we use a threshold to reduce the number of bad attributes. Attribute rank can not go lower than the threshold. Such threshold is selected to maximize the accuracy of our approach.

Since the number of instances in each frequent graph is not the same, rank values might be quite different for different frequent graphs. In order to compare the correlation of attributes among frequent graphs as well as reduce calculating cost in later calculations, we use attribute weight instead of rank. Attribute weight is calculated from rank as follows:

$$weight_i = \frac{rank_i}{N(X)}, \tag{1}$$

where $rank_i$ is the rank of the ith attribute, and $N(X)$ is the number of instance graphs of frequent graph X. In the next section, we use the above feature attributes and their weights for finding hidden relations.

D. Distance Estimation

Here, a graph lacking an entity is a graph that is missing one entity compared with some frequent graph. A graph missing more than one entity can be dealt with recursively. That is, after we find the first entity, we look for the second entity, and so on. In Figure 1, the instance graph y consists of two entities a_y and b_y, and a missing entity c_y. Our task is to find c_y in data set C, where C is the set of entities having the same domain as c_y. In fact, entity c_y may not exist in C.

Words in the feature attributes of entities are extracted. Entities such as a_y and b_y existed in the instance graph

 ISBN: 978-1-61208-104-5

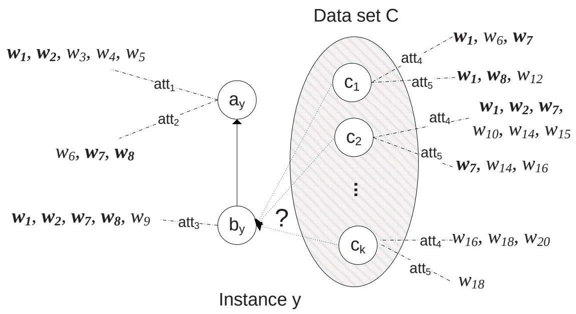

Figure 3. Words appearing in feature attributes of entities

y in Figure 1, and so the words in these entities are fewer and easier to extract. However, c_y has not yet been identified, and so we have to list all the words in each feature attribute belonging to data set $C(c_1, c_2, \ldots, c_k)$. This process consumes much time and computing resources. To reduce the burden, we index words in C and only extract words that appear in both a_y and b_y. After extracting words from the feature attributes, we remove words that do not exist in all entities. This means that we only keep words that appear in all entities of the instance graph. Note that we propose another solution for the case in which no such word exists (see the end of this section). Figure 3 illustrates words extracted from the feature attributes, $\{att_1$ (domain A), att_2 (domain A), att_3 (domain B), att_4 (domain C), att_5 (domain C)$\}$, in the instance graph y and entities in data set C.

Entity a_y and entity b_y share the word set $\{w_1, w_2, w_7, w_8\}$. Thus, entity c_y that will be detected must store the word subset $\{w_1, w_2, w_7, w_8\}$ in the content of its feature attributes. Let S_t be the set of words appearing in all entities of the graph after entity t has been inserted. In Figure 3, we have $S_1 = \{w_1, w_7, w_8\}$; $S_2 = \{w_1, w_2, w_7\}$; $\ldots$; $S_k = \{\emptyset\}$. The distance is estimated using the attribute weights and the number of words stored in S_t after S_t is projected in turn onto these attributes. For example, the set S_1 after being projected onto att_4 becomes the set $\{w_1, w_7\}$, and so the number of words in the projected S_1 is 2. For each entity c_t in data set C, the distance from it to the graph is defined as

$$l(c_t) = \frac{1}{\sum_{i=1}^{n}[weight_i \times N(\pi_{att_i}(S_t))]}, \quad (2)$$

where *n* is the number of feature attributes in the discovered domain, $weight_i$ is the weight of the *i*th feature attribute, $\pi_{att_i}(S_t)$ is the projection of the set S_t onto attribute att_i, and $N()$ is a function to count the number of words in the projected S_t. Because S_t contains words extracted from many different attributes, the projection $\pi_{att_i}(S_t)$ is a way to pick out words only from attribute att_i. Accordingly, the

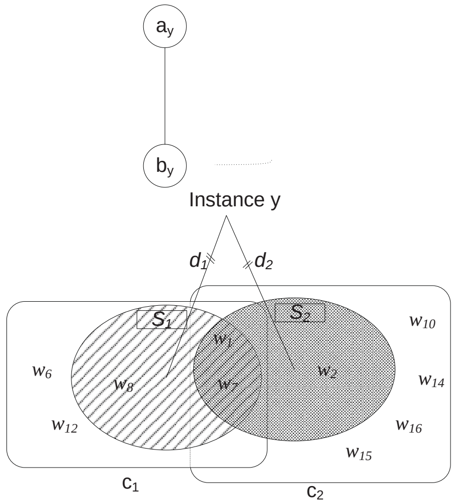

Figure 4. Two entities having the same distance to a graph

shorter the distance l of the entity is, the more suitable the entity will be to link to the graph. However, there are likely entities in C that have equal shortest distances. Therefore, we have to decide which among them should be linked to the graph next. Figure 4 shows an example of this problem. The entities c_1 and c_2 have equal distances l (i.e., $l(c_1) = l(c_2)$). Thus, we need to determine which, c_1 or c_2, is more suitable for connecting to the graph.

Words appearing in a set of feature attributes are not involved in the calculation if they do not occur in all entities of the graph. In Figure 4, these words are $w_6, w_{10}, w_{12}, w_{14}, w_{15}$, and w_{16}. These words can cause entities to become irrelevant. This means that the entity containing more words not in S_t will have a larger distance. The above considerations motivated us to use the following function:

$$d(c_t) = l(c_t) - \varepsilon \frac{1}{N(\overline{S_t})}, \quad (3)$$

where ε is a small positive number such that this measure does not affect the main distance $l(c_t)$. The resulting set from using distance d does not add any entities beyond those added using distance l. $\overline{S_t}$ is the complement of S_t (i.e., words in the content of the feature attributes of c_t do not appear in the whole graph). From this definition, we can see that the shorter the distance d of the entity is, the more suitable the entity will be for linking to the graph. Note that after this procedure, if there are still many entities with the same shortest distance d, then we must choose among them randomly or manually. Our experiment showed that this approach improved accuracy in comparison with simply using the distance l in Equation 2.

Next, we resolve the problem of entities in the graph that do not share words in different domains because of irrelevant feature attributes. In this case, the distance d from each entity

Algorithm 2 DHR-DE($g, C, P, pWeight, g'$)

Input: a graph g, a dataset C for finding entity, and a set of feature attributes P and their weight $pWeight$.
Output: a graph g' which added found entity in C to g.

1) Set current shortest distance $min\_d = -1$;
2) **for each** $c_t \in C$ **do**
3) Extend g to g' by adding c_t in graph g;
4) S_1 = extract *words* stored in feature attributes of g' that appear in all entity of g';
5) S_2 = extract *words* stored in feature attributes of g' that do not appear in all entity of g';
6) **if** $S_1 \neq \{\emptyset\}$ **then**
7) Calculate distance d using S_1 and S_2.
8) **if** $min\_d = -1$ or $min\_d > d$ **then**
9) $r = \{\emptyset\}$; $min\_d = d$;
10) Insert c_t into r;
11) **if** $min\_d = d$ **then**
12) Insert c_t into r;
13) **else**
14) **if** (graph g contains one node) **then return**;
15) **else**
16) Split graph g into subgraphs and execute lines 4 to 7 for each distance from a subgraph;
17) Calculate distance d' from each subgraph distance;
18) Insert c_t into r if distance d' is less than or equal to min_d;
19) **if** (r contains more than one entity) **then**
20) Choose and entity randomly;
21) Extend g to g' by adding such entity in graph g;
22) **return;**

in data set C to the graph is zero. One idea is to consider each entity in a graph as a separate subgraph and the distance from an entity in data set C to the graph equals the sum of distances from it to the subgraphs:

$$d'(c_t) = \sum_{j=1}^{m} d_j(c_t), \tag{4}$$

where m is the number of entities in the graph, and d_j is the distance from c_t to a subgraph that stores only one *j*th entity. Suppose that entity a_y and entity b_y in Figure 3 do not share any word, and so S is always empty. To estimate the distance, we view instance graph y from a different angle: y includes two subgraphs, one storing entity a_y and one storing entity b_y. Consequently, the distance from c_t to the graph is the sum of distances d from c_t to the subgraph storing a_y and from c_t to the subgraph storing b_y. Algorithm 2 illustrates the process used to find the most suitable entity in dataset C using distance functions from Equation 2 and 4.

V. Evaluation

We evaluated the proposed algorithm on real data sets. The data sets were derived from four publicly available geographic information sources:

The U.S. Census data is provided by the Census Bureau. The Census data comprises population statistics at various geographic levels, from the United States as a whole, to state, county, sub-county (roughly, cities and incorporated towns), so-called “census data places”, ZIP Code Tabulation Areas (ZCTAs, which approximate ZIP codes), and even deeper levels of granularity. The data set contains around 3,200 counties, 36,000 towns, 16,000 villages, and 33,000 ZCTAs [17].

GeoNames gathers geographical data, such as names of places in various languages, elevations, and populations, from various sources. All lat/long coordinates are in WGS84 (World Geodetic System 1984). It contains over 8 million geographical names and consists of 7 million unique features including 2.6 million populated places and 2.8 million alternate names [18].

The DBpedia data set is a large multi-domain ontology which has been derived from Wikipedia. The DBpedia data set contains geo-coordinates for 392,000 geographic locations [19].

The World Factbook provides information about the history, people, government, economy, geography, communications, transportation, military, and transnational conflicts of 266 world entities [20].

The above data sources were linked together with *owl:sameAs* links, creating about 100,000 connected graphs. Note that not every entity had *owl:sameAs* links; these formed empty graphs, and we did not include them in our graph set. We applied our modified cSpan to find frequent graphs in the graph set. A 20% link threshold between datasets was used. With these settings, we derived 13 frequent graphs patterns. These frequent graphs were used in the following evaluations.

To test the quality and validity of our distance measure based on feature attributes, we compared our algorithm for detecting hidden relations with a naïve algorithm. The naïve algorithm used only information about the URI name to make a prediction. We used a *k-fold* cross-validation method with $k = 10$ [21] to construct the training and test sets. That is, the dataset was split into 10 equal groups. In turn, each group was used for testing and the remaining groups were used for training. The final result is an average over choices. For each instance of a frequent graph , we evaluated the accuracy by removing one entity and attempting to find it again. Note that not all frequent graphs included all four domains (i.e., US Census, GeoNames, DBpedia, and World Factbook), so the choice of entity to be removed depended on whether it existed in the graph. Also note that the frequent graphs were directed graphs and did not have any ambiguities. Figure 5 lists the frequent graphs with the number of instances.

Figure 6 compares the accuracies of our algorithm and the naïve algorithm. Accuracy is the precision of prediction, i.e., the percentage of found entities that were correct. In the case in which we removed an entity belonging to the US Census, GeoNames, or DBpedia domain, our method

 ISBN: 978-1-61208-104-5

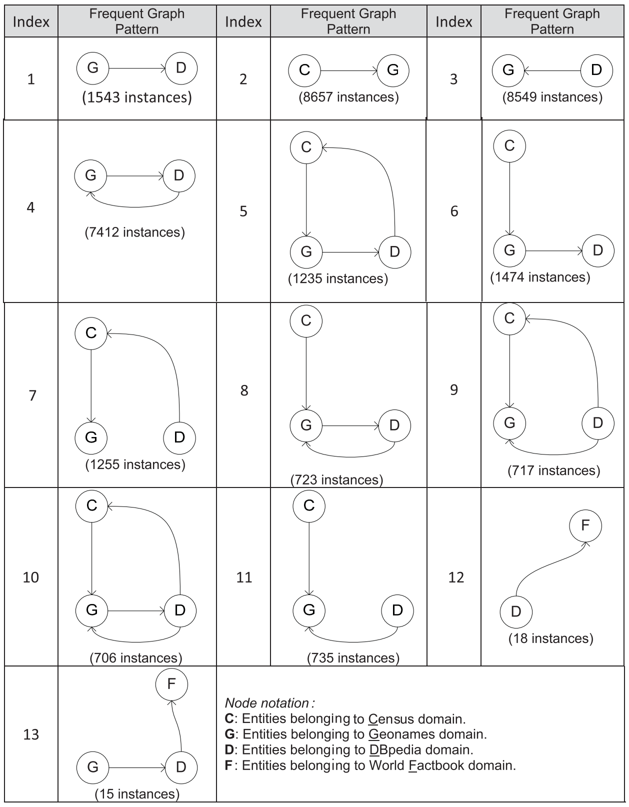

Figure 5. Closed frequent graphs

gave a better result than the naïve method. For the World Factbook, however, our method gave worse results because the names in the URIs were too well matched. For example, http://dbpedia.org/resource/Nauru and http://www4.wiwiss.fu-berlin.de/factbook/resource/Nauru match "Nauru". In our algorithm, information extracted from other feature attributes caused significant noise. However, we are only interested in the general case wherein the attributes of entities do not yield very similar information. In addition, the results for the first and third frequent graphs patterns were quite low. The reason is that if one of the entities is missing, then the information gained from the feature attributes of the other entities is not enough to detect the missing one.

Finally, we considered graphs of data sets that really were missing entities in the data and tried to predict new entities that could be linked to them. Our algorithm was able to find new entities even though the number of such graphs was very small. This task was difficult because the entities may not exist in the data set. For example, in the case of http://sws.geonames.org/5879092/ and http://www.rdfabout.com/rdf/usgov/geo/us/ak in the linked data, the two entities are linked by *owl:sameAs* and do not link to any other entity. They both refer to Alaska. Our method found a new entity in DBpedia which can link to them: http://dbpedia.org/resource/Alaska

Looking at the results, we can see that our method generally increased the completeness of the linked data. Although it was far from perfect, it easily incorporated new knowledge with few mistakes.

VI. Conclusion

We presented an approach to detecting hidden *owl:sameAs* relations in geographical data sets, such as those of the U.S. Census, GeoNames, DBpedia, and World Factbook. Since feature attributes play an important role in describing a resource, we can carry over relationships between resources. Our approach uses supervised learning to train a feature attribute set and uses the set for detecting relations. We compared the outcomes of ours and a naïve approach using only URI name data for discovering hidden relations and found that our approach has higher accuracy in most cases, especially for resources in which there are not too many matching feature attributes.

There are still many interesting aspectss to be studied in detecting relations. One of them is noise. Besides useful information, there is also superfluous information, or noise. Such noise does not describe resources, and so it makes the distance estimation worse. For example, articles, prepositions, and auxiliary verbs occur frequently, but they do not help in detecting hidden relations.

VII. Acknowledgment

We would like to thank members in NII for discussing our research and giving us valuable recommendations. We also thank the reviewers for their comments.

References

[1] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data - the story so far," *International Journal on Semantic Web and Information Systems*, vol. 5, pp. 1–22, 2009.

[2] J. Mi, H. Chen, B. Lu, T. Yu, and G. Pan, "Deriving similarity graphs from open linked data on semantic web," in *Proceedings of the 10th IEEE International Conference on Information Reuse and Integration*, 2009, pp. 157–162.

[3] P. Cudre-Mauroux, P. Haghani, M. Jost, and K. A. H. de Meer, "idMesh: Graph-based disambiguation of linked data," in *Proceedings of the 18th International World Wide Web Conference*, 2009, pp. 591–600.

[4] L. Getoor, "Link mining: A new data mining challenge," *ACM SIGKDD Explorations Newsletter*, vol. 5, pp. 84–89, 2003.

[5] Linked Data Community, "Statistics on data sets of LOD," http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/Statistics.

[6] E. Hyvonen, R. Lindroos, T. Kauppinen, and R. Henriksson, "An ontology service for geographical content," in *Proceedings of the 6th International and 2nd Asian Semantic Web Conference*, 2007, pp. 33–34.

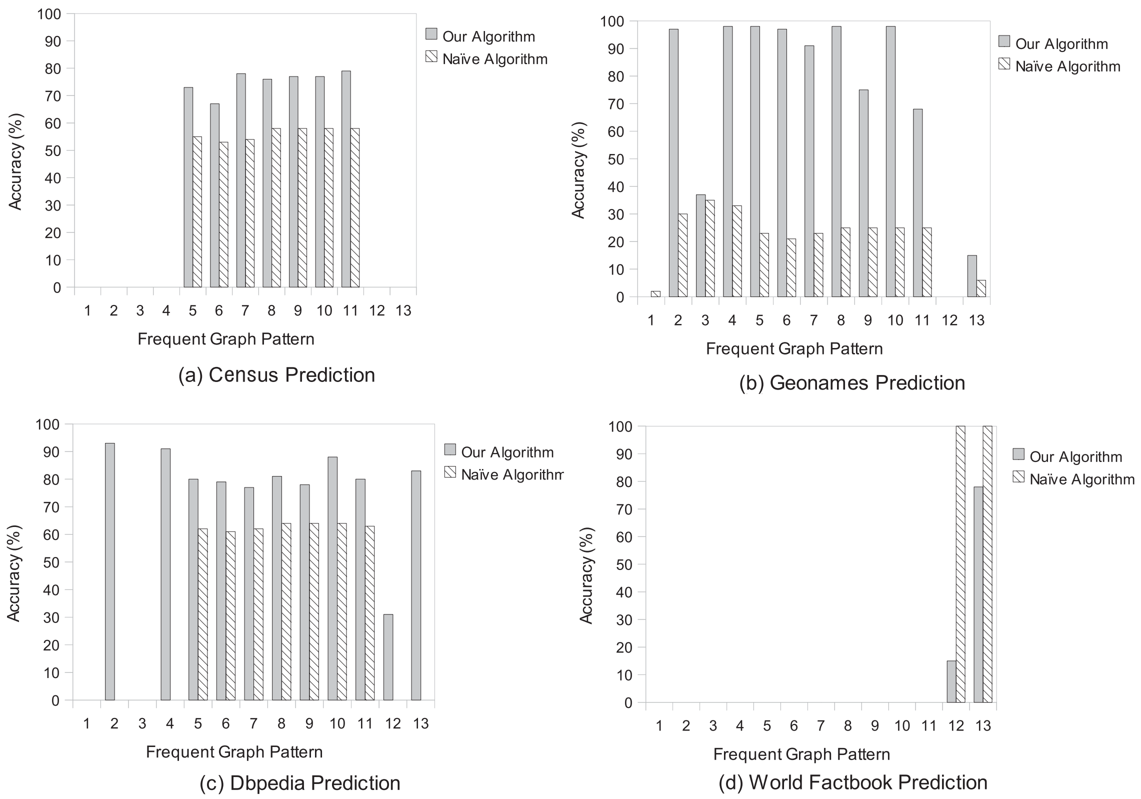

Figure 6. Accuracies of our algorithm and naïve algorithm

[7] R. Volz, J. Kleb, and W. Mueller, “Towards ontology-based disambiguation of geographical identifiers,” in *Proceedings of the WWW2007 Workshop i3: Identity, Identifiers, Identification*, 2007.

[8] O. Hassanzadeh and M. Consens, “Linked movie data base,” in *Proceedings of the WWW2009 Workshop on Linked Data on the Web*, 2009.

[9] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, “Silk - a link discovery framework for the web of data,” in *Proceedings of WWW2009 Workshop on Linked Data on the Web*, 2009.

[10] A. Jaffri, H. Glaser, and I. Millard, “URI disambiguation in the context of linked data,” in *Proceedings of the WWW2008 Workshop on Linked Data on the Web*, 2008.

[11] D. J. Cook and L. B. Holder, *Mining Graph Data*. John Wiley and Sons, 2007.

[12] X. Yan, X. J. Zhou, and J. Han, “Mining closed relational graphs with connectivity constraints,” in *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005, pp. 324–333.

[13] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, “Discovering frequent closed itemsets for association rules,” in *Proceedings of the 7th International Conference on Database Theory*, 1999, pp. 398–416.

[14] J. Pei, J. Han, and R. Mao, “Closet: An efficient algorithm for mining frequent closed itemsets,” in *Proceedings of 2000 ACM-SIGMOD International Workshop Data Mining and Knowledge Discovery*, 2000, pp. 11–20.

[15] X. Yan, J. Han, and R. Afshar, “Clospan:mining closed sequential patterns in large datasets,” in *Proceedings of 3rd SIAM International Conference on Data Mining*, 2003, pp. 166–177.

[16] M. J. Zaki and C.-J. Hsiao, “Charm: An efficient algorithm for closed itemset mining,” in *Proceedings of 2nd SIAM International Conference on Data Mining*, 2002, pp. 457–473.

[17] J. Tauberer, “The U.S. census data,” http://www.rdfabout.com/demo/census/, 2007.

[18] M. Wick, “The GeoNames geographical database,” http://www.geonames.org/.

[19] DBpedia Team, “The DBpedia database,” http://wiki.dbpedia.org/, 2009.

[20] CIA Factbook D2R Server, “The World Factbook database,” http://www4.wiwiss.fu-berlin.de/factbook/.

[21] G. J. McLachlan, K.-A. Do, and C. Ambroise, *Analyzing microarray gene expression data*. John Wiley and Sons, 2005.