Ontological Networks: Mapping Ontological Knowledge Bases into Graphs

Lucas Fonseca Navarro
Federal University of Sao Carlos
Sao Carlos - SP, Brazil
navarro.lucasf@gmail.com

Estevam Rafael Hruschka Jr.
Federal University of Sao Carlos
Sao Carlos - SP, Brazil
estevam@dc.ufscar.br

Ana Paula Appel
IBM Research
Sao Paulo - SP, Brazil
apappel@br.ibm.com

ABSTRACT

With the exponentially growing amount of available data on the Web over the last years, several projects have been created to automatically extract knowledge from this information set. As the data domains on the Web are too wide, most of these projects store the acquired knowledge in ontological knowledge bases (OKBs). Mapping it into graph-based representation makes possible to apply graph-mining techniques to extract implicit information. However most of these projects treat the mapping process using different adjustments in several ways, thus, there is not a standard mapping process or a formal way defined to do this task. In this paper we formally describe a graph structure called Ontological Network and how it can be used to map an Ontological Knowledge Base. We also show some graph-mining based algorithms to find new facts and to extend the ontology of an OKB while mapped into an Ontological Network as example.

CCS Concepts

•**Computing methodologies** → **Ontology engineering;** *Machine learning algorithms;*

Keywords

Ontology engineering; Machine Learning; Graph Mining; Ontological Knwoledge Bases

1. INTRODUCTION

Over the last few years, many research projects focused on building large scale ontological knowledge bases(OKB) have been developed, such as Knowledge Vault [5], Freebase [3], YAGO [15] and a continuously learning program called NELL (Never Ending Language Learner) [4]. These projects store their knowledge using what we call Ontological Knowledge Bases (OKBs). Considering the wide variety of different domains present in the knowledge bases, an ontology can be used for better organization.

An OKB can be divided in two parts: the ontological model, that determines a set of categories (e.g $\{athlete$, $sportsTeam$, $fruit$, $emotion$, ...$\}$) and how they can be related to each other (e.g $\{athletePlaysForTeam(athlete$, $sportsTeam\}$). The second part is the set of facts, which are the set of instances following the ontological model (e.g $\{fruit(apple)$, $athlete(Ward)$, $sportsTeam$ $(Pittsburgh$ $Steelers)$, $athletePlaysForTeam(Ward$, $Pittsburgh$ $Steelers)\}$).

Due to the power of graph-based algorithms, it is common to find different approaches to map data from KBs (and also OKBs) into graphs, and then, to apply graph-mining techniques that is generally used to find implicit information that are already on the KB. Most of the time a human being is capable to infer such implicit information, but this is a hard task for a machine (such as learning programs that populates the KB from corpora or the Web).

One of the most studied and disseminated topics of graph-mining is the link-prediction task, in which the goal is to estimate the likelihood of future existence of an edge between two nodes, based on the current graph information[7]. This task is most applied in social networks, that also are mapped as graphs, to recommend new friends like "People you may know". This task can also be used to find implicit knowledge from a graph mapping a KB.

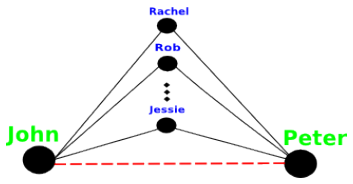

Figure 1: Link-Prediction example

In **Figure 1** we show an example of how link-prediction is used to recommend new friends to users of a social network. In the example we can see that the users (nodes of the graph) John and Peter have several friends in common, so there is a higher probability of both of them being (or becoming) friends in real life. Thus, one can be recommended as friend to the other. This is the idea of the common-neighbors predictor[12].

Most of the times a KB is mapped into a graph to execute link-prediction techniques to find new edges(e.g Figure 1) that is the same as new facts to the KB. When using

WWW'16 Companion, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2890587

an OKB properly mapped into a graph model, we could use link-prediction to find new facts[10], but also to find new categories[16] and relations[1] to extend the ontological model as well.

In **Figure 2** we show an example of how link-prediction can be used to find a new possible relations to extends the ontological model. The idea is the same as the one used when finding new facts, but counting edges from groups of nodes instead of single nodes: a lot of nodes of the category *athlete* are related to nodes of the category *sport* that are related to nodes of the category *sportsLeague*, so probably there's a relation between *athletes* and *sportsLeagues*, in this case the relation *athletePlaysLeague*.

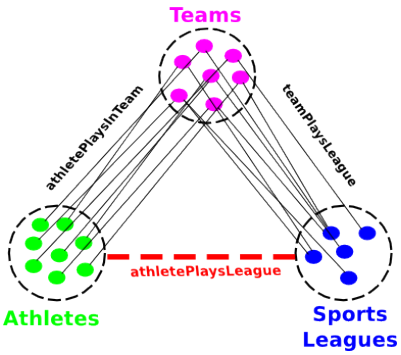

Figure 2: Link-Prediction to find new relations

Despite some projects already defined a way to map OKBs into graphs to use graph mining techniques, most of these projects did it using different adjustments in several ways. In addition, there are not many works describing this process formally, as there is not a standard to this mapping process.

In this paper we present a formal definition of an Ontological Network, using graphs to map ontological knowledge bases, to apply graph-mining techniques, not just to find new facts(instances) for the KB, but also to extend its ontological model finding new categories and new relations. We'll also present some algorithms over the proposed model to consolidate our approach.

2. RELATED WORKS

Description logics[2] is a formal language that can be used to describe ontologies, see and example above:

- Constants: Neymar, Messi, Soccer
- Predicates(arity): Sport(1), Athlete(1), plays(2), teamates(2)
- Variables: x, y, z
- Operators: $\wedge, \vee, \neg, \Rightarrow, \ldots$
- Quantifiers: $\forall, \exists$
- Formulas:
  - Athlete(Neymar), Sport(Soccer), plays(Neymar, Soccer), teamates(Neymar, Messi)
  - Athlete(x), plays(x,y) $\wedge$ teamates(x,z) $\Rightarrow$ plays(z,y)

This model is suitable to represent a simple ontology, the categories and relations could be represented by the predicates, while the instances would be the constants. The model we will present in the next sections (which we call Ontology Networks) uses some logic notations and actually it could be presented using this language. But the characteristics of our model are more specific to work with ontologies as graphs, thus, it is designed to be easier to learn and simpler to use (more details are given in Section V).

There are libraries such as graphOnt[8] that maps ontologies into graphs for better visualization or analysis, and also a class called OWLGraphWrapper[1] from OWLTools. These libraries are very useful to directly apply some graph-like operations over ontological knowledge bases, but none of them are described as a formal model.

3. BACKGROUND

The most disseminated techniques of graph-mining to find implicit knowledge are related to link-prediction. Most of these methods are based on graph structural properties [11] in which the goal is to assign connection values, called *score* $(u; w)$, to pairs of nodes $\langle u, w\rangle$ based on a graph G. Traditionally, the assigned scores are later ranked in a list in decreasing order of $score(u; w)$, and then, predictions are made according to this list. For a node u in the graph G, let $\Gamma(u)$ denote the set of neighbors of u in G. A number of link prediction approaches are based on the idea that two nodes in G (e.g u and w) are more likely to link to each other, in the future, if their sets of neighbors $\Gamma(u)$ and $\Gamma(w)$ largely overlap.

The most straightforward implementation of this link-prediction idea is the common-neighbors metric [12], under which the *scores* are defined as $score(u; w) := |\Gamma(u) \cap \Gamma(w)|$. A common-neighbors predictor captures the basic notion, inspired in social networks, that two strangers who have a common friend may be introduced by that common friend and, thus, become friends themselves. When analyzing such *introduction* act in a graph-based context, it has the effect of "closing a triangle" in the graph and feels like a common mechanism in real life [17].

Community detection is also a very famous approach used in graph mining area. It is used to find communities (also called clusters) in a graph most of the times based on its topology, for example using graph cliques (sub-graphs in which every node is connected to every other node in the clique), a review on these algorithms can be read in [9].

4. DEFINITIONS

Let $G = (V, E)$ be an undirected graph with a set of nodes V and a set of edges E. $\Gamma(u) := \{v \in V : \exists\{v, u\} \in E\}$ of node u is defined to be the set of nodes in V that are adjacent to u. The number of common neighbors between two nodes (u and v) can be defined as $\aleph(u, v) = |\Gamma(u) \cap \Gamma(v)|$.

A closed triangle $\Delta(u, v, w)$ of a graph $G = (V, E)$ is a set of three complete connected nodes where $u, v, w \in V$ and $\Delta(u, v, w) = \{< u, v >, < v, w >, < w, u >\} \in E$. An open triangle $\Lambda(u, w)$ of a graph $G = (V, E)$ is three connected node where $\Lambda(u, w) = \{(u, v), (v, w)\} \in E \wedge \{u, w\} \notin E$. The $\Delta_c(c_1, c_2, c_3)$ represents all the closed triangles composed by node's of categories c_1, c_2 and c_3 and $\Lambda_c(c_1, c_2)$ represents all the open triangles composed by node's categories of c_1 and c_2 (in this case, the middle nodes categories doesn't matter). Any Δ_c is called a **closed triangles category group** and

[1]http://owltools.googlecode.com/svn/trunk/docs/api/owltools/graph/OWL GraphWrapper.html

any Λ_c is called a **open triangles category group**. The c_u of a node u is the set of the categories that u belongs to.

An Inference Rule (or just **rule** for simplicity) is a logical form, consisting of a conclusion r, and premises $p_1, p_2, ..., p_n$. One of the possible representations is $r \Longleftarrow p_1 \wedge p_2 \wedge ... \wedge p_n$. The premises and the conclusion are literals than they can be predicates (p), a logical function $p(x_1, x_2, ..., x_n)$ that can only return true of false ($x_1, x_2, ..., x_n$ are variables)

5. BUILDING AN ONTOLOGICAL NETWORK

5.1 Ontological Knowledge Base (OKB)

To define an ontological network created from an OKB, we need to formally define an OKB first. In this section an OKB format we call KB_o^2 is defined. To do so it is used predicate logic elements.

DEFINITION 1. *An Ontological Knowledge Base $KB_o^2 = (C, H, R, I)$ is composed by four components: a set of categories (C), a set (H) of predicates "ako(a kind of)" that express the hierarchy against the categories, a set (R) of predicates that express relation among the categories.*

- *$\forall$ predicates $p(t_1, t_2) \in H, R$ the arity of p is 2 ($p/2$) and $t_1, t_2 \in C$;*
- *The predicate "ako" $\in H$ are transitive: if $ako(t_1, t_2)$, $ako(t_2, t_3) \in H$, then $ako(t_1, t_3) \in H$;*
- *It must not exist predicates with equal terms in H: $ako(t, t) \notin H$;*

At last, a set of instances(I) of the categories(C) and relations(R). The instances set(I) are the facts of the KB_o^2. The set (I) can be divided into two subsets, the categorization set and the relational set ($I = I_c \cup I_r$)

- *The categorization set (I_c) is composed by predicates $p_c(t)$ with names of the categories in C (p_c in C) and the arity of p_c is 1 ($p_c/1$).*
- *The relational set (I_r) is composed by predicates $p_r(t_1, t_2)$ such that: $p_r(c_1, c_2) \in R$ and $c_1(t_1), c_2(t_2) \in I_c$.*

The number 2 in the name of our OKB model (KB_o^2) was chosen because all relations on this model have arity 2. Below we present an example to better illustrate how KB_o^2 works.

EXAMPLE 1. *Consider an ontological knowledge base $KB_o^2 = (C, H, R, I)$ used to store data from a social network that have different types of relations.*

- *C={Person, MaleP, FemaleP};*
- *H={ako(MaleP, Person), ako(FemaleP, Person)}*
- *R={friends(Person, Person), fatherOf(MaleP, Person), motherOf(FemaleP, Person), descendant(Person, Person), relationship (MaleP, FemaleP), relationship (FemaleP, MaleP)}*
- *$I = I_c \cup I_r$ = {Person(Lucas), MaleP(Lucas), Person(Jose), MaleP(Jose), Person(Raquel), FemaleP(Raquel), Person(Julia), FemaleP(Julia), Person(Vinicius), MaleP(Vinicius), Person(Leo), ...} $\cup$ {friends (Lucas, Leo), friends(Vinicius, Leo), friends (Julia, Raquel), relationship (Lucas, Julia), relationship(Raquel, Jose), relationship(Jose, Raquel), descendant(Lucas, Jose), descendant(Vinicius, Jose), descendant(Vinicius, Raquel), fatherOf(Jose, Lucas), motherOf(Raquel, Lucas), motherOf(Raquel, Vinicius), ...}*

5.2 Ontological Networks (N_o)

In this subsection we'll define an ontological network (N_o) created from an arbitrary ontological knowledge base KB_o^2.

DEFINITION 2. *An Ontological Network $N_o = (G_m, G_i)$ is composed by two graphs, an ontological model graph (G_m) and an ontological instances graph (G_i).*

An Ontological Model Graph $G_m = (V_m, E_m)$ has the same purpose of the model parts on the KB_o^2, to determine the categories that the instances can be and the possible relationships between categories. The set of nodes V_m is composed by labeled nodes that represents categories, and the set of edges E_m is composed by labeled edges that represents a relation name.

DEFINITION 3. *Given an Ontological Knowledge Base $KB_o^2 = (C, H, R, I)$, an Ontological Model Graph $G_m = (V_m, E_m)$ can be created to map the ontological model of KB_o^2 such as:*

- *$V_m \equiv C$: $\forall\, c \in C\ \exists\, v \in V_m \mid v.label = c$;*
- *$E_m \equiv H \cup R$: $\forall\, p(t_1, t_2) \in H \cup R\ \exists\, e \in E_m \mid e = p < t_1, t_2 >$;*

To better illustrate an ontological model graph, an example is present below, mapped from the KB_o^2 of Example 1.

EXAMPLE 2. *Given the KB_o^2 of Example 1, the ontological model graph $G_m = (V_m, E_m)$ mapped by it would be:*

- *V_m = {Person, MaleP, FemaleP};*
- *E_m = {ako < MaleP, Person >, ako < FemaleP, Person >, friends< Person, Person >, fatherOf< MaleP, Person >,...};*

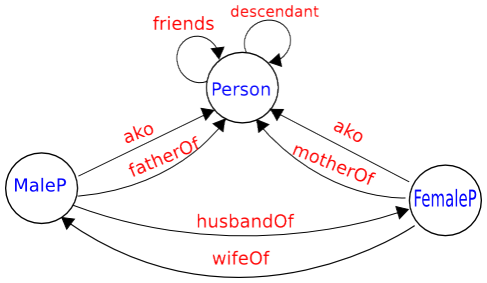

Figure 3: Ontological Model Graph Example

Figure 3 is the graphic representation of the graph presented in Example 2.

An Ontological Instances Graph $G_i = (V_i, E_i, X)$ is the graph that will be the network of the instances of the KB. The set of nodes V_i is composed by labelled nodes, representing the instances (the parameters of the predicates $\in I$). The set of edges E_i is composed by labeled edges that represents relations among two instances. The set X is the category list, having the category information for each node.

DEFINITION 4. *Given an Ontological Knowledge Base* $KB_o^2 = (C, H, R, I_c \cup I_r)$ *and Ontological Model Graph* $G_m = (V_m, E_m)$ *mapped from* KB_o^2*, an ontological instances graph* $G_i = (V_i, E_i, X)$ *can be created to map the instances of* KB_o^2 *such as:*

- $\forall\, p_c(t) \in I_c\ \exists\, v \in V_i \mid v.label = t;$
- $E_i \equiv I_r$: $\forall\, p_r(t_1, t_2) \in I_r\ \exists\, e \in E_i \mid e = p_r < t_1, t_2 >;$
- $X \equiv I_c$: $\forall\, p_c(t) \in I_c\ \exists\, x \in X \mid x = (t, p_c);$

To better illustrate an ontological instances graph, an example is present below, mapped from the KB_o^2 of Example 2.

EXAMPLE 3. *Given the* KB_o^2 *of Example 1, the ontological instances graph* $G_i = (V_i, E_i)$ *mapped by it would be:*

- V_i = {*Lucas, Jose, Raquel, Vinicius, Leo, Julia, ...*};
- E_i = {*friends(Lucas, Leo), descendant(Lucas, Jose), fatherOf(Jose, Vinicius), relationship(Lucas, Julia), ...*};
- X = {*(Lucas, Person), (Lucas, MaleP), (Raquel, Person), (Raquel, FemaleP), (Leo, Person), ...* }

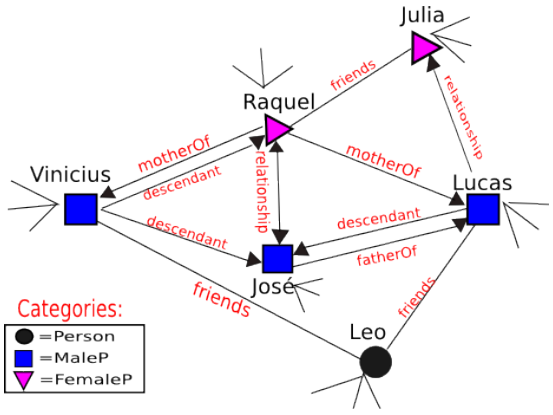

Figure 4: Ontological Instances Graph Example

Figure 4 is the graphic representation of the portion of the graph presented in Example 3 mapped from I of Example 1. The categorization set X is represented by the node colours and shapes.

It's an alternative format for a N_o that can be also very interesting to apply graph-mining algorithms, a format with just one unified graph:

$$N_o^{'} = (V_m \cup V_i, E_m \cup E_i \cup X)$$

In $N_o^{'}$ there's just one graph, unifying the ontological model with the instances, in practice this network generally will look like a small world network[13], with the category nodes being the hubs.

We use an KB_o^2 to build (and define) and Ontological Network, but an N_o can exists by itself, and not just mapped from a OKB. Actually the KB_o^2 and the N_o are equivalent models, it's possible to represent the two of them using the other one.

6. APPLICATIONS

When mapping an OKB into an Ontological Network (N_o), becomes possible to apply graph-mining techniques, not just to find facts for the OKB, but to extend its ontology as well, finding new relations and new categories. The purpose of this section is to consolidate the importance of having a well defined model of N_o to map an OKB, by showing some briefly examples and ideas of algorithms to extract implicit facts and ontology information by mining the N_o.

Next, we present three applications of graph-mining algorithms to extend an ontological network. An algorithm to find new relations, an idea of how to use community detection to find new categories, and an algorithm to find inference rules that uses the ontology information.

6.1 Finding New Relations

Link-prediction techniques, can be referred as the task to find edges that would appear in a near future of a graph. In the context of graphs mapping knowledge bases these techniques are mostly used to find new facts, but they can be used as well to find new edges on the ontological model graph that is equivalent to new relations to an OKB as well, using the model of N_o proposed.

Algorithm 1 Relations-Predictor

Require: $N_o(G_m, G_i) \mid G_m = (V_m, E_m), G_i = (V_i, E_i, X)$
Ensure: New edges on E_m
1: Find all $\Lambda((u,p),(w,q))$ in G_i
2: **for all** open triangle $\Lambda((u,p),(w,q))$ **do**
3: Calculate $score_{CN}(u,w) = |\Gamma(u) \cap \Gamma(w)|$
4: Group $\Lambda((u,p),(w,q))$ in $\Lambda_c((c_u,p),(c_w,q))$
5: **end for**
6: **for all** $\Lambda_c((c_u,p),(c_w,q))$ **do**
7: Calculate

$$score^c_{CN}(c_u, c_w) = \sum_{\forall \Lambda(u,w) \in \Lambda_c((c_u,p),(c_w,q))} score_{CN}(u,w)$$

8: **if** $score^c_{CN}(c_u, c_w) \geq \xi$ **then**
9: Create the edge: $rel < c_u, c_w >$ in E_m
10: **end if**
11: **end for**

The Relations-Predictor Algorithm is a generic algorithm created just to exemplify the task of finding new relations using link-prediction techniques in an N_o. First it finds all open triangles $\Lambda((u,p),(w,q))$ present on the graph, calculates its common-neighbours score[12] and group then into the respective open triangle category group $\Lambda_c((c_u,p),(c_w,q))$. After that, for each category group, it computes the sum of all CN scores ($score^c_{CN}(c_u, c_w)$) and if this value is greater or equal then a given threshold(ξ), it creates a new edge $rel < c_u, c_w >$ that is a relation between the categories of the group.

In the **Figure 5** we use our imaginary social network to illustrate how the Relations-Predictor algorithm can be used, in that case, if it finds too much open triangles with that pattern, where the two edges are the relation "descendant" and the the nodes of the category "Person" (Λ_c((Person,descendant), (Person,descendant))), it may find that exists a relation, in this case "*brothers* $<$ *Person, Person* $>$". We can observe that in the algorithm the relations found have generic names, another program or human supervision are needed to name the relation like in the example.

The Prophet[1] is a component of NELL[4] that finds new

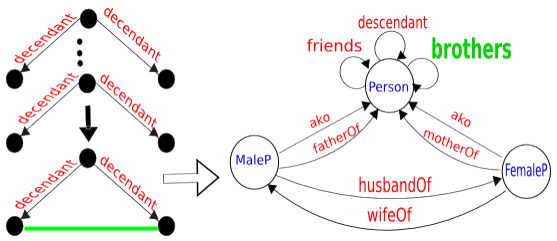

Figure 5: Example: Finding a new relation

relations among two categories to extend its ontology by mapping NELL's KB into a graph. Additionally it finds instances of the new found relations(new facts) and misplaced edges on the graph (wrong facts on NELL's OKB). Its algorithm is present in[1], it's very similar to the Relations-Predictor Algorithm, it uses a link-prediction index called extra-neighbours, and groups the open triangles in category groups either. If we uses the definition of N_o would be possible to better describe Prophet's algorithm. For curiosity purpose, currently Prophet is working with OntExt[14], prophet finds pairs of categories that might be related and OntExt finds names to these pairs.

6.2 Finding New Categories

Another disseminated topic of graph-mining the graph clustering, sometimes called community detection algorithms[6]. Briefly explaining, this task uses the topology information of the graph to divide it in clusters of nodes.

To find new categories in an N_o, we can use community detection algorithms in a similar way of the link-prediction. We apply the algorithm in the graph, then analyses each community that were found (such as the triangle category groups), and if the community attends to a given condition we can create a new category from it. Another option would be if the communities shares common features(same set of relations for example), then each community could be an instance of a new possible category (see Figure 6).

In the **Figure 6** we exemplify how we can find a new category using community detection in our imaginary social network. Imagine that we apply a community detection algorithm that returns communities where nodes inside of them are mostly related by the following relations: *descendant, fatherOf, motherOf, brothers* (Figure 6 (a)). Observing this pattern, we can infer that this communities are families, so the category "family" can be added to our N_o (Figure 6 (b)).

6.3 Finding Inference Rules (new facts)

We're currently working in a new component of NELL[4] to find inference rules through graph-mining that we call the Graph Rule Learner(GRL), it uses Prophet's link-prediction metric, the extra-neighbours index to find and make a threshold to validate the rules and choose what predicate will be the conclusion(head) of the rule.

In **line 1**, the algorithm will find and list all the closed triangles $\Delta(u, v, w)$ of the graph G, the number of neighbours $\aleph$ between each pair of nodes of the triangle are calculated (e.g $\aleph(u, v)$), and grouped in the respective open triangle category group Λ_c[2] (e.g $\Lambda(c_u, c_v)$). The closed triangle is also grouped in the closed triangle category group

[2]Despite the three nodes are connected. It is considered that the edge between the pair of parameters of *Lambda* doesn't exist in each group

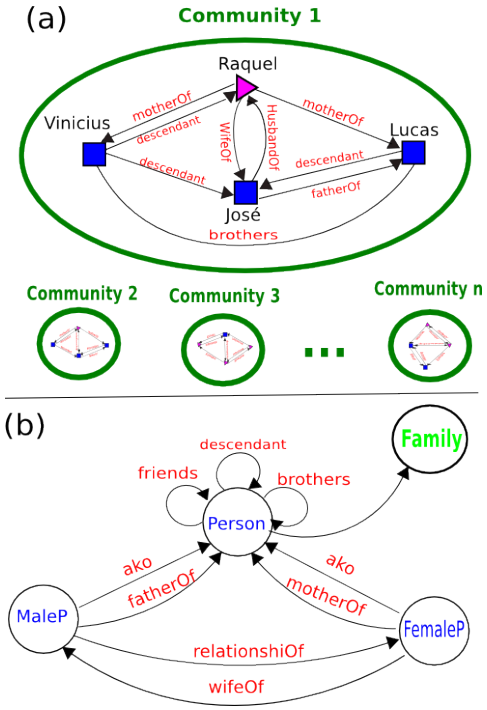

Figure 6: Example: Finding a new category

$\Delta_c(c_u, c_v, c_w)$. Next in **line 8**, for each open triangle category group $\Lambda_c(c_i, c_j)$, the number of extra neighbours $\aleph_c$ will be calculated. Then in **line 11**, for each close triangle category group $\Delta_c(c_u, c_v, c_w)$, the pair of categories with the highest extra neighbour value $\aleph_c$ will be selected, e.g (c_u, c_v). Then, if the extra neighbour value of this pair is greater or equal then a given threshold ξ, validate the rule $r_{c_u c_v}(c_u, c_v) \Longleftarrow r_{c_u c_w}(c_u, c_w) \wedge r_{c_w c_v}(c_w, c_v)$.

One literal $r_{c_x c_y}(c_x, c_y)$ indicates a relation(the predicate r) $r_{c_x c_y} \in E_c$ between the categories c_x and c_y, so the paremeters of the predicate $r_{c_x c_y}$ need to be instances of the categories c_x and c_y respectively.

To exemplify this in using our imaginary social network after adding the relation $brother < Person, Person >$, we could apply GRL and possibly obtain the rule above:

$$brothers(Person_X, Person_Z) \Longleftarrow$$
$$descendant(Person_X, Person_Y)$$
$$\wedge\, descendant(Person_Z, Person_Y)$$

This rule can be used to find more instances of the relation brother to the N_o, adding edges to G_i (the same of facts to the OKB).

After the application of the graph-mining algorithms presented in the last three subsection in our imaginary social network, in **Figure 6 (b)** the final ontological model graph is present, and in **Figure 7** we present how the ontological instances graph could be augmented with that.

7. DISCUSSIONS AND CONCLUSION

In this work we presented a structure called ontological network(N^o) that can be used to map and ontological

Algorithm 2 The GRL

Require: $G_i = (V, E, X)$
Ensure: List of Inference Rules
1: Find all $\Delta(u, v, w)$ in G
2: **for all** closed triangle $\Delta(u, v, w)$ **do**
3: Calculate $\aleph(u, v)$, $\aleph(v, w)$ and $\aleph(w, u)$
4: Group $\Delta(u, v, w)$ in $\Delta_c(c_u, c_v, c_w)$
5: Group $\Lambda(u, v)$ in $\Lambda_c(c_u, c_v)$, $\Lambda(v, w)$ in $\Lambda_c(c_v, c_w)$ and $\Lambda(w, u)$ in $\Lambda_c(c_w, c_u)$
6: **end for**
7: **for all** $\Lambda_c(c_i, c_j)$ **do**
8: Calculate

$$\aleph_c(c_i, c_j) = \sum_{\forall \Lambda(u,v) \in \Lambda_c(c_1, c_2)} (\aleph(u, v) - 1)$$

9: **end for**
10: **for all** $\Delta_c(c_u, c_v, c_w)$ **do**
11: Find the category pair with highest $\aleph_c$: $(c_i, c_j) = MAX(\aleph_c(c_u, c_v), \aleph_c(c_v, c_w), \aleph_c(c_w, c_u))$
12: **if** $\aleph_c(c_i, c_j) \geq \xi$ **then**
13: Validate the rule: $r_{c_i c_j}(c_i, c_j) \Longleftarrow r_{c_i c_k}(c_i, c_k) \wedge r_{c_k c_j}(c_k, c_j)$
14: **end if**
15: **end for**

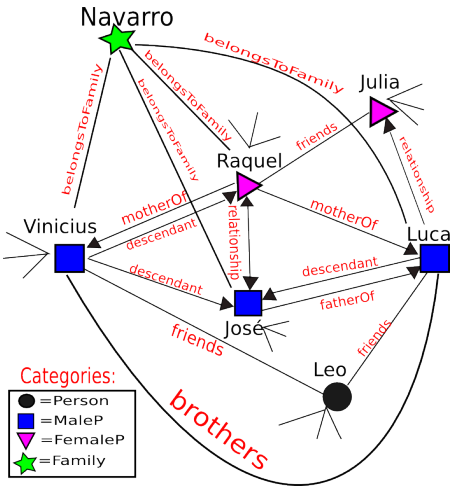

Figure 7: New Ontological Instances Graph

knowledge base(OKB) for the execution of graph-mining algorithms to extract implicit information from it. The goal we wanted to achieve with this article(and the N^o structure) is to formally define a structure to be used assisting another projects that uses ontological knowledge bases and intends to map it into graphs to apply graph-mining algorithms on it. We also wanted to present some ideas and demonstrate by using simple algorithms that graph mining techniques can be very useful to extends OKBs.

We already implemented a version of Prophet within OntExt and the Graph Rule Learner, both functional to work with NELL. We also start to work on the task of find new categories. The structure (N^o) was widely used in the design and documentation of all these algorithms. This project still lack of proof of how this structure is really usefull and measure it's usefullness and other characteristics of it, but we plan to work on that in a near future.

8. REFERENCES

[1] A. P. Appel and E. R. Hruschka Junior. Prophet – a link-predictor to learn new rules on nell. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, ICDMW '11, pages 917–924, Washington, DC, USA, 2011. IEEE Computer Society.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Pate-Schneider. *The description logic handbook: Theory, implementation, and applications*. Cambridge University Press, 2 edition edition, 2010.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *In Proceedings of SIGMOD*, 2008.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of AAAI*, 2010.

[5] I. L. Dong, K. Murphy, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion, 2014.

[6] S. E. S. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[7] L. Getoor and C. P. Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7:3–12, 2005.

[8] S. Kelley, M. Goldberg, M. Magdon-Ismail, K. Mertsalov, W. Wallace, and M. Zaki. graphont: An ontology based library for conversion from semantic graphs to jung. *Intelligence and Security Informatics, ISI'09*, pages 170–172, 2009.

[9] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5), 2009.

[10] N. Lao, T. Mitchell, and W. W. Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[11] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of CIKM*, pages 556–559, New York, NY, USA, 2003. ACM.

[12] F. Lorrain and H. White. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, 1:49–80, 1971.

[13] S. Milgram. Tje small-world problem. *Psychology Today*, pages 62–67, 1967.

[14] T. Mohammed, E. R. Hruschka Jr., and T. M. Mitchell. Discovering relations between noun categories. *EMNLP*, ACL:1447–1455, 2011.

[15] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *In Proceedings of WWW*, 2007.

[16] P. Velardi, S. Faralli, and R. Navigli. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics Journal*, 2013.

[17] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 1998.