Finding Inference Rules using Graph Mining in Ontological Knowledge Bases

Lucas Fonseca Navarro*, Estevam R. Hruschka Jr.* and Ana Paula Appel†
*Federal University of Sao Carlos
Rodovia Washington Lus, Km 235 - SP 310, Sao Carlos - SP, CEP 13565-905 Tel.: +55 16 3351-8111
E-mails: navarro.lucasf@gmail.com, estevam@dc.ufscar.br
†IBM Research Brazil
Rua Tutoia, 1157 - TUT05, Sao Paulo, SP, Brazil - CEP 04007-900 - E-mail: apappel@br.ibm.com

***Abstract*—The exponentially grow of Web and data availability, the semantic web area has expanded and each day more data is expressed as knowledge bases. Knowledge bases (KB) used in most projects are represented in an ontology-based fashion, so the data can be better organized and easily accessible. It is common to map these KBs into a graph when trying to induce inference rules from the KB, thus it is possible to apply graph-mining techniques to extract implicit knowledge. One common graph-based task is link prediction, which can be used to predict edges (new facts for the KB) that will appear in a near future. In this paper, we present Graph Rule Learner (GRL), a method designed to extract inference rules from ontological knowledge bases mapped to graphs. GRL is based on graph-mining techniques, and explores the combination of link prediction metrics. Empirical analysis reveled GRL can successfully be applied to NELL(Never-Ending Language Learner)[1] helping the system to infer new KB beliefs from existing beliefs (a crucial task for a never-ending learning system).**

I. Introduction

In the last years a number of different research projects focused on building large scale ontological knowledge bases (also called ontologies), such as Knowledge Vault [1], Freebase [2], YAGO [3] and a continuously learning program called NELL (Never Ending Language Learner) [4]. Traditionally, an ontological knowledge base (OKB) organizes and stores knowledge in two different parts, namely: i) an ontological model, where categories (*city, company, person*, etc.) and relations (**worksFor**(*person*, *company*), **headQuarteredIn**(*company*, *city*))) are defined, and ii) a set of facts which are instances of categories (**city** (*New York*), **company**(*Disney*), **person**(*Walt Disney*)) and instances of relations (**headQuarteredIn**(*Disney*, *Orlando*))

NELL is the main motivation of this work, mainly because of its never-ending learning characteristics. To build its continuously growing knowledge base, NELL reads the web 24 hours a day, 7 days a week with two specific goals: *i) extend its own knowledge base* and, *ii) improve its own learning abilities*. To allow NELL performing these tasks, different components, based on different approaches are coupled together following the *Never-Ending Learning Principles*[4] (most of these approaches are listed in NELL's website[2]). NELL's different components work together extracting information and learning from the web, as well as from NELL's own knowledge base.

Even in a never-ending learning approach, the question of how to develop methodologies to help populating ontological KBs and improving their coverage is still a challenge [5]. Thus, the use of a rule-based inference approach (here called *Rule Learner* - RL) can have relevant impact in the KB population task. In general, the goal of a RL is to induce inference rules from structured or unstructured data[6]. As an example, consider two facts (represented in a KB): *the athlete* ***Neymar Jr.*** *plays in sports team* ***Barcelona*** and the same *athlete* ***Neymar Jr.*** *plays on sports league* ***Champions League***. From both facts, we can infer the implicit statement (new fact): *the sports team* ***Barcelona*** *plays in the sports league* ***Champions League***. Such new fact tends to be easily inferred by humans, but not by a machine (without specific inference capabilities). Therefore, a RL can help finding patterns and creating inference rules like: if an athlete **Y** plays for a sports team **X** and athlete **Y** plays in sports league **Z**, then sports team **X** plays in sports league **Z**.

According to [6], there are two problems with most of the existing inference rule learners: they do not scale when based on large corpora and they tend to assume that the training data is largely accurate and complete. However, to be coupled to a never-ending learning system, such as NELL, a RL must overcome both issues. It happens mainly because NELL's KB is continuously growing and continuously being updated and revised by NELL's components. In this paper, we present Graph Rule Learner (GRL), a method designed to allow inference rules induction (or extraction) from ontological knowledge bases (represented as graphs). GRL is based on graph-mining techniques, and explores different link prediction metrics to approach what is called extra-neighbor assessment[7]. Empirical results show that GRL can be integrated with NELL as a new component, thus, it can use NELL's KB to find inference rules and help populating NELL's KB. Also, GRL can be used as a generic inference rule induction component to be coupled to other KBs.

GRL assumes that the training data is mostly (but not completely) accurate and complete. However, it is not a problem if the KB is either imperfect, or incomplete. Actually, link prediction algorithms assume that the missing links are

[1]http://rtw.ml.cmu.edu
[2]http://rtw.ml.cmu.edu/rtw/publications

due to the KB evolution in the near future, thus it is currently incomplete. To take advantage of more accurate knowledge, GRL limits its learning process to a specific part of NELL's KB called the set of *beliefs*, that is composed just by high confidence facts. Regarding scalability, GRL scales with large graphs (the same as large KB's), using a graph disk structure called GraphDB-Tree [8].

The main contributions of this paper are: i) proposing a link-prediction-based method to extract inference rules from a continuously growing ontological KB; ii) empirically showing that link-prediction metrics adapts well to the problem of inference rules extraction even when the input KB is *incomplete* and somehow noisy (having wrong facts stored).

II. Related Works

Rule induction from data is not a novel task and many different approaches have been proposed. Due to space constraints, in this section we focus on more recent approaches and which are closely related to GRL. The *Online Rule Learner* (ORL) [6] mines inference rules from explicit information extracted from large corporas using automated information extraction (IE) systems [9], [10]. ORL is similar to GRL in the sense that it maps input corpora into a graph-based representation. Differently from GRL, however, ORL uses the topology of the created graph to extract rules, instead of link prediction techniques used in GRL.

The Universal Schema proposed in [11] focuses on the benefits of using latent features for increasing coverage of KBs. Key differences between that approach and the one proposed in the work described in our paper include our use of graph-based link prediction measurements as opposed to surface-level patterns in theirs, and also the ability of the proposed GRL method to generate useful (and comprehensible) inference rules which is beyond the capability of the matrix factorization approach.

A traditional approach to extract inference rules is the inductive logic programming (ILP), which deduces rules from ground facts. According to [12], current ILP systems cannot be applied to KBs who gathers data from web with a large scope of categories (anything in the world), such as NELL, mainly because they usually require negative statements as counter-examples, and these projects just hold instances that they consider correct or have some confidence[3]. Also, the ILP-based approach don't scale to the huge amount of data that these kind of KBs store.

Regarding NELL's, when considering its KB as input to induce inference rules, there are other previously proposed approaches. In [13] a Markov Logic approach is used to allow inference over subsets of categories and relations. PRA [5] is the graph-based approaches. PRA (Path Ranking Algorithm) uses a combination of constrained, weighted, random walks through NELL's KB graph to reliably infer new beliefs for it's KB. PRA performs such inference by automatically learning semantic inference rules over the KB.

[3]Ontology properties such as mutual exclusion can be used to solve part of this problem as done in [3]

III. Definitions

Let $G = (V, E)$ be an undirected graph with a set of nodes V and a set of edges E. In addition, n represents the number of nodes and m represents the number of edges in G. $\aleph(u, v)|u, v \in V$ represents the number of common neighbors between u and v, and is defined as the set of nodes in V that are simultaneously adjacent to u and v and $\Gamma(u)$ denote the set of neighbors of u in G. ($\aleph(u, v) = |\Gamma(u) \cup \Gamma(w)|$).

An ontological knowledge base (OKB) can be mapped into a graph, called an ontological graph $G = (V, E, X)$, where each $x \in X$ is a pair composed by a node $v \in V$ and a category c from the set of categories of the OKB, so X has the list of categories for each node. And each edge in E is rotuled with a name (e.g athletePlaysSport).

A closed triangle $\Delta(u, v, w)$ of a graph $G = (V, E)$ is a set of three completely 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 formed by three connected nodes where $\Lambda(u, w) = \{(u, v), (v, w)\} \in E \wedge \{u, w\} \notin E$. In an ontological graph, $\Delta_c(c_1, c_2, c_3)$ 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 don't matter). Any Δ_c is called a **closed triangles category group** and any Λ_c is called a **open triangles category group**.

The extra neighbors[7] value between two categories $\aleph_c(c_1, c_2)$, indicates how related these two categories of nodes are. This value is calculated using:

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

It can be associated as a measure between two categories of nodes ($score_E N(c_1, c_2)$) instead of a measure between two nodes, as in most of Link Prediction metrics.

A commonly used similarity measure to calculate *score functions* in LP tasks is the Jaccard coefficient: $score_{Jac}(u; w) := \frac{|\Gamma(u) \cap \Gamma(w)|}{|\Gamma(u) \cup \Gamma(w)|}$. [14], it measures the probability that both u and w having a feature f (for a randomly selected feature f that either u or w has). Another traditional LP metric is the Salton index[15], commonly known as the cosine similarity. Formally, the Salton index is: $score_{Sal}(u; w) := \frac{|\Gamma(u) \cap \Gamma(w)|}{|\Gamma(u)| \times |\Gamma(w)|}$. Both Salton and Jaccard indexes are used in experiments with GRL (see in Section V).

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

IV. The Graph Rule Learner

The Graph Rule Learner (GRL) is an algorithm designed to extract inference rules from ontological knowledge bases. GRL uses a link-prediction metric called extra-neighbors[7] to rank possible rules, and also to determine the antecedents and consequents of each induced rule.

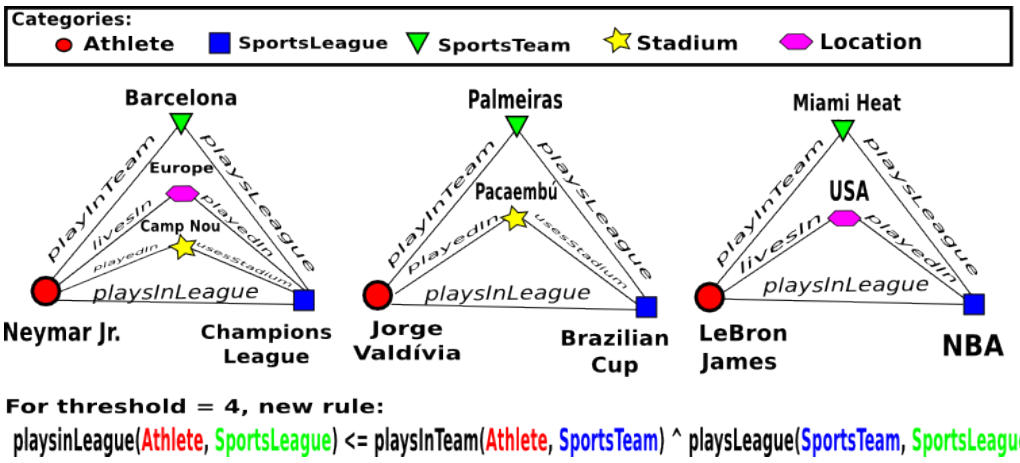

Fig. 1. GRL running example

A. GRL Algorithm

GRL needs an ontological graph as input, and its output is a list of induced inference rules.

Algorithm 1 The GRL

Require: $G = (V, E, X)$
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 $\aleph_c(c_i, c_j)$
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**

In **line 1**, GRL finds and lists all closed triangles $\Delta(u, v, w)$ present in graph G. Then, for each triangle, the number of neighbors $\aleph$ between each pair of nodes is calculated (e.g $\aleph(u, v)$), and grouped in the respective open triangle category group Λ_c[4] (e.g $\Lambda(c_u, c_v)$). The closed triangle is also grouped in the closed triangle category group $\Delta_c(c_u, c_v, c_w)$. In **line 8**, for each open triangle category group $\Lambda_c(c_i, c_j)$, the number of extra neighbors $\aleph_c$ is calculated. $\aleph_c$ is the sum of the $\aleph - 1$ of all instances $\Lambda(i, j)$ in the group. If $\aleph_c(c_i, c_j) = 0$ it indicates that all pair of nodes $\Lambda(i, j)$ in the group have only one neighbor in common. In **line 11**, for each closed triangle category group $\Delta_c(c_u, c_v, c_w)$, the pair of categories with the highest extra neighbors value $\aleph_c$ will be selected (e.g (c_u, c_v)). Then, if the extra neighbor value of this pair is greater or equal than a given threshold ξ, the rule $r_{c_u c_v}(c_u, c_v) \Longleftarrow r_{c_u c_w}(c_u, c_w) \wedge r_{c_w c_v}(c_w, c_v)$ is validated. One literal $r_{c_x c_y}(c_x, c_y)$ indicates a relation(predicate) $r_{c_x c_y} \in E_c$ between the categories c_x and c_y, and its parameters must be instances of categories c_x and c_y respectively.

In **Figure 1**, a simple example of the GRL algorithm for an arbitrary graph is presented.

We have the closed triangle category group Δ_c $(Athlete,$ $SportsTeam,$ $Sports$ $League)$, and its three instances: Δ(Neymar Jr., Barcelona, Champions League), Δ(Jorge Valdivia, Palmeiras, Brazilian Cup), and Δ(Lebron James, Miami Heat, NBA).

The pair of categories of the group that has the greatest extra neighbor value (the consequent of the rule) is *(Athlete, SportsLeague)*: $\aleph_c(Athlete,\ SportsLeague) = 4$, against *(Athlete, SportsTeam)* and *(SportsTeam, SportsLeague)* that have both 0.

To validate the rule, $\aleph_c$ has to be greater or equal than the given threshold that is equal to four. In this example the rule will be created, and it says that if an athlete **X** plays for a sports team **Y** and this sports team **Y** plays in sports league **Z**, then athlete **X** plays in sports league **Z**. In this example, there is also two other possible closed triangle category groups: Δ_c $(Athlete,\ Location,\ SportsLeague)$ and Δ_c $(Athlete,$ $Stadium,\ SportsLeague)$, but no pair (Λ) of this groups achieves an $\aleph_c$ greater than 4.

In a closed triangle category group, more than one triple of relations is possible, for instance: The group Δ_c $(Athlete,$ $SportsTeam,\ SportsLeague)$, we can have the instances: Δ $(NeymarJr.,\ Barcelona,\ Champions\ League)$ and Δ $(NeymarJr.,\ RealMadrid,\ ChampionsLeague)$. The triple of relations for the first one is (playsIn, playsLeague, playsInLeague) and for the second is (playsAgainst, playsLeague, playsInLeague). Having multiple triples of relations inside the same group, indicates that multiple rules can be created, so it is needed to decide how to choose among these multiple triples of relations. For now the GRL just counts the occurrence of each triple inside a closed triangle category group and pick just the one that occurred more frequently,

[4]Despite the three nodes are connected, GRL considers that the edge between the pair of parameters of Λ does not exist in each group

but there are other ways to treat this problem, in the example above the two relation triples could generate correct rules. In the future we plan to explore other possibilities.

When using NELL's KB (as well as any other ontological KB) as input, it is expected to find several repeated rules in terms of predicates, but with different categories as parameters. This is expected mainly because, in such KBs, there are hierarchy among categories and multi-categorized instances. See the example below of GRL's output on NELL's graph:

$$teamPlaysSport(sportsTeam, sport) \Longleftarrow athletePlaysSport(personAsia, sport) \wedge athletePlaysForTeam(personAsia, sportsTeam)$$
$$teamPlaysSport(sportsTeam, sport) \Longleftarrow athletePlaysSport(personUsa, sport) \wedge athletePlaysForTeam(personUsa, sportsTeam)$$

Following along these lines, a **grouping** and **ranking** process is applied on the GRL output. This process consists to simply group all rules sharing repeated predicates in one single generic rule with variables (I, J and K) as parameters, and rank these rules by the number of occurrences on GRL's output list. After this process, we can use this $rank$ to increase confidence in some of the rules. Experiments over this process are present in Section V.

B. GRL Implementation

One common problem when implementing a graph mining algorithm is scalability, mainly when working with graphs having a growing number (from hundreds to millions and sometimes billions) of nodes and edges. To cope with this scalability issue, GRL stores the graph representation in disk using a structure called GraphDB-Tree [8]. GraphDB-Tree is a data structure designed to fast storage and recovery of a graph on secondary memory. The complexity to recover the neighbor list for any node is $O(1)$, so it's very efficient to algorithms that uses just the locality of the nodes (e.g., find graph cliques (such as triangles), calculating some LP metrics: common-neighbors, extra-neighbor, Jaccard, Salton, etc).

V. Experiments

In this section we show some results (inference rules) of a GRL experiment using both NELL's KB, as well as YAGO's KB as input. Also, we present experiments running GRL based on different link-prediction scores in place of extra-neighbors to validate rules[5].

A. Finding Inference Rules with GRL

1) GRL applied to NELL's KB: In this experiment, NELL's KB, also called *rtwgraph*, was used as input for GRL. NELL's KB is automatically extended and populated in a iterative fashion. For this experiment we use the KB from iteration 820, (*rtwgraph* had around 700.000 nodes and 500.000 edges).Using threshold equal to ten ($xi = 10$), the output rule list rl_1 contains 3.780 rules before the grouping process. Examples:

$$R_1.\ teamplayssport(sportsteam, sport) \Longleftarrow athleteplayssport(sport, personUsa) \wedge athleteplaysforteam(personUsa, sportsteam)$$
$$R_2.\ headquarteredin(city, company) \Longleftarrow atlocation(company, buildingfeature) \wedge atlocation(buildingfeature, city)$$

2) Grouping and Ranking Repeated Rules.: As we are working with an ontological graph, lots of rl_1 rules are repeated, with only the parameters of relations being different. Thus, as previously mentioned, one extra grouping step is needed. Grouping this rules into generic ones and ranking them (based on the number of times it is repeated in rl_1) generates a new rule list rl_2, with 870 rules. Two of the top ranked rl_2 rules are presented below, see more in the appendix.

$$R_1.\ athleteplayssport(X, Z) \Longleftarrow teammate(X, Y) \wedge athleteplayssport(Y, Z)$$
$$R_2.\ animalistypeofanimal(X, Z) \Longleftarrow animalistypeofanimal(X, Y) \wedge animalistypeofanimal(Y, Z)$$

3) GRL applied to YAGO KB: As it was mentioned before, YAGO[3] is an OKB such as NELL, that mined data from repositories such as wordnet[6] and wikipedia[7]. In this experiment, YAGO's KB was used as input for GRL, the only problem we had was that YAGO's ontology has a big hierarchy with tens of thousands of categories (while NELL has less than a thousand)[8], so, GRL grouping process was not very effective. We could extract around 350.000 categorized nodes and 550.000 edges from YAGO's(1) KB. Using threshold equal to ten ($xi = 10$), the output rule list rl_1 contains 286 rules before the grouping process. After the grouping and ranking process, the output rule list rl_2 contains 88 rules. Examples:

$$R_1.\ locatedIn(X, Z) \Longleftarrow hasCapital(X, Y) \wedge locatedIn(Y, Z)$$
$$R_1.\ hasPredecessor(X, Z) \Longleftarrow hasPredecessor(X, Y) \wedge hasPredecessor(Y, Z)$$

4) Validating rules: GRL algorithm (and most other link-prediction algorithms) does not present 100% precision. We add the group process to make the output rules more generic, and we use the rank given on this process to help enhance confidence in some rules. In table I the precision curve over this rank is present for NELL and YAGO experiment.

[5]All the experiments were performed using a personal computer with $Intel(R)\ Core^{TM}\ i7 2.49Hz$ with $6GB$ of RAM and on Linux Ubuntu 12.04 (32 bits)

[6]https://wordnet.princeton.edu/

[7]http://en.wikipedia.org/

[8]Yago has very specific categories, such as: "Bob Dylan albuns" and "String quartets by Ludwig van Beethoven"

TABLE I
APPLYING RANK AS THREESHOLD (GRL'S WITH $\xi = 10$)

GRL's output precision by rank				
	NELL		YAGO	
rank $\geq$	t.r	c.r	t.r	c.r
20	13	61.64%	3	100.00%
15	20	60.00%	3	100.00%
10	31	54.84%	7	71.42%
9	38	60.53%	8	62.25%
8	44	61.36%	9	66.66%
7	51	56.86%	9	66.66%
6	74	51.35%	12	58.33%
5	100	47.00%	15	53.33%
4	141	43.97%	17	52.94%

Table I contains statistics captured by selecting rules on the grouped list by the $rank$ given on the group process. Despite the fact that the proposed rank can help to give more confidence to some rules(since for lower ranked rules the precision tends to fall as shown in Table I), it is easy to see that using the rank as a fixed threshold tends to promote the extraction of wrong rules. It is important to recall that in a never-ending learning environment (such as NELL), wrong knowledge (generated by wrong rules) can be propagated and deteriorate the whole KB (because of semantic drift). To manually classify rules is a valid option and in the future to automate the identification of correct and wrong rules it is possible to train a simple classification model using these manually classified rules.

B. Using different link-prediction metrics

In addition to the extra-neighbors (EN) metric, we've also performed experiments using *rtwgraph* as input, and common-neighbors (CN), Jaccard(Jac) and Salton(Sal) metrics during the rule extraction process. In **Table II** we present the precision of each metric selecting grouped rules using $rank$ as threshold.[9]

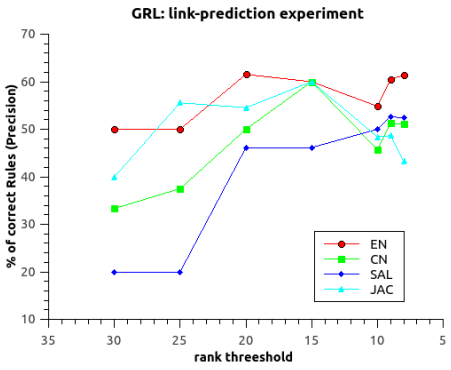

Fig. 2. Precision curve using different LP metrics

Figure 2 depicts the precision curve over the rank threshold for each used metric. The extra-neighbors(EN) is in red, the common-neighbors(CN) in green, the Jaccard(Jac) in light blue and the Salton(Sal) in dark blue. Observing **Table II** and **Figure 2** numbers we can see that the extra-neighbors metric achieved the best overall precision. If we compare by the number of correct rules, this greater precision may not look like a big deal, but when the number of rules grows such difference can be more relevant.

C. Comparing GRL with simmilar state of the art Rule Learners

In this subsection, there's a comparison of GRL with state-of-the-art Rule Learner approach AMIE[12] and also with PRA[5] (because it is a component of NELL[10])

In Table III we present results of GRL, AMIE and PRA running in NELL's OKB iteration 885[11]. To evaluate the precision we randomly pick 100 rules from each of the lists, and manually classify them as correct or incorrect.

To have an idea of how good the rules are to generate new facts to NELL, we apply the correct ones classified among the 100 used to calculate precision to NELL's OKB, then we divide the total facts found by this values to get a normalized measure. GRL rules generate on average more facts than AMIE's[12].

It's possible to see that GRL has only one format and we consider that an advantage, mainly because it produces very readable rules that are also easy to apply to the OKB. AMIE has the same format as GRL and also another one that have only one relation at the body of the rule, this might be interesting too, with rules such as the one above:

$$AMIE.1\, ismultipleof(X,Y) \Longleftarrow animalistypeofanimal(X,Y)$$
$$AMIE.2.\, synonymfor(X,Y) \Longleftarrow synonymfor(Y,X)$$

The first rule above presents the generalization concept, we could say that ismultipleof is a generalization(parent node) of animalistypeofanimal, and the second presents the symmetry concept, it represents the fact that relation synonymfor is symmetric, thus if X is related to Y them Y is related to X.

PRA itself, has the most non-standarized rule format. We found a correct rule created by both PRA and GRL, such rule generates the same fact (has the same head), but with a different body:

$$PRA.\, trophyisthechampionshipgameofsport(X,Z) \Longleftarrow trophywonbyteam(X,Y) \wedge agentcontrols(W,Y) \wedge athleteledteam(W,Y) \wedge teamplaysport(Y,Z)$$
$$GRL.\, trophyisthechampionshipgameofsport(X,Z) \Longleftarrow athletewontrophy(Y,X) \wedge athleteplaysport(Y,Z)$$

[9]The threshold value used in GRL for each metric was ajusted because of the different magnitude of each formula

[10]And these three algorithms find inference rules from ontological knowledge bases

[11]PRA found a larger amount of rules than the other two, because the version we used is coupled with NELL and uses the whole OKB and not just the (much smaller) set of beliefs

[12]PRA was not used in this experiment because it's rules generally have relations in the body that doesn't have instances in beliefs set

TABLE II
EXPERIMENTING DIFFERENT LP METRICS: PRECISION STATISTICS USING RANK AS THREESHOLD

-	Extra-Neighbors		Common-Neighbors		Jaccard		Salton	
rank $\geq$	Rules	Precision	Rules	Precision	Rules	Precision	Rules	Precision
30	6	50.00%	6	33.33%	5	40.00%	5	20.00%
25	8	50.00%	8	37.50%	9	55.56%	5	20.00%
20	13	61.54%	12	50.00%	11	54.54%	13	46.15%
15	20	60.00%	20	60.00%	15	60.00%	13	46.15%
10	31	54.84%	35	45.71%	31	48.39%	30	50.00%
9	38	60.53%	41	51.22%	37	48.65%	38	52.63%
8	44	61.36%	47	51.06%	44	43.18%	42	52.38%

TABLE III
COMPARISON BETWEEN GRL, AMIE AND PRA

-	$RulesFound$	$Precision$	$Gen.Facts$	$Format$
GRL	459	$42/100 = 42\%$	115.59	$r \Longleftarrow p_1 \wedge p_2$
AMIE	982	$35/100 = 35\%$	89.25	$r \Longleftarrow p_1 \wedge p_2 \quad \vee \quad r \Longleftarrow p$
PRA	49966	$33/100 = 33\%$	–	$r \Longleftarrow p_1 \wedge p_2 \ldots \wedge p_n \quad (n \geq 1)$

Looking at these two rules it is possible to see GRL rule is more readable and, also produces the same fact, but needing a simpler "condition" (which means less computational effort on generating facts).

D. Scalability

Experiments evaluating scalability performance of the GRL implementation were not performed. Considering that most of GRL computational effort is related to the task of finding all closed triangles present in the graph, we can base our scalability analysis upon results presented in [8].

VI. CONCLUSION

One of the biggest challenge in current research, focused on automatically building Knowledge Bases from data, is how to populate such Knowledge Bases to allow them to have enough coverage, diminishing sparsity issues. In this sense, even approaches based on never-ending learning principles suffer from KB lack of coverage. Graph Rule Learner was designed and implemented to be used having NELL's KB as input. Its algorithm considers that the input data is sufficiently accurate and complete, even though being in constant evolution.

GRL, differently from most of the other current approaches, explores ontological knowledge to get better results in inference rule extraction precision and scalability. Empirical results show that GRL can cope with NELL's KB characteristics of being a big and never-ending growing KB, thus, not being noise free, neither being complete. Also, GRL can be considered generic enough to be used having other ontological knowledge bases as input and it achieved great results when compared with state of art Rule Learners such as AMIE and PRA running with NELL base.

REFERENCES

[1] 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.

[2] 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.

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

[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] M. Gardner, P. P. Talukdar, B. Kisiel, and T. Mitchell, "Improving learning and inference in a large knowledge-base using latent syntactic cues," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, 2013.

[6] S. Raghavan and R. J. Mooney, "Online inference-rule learning from natural-language extractions," *Statistical Relational Artificial Intelligence*, vol. AAAI 2013 Workshop, pp. 57–63, 2013.

[7] 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*, ser. ICDMW '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 917–924. [Online]. Available: http://dx.doi.org/10.1109/ICDMW.2011.142

[8] L. F. Navarro, A. P. Appel, and E. R. Hruschka Jr., "Graphdb - storing large graphs on secondary memory," *ADBIS Special session on Big Data: New Trends and Applications (BiDaTA) in conjunction with the 17th East-European Conference on Advances in Databases and Information Systems (ADBIS)*, vol. 17, pp. 177–186, 2013.

[9] J. Cowie and W. Lehnert, "Information extraction," *CACM*, vol. 39(1), p. 8091, 1996.

[10] S. Sarawagi, "Information extraction," *Foundations and Trends in Databases*, vol. 1(3), p. 261377, 2008.

[11] S. Riedel, L. Yao, A. McCallum, and B. M. Marlin, "Relation extraction with matrix factorization and universal schemas," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 74–84. [Online]. Available: http://www.aclweb.org/anthology/N13-1008

[12] L. Galarraga, C. Teflioudi, K. Hose, and F. M. Suchanek, "Amie: Association rule mining under incomplete evidence in ontological knowledge bases," *International World Wide Web Conference(WWW')*, 2013.

[13] R. G. S. Dos Santos and E. R. Hruschka Jr., "Markov logic scalability in a never-ending language learning system," in *Proceedings of the NewsKDD Workshop: Data Science for News Publishing. Workshop at the 20th ACM-SIGKDD Conference on Knowledge Discovery and Data Mining - KDD2014*, 2014, pp. 21–25.

[14] P. Jaccard, "Etude comparative de la distribution florale dans une portion des alpes et des jura," *Bulletin de la Socit Vaudoise des Sciences Naturelles*, vol. 37, p. 547579, 1901.

[15] G. Salton and M. M. J., "Introduction to modern information retrieval," *McGraw-Hill, Inc., New York, USA*, p. 35, 1986.