Towards Transfer Learning of Link Specifications

Axel-Cyrille Ngonga Ngomo
Universität Leipzig
AKSW Research Group
http://aksw.org

Jens Lehmann
Universität Leipzig
AKSW Research Group
http://aksw.org

Mofeed Hassan
Universität Leipzig
AKSW Research Group
http://aksw.org

***Abstract*—Over the last years, link discovery frameworks have been employed successfully to create links between knowledge bases. Consequently, repositories of high-quality link specifications have been created and made available on the Web. The basic question underlying this work is the following: Can the specifications in these repositories be reused to ease the detection of link specifications between unlinked knowledge bases? In this paper, we address this question by presenting a formal transfer learning framework that allows detecting existing specifications that can be used as templates for specifying links between previously unlinked knowledge bases. We discuss both the advantages and the limitations of such an approach for determining link specifications. We evaluate our approach on a variety of link specifications from several domains and show that the detection of accurate link specifications for use as templates can be achieved with high reliability.**

I. Introduction

Link Discovery has gained significant momentum over the last years due to the growth of the Linked Data Web and the use of Semantic Web technologies across manifold applications including question answering [16], federated querying [13], large-scale inferences [17] and data integration [2]. Over the last years, several tools and libraries have been developed with the main aim of efficiently supporting the whole of the link discovery process [3], [15], [12]. In general, this process can be modeled as consisting of two steps: Once provided with a source and target set of instances, the first step consists of discovering a link specification for retrieving high-quality links. This step is of crucial importance the precision and recall of the link discovery process depend heavily on the link specification used. Once a specification has been decided upon, it has to be carried out. Several frameworks such as LIMES [7] and SILK [4] have been developed to address the quadratic runtime of link discovery. With the ongoing growth of the Linked Data Web, these tools and libraries have been employed successfully to create links between the different knowledge bases on the Linked Data Web. Consequently, different repositories of high-quality link specifications have been created and made publicly available on the Web. For example, the LATC project[1] generated 170 specifications for linking knowledge bases across several domains including persons, organizations and geo-spatial entities.

The basic observation underlying this paper is the following: While several approaches for learning link specifications have been developed over the past years (e.g., [9], [10]), the discovery of a specification to link two datasets has been regarded as an isolated process. Hence, to the best of our knowledge, none of the previous approaches to detecting link specifications has made use of the already available knowledge available in repositories for link specifications. The primary aim of this paper is consequently to explore how this knowledge can be reused within the framework of transfer learning. More specifically, our goal is to present a formal framework for the transfer learning of link specifications with the aim of improving the computation of link specifications.

The rest of this paper is structured as follows: We first begin by presenting formally the problem of link specification and the idea that underlie transfer learning. Thereafter, we present a formal framework that combines both ideas and allows implementing transfer learning for link specifications. We then present the heuristics we implemented. Finally, we conclude with an overview of possibilities that arise in this novel field of research. Note that throughout this paper, we use RDF prefixes as available at http://prefix.cc. We refer to [6] for full details of the algorithm, evaluation and related work.

II. Preliminaries and Notation

A. Link Discovery

The link discovery problem, which is similar to the record linkage problem, is an ill-defined problem and is consequently difficult to model formally [1]. In this work, we expand the formalization presented in [8]. The goal of link discovery can be described as follows: Given two sets of resources S (source) and T (target) as well as a relation ρ, compute the set M of pairs of instances $(s,t) \in S \times T$ such that $\forall (s,t) \in M : \rho(s,t)$. The sets S resp. T are usually (not necessarily disjoint) subsets of the resources contained in two (not necessarily disjoint) knowledge bases $\mathcal{K}_S$ resp. $\mathcal{K}_T$. In most frameworks, the computation of whether $\rho(s,t)$ holds for two elements is carried out projecting the elements of S and T based on their properties in a similarity space $\mathfrak{S}$ and setting $\rho(s,t)$ iff some similarity condition $\sigma(s,t) \geq \tau$ is satisfied, where $\sigma : S \times T \to [0,1]$ is a similarity function and $\tau \in [0,1]$. The specification of the sets S and T and of this similarity condition is usually carried out within a *link specification*. In general, a link specification consists of three parts:

1) two sets of restrictions $\mathcal{R}_1^S$ ... $\mathcal{R}_m^S$ resp. $\mathcal{R}_1^T$ ... $\mathcal{R}_k^T$ that specify the sets S resp. T,

[1] http://latc-project.eu

2) a specification of a complex similarity metric σ via the combination of several atomic similarity measures σ_1, ..., σ_n and
3) a set of thresholds τ_1, ..., τ_n such that τ_i is the threshold for σ_i.

A restriction $\mathcal{R}$ is generally a logical predicate. Typically, restrictions in link specifications state the `rdf:type` of the elements of the set they describe, i.e., $\mathcal{R}(x) \leftrightarrow x$ `rdf:type someClass` or the features the elements of the set must have, e.g., $\mathcal{R}(x) \leftrightarrow (x$ `someProperty someValue`$)$. Each $s \in S$ must abide by each of the restrictions $\mathcal{R}_1^S$ ... $\mathcal{R}_m^S$, while each $t \in T$ must abide by each of the restrictions $\mathcal{R}_1^T$ ... $\mathcal{R}_k^T$. Note that the atomic similarity functions $\sigma_1, ..., \sigma_n$ can be combined to σ by different means. Also note that this formal model allows regarding link specifications as binary classifiers that assign the class $+1$ to pairs such that $\rho(s,t)$ and -1 to pairs that should not. This view of link specifications and the existence of repositories for link specifications automatically raises the question whether transfer learning can be applied to alleviate the costs necessary to create them. Addressing this issue is the core idea behind transfer learning.

B. Transfer Learning

Our formalization of machine learning in general and transfer learning in particular is based on [11]. The goal of most machine learning approaches is to find a predictive function $f : \mathcal{X} \rightarrow \mathcal{Y}$ which can compute the right classification $f(x_i) = y_i$ when given the input data x_i. We call the set $\mathcal{X}$ the *domain* in which we are to learn the right way to classify, while we dub the pair (f, Y) the *task* t of the machine learning approach at hand and write $t = (f, \mathcal{Y})$. In the case of link discovery, $\mathcal{X} = S \times T$ while $\mathcal{Y} = \{+1, -1\}$ with $f(x_i) = +1$ if $\rho(s,t)$ and $f(x_i) = -1$ in all other cases. Finding the function f for link discovery tasks is generally very costly, as it requires either (mostly manually) labeled training data [9] or a significant amount of computation [10]. The idea behind *transfer learning* (also coined *knowledge transfer*) [11] can be broadly described as follows: Given other machine learning tasks t' with known or unknown classification functions f' that are somehow "related" to f, use the functions f' or the domain knowledge available for determining f' (i.e., transfer the knowledge from the tasks t') to improve the process of finding $(f, \mathcal{Y})$.

In general, three categories of transfer learning from a task $t' = (f', \mathcal{Y}')$ to a task $t = (f, \mathcal{Y})$ can be differentiated: inductive, transductive and unsupervised transfer learning. *Inductive learning* assumes that training data for learning f is available and aims either to reuse labeled data available for learning f' (in which case inductive learning reduces to multi-task learning) or to use the data on which f' is to be learned (self-taught learning). *Transductive learning* assumes that labeled data is only available for the task t' and aims at reusing that data for learning f. If $\mathcal{X} = \mathcal{X}'$, then transductive learning reduces to adapting $(f', \mathcal{Y}')$ to a new task. Else, if the domain and task are the same, we are confronted with a machine learning task which requires dealing with sample selection biases and covariance shift. The last category of transfer learning, *unsupervised transfer learning*, is concerned with learning when labeled data is available for learning neither $(f, \mathcal{Y})$ nor $(f', \mathcal{Y}')$. In this paper, we address the following transductive transfer learning task: Given a set of functions f_i' computed for domains X_i', improve the determination of the function f for solving task $(f, \mathcal{Y})$ on the domain $\mathcal{X}$ with $\forall i : \mathcal{X} \neq \mathcal{X}_i'$.

III. Formal Framework for Transfer Learning of Link Specifications

The idea behind our approach to the transfer learning of link specifications is sketched in Figure 1. Instead of regarding the learning of link specifications as an isolated task (see left side of the figure), we aim to reuse existing specifications (see right side of the figure). Based on a repository of existing link specifications, the aim of our approach is thus to extract knowledge out of the existing (and often validated) specifications and transfer it to a link specification, which can be used as a template (i.e., an initial solution f) for the linking task at hand. Let us assume that the current domain $\mathcal{X} = S \times T$ is fully known, i.e., that the set of restrictions that describe both S and T has already been computed[2]. In link discovery, the elements of the domain $\mathcal{X}$ are usually a set of pairs of points $(s,t) \in S \times T$ from sets of instances S and T described in a similarity space $\mathfrak{S}$. This similarity space is defined over a subset of $P_L \times P_L'$ of source properties P_L and target properties P_L'. Consequently, a link specification can be described as a binary classifier that maps elements of $(P_L \times P_L')^n$ to the classes +1 and -1.

One of the challenges of transferring knowledge from a link specification to another is that link specifications can be very complex in general since it combines restrictions, arbitrarily nested complex similarity metrics and thresholds. Still, given the formalization of link specifications presented in Section II-A, we can tackle the transfer learning of link specifications by reducing it to a set of three simpler problems: The measurement of restriction similarity, property similarity as well as determining the accuracy of link specifications. The *restriction similarity* ensures that the domains $\mathcal{X}'$ of known link specifications and the domain $\mathcal{X}$ of the link specifications to devise are as related as possible. Here the basic idea is that the more related $\mathcal{X}$ and $\mathcal{X}'$ are, the higher the probability that the function f' is a good initial value for f. The *property similarity* ensures that the similarity space $\mathfrak{S}'$ used in $\mathcal{X}'$ can be mapped to a similarity space $\mathfrak{S}$ that describes the elements of $\mathcal{X}$. Therewith, we ensure that the transfer of $(f', \mathcal{Y}')$ to $(f, \mathcal{Y})$ is possible by mapping each of the dimensions of $\mathfrak{S}'$ to $\mathfrak{S}$. Finally, we assume that not all tasks were solved with the same accuracy. Thus we also measure the *accuracy* of each task $(f', \mathcal{Y}')$ so as to learn from the better ones. In the following, we describe formally how those three functions can

[2]Computing such restrictions is the object of ontology matching, which is out of the scope of this paper. A good overview of approaches for achieving such computations can be found in [14].

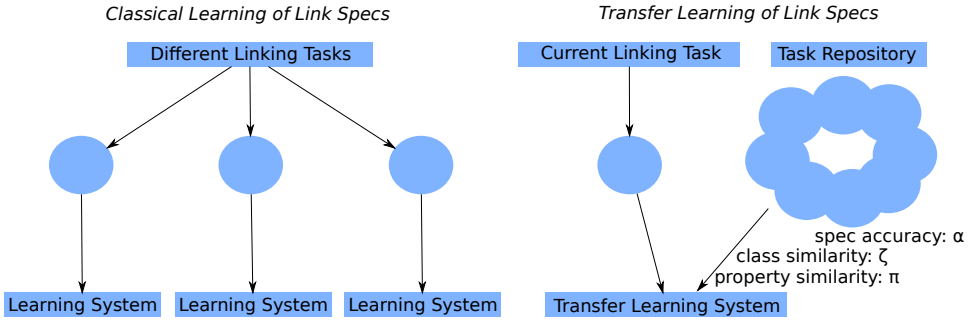

Figure 1. Overview of transfer learning of link specifications.

be used for transfer learning, whereas in the next section, we describe implementations of those functions.

Transfer learning on link specifications assumes that 2 sets of restrictions are given and aims to find the right classifier based on known classifiers for other source and target classes mapped by known classifiers. As pointed out in Section II-A, these restrictions are mostly class restrictions of the form s `rdf:type someClass`. Thus, in the following, we will reduce the similarity of restrictions to the similarity of sets of classes. Note that this reduction does not diminish the applicability of our transfer learning framework as even triples of the form s `someProperty someValue` can be considered to be class expressions. In the following, we denote the set of all classes as C, the set of all properties as P and the set of all link specifications as Q. Lowercase variants of these letters are used to indicated elements of the respective sets.

The input of a transfer learning problem are two sets of classes $\mathcal{C} \subseteq C$ and $\mathcal{C}' \subseteq C$, which should be mapped. Furthermore, we assume a set $\{q_1, \ldots, q_m\}$ of existing link specifications from which we want to transfer knowledge. In addition, we assume the existence of two sets P_L and P'_L denoting the sets of properties relevant for the current linking task. Those sets can be computed automatically by determining the properties associated with instances of $\mathcal{C}$ and $\mathcal{C}'$ or provided manually.

Using these preliminaries, we can define the main functions for implementing transfer learning: $\zeta : 2^C \times 2^C \mapsto [0,1]$ compares the similarity of two input class sets, possibly using background knowledge about those classes available in the underlying triple stores, and returns a value between 0 and 1, where 0 indicates no similarity and 1 indicates identity. Similarly, we assume the existence of a function $\pi : P \times P \mapsto [0,1]$, which works analogously for properties. The set all such property similarity functions is denoted as Π. For each specification in the background knowledge, an assessment function $\alpha : Q \mapsto [0,1]$ can be used to judge its quality. The aim of this function is to prioritize high quality link specifications. Specifying α is optional, i.e. it can be omitted by dropping it from Equation 1.

In order to implement transfer learning, the basic framework implements a set of methods: $r : Q \times P \times \Pi \mapsto T$ replaces each source property in a link specification with the most similar property in P according to the given similarity function π. Analogously, $r' : Q \times P \times \Pi \mapsto T$ performs the same replacement for target properties. Moreover, we assume the existence of two helper functions $\psi : Q \mapsto 2^C$ and $\psi' : Q \mapsto 2^C$, which return the source and the target classes in a link specification, respectively. Note that the formal specification of transfer learning can be used in all linking tools if the above four functions are implemented.

Combining all of the above notions, we can formally define the equation which allows computing a score ω for how well a link specification task t' (with link specification q') is suited to be for transfer learning for the task $t = (f, \mathcal{Y})$ at hand:

$$\omega(t, t') = \alpha(q')\zeta(\psi(q'), \mathcal{C}) \cdot \zeta(\psi'(q'), \mathcal{C}') \cdot r'(r(q', P_L, \pi), P'_L, \pi) \quad (1)$$

Intuitively, the equation computes the product of the different similarity functions defined above and this depends on the quality of existing specifications and their similarity to the specification to devise. Furthermore, it maps properties and classes in existing specifications to those relevant for the current linking task. This weight function can be used in manifold ways. For example, it can be used to compute a weighted sum over all available link specifications. It can also be used to sort and rank the available link specifications and present them to the user in charge of devising the link specification for the task at hand. Manifold approaches can be used to implement the appropriate similarity functions σ, π and α for computing the similarity of properties and classes as well as the accuracy of link specifications. In the following sections, we present such functions.

IV. Heuristics

A. Accuracy of Specifications

The most common method to assess precision and recall of link specifications is to manually label created links as correct or incorrect. Several link discovery tools such as LIMES [9] as well as the LATC[3] and SAIM[4] platforms integrate user feedback directly and let users evaluate links. Furthermore, game-based approaches such as VeriLinks [5] have been devised to evaluate links. In the LATC repository in particular (which we use as a base for our evaluation) 66.47% of the link specifications are associated with reference positive and negative links. To estimate the quality of a link specification, we can calculate the precision of the evaluated links as the

[3]http://latc-project.eu
[4]http://saim.sf.net

number of correct links c divided by the total number n of evaluated links. The assumption that the precision of the verified links can be used as approximation of the precision of all links generated by the link specification would reflect current practice. However, such an approximation would not take into account that a higher number of evaluated links usually leads to a higher confidence estimate. For this reason, we define the specification accuracy as the center of the 95% confidence interval of the proportion of correct links.

One weakness of our approach to compute specification accuracy is that we cannot take any knowledge into account about how the reference links were evaluated. The above approach assumes that they were randomly drawn from the set of generated links. However, in some cases algorithms may have just presented the user problematic links with unclear classification which is common in active learning approaches [8]. In that case, our specification accuracy estimate tends to be too pessimistic.

B. Similarity of Classes

In addition to measuring the accuracy of link specifications, it is central measure how similar the domain $\mathcal{X} = S \times T$ at hand is to domains $\mathcal{X}' = S' \times T'$ for which classifiers are known. As the domains are defined by a set of restrictions, measuring the similarity between two domains can be carried by measuring the similarity between the restrictions that define them. Most commonly, each source restriction $\mathcal{R}_i^S$ is a s `rdf:type` c_i, where c_i is the class type of s.. The call the set $\mathcal{R}(S) = \bigcup_i \{c_i\}$ the *restriction set* for a source S. We define $\mathcal{R}(T)$ analogously. Given this model, we implemented two different approaches to computing the similarity of classes (which we will denote $\zeta(S, S')$ for the sake of brevity): a *label-based* approach and a *data-centric* approach. The idea behind the label-based similarity is that two sets of instances are similar if the labels in the elements of their restriction sets are similar. We thus defined the first similarity $\zeta_l(S, S')$ as

$$\sum_{c_i \in \mathcal{R}(S)} \max_{c'_j \in \mathcal{R}(S')} sim(label(c_i), label(c'_j)). \quad (2)$$

Here, the function $label(c)$ returns the label of a resource while sim is a string similarity function.

Given that the label of resources is stored in endpoints that are not always available, we extended the similarity measure ζ_l by considering the local name $name$ of each resource as its label. We thus extended the similarity function ζ_l to ζ_u as follows:

$$\zeta_u(S, S') = \sum_{c_i \in \mathcal{R}(S)} \max_{c'_j \in \mathcal{R}(S')} sim(name(c_i), name(c'_j)). \quad (3)$$

The similarity of properties $\pi(p_i, p'_j)$ can be defined analogously. We implemented several property similarity functions π_l, π_u and π_d. Note that our approach can easily be extended to any type of restriction. The evaluation of the similarities can be found in[6].

V. Conclusions and Future Work

In this paper, we presented a formal model for the transfer learning on link specifications. (Semi-)Automatic ways for determining the right thresholds can be derived from previous algorithms. Thus, learning link specifications can be combined with our approach. For example, specification templates can be used to seed genetic programming algorithms [10] such as to accelerate their convergence. In addition, knowing which template to use can help when choosing the right deterministic model (Boolean classifier, linear classifier) as well as its initialization for these models [8]. The main question behind our future work will thus be twofold: First, we will aim to combine existing approaches to learning link specications to transfer learning and measure how much labeling effort can be saved when applying transfer learning to the detection of link specifications. Moreover, we will aim to combine our transfer learning approach with more sophisticated class and property similarity approaches to see how they affect our mean reciprocal rank score.

References

[1] A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *SIGMOD Conference*, pages 783–794, 2010.

[2] D. Ben-David, T. Domany, and A. Tarem. Enterprise data classification using semantic web technologies. In *ISWC*, 2010.

[3] A. Hogan, A. Polleres, J. Umbrich, and A. Zimmermann. Some entities are more equal than others: statistical methods to consolidate linked data. In *NeFoRS*, 2010.

[4] R. Isele, A. Jentzsch, and C. Bizer. Efficient Multidimensional Blocking for Link Discovery without losing Recall. In *WebDB*, 2011.

[5] J. Lehmann, Q. Nguyen, and T. Ermilov. Can we create better links by playing games? In *7th IEEE International Conference on Semantic Computing, September 16-18, 2013, Irvine, California, USA*, 2013.

[6] A.-C. N. Ngomo, J. Lehmann, and M. Hassan. Transfer learning of link specifications. Technical report, University of Leipzig, 2013. http://svn.aksw.org/papers/2013/JIIS_Transfer_Learning/public.pdf.

[7] A.-C. Ngonga Ngomo. A time-efficient hybrid approach to link discovery. In *Proceedings of OM@ISWC*, 2011.

[8] A.-C. Ngonga Ngomo, J. Lehmann, S. Auer, and K. Höffner. Raven: Towards zero-configuration link discovery. In *Proceedings of OM@ISWC*, 2011.

[9] A.-C. Ngonga Ngomo and K. Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In *Proceedings of ESWC*, 2012.

[10] A. Nikolov, M. D'Aquin, and E. Motta. Unsupervised learning of data linking configuration. In *Proceedings of ESWC*, 2012.

[11] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

[12] G. Papadakis, E. Ioannou, C. Niederee, T. Palpanasz, and W. Nejdl. Eliminating the redundancy in blocking-based entity resolution methods. In *JCDL*, 2011.

[13] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. Fedx: optimization techniques for federated query processing on linked data. In *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ISWC'11, pages 601–616, Berlin, Heidelberg, 2011. Springer-Verlag.

[14] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, PP(99), 2011.

[15] J. Sleeman and T. Finin. Computing foaf co-reference relations with rules and machine learning. In *Proceedings of SDoW*, 2010.

[16] C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano. Sparql template-based question answering. In *Proceedings of WWW*, 2012.

[17] J. Urbani, S. Kotoulas, J. Maassen, F. van Harmelen, and H. Bal. Owl reasoning with webpie: calculating the closure of 100 billion triples. In *Proceedings of the ESWC 2010*, 2010.