Structuring Linked Data Search Results Using Probabilistic Soft Logic

Duhai Alshukaili(✉), Alvaro A.A. Fernandes, and Norman W. Paton

School of Computer Science, University of Manchester,
Oxford Road, Manchester M13 9PL, UK
{dhahi.alshekaili,a.fernandes,norman.paton}@manchester.ac.uk

Abstract. On-the-fly generation of integrated representations of Linked Data (LD) search results is challenging because it requires successfully automating a number of complex subtasks, such as structure inference and matching of both instances and concepts, each of which gives rise to uncertain outcomes. Such uncertainty is unavoidable given the semantically heterogeneous nature of web sources, including LD ones. This paper approaches the problem of structuring LD search results as an evidence-based one. In particular, the paper shows how one formalism (viz., probabilistic soft logic (PSL)) can be exploited to assimilate different sources of evidence in a principled way and to beneficial effect for users. The paper considers syntactic evidence derived from matching algorithms, semantic evidence derived from LD vocabularies, and user evidence, in the form of feedback. The main contributions are: sets of PSL rules that model the uniform assimilation of diverse kinds of evidence, an empirical evaluation of how the resulting PSL programs perform in terms of their ability to infer structure for integrating LD search results, and, finally, a concrete example of how populating such inferred structures for presentation to the end user is beneficial, besides enabling the collection of feedback whose assimilation further improves search result presentation.

Keywords: Linked data search · Linked data integration

1 Introduction

The idea of linked data (LD) underpins the attempt to transfer the strengths of the web of documents to data: data can be shared, searched for and browsed, building on standards for data identification, description and linking that provide low barriers to entry, facilitating new applications in areas such as data science and open government. Along with a basic model that publishing, it can be anticipated that popular types of tool can transfer successfully from the web of documents to the web of data, supporting activities such as searching and browsing. However, data and documents have important differences, and direct translations of techniques that have been successful in the web of documents can seem rather less intuitive in the web of data.

P. Groth et al. (Eds.): ISWC 2016, Part I, LNCS 9981, pp. 3–19, 2016.
DOI: 10.1007/978-3-319-46523-4_1

For example, keyword search has been an important technology in the web of documents: searches give rise to ranked lists of documents, through which the user can browse. In such approaches, the user accesses the published document directly. Such an approach has been transferred to the web of data, as represented by LD search engines such as Falcons [3], Sindice [11] and Swoogle [5]. Although LD search engines are useful, their results, which take the form of collections of RDF resources, present to users the data *as published*. Thus, LD search engines tackle the question *What resources are out there that match the search?*, and not so much *What data is out there that matches the search?*. For example, searching for `"Godfather actors"` returns results (see Fig. 1), among others, that are about two distinct *films* in whose name the string `Godfather` occurs, as well as about *actors* that have appeared in those films. Assuming for the moment that the user is looking for data about actors in the US film named `The Godfather` (en.wikipedia.org/wiki/The_Godfather), released in 1972, filtering and structuring the results in different tables, as shown in Fig. 2, would be desirable since it distinguishes between films and actors and provides structure to the presentation of films and of actors that have appeared in them.

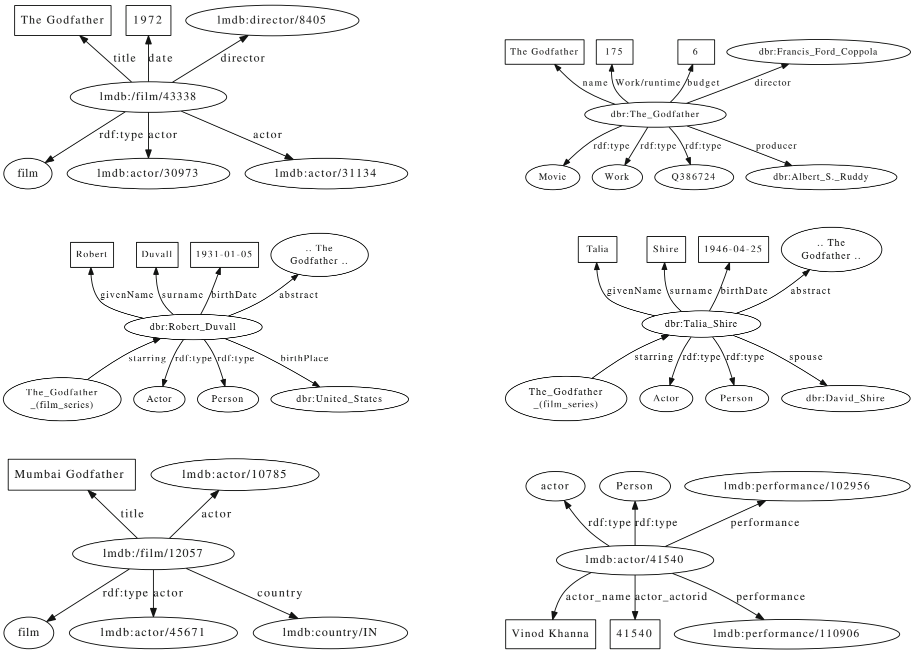


Fig. 1. Example LD search results for the term `"Godfather actors"`

As such, there is an opportunity to complement the work to date on search engines for LD by devising techniques to infer a structure from the returned resources. This would insulate the user from many of the heterogeneities that LD resources typically exhibit due to weak constraints on publication.

Movie				
name	**date**	**director**	**budget**	**runtime**
The Godfather	1972	Francis Ford Coppola	6	172

Actor				
givenName	**surname**	**birthDate**	**birthPlace**	**spouse**
Robert	Duvall	1931-01-05	United States	null
Talia	Shire	1946-04-25	null	David Shire

Fig. 2. Results in Fig. 1 integrated and reported as tables

The integration of search results by means of structure inference is by no means a straightforward task because there is no a *priori* target model to which the results are to be mapped, the available evidence about the relationships between different resources may be partial or misleading, and the integration must be carried out automatically, or at most with small amounts of feedback from the user. Our hypothesis is that LD search result integration can benefit from (i) combining different sources of evidence to inform the integration process, (ii) systematically managing the uncertainty associated with these sources, and (iii) making cost-effective use of feedback. To test this hypothesis, this paper investigates the use of probabilistic soft logic (PSL) [1] for combining three sources of evidence, viz., syntactic matching, domain ontologies and user feedback.

The contributions of this paper are:

1. The characterization of the LD search result integration task as one in which different sources of (partial and uncertain) evidence are brought together to inform decision making.
2. The instantiation of this characterization of the problem using PSL to combine, in a principled and uniform way, evidence from syntactic matching, from domain ontologies and from user feedback.
3. The empirical evaluation of this approach, in which it is shown how the principled, uniform use of different types of evidence improves integration quality for the end user.

2 Related Work

LD search engines such as Falcons [3] and Sindice [11] return a list of resources that match the given search terms ranked by their estimated relevance but without attempting identify their underlying structure. Our contributions build on such search engines as we infer a tabular structure over the returned results. Sig.ma [15] also builds on returned search results. Differently from us, it assumes that the search terms describe one entity (hence a singleton, never a set). Similarly to us, Sig.ma aims to build a profile (essentially a property graph, not a tabular structure as we do) that characterises the searched-for entity on the basis of the results returned by Sindice. Sig.ma uses heuristics whose only input is syntactic evidence, and accumulates information about the entity from different LD resources without using a principled evidence assimilation technique.

In contrast, we use a probabilistic framework to uniformly assimilate different kinds of evidence in a principled manner. Sig.ma does respond to feedback: the user can decide whether a resource is allowed to contribute data to the generated profile or not. However, such feedback does not affect future searches, whereas, in our approach, feedback given is assimilated as evidence and therefore influences positively the quality of future interactions.

Inferring a structure from instance-level RDF data, but not search results *per se*, has received much attention by the research community. However, this activity has focussed on finding ontology axioms such as range restriction [4,16], domain restriction [4,16], subsumption [14,16,18], and equivalence axioms [6,14,16] that apply to the data. In contrast, our approach uses PSL to populate data tables with returned search results, which are more heterogeneous, structurally and semantically, than the data the approaches cited normally target. For example, searching for a film title (e.g. The Godfather) with Falcons returns results from at least three LD resources: DBPedia, LinkedMDB and Yago. This makes the structure inference task harder, as conflicts between the vocabularies used by different datasets need to be resolved.

Given input ontologies, inductive logic programming, a statistical relational learning paradigm, is used in [6,14] to extend the ontologies with new axioms given data defined in terms of the given ontologies. In contrast, no background knowledge is needed in some other approaches. Hierarchical clustering was used in [4] to find schematic patterns, whereas association rule mining [16] and Bayesian networks [18] have also been used to infer descriptions of concepts. Our contributions also use background knowledge in the form of a PSL program resulting from a supervised learning stage in which logical rules are assigned weights. However, we view ontologies as semantic evidence, since we aim to populate a tabular structure of search results rather than extend a given ontology.

Recently, there has been a growing interest in applying statistical relational learning (SRL) techniques such Markov logic networks (MLNs) and PSL to problems where relational dependencies play a crucial role (e.g., inference over social network data) or where principled integration requires assimilation of multiple sources of evidence. In the Semantic Web context, Niepert *et al.* [10] exploited MLNs to match ontologies by modelling axioms as hard constraints that must hold for every alignment. They use similarities of concepts in the input ontologies as evidence, taking them as a seed alignment, and apply integer linear programming to maximise the probability of alignment based on the seed alignments as constrained by the ontological axioms. Their work showed that syntactic and semantic evidence can be combined using SRL techniques. Differently from them, we do not represent ontological knowledge as hard constraints since we target a tabular structure. In our approach, ontologies are used as evidence for inferring an instance of out target structure in the returned search results. This means that we treat ontological knowledge as uncertain. Similarly to us, Pujara et al. [13] used PSL to infer knowledge graphs about real-world entities from noisy extractors, and, in particular, the assimilation of lexical similarities and ontological evidence proves to be crucial in de-noising extracted graphs.

3 A Brief Introduction to PSL

In this paper, the formalism used to infer an integrated structure over returned LD search results is PSL [1], a general-purpose learning and inference framework for reasoning with uncertainty in relational domains.

In Sect. 4, we contribute a set of PSL rules and a target metamodel. The rules express probabilistic relationships between triples in search results. After the rules have been converted into a PSL program, the PSL implementation processes the returned results so as to instantiate the target metamodel, thereby yielding an integrated, structured view over the results.

A PSL program is a set of weighted first-order-logic formulae of the form $w : A \leftarrow B$, where w are non-negative weights, B is a conjunction of literals and A is a single literal. Consider the following example PSL program (adapted from [8], where rules are written as $w : B \rightarrow A$):

$$0.3 : votesFor(B,P) \mathrel{\tilde{\leftarrow}} friend(B,A) \mathbin{\tilde{\wedge}} votesFor(A,P) \tag{1}$$

$$0.8 : votesFor(B,P) \mathrel{\tilde{\leftarrow}} spouse(B,A) \mathbin{\tilde{\wedge}} votesFor(A,P) \tag{2}$$

The semantics, including the tilde-capped connectives, are briefly explained below but, intuitively, this PSL program states that, given any individuals a, b and p, instantiating (resp.) the logical variables A, B and P, a claim is made that if b is either a friend or a spouse of a and a votes for party p, then, with some likelihood, b votes for p. The respective weights assert that the influence of spouses on what party b votes for is larger than that of friends.

Softness in PSL arises from the fact that truth values are drawn from the continuous interval $[0,1]$, i.e., if $\mathcal{A}$ is the set $\{a_1,\ldots,a_n\}$ of atoms, then an interpretation is a mapping $I : \mathcal{A} \rightarrow [0,1]^n$, rather than only to the extreme values, i.e., either 0 (denoting falsehood) or 1 (denoting truth).

To capture the notion that different claims (expressed as rules) may have different likelihoods, a probability distribution is defined over interpretations, as a result of which rules that have more supporting instantiations are more likely. In the case of Rules (1) and (2) above, interpretations where the vote of an individual agrees with the vote of many friends, i.e., satisfies many instantiations of Rule (1) are preferred over those that do not. Moreover, where a tradeoff arises between using agreement with friends or with spouses, the latter is preferred due to the higher weight of Rule (2).

Determining the degree to which a ground rule is satisfied from its constituent atoms requires relaxing (with respect to the classical definitions) the semantics of the logical connectives for the case where terms take soft truth-values. To formalize this, PSL uses the Łukasiewicz t-norm and its corresponding co-norm, which are exact (i.e., coincide with the classical case) for the extremes and provide a consistent mapping for all other values. The relaxed connectives are notated with a capping tilde, i.e., $\tilde{\wedge}$, $\tilde{\vee}$, and $\tilde{\neg}$, with $A \mathrel{\tilde{\leftarrow}} B \equiv \tilde{\neg} B \mathbin{\tilde{\vee}} A$.

Atoms in a PSL rule can be user-defined. Thus, a unary predicate `IsDictWord` might be defined to have truth value 1 if the individual of which it is predicated is a dictionary word and 0 otherwise. Atoms in a PSL rule can also capture user-defined relationships between sets of individuals. Thus, the truth value of `S.interests` `SimSetEq`[] `T.interests` is the similarity of the respective sets of interests of the individuals `S` and `T` as computed by the user-provided definition of set similarity `SimSetEq`[].

In summary, the basic idea is to view logical formulas as soft constraints on their interpretations. If an interpretation does not satisfy a formula, it is taken as less likely, but not necessarily impossible. Furthermore, the more formulas an interpretation satisfies, the more likely it is. In PSL, this quantification is grounded on the relative weight assigned to each formula. The higher the weight, the greater the difference between the likelihood of an interpretation that satisfies a formula and the likelihood of one that does not.

The key tasks supported by PSL implementations are learning and inference. The rules in a PSL program can be either given (i.e., asserted) or learned from sample (or training) data. In this paper, rules (shown later) are given. Furthermore, the weight of each rule can also be given or learned from sample data. In this paper, rule weights are learned from sample data (as detailed later). The PSL weight learning process takes a PSL program (possibly with initial weights), a specification of both evidence and query predicates, and sample data. A predicate is said to be an *evidence predicate* if all its ground atoms have known truth values by observation. A predicate is said to be a *query predicate* if one or more of its ground atoms have unknown truth values. The process returns a relative non-negative weight for each rule. A positive weight denotes that a rule is supported by the sample data whereas the magnitude indicates the strength of that support. A weight of zero denotes lack of support in the sample data for that rule (but since weights are relative it does not entail impossibility).

The main purpose of a PSL program is inference. The PSL inference process takes a PSL program, evidence as data, and a query. It then computes the most probable assignment of soft truth-values to the query, i.e., the probability that the given query atom is true. A major strength of PSL is that implementations perform inference by constructing a corresponding convex optimization problem for which a solution can be efficiently computed even for large-size inputs. For detailed descriptions of the PSL learning and inference algorithms, see [1].

4 Structure Inference Over LD Search Results

Our approach can be briefly summarized as follows. Firstly, we have defined a metamodel (see Fig. 3) that characterizes the type of structure to be inferred (i.e., populated with resources returned by LD searches). Every instances of the metamodel is a tabular representation of search results that we refer to as a *report*. Thus, given the search returns, our goal is to infer that some resources are entity types and some are entities (i.e., elements in the extent of an inferred entity type), some other resources are properties of some inferred entity type and some are property values (i.e., elements in the domain of an inferred property).

Next, we have expressed the semantics of the metamodel as a set of unweighted PSL rules, which we refer to as *the baseline model*, denoted by $\mathbb{B}$

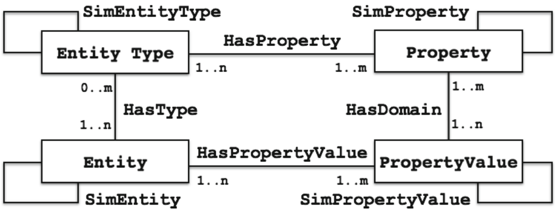


Fig. 3. The metamodel for population with LD search results.

(see Rules R1-R17 in Fig. 4). The rules in $\mathbb{B}$ build on syntactic evidence alone. In order to assimilate semantic evidence, we have added rules to $\mathbb{B}$ that build on additional ontological evidence. We refer to this as *the semantic model*, denoted by $\mathbb{S}$, where $\mathbb{S} \supset \mathbb{B}$ (see examples in Rules R18-R23 in Fig. 4). Next, using sample data from search results and from ontologies, we use a PSL implementation (github.com/linqs/psl) to learn weights for the rules in $\mathbb{S}$ and obtain the corresponding PSL program $PSL(\mathbb{S})$. This also yields $PSL(\mathbb{B})$, i.e., the subset of $PSL(\mathbb{S})$ where we only retain those rules that occur in $\mathbb{B}$. Then, given the returned results of any LD search, we can use PSL inference from $PSL(\mathbb{S})$ to instantiate the metamodel with the most probable characterization of the returned resources. In order to assimilate evidence from user feedback, we have defined a separate set of rules, which we refer to as *the feedback model*, denoted by $\mathbb{F}$ (see examples in Rules R24-R27 in Fig. 4). To obtain the weights for the rules in $\mathbb{F}$ and thus obtain the PSL program $PSL(\mathbb{F})$, we generated synthetic feedback and used the same PSL implementation. The PSL program $PSL(\mathbb{S}) \cup PSL(\mathbb{F})$ integrates LD search results given syntactic, semantic and feedback evidence.

The Baseline Model: Assimilating Syntactic Evidence. Our approach is grounded on the hypothesis that, with some likelihood, there exist relationships between RDF triples returned by search and instantiations of our metamodel that can be captured by a PSL program. Correspondingly, the first step in the construction of the baseline model $\mathbb{B}$ is to map, as a purely syntactic operation, RDF triples onto predicates that assert membership of metamodel constructs. For example, on the evidence of an RDF triple such as `U rdf:type U'`, where `U` and `U'` are URIs, we may conclude that `RDFIsInstOf(U,U')`, i.e., that there is an instance-of relationship between an individual `U` and a type `U'`.

Such mappings are inherently uncertain (i.e., only hold with some likelihood) because it is impossible to capture a publisher's intentions. So, e.g., if `RDFType(Q386724)` is not a user-level type then perhaps it should not be considered to denote an entity type, and therefore not suitable to be reported as a table to the end user. Another source of uncertainty is the search itself (i.e., its associated precision and recall). For example, the RDF individual labeled `"Vinod Khana"` may be returned but should not be listed in a table describing the actors of the 1972 US-produced Godfather film. Rule uncertainty is reflected in rule weights that are learned from sample data as described in Sect. 5.

The baseline model $\mathbb{B}$ consists of Rules R1-R17 in Fig. 4. There are two subsets in $\mathbb{B}$: Rules R1-R9 infer membership for all the constructs in the metamodel in Fig. 3 except for similarity relationships, which are inferred through Rules R10-R17, comprising the second subset. The bodies of Rules R1-R9 show how RDF triples, in raw (i.e. `Triple` predicate in Fig. 4) or in mapped form (e.g. `RDFIsInstOf` in Fig. 4), provide evidence for metamodel membership predicates. For example, in Rule R1, a returned RDF triple of the form `S rdf:type T` counts as evidence that `T` is an entity type. As expected, rules heads (i.e., inferred predicates) may appear in rule bodies as further evidence. For example, in Rule R7, a returned RDF triple `(S,P,O)`, where `S` has been inferred to be an entity and `P` has been inferred to be a property, counts as evidence for inferring that `O` has domain `P`. User-defined predicates (such as `IsDictWord`) also count as evidence, as shown in Rule R2. Lexical similarity (in Rules R10-R17) also counts as evidence. For example, in Rule R8, the following count as evidence that `P` is a property of the entity type `T`: `P` appears as a predicate, and `T` as type, in the returned results, and the set of URIs of which `P` is predicated is similar to the set of URIs which are said to be of type `T`.

The bodies of Rules R10-R17 show how similarity relationships can be inferred from user-defined predicates (such as `LexSim` in, among others, Rule R10) and from user-provided definitions (possibly parametrized) of set similarity (such as `SimSetEq`[] in, among others, Rule R11). A value for `LexSim` is computed using cosine similarity for strings with length greater than fifty and Levenshtein (or edit) distance otherwise. In the case of URIs, we use the label (as given by `rdfs:label`) of the dereferenced resource the URI points to, or else the local name if no label is found. Rule R14 treats an entity as a set of property values and infers similarity between entities from the overlap of property value sets. Rule R16 uses object overlap (a metric that is commonly used to align properties [7,17]) to infer similarity of properties.

Consider again the example search results in Fig. 1. Given the baseline model, if the rules had equal weights, `EntityType(Movie)` would be inferred with a higher probability than `EntityType(Q386724)` because the former satisfies Rules R1 and R2, whereas, `Property(birthDate)` would more probable than `Property(spouse)` because the latter is a predicate of fewer resources. Instances of the `HasProperty` relationship are inferred based on their co-occurrence in RDF resources computed by set similarity predicates in PSL. Based on our running example, the probability of `HasProperty(Person, surname)` is greater than that of `HasProperty(Person, spouse)` because `surname` is a predicate of more resources than `spouse` is.

The Semantic Model: Adding Ontological Evidence. The baseline model only assimilates syntactic evidence. We can extend it with rules that assimilate evidence from ontologies to yield the model we call semantic.

Ontologies are computational artefacts that formally describe concepts and relationships in a given domain. Thus, in terms of the metamodel we target, they describe entity types and their properties and provide evidence that complements the baseline model described above. Our approach is to make use of statements in ontologies about types and properties. This is then used to ground predicates that represent ontological evidence. Table 1 shows the different kinds of evidence we extract from ontologies and how it is mapped into PSL predicates.

```
/*  Rules R1-R9: Inference of metamodel instantiations */
R1: EntityType(T)         ←̃ RDFSubject(S) ∧̃ RDFIsInstOf(S,T)
R2: EntityType(T)         ←̃ RDFSubject(S) ∧̃ RDFIsInstOf(S,T) ∧̃ IsDictWord(T)
R3: Entity(S)             ←̃ RDFIsInstOf(S,T) ∧̃ EntityType(T)
R4: Property(P)           ←̃ RDFSubjPred(S,P) ∧̃ Entity(S)
R5: PropertyValue(O)      ←̃ Triple(S,P,O) ∧̃ Entity(S) ∧̃ Property(P)
R6: HasType(S,T)          ←̃ RDFIsInstOf(S,T) ∧̃ EntityType(T)
R7: HasDomain(O,P)        ←̃ Triple(S,P,O) ∧̃ Entity(S) ∧̃ Property(P)
R8: HasProperty(T,P)      ←̃ RDFPredicate(P) ∧̃ RDFType(T) ∧̃
                  {P.RDFSubjPred(inv)} SimSetEq[URI] {T.RDFIsInstOf(inv)}
R9: HasPropertyValue(S,O) ←̃ Triple(S,P,O) ∧̃ Entity(S) ∧̃ Property(P)

/*  Rules R10-R17: Inference of similarity relationships */
R10: SimEntityType(T1,T2)    ←̃ EntityType(T1) ∧̃ EntityType(T2) ∧̃
                                LexSim(T1,T2)
R11: SimEntityType(T1,T2)    ←̃ EntityType(T1) ∧̃ EntityType(T2) ∧̃
                  {T1.HasProperty} SimSetEq[] {T2.HasProperty}
R12: SimEntity(E1,E2)        ←̃ Entity(E1) ∧̃ Entity(E2) ∧̃ Label(E1,L1) ∧̃
                                Label(E2,L2) ∧̃ LexSim(L1,L2)
R13: SimEntity(E1,E2)        ←̃ Entity(E1) ∧̃ Entity(E2) ∧̃ Name(E1,N1) ∧̃
                                Name(E2,N2) ∧̃ LexSim(N1,N2)
R14: SimEntity(E1,E2)        ←̃ Entity(E1) ∧̃ Entity(E2) ∧̃
                  {E1.HasPropertyValue} SimSetEq[] {E2.HasPropertyValue}
R15: SimProperty(P1,P2)      ←̃ Property(P1) ∧̃ Property(P2) ∧̃ LexSim(P1,P2)
R16: SimProperty(P1,P2)      ←̃ Property(P1) ∧̃ Property(P2) ∧̃
                  {P1.HasDomain(inv)} SimSetEq[] {P2.HasDomain(inv)}
R17: SimPropertyValue(V1,V2) ←̃ PropertyValue(V1) ∧̃ PropertyValue(V2) ∧̃
                                LexSim(V1,V2)
/* ... */
/*  Rules R18-R20: Extending inference with ontological evidence (OE) */
R18: Entity(S) ←̃ RDFIsInstOf(S,T) ∧̃ OntType(T)
R19: Entity(S) ←̃ RDFIsInstOf(S,T) ∧̃ LexSim(T,OT) ∧̃ OntType(OT)
R20: Entity(S) ←̃ RDFIsInstOf(S,T) ∧̃ OntEqType(T,OT)
/*  ...  */
/*  Rules R21-R23: Extending similarity relationships with OE */
R21: SimEntityType(T1,T2) ←̃ EntityType(T1) ∧̃ EntityType(T2) ∧̃
                             OntEqType(T1,T2)
R22: SimEntityType(T1,T2) ←̃ EntityType(T1) ∧̃ EntityType(T2) ∧̃ OntType(OT) ∧̃
                             LexSim(T1,OT) ∧̃ LexSim(T2,OT)
R23: SimEntityType(T1,T2) ←̃ EntityType(T1) ∧̃ EntityType(T2) ∧̃ OntType(OT) ∧̃
                  {T1.HasProperty}  SimSetEq[Property] {OT.OntHasProperty} ∧̃
                  {T2.HasProperty} SimSetEq[Property] {OT.OntHasProperty}
/*  ...  */
/*  Rules R24-25: Extending inference with feedback evidence */
R24:  Entity(S)      ←̃ RDFIsInstOf(S,T) ∧̃ EntityTypeFB(T,"yes")
R25: ¬̃Entity(S)      ←̃ RDFIsInstOf(S,T) ∧̃ EntityTypeFB(T,"no")
R26:  EntityType(S) ←̃ ∧̃ EntityTypeFB(S,"yes")
R27: ¬̃EntityType(S) ←̃ EntityTypeFB(S,"no")
/*  ...  */
```

Fig. 4. PSL rules used in structure inference

The main idea here is that being defined as a concept or property in some pertinent ontology counts as additional evidence that a returned result (e.g., a resource URI) is an entity type or property, respectively, in terms of our metamodel. Thus, we use the PSL predicates in Table 1 to construct PSL rules such as R18-R20 in Fig. 4, which assimilate ontological information as evidence that a given resource `S` is an entity. For example, if `"The Godfather"` appears in a triple in the returned results as an instance of `Movie`, where `Movie` is asserted to be a concept (e.g., in Movie Ontology (www.movieontology.org/)) this adds strength to the belief that `"The Godfather"` is an instance of `Entity` in our metamodel (see R18 in Fig. 4). We also use ontological statements to

Table 1. Mapping ontological statements into PSL predicates

`Ontological statement`	`PSL predicate`
`T rdf:type rdfs:Class`	`OntType(T)`
`T rdf:type owl:Class`	
`P rdf:type rdf:Property`	`OntProperty(P)`
`P rdf:type owl:DatatypeProperty`	
`P rdf:type owl:ObjectProperty`	
`T1 owl:equivalentClass T2`	`OntEqType(T1,T2)`
`P1 owl:equivalentProperty P2`	`OntEqProperty(P1,P2)`
`T1 owl:disjointWith T2`	`OntDisjointType(T1,T2)`
`P rdfs:domain T`	`OntHasProperty(T,P)`

supplement evidence for similarity relationships as exemplified by Rules R21-R23 in Fig. 4. Rule R21 exemplifies the direct use of ontological evidence. Rule R22 exemplifies the use of ontological evidence mediated by lexical similarity. Rule R23 exemplifies the use of set-similarity predicates where one of the sets contains elements from ontological statements. In this case, ontological statements help reconcile heterogeneity in the returned results.

Subsuming the baseline model, the semantic model $\mathbb{S} \supset \mathbb{B}$ contains Rules R1-R17 plus such rules as R18-R20 and R21-R23 that assimilate additional evidence to the predicates in the baseline model.

The Feedback Model: Assimilating User-Provided Evidence. The semantic model assimilates syntactic and semantic evidence. We can extend it with rules that assimilate user-provided evidence in the form of feedback.

There has been a growing interest in assimilating user feedback in data integration (see [2] for a general proposal, and [12] for a general methodology and recent work in the area). User feedback is even more important in environments that are characterized by large-scale heterogeneity and highly autonomous data sources as is the case in the Web of Data(WoD) [9]. Also, with recent advances in crowdsourcing, feedback for solving complex data integration challenges can now be obtained more cost-effectively. In this paper, we assume that the user knows the domain of the search term and is motivated by obtaining high-quality results from the search, which we believe to be reasonable in the context of search results personalization.

We use feedback in two ways. The first is to refine the report presented to the user. When the returned results have been structured and integrated using PSL, a report is presented to the user (as described in Sect. 6). Feedback can be provided that results in refinement of the report. For example, a prior state of the report shown in Fig. 2 could have contained another row in the `Movie` tables referring to the `Mumbai Godfather` film which the user ruled out as a false positive. In this case, feedback on inference results powers up a form of filtering, i.e., of data cleaning, generating an incentive for users to provide feedback in the first place. The second, and more significant, way we use feedback is to take advantage of accumulated, user-provided feedback to improve the inference results before a report is produced. In this case, feedback on inference results is another type of evidence that can be assimilated.

Table 2. PSL predicates used to gather user feedback (with examples)

PSL feedback predicate	Example
EntityTypeFB(uri,term)	EntityTypeFB(Q11424,no)
HasTypeFB(uri,uri,term)	HasType(dbr:The_Godfather,CreativeWork, yes)
HasPropertyFB(uri,uri,term)	HasPropertyFB(CreativeWork, author, yes)
SimEntityTypeFB(uri,uri,term)	SimEntityType(Film, Movie, yes)
SimPropertyFB(uri,uri,term)	SimPropertyFB(duration, runtime, yes)
SimEntityFB(uri,uri,term)	SimEntityFB(dbr:The_Godfather, lmdbr:film/43338, yes)

Table 2 shows the feedback predicates used in our model. The feedback is simply a comment from the user on the correctness of the inferred query predicate. Thus, we use the PSL predicates in Table 2 to construct such PSL rules as R24-R27 in Fig. 4, which assimilate user-provided feedback as evidence that a given resource `S` is, or is not, an entity. For example, if `"War and Peace"` appears in a triple in the returned results as an instance of `Movie`, where `Movie` is confirmed by feedback to be an entity type, this adds strength to the belief that `"War and Peace"` is an instance of `Entity` in our metamodel.

Weight Learning for PSL Program Generation. We generated two PSL programs $PSL(\mathbb{S})$ and $PSL(\mathbb{F})$ for inference by learning weights for the rules in $\mathbb{S}$ and $\mathbb{F}$. Firstly, we obtained search results using Sindice [11] and Falcons [3]. The search terms and vocabularies used were: for the `Films` domain, `"The Godfather"`, `"Godfather actors"`, `"Casablanca"` and the Movie ontology; for the `Cities` domain, `"Berlin"`, `"Manchester"` and the GeoFeatures (www.mindswap.org/2003/owl/geo/) and GeoNames (www.geonames.org/) ontologies; for the `People` domain, `Tim Berners-Lee`, `Chris Bizer` and the FOAF and SWRC (ontoware.org/swrc) ontologies. We collected the top 20 results from each search engine for each search term. The total number of typed individuals in the corpus was 304 and the total number of triples was 12,160. The corpus contained, resp., 180 and 502 distinct domain types and properties.

Secondly, we annotated the corpus with the ground truth. In doing so, we took into account the relevant domain. For example, searching for `"Casablanca"` return results about the city and not just the film of that name. Given the sample data from search results and vocabularies, we used PSL to learn weights for the rules in the semantic model and yield the $PSL(\mathbb{S})$ program. We learn the weights discriminatively using maximum-pseudo likelihood. To reduce overfitting, we use 5-fold cross-validation and we average the weights of each rule.

To learn weights for the rules in feedback model $\mathbb{F}$ and yield the PSL program $PSL(\mathbb{F})$, we proceeded as follows. One might crowdsource sample feedback instances, but we simulated feedback acquisition to give rise to synthetic sample. The synthesis procedure we used is as follows. We take as input a sample of inferences returned by $PSL(\mathbb{S})$ and the number of users providing feedback (100, in this paper). For each worker, we randomly generate 50 feedback instances, where a feedback instance is a true/false annotation on the inference returned for query predicates. In this paper, we assume that feedback is reliable. However, introducing a per-user degree of unreliability only requires a simple change.

5 Experimental Evaluation

The goal of the experimental evaluation is to measure how well the various PSL programs, generated as described above, infer a structure from LD search results that conforms to the adopted metamodel. To provide comprehensive information about the experiments, we have made the datasets, the code, and documentation available on GitHub [1].

Exp. 1: Inference Using Syntactic and Semantic Evidence. The goal of Exp. 1 is to measure the quality of $PSL(\mathbb{B})$, where only syntactic evidence is assimilated, and then measure the quality of $PSL(\mathbb{S})$, where semantic evidence is also assimilated, thereby allowing us to measure the impact of using ontologies on the quality. We measure the quality of our program using the area under precision-recall curve (AUC) for each query predicate in our PSL model. The precision/recall curve is computed by varying the probability threshold above which a query atom is predicted to be true. This means that the measurement does not depend on setting any threshold.

We first performed PSL inference on $PSL(\mathbb{B})$ on the search results, for each of three domains in turn. We denote the measured quality as $AUC(\mathbb{B})$ with some abuse of notation. We then added semantic evidence extracted from the relevant vocabularies to the search results and performed inference on $PSL(\mathbb{S})$. We denote the measured quality by $AUC(\mathbb{S})$. We then calculate the quality impact of using semantic evidence as $\Delta = AUC(\mathbb{S}) - AUC(\mathbb{B})$. Columns 2, 3 and 4 in each subtable in Table 3 list all the measurements obtained in Exp. 1 for the corresponding domain.

Discussion. As measured in terms of the AUC, across the domains, the quality of $PSL(\mathbb{B})$ is good on average (around 0.65) if we discount the similarity relationships, which are inherently dependent on semantic evidence. This suggests that $PSL(\mathbb{B})$ uses syntactic evidence effectively but that the approach might benefit from assimilating other forms of evidence.

Assimilating ontological evidence indeed leads to improvement in the AUC, particularly w.r.t. similarity relationships, with a knock-on positive effect on the quality of the tabular representation. Thus, the corresponding average $AUC(\mathbb{S})$ increases to close to 0.7. The degree of improvement varies across

[1] https://github.com/duhai-alshukaili/StructuringLDSearchResults.

Table 3. AUC results for our PSL models with datasets in the test collection

Query Predicate	$AUC(\mathbb{B})$	$AUC(\mathbb{S})$	$\varDelta$	$AUC(\mathbb{S} \cup \mathbb{F})$	$\varDelta'$
`Entity`	.726	.736	.010	.827	.091
`EntityType`	.749	.812	.063	.954	.142
`HasProperty`	.512	.578	.057	.614	.036
`HasType`	.554	.604	.050	.709	.105
`Property`	.774	.774	.000	.779	.005
`SimEntity`	.083	.108	.025	.322	.214
`SimEntityType`	.320	.351	.030	.517	.166
`SimProperty`	.159	.184	.025	.621	.437

(a) Films

Query Predicate	$AUC(\mathbb{B})$	$AUC(\mathbb{S})$	$\varDelta$	$AUC(\mathbb{S} \cup \mathbb{F})$	$\varDelta'$
`Entity`	.645	.751	.105	.789	.039
`EntityType`	.828	.844	.017	.844	.000
`HasProperty`	.457	.511	.054	.559	.048
`HasType`	.557	.731	.174	.732	.000
`Property`	.683	.683	.000	.685	.001
`SimEntity`	.086	.583	.497	.612	.030
`SimEntityType`	.869	.891	.022	.891	.000
`SimProperty`	.257	.230	-.027	.537	.307

(b) People

Query Predicate	$AUC(\mathbb{B})$	$AUC(\mathbb{S})$	$\varDelta$	$AUC(\mathbb{S} \cup \mathbb{F})$	$\varDelta'$
`Entity`	.525	.651	.126	.882	.231
`EntityType`	.776	.793	.017	.793	.000
`HasProperty`	.470	.485	.015	.541	.056
`HasType`	.631	.668	.037	.820	.152
`Property`	.774	.780	.006	.794	.014
`SimEntity`	.082	.344	.261	.411	.068
`SimEntityType`	.648	.662	.014	.662	.000
`SimProperty`	.484	.386	-.098	.687	.302

(c) Cities

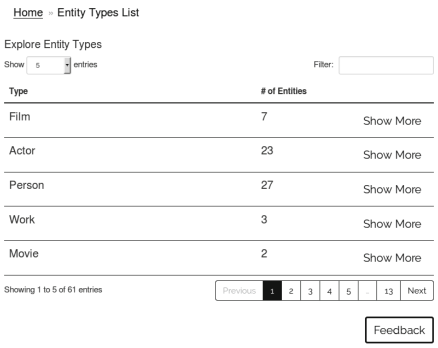
domains depending on the coverage of the ontologies used. For example, the $\mathbb{B}$-probability of `HasProperty(dbo:Person, dbo:spouse)` is 0.04, whereas its $\mathbb{S}$-probability is much higher, at 0.80, as the DBPedia ontology explicitly defines this relationship. In other cases, explicit assertion of type disjointness has a significant effect too. Thus, the $\mathbb{B}$-probability of `SimEntity(dbr:Casablanca, dbr:Casablanca_(film))` is 0.55, whereas its $\mathbb{S}$-probability is much lower, at 0.01, because `dbo:Work` and `dbo:wgs84_pos:SpatialThing` are disjoint in the DBpedia ontology. In the case of `Entity`, the assertion of a type by an ontology acts as a reliable anchor for individuals returned in the search results. For example, the $\mathbb{B}$-probability of `Entity(lmdb:film/43338)` is 0.22, whereas its $\mathbb{S}$-probability is higher, at 0.38, because its type, `lmdb:film`, matches the type `dbo:Film`, in the MO. As hinted above, improvements in the inference of metatypes (e.g., `Entity`) has a knock-on effect on the corresponding set-similarity relationship (`SimEntity` in this example). In the case of `SimEntity` the improvement is more significant the `People` and `Cities` domain than for `Films` because of inherent type ambiguity. For example, searching with `"Casablanca"` returns films, organizations, and a city. Type ambiguity is perhaps best solved with user feedback, as the next experiment shows.

Exp. 2: Inference Using Feedback. The goal of Exp. 2 is to measure the quality of $PSL(\mathbb{S} \cup \mathbb{F})$, where feedback evidence is also assimilated, thereby allowing us to measure the impact of using feedback on the quality. We simulated the feedback evidence as being provided for the top 5 % of the inference results for each feedback target. This assumes a strategy in which feedback is targeted at removing false positives. We denote the measured quality by $AUC(\mathbb{S} \cup \mathbb{F})$. We then calculate the quality impact of using semantic evidence as $\varDelta' = AUC(\mathbb{S} \cup \mathbb{F}) - AUC(\mathbb{S})$. Columns 5 and 6 in each subtable in Table 3 list all the measurements obtained in Exp. 2 for the corresponding domain.

Discussion. As measured in terms of the AUC, across the domains, the quality of $PSL(\mathbb{S} \cup \mathbb{F})$ is quite good in average (around 0.7) even if we include the similarity relationships. In other words, user feedback seems to address the ambiguity issues that caused the quality of $PSL(\mathbb{S})$ to be lower for similarity relationships. Thus, although the impact of feedback is not uniformly high, it seems complementary and corrective, i.e., it improves the most where the most improvement is needed, viz., the similarity relationships, where we can observe AUC improvements that can reach 80 %, 130 %, up to almost 240 %. This, the highest improvement, was observed for `SimProperty` in the `Films` domain. The reason for this is that $PSL(\mathbb{S})$ produces many false positives for `SimProperty`. One possible reason is that property names (e.g., `name`) are often reused without qualification for very different concepts. Combining syntactic and ontological evidence seems insufficient.

6 A Deployment Case Study

Imagine a user gives `"Godfather actors"` as the search term. Relevant returned results come predominantly from the Linked Movie Database and the DBpedia. A few, less relevant, results come from Linked WordNet, BookMashup, and MusicBrainz. The PSL program uses the results to make inferences as to how to instantiate the target metamodel in a way that integrates the returned results into a tabular report.



Home » Entity Types List

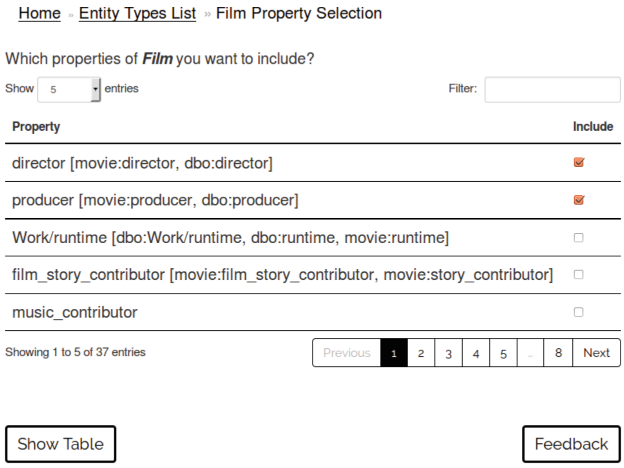
Explore Entity Types

Show 5 entries Filter:

Type	# of Entities	
Film	7	Show More
Actor	23	Show More
Person	27	Show More
Work	3	Show More
Movie	2	Show More

Showing 1 to 5 of 61 entries

Fig. 5. Type selection



Home » Entity Types List » Film Property Selection

Which properties of ***Film*** you want to include?

Show 5 entries Filter:

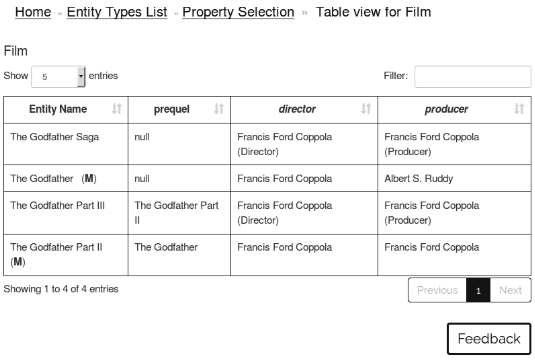
Property	Include
director [movie:director, dbo:director]	☑
producer [movie:producer, dbo:producer]	☑
Work/runtime [dbo:Work/runtime, dbo:runtime, movie:runtime]	□
film_story_contributor [movie:film_story_contributor, movie:story_contributor]	□
music_contributor	□

Showing 1 to 5 of 37 entries

Fig. 6. Property selection

As depicted in Fig. 5, our PSL-driven user interface then provides the user with a list of postulated entity types for the given search term. The PSL query predicate behind Fig. 5 is `HasType`, with rows ordered by `EntityType` probability. At this point, the user can express an interest in one of the listed types by pressing on `Show More` which lists the inferred properties of the selected type. Figure 6 shows the list of inferred properties of the postulated entity type Film. The PSL query predicate behind Fig. 6 is HasProperty, with rows ordered by Property probability.

At this point, the user can tick some properties to obtain the final tabular representation by clicking on Show Table. Figure 7 shows the table produced for entity type Film by selecting the properties prequel, director, and producer. Some properties shown (e.g. producer) are candidates for fusion and some entities (e.g. The Godfather) are candidates for deduplication. At each stage in this process, the user can intervene by clicking on the Feedback button to contribute feedback, which becomes evidence for use in future searches.


Home » Entity Types List » Property Selection » Table view for Film
Film
Show 5 entries
Filter:
Previous 1 Next
Feedback

Entity Name	prequel	*director*	*producer*
The Godfather Saga	null	Francis Ford Coppola (Director)	Francis Ford Coppola (Producer)
The Godfather (**M**)	null	Francis Ford Coppola	Albert S. Ruddy
The Godfather Part III	The Godfather Part II	Francis Ford Coppola (Director)	Francis Ford Coppola (Producer)
The Godfather Part II (**M**)	The Godfather	Francis Ford Coppola	Francis Ford Coppola

Showing 1 to 4 of 4 entries

Fig. 7. Data table

Note, therefore, that the use of a probabilistic framework allows us not only to structure and integrate the results but also to improve presentation (e.g., by ordering rows by likelihood) and to obtain targeted feedback. Note also that since the underlying PSL program models similarity relationships, the interface can make principled, uniform choices regarding deduplication (using SimEntity and SimProperty) and data fusion (using SimProperty and SimPropertyValue). In the example screenshots, some candidate property values (e.g., of Director) have been fused on the basis of SimEntity. Without this, the final table might be more heavily polluted by the natural redundancy one expects in search results.

7 Conclusions

This paper has provided empirical backing for the research hypothesis that assimilating different sources of (partial and uncertain) evidence is effective in inferring a good quality tabular structure over LD search results. The paper has described how a PSL program has been constructed with which the different sources of evidence can be assimilate in a principled and uniform way, where such sources are syntactic matching, domain ontologies and user feedback. It was shown how the PSL program can drive a user interface by mean of which the user can provide feedback that improves future quality, in a pay-as-you-go style. Moreover, the expressiveness of PSL allows the program to express similarity relationships from which, as shown, it is possible to perform immediate duplicate detection and data fusion prior to showing cleaner results to the user.

Acknowledgement. This work has been made possible by funding from the Omani National Program for Graduate Studies. Research on data integration at Manchester is supported by the VADA Programme Grant of the UK Engineering and Physical Sciences Research Council, whose support we are pleased to acknowledge.

References

1. Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: Hinge-loss Markov random fields and probabilistic soft logic. CoRR abs/1505.04406 (2015)
2. Belhajjame, K., Paton, N.W., Fernandes, A.A., Hedeler, C., Embury, S.M.: User feedback as a first class citizen in information integration systems. In: CIDR. pp. 175–183 (2011)
3. Cheng, G., Qu, Y.: Searching linked objects with falcons: approach, implementation and evaluation. Int. J. Semant. Web Inf. Syst. (IJSWIS) **5**(3), 49–70 (2009)
4. Christodoulou, K., Paton, N.W., Fernandes, A.A.A.: Structure inference for linked data sources using clustering. Trans. Large-Scale Data- Knowl.-Centered Syst. **19**, 1–25 (2015)
5. Ding, L., Finin, T.W.: Boosting semantic web data access using swoogle. In: Proceedings of the 20th NCAI, pp. 1604–1605 (2005)
6. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 107–121. Springer, Heidelberg (2008)
7. Gunaratna, K., Thirunarayan, K., Jain, P., Sheth, A., Wijeratne, S.: A statistical and schema independent approach to identify equivalent properties on linked data. In: Proceedings of the 9th International Conference Semantic Systems, pp. 33–40. ACM (2013)
8. Kimmig, A., Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: A short introduction to probabilistic soft logic. In: NIPS/Probabilistic Programming (2012)
9. Madhavan, J., Cohen, S., Dong, X.L., Halevy, A.Y., Jeffery, S.R., Ko, D., Yu, C.: Web-scale data integration: you can afford to pay as you go. In: Proceedings of the CIDR 2007, pp. 342–350 (2007)
10. Niepert, M., Noessner, J., Meilicke, C., Stuckenschmidt, H.: Probabilistic-logical web data integration. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 504–533. Springer, Heidelberg (2011)
11. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: a document-oriented lookup index for open linked data. IJMSO **3**(1), 37–52 (2008)
12. Paton, N.W., Belhajjame, K., Embury, S.M., Fernandes, A.A.A., Maskat, R.: Pay-as-you-go data integration: experiences and recurring themes. In: Freivalds, R.M., Engels, G., Catania, B. (eds.) SOFSEM 2016. LNCS, vol. 9587, pp. 81–92. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49192-8_7
13. Pujara, J., Miao, H., Getoor, L., Cohen, W.: Knowledge graph identification. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 542–557. Springer, Heidelberg (2013)
14. Sazonau, V., Sattler, U., Brown, G.: General terminology induction in owl. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 533–550. Springer, Heidelberg (2015)

15. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.ma: live views on the web of data. Web Semant. **8**(4), 355–364 (2010)
16. Völker, J., Niepert, M.: Statistical schema induction. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 124–138. Springer, Heidelberg (2011)
17. Zapilko, B., Mathiak, B.: Object property matching utilizing the overlap between imported ontologies. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 737–751. Springer, Heidelberg (2014)
18. Zhu, M., Gao, Z., Pan, J.Z., Zhao, Y., Xu, Y., Quan, Z.: Ontology learning from incomplete semantic web data by BelNet. In: 25th ICTAI, pp. 761–768. IEEE (2013)