Interpreting linked data search results using Markov Logic

Duhai Alshukaili
School of Computer Science
University of Manchester
Oxford Road, Manchester
M13 9PL, UK
duhai.alshukaili@gmail.com

Alvaro A.A. Fernandes
School of Computer Science
University of Manchester
Oxford Road, Manchester
M13 9PL, UK
alvaro@cs.man.ac.uk

Norman W. Paton
School of Computer Science
University of Manchester
Oxford Road, Manchester
M13 9PL, UK
norm@cs.man.ac.uk

ABSTRACT

Linked Data (LD) follows the web in providing low barriers to publication, and in deploying web-scale keyword search as a central way of identifying relevant data. As in the web, searches initially identify results in broadly the form in which they were published, and the published form may be provided to the user as the result of a search. This will be satisfactory in some cases, but the diversity of publishers means that the results of the search may be obtained from many different sources, and described in many different ways. As such, there seems to be an opportunity to add value to search results by providing users with an integrated representation that brings together features from different sources. This involves an on-the-fly and automated data integration process being applied to search results, which raises the question as to what technologies might be most suitable for supporting the integration of LD search results. In this paper, we investigate the use of Markov Logic, which brings together first order logic and probabilistic graphical models to support both learning and inference in uncertain domains. Specifically, we: (i) characterise key features of LD search results that are relevant to their integration; (ii) discuss how these motivate the use of an approach based on Markov Logic; (iii) describe some initial experiences in the use of Markov Logic for interpreting search results; and (iv) present some avenues for future investigation.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed databases; H.3.5 [**Online Information Services**]: Data sharing; H.3.5 [**Online Information Services**]: Web-based services

1. INTRODUCTION

Linked Data (LD) seeks to do for data what the web did for documents. In essence, LD involves publication of data according to a small collection of principles that indicate how data resources are identified and represented, and that encourage the creation of links [1]. Publishers make data available following the principles, and users access or process the resulting data using either generic tools or bespoke applications.

As in the web of documents, keyword search engines are an important element in the tool set. However, although search results in the web of documents provide a result that was designed for human use, search results in the web of data tend to be collections of RDF resources that may be time consuming and cumbersome to explore. Furthermore, the results of a search may involve values of different types, which are interleaved in a search result. For example, a search for *Bob Dylan* using the *Sindice* search engine [22] returns results that represent a person, (several) albums and (several) songs. A manually created report over such a search result might pull together the properties of the individual *Bob Dylan* from several resources into a heading and a list of properties, and then might provide separate tables for the collections of albums and songs about which information was retrieved.

Could such a report be generated automatically? The creation of such a report requires, at a minimum, identifying:

- the (real-world) entity types that are represented in the search result;
- the individuals entity instances that are represented in the search result, and in particular which ones are individuals and which ones belong to collections; and
- the properties of each of the relevant entity instances.

Although this is a data integration problem, it is quite unlike classical data integration, in which typically there is a known target (or global) schema to which source data should be mapped, and there is some level of human engagement in the data integration. This raises the question as to what approaches might be suitable for interpreting and integrating search results.

We know of one previous proposal to address this problem, namely Sig.ma [24], which generated a report that integrated the top hits from the Sindice search engine. In Sig.ma, several steps were followed to integrate the data, as discussed more fully in Section 2, but the overall approach assumed that the result of the search described a single entity instance. Although Sig.ma was important in identifying the opportunity and in developing an initial realisation of the vision, the data integration component of Sig.ma was quite restrictive and seemed rather *ad hoc*. As such, there seems to be scope to explore additional approaches that seek to make the result integration both more general and more systematic. We would expect there to be several very different ways of addressing this problem, one of which is explored in this paper.

Here we discuss an approach based on Markov Logic [6], in which we: (i) postulate rules that capture relationships that exist within search results, which are expressed using logic; (ii) learn weights for these rules that represent their strengths as constraints; and (iii) use the resulting weighted rules over search results to infer (with uncertainty) the entity types and instances that are represented in the result. We are motivated to use Markov Logic as it provides a well founded approach to integrating evidence of different types to support conclusions that can inform data integration. Markov logic has been applied to a range of tasks of relevance to data integration, including classification, entity resolution and knowledge-base construction, as discussed more fully in Section 2.

This paper investigates the application of Markov Logic to the integration of LD search results. In so doing, it uses the following structure. In Section 2, we review related work in linked data search, linked data integration and applications of Markov Logic. In Section 3 we provide an overview of Markov Logic, and in Section 4 we describe how it can be applied for identifying entity types and instances in linked data search results. In Section 5 we draw some conclusions and outline directions for future work.

2. RELATED WORK

In this section we discuss work that is relevant to the problem of linked data search result integration, specifically by reviewing results on *Linked Data Search*, *Data Integration for Linked Data* and *Markov Logic for Data Integration.*

Linked Data Search. The increase in the amount of RDF data published in the web has given rise to a number of LD search proposals. Swoogle [5] is an early LD search engine that indexed RDF(s) and OWL ontologies using inverted indexes. Given a set of terms, Swoogle returns ontologies that mention these terms. As such, Swoogle adopts a document-centric approach for indexing LD ontologies [12]. Unlike Swoogle, the Falcons [3] search engine was tailored towards the search of arbitrary data while providing an entity-centric search approach in which the objective is to identify the most relevant RDF resources rather than the most relevant documents that contain them. Falcons included components for crawling, parsing, organizing, ranking, sorting and querying RDF data. In order to extract terms from RDF documents, it employed the notion of a virtual document as an intermediate representation that enables the consolidation of data from multiple sources. This consolidation is based on URI reuse and the mention of an entity in different sources. Falcons also includes a reasoning component aimed at inferring class hierarchies of indexed entities. These hierarchies provide a means for restricting the search result based on the types of resources. Sindice [22] adopts a document-centric approach and aimed to provide a range of search services over RDF documents. The services offered include keyword search over RDF, searching for entities (classes, properties and individuals) matching a term in RDF documents, and providing APIs to expose search services to software agents. Sindice consolidates entities while indexing, based on inverse functional properties. It also implements a localized reasoning component for discovering additional information about entities. From this brief review, it can be seen that LD search engines often carry out some preliminary result consolidation tasks, but falls short of a concerted approach to result integration.

Sig.ma [24], however, does address result integration in a more comprehensive manner. Sig.ma uses search results from Sindice [22] to collect RDF data and build an aggregated view of the results in the form of entity profiles. Sig.ma uses a recursive search step that collects RDF data which contains resource identifiers that match a search term. A first search step collects the source URLs that contain the search terms. A second step is performed to search for sources containing the URI identifiers found in the results of the first step. The collected RDF data is decomposed into chunks (called resource descriptions) that describe distinct entities, and ranked against the search term. Sig.ma collects additional data when it encounters an *owl:sameAs* predicate. The resulting resource descriptions are consolidated by combining the values of lexically similar attributes. Hand-crafted rules such as the removal of “has” from “has title” or replacement of attributes that may share similar values such as “web page” and “homepage” with the term “Web page” are applied in the consolidation step. In addition, Sig.ma allows users to interact with the resulting entity profiles, either through navigating to other sources of information, or by refining the results. The refinement capabilities allow users to reject or accept the sources that contribute data to the generated entity profiles. However, Sig.ma does not provide any means for resolving semantic heterogeneities in the data before attempting to construct the integrated view. For example, if a user is interested in information about *Manchester University*, Sig.ma combines data on the university, on Manchester Grammar School, and on a railway station. While it brings together lots of correct information, some incorrect data is often included. Our aim is to develop a more principled approach that enables us to resolve such heterogeneities through the identification entities and their types using Markov Logic.

Data Integration for Linked Data. Low barriers to publication, as well as diversity of publishers without central coordination, have led to LD being published with inconsistencies both at the conceptual and at the instance levels. At the conceptual level, this comes in the form of different conceptualizations of the same domain, inconsistencies in the structural representation of concepts in LD terminologies, etc. At the instance level, there may be many different resources that describe the same real world entities, redundant information, and contradictory attribute values. There is a plethora of approaches that deal with the alignment and matching of RDF sources. These approaches can be divided into two broad categories: ontology matching and instance matching approaches [2]. The goal of ontology matching approaches is to align schema level elements of RDF sources using information from the schema level, the instance level, or both [7]. On the other hand, instance level approaches try to resolve the multiplicity of data resources that describe the same real world entity [8]. RDF instance matching tools such Silk [16, 15], ObjectCoref [14], and LIMES [18] discover *owl:sameAs* identity links at the instance level. Our work aims not just to resolve identities between pairs of resources in the search results but also to infer a ER schema structure for the results and populate with data provided in the instances of the search. In this regard, there have been a number of proposals for structure inference from RDF sources [4, 27, 28, 25]. Such approaches take as an input a data graph and produce a structural summary that is homomorphic to the original data graph using techniques such as hierarchical clustering [4, 28], association rule mining [25], and inference using Bayesian Networks [27]. However, these approaches are often evaluated on a specific dataset at time (i.e. Magnatune or DBpedia). This is different from inferring a structure from search results because the relevant sources in the results often vary in terms of the datasets they originate from. For example, a search for a film title (e.g. Godfather) on Sindice [22] returns results from at least three datasets: DBpedia[1], freebase[2], and LinkedMDB [3]. This makes the structure inference problem harder as mappings between the dataset need to be inferred or incorporated as evidence.

In addition to research in linking RDF at the conceptual and instance levels, there have been some studies on mapping properties across RDF sources [9, 26, 10]. The aim of such approaches is to find similar [9] or equivalent [26, 10] properties using statistical measures that utilize subject and object overlap of properties. These approaches, as well as approaches that map between ontology concepts and instances across datasets can be utilized as additional sources of evidence in in our approach (see Section 5).

Markov Logic for Data Integration. Markov Logic brings together two prominent paradigms within artificial intelligence, namely first-order logic and Markov networks [6]. The basic idea is that the syntax of first order logic is used to describe constraints that hold on the set of possible worlds, but that the constraints are no longer necessarily hard. Instead, each formula is associated with a weight that indicates how strong the constraint is; the higher the weight the more likely a constraint is to hold. A set of weighted formulas is referred to as a Markov logic network (MLN).

Markov Logic (ML) has been applied to data integration tasks such as entity resolution [23], knowledge base construction [20] and ontology matching [19]. Sigla and Domingos [23] described a domain-specific MLN that performs an entity resolution task on bibliography entries. They used an MLN to encode knowledge about the similarity of the publications based on the similarity of the venue, authors and titles. They demonstrated that such an MLN, combined with predicate equivalence and reverse predicate equivalence axioms, can achieve superior results to established approaches.

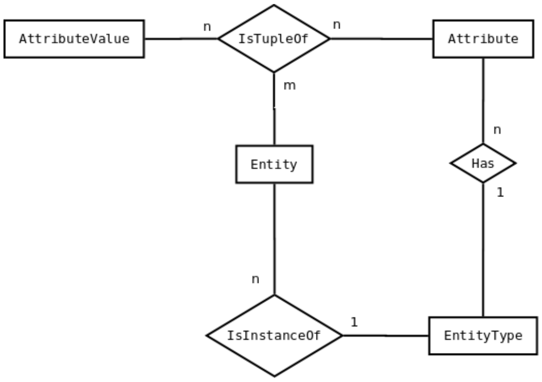

Figure 1: The ER metamodel that is to be populated from search results.

In contrast with this approach, the rules we use in our MLN do not currently encode domain-specific knowledge. In seeking to integrated data from multiple sources, our work is similar to Elementary [20]. The goal of Elementary is the construction of a knowledge base of entities based on information extracted from web pages. It uses an MLN inference engine for discovering co-referent entity mentions of people and organizations. It also links between such entities by using the co-occurrence pattens as evidence in the inference process. Elementary combine various forms of evidence, such as data extracted from standard NLP toolkits, domain knowledge, lexical matching, and user feedback in the inference process. In the context of LD, Niepert *et al.* [19] utilized ML for ontology and RDF instance matching problems. Their approach to ontology matching was combining logical axioms expressed in ontologies with lexical similarities to map between the concepts and attributes in different ontologies. In their instance matching problem they utilized a similarity metric [21] to infer matches between instances. They used the constraints defined in the ontologies of these instances to prevent the MLN from making incorrect inferences. While this approach demonstrates how semantic and syntactic evidence can be combined using ML, this approach assumes that ontology definitions are available with the instance data, which is not necessarily the case for LD search results.

3. MARKOV LOGIC

Markov Logic combines *first-order logic* and Markov networks in a unifying representation for the definition of probabilistic models.

Formally, a Markov logic network (MLN) is a set of pairs (F_i, w_i), where F_i is a first-order logic formula and w_i is a real value representing its weight. Formulas can be seen as constraints on a set of possible worlds. The higher the weight, the stronger the constraint is, and therefore, the less probable is a world that violates the constraint. In an MLN, a formula with a negative weight w can be replaced with its negated formula with a weight of $-w$. A formula can also be assigned an infinite weight to indicate a constraint that should not be violated. Given as set of constants in some domain, an MLN defines a ground Markov network where the nodes correspond to ground predicates.

[1] http://www.dbpedia.org
[2] http://rdf.freebase.com
[3] http://data.linkedmdb.org/

```
//Evidence Predciates
Triple1(uri,uri,uri)
Triple2(uri,uri, literal )

//Query Predicates
Entity(uri)
EntityType(uri)
Attribute(uri)
AttributeValue( literal )
LnkAttributeValue(uri)
Has(uri,uri)
IsInstanceOf(uri,uri)
```

(a) Predicates

Triple1(s,rdf:type,o) => Entity(s) ^ EntityType(o) ^ IsInstanceOf(s,o)	R_1
Triple2(s,p,o) ^ Attribute(p) ^ AttributeValue(o) => Entity(s)	R_2
Triple1(s,p,o) ^ Attribute(p) ^ LnkAttributeValue(o) => Entity(s)	R_3
Triple2(s,p,o) ^ Entity(s) ^ AttributeValue(o) => Attribute(p)	R_4
Triple1(s,p,o) ^ Entity(s) ^ LnkAttributeValue(o) => Attribute(p)	R_5
Triple1(s,p,o) ^ Entity(s) ^ Attribute(p) => LnkAttributeValue(o)	R_6
Triple2(s,p,o) ^ Entity(s) ^ Attribute(p) => AttributeValue(o)	R_7
Triple1(s,rdf:type,type) ^ Triple2(s,p,o) ^ EntityType(type) ^ Attribute(p) => Has(type,p)	R_8
Triple1(s,rdf:type,type) ^ Triple1(s,p,o) ^ EntityType(type) ^ Attribute(p) => Has(type,p)	R_9

(b) Rules

Figure 2: An MLN rule set that uses RDF triples to derive ER construct extensions

An MLN formula is defined over a set of predicates. The predicates can be categorized into *query* and *evidence* predicates. The MLN formulas define relationships using these predicates. The MLN in Figure 2 describes relationships between *evidence* predicates that represent search results and *query* predicates that represent the constructs of the entity relationship (ER) meta model in Figure 1. The evidence and query predicates are then related to each other by formulas such as:

$$Triple1(s, ”rdf : type”, o) \implies EntityType(s).$$

In this formula, s and o are variables that represent the subject and object of a triple in a search result. The formula states that where there is a triple in which s is related to o by $rdf : type$, we can infer that s is an $EntityType$. Now, in fact, this may not always be the case, and the weight associated with the formula in the MLN captures the strength of the constraint represented by the rule. We discuss the evidence and query predicates, as well as the formulas in our MLN model, in more detail in Section 4.

Given a domain of interest, there are three tasks to be performed by the modeller: structure learning, weight learning and inference. An overview of these tasks now follows.

Structure Learning

Given a set of predicates and example data for the domain of interest, the ML structure learning process learns first-order logic formulas that define the relationships between the given declared predicates from the evidence provided in the form of example data in the domain. The structure learning process uses a beam search strategy to find the best clauses to add to the MLN [6]. In theory, structure learning provides an alternative to relying on domain experts to write rules that capture the semantics of the domain. However, the structure learning process is known to give rise to scalability issues for large datasets [17]. Given that, in this case, the presumed MLN structure is known *a priori*, i.e., we have developed a meta-model of ER models, and written rules for populating this meta-model based on the evidence in the form of triples from the search result. Thus, we have not performed structure learning and hence do not report any results in this respect.

Weight Learning

Given a set of rules and a database of evidence from the domain of interest, the weights of the rules can be learned. In this process, one or more predicates whose truth values are unknown are designated as query predicates. The learning procedure optimizes the learned weights with respect to such predicates assuming that all the truth values of the remaining predicates are given. The weight produced for each rule can be either positive, negative, or zero. A positive weight is an indication that a rule is supported in the domain given the evidence. On the other hand, a learned negative weight $-w$ is an indication that a rule is not supported by domain evidence, and in fact its negation is supported in the domain with weight w. Finally, a weight 0 suggests that a rule has no evidence (either for or against) in the domain. This occurs when groundings of the rule are not provided in the evidence.

Inference

The inference process takes as input a weighted MLN and a database consisting of ground evidence predicates, and outputs the marginal probabilities of query predicates of the most likely world given the evidence. This involves finding the truth assignment that maximises the sum of the weights of satisfied clauses [6].

4. INTERPRETING LINKED DATA SEARCH RESULTS

The problem we address is that of inferring, from search results, the entities, entity types, attributes and relationships that are described in the results. By these terms, we mean the constructs that are familiar from entity-relationship (ER) conceptual modelling. Identifying what entities and attributes are described from the user's point of view is a task with uncertain outcomes. This is because not every resource described using RDF is seen as an entity by the user. For example, the RDF resource depicted in Figure 3(a) that describes an organization named "Universities UK" may not be considered an entity in a search for members of a collection of UK universities. Also, not all RDF predicates that are used in the description of a resource can be seen as attributes of interest to the end user. An example of this is *dbo:wikiPageID*, which denotes a Wikipedia page identifier from the DBpedia dataset.

To model such uncertainty, we use ML to define a number of hypotheses as to what the URIs and literals in the resources

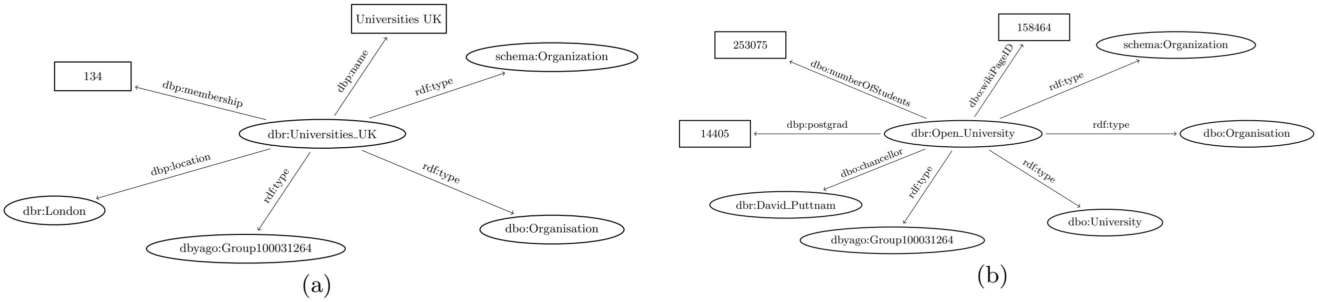

Figure 3: An Example of search result for a search of "Universities UK"

denote in terms of ER constructs. Inference from such hypotheses allows us to build an ER view of the data in the search results, and thereby to organize the results in a form that is suitable for the user. A key advantage of using ML for this task is that it provides an opportunity for incorporating different forms of evidence, including using feedback as typical of the *pay-as-you-go* approach to data integration [11]. We now describe an MLN for learning the uncertainty of constructing such ER views from the search results. We also describe the learning and inference processes. Finally, we present the initial experimental results of our investigation.

4.1 An MLN for search results

We now describe the MLN in Figure 2 that we use to interpret RDF search results. The evidence predicates represent observed variables with known truth values in the ground Markov network. The query predicates represent the unobserved variables for which the inference process estimates a probability distribution given the evidence. In this MLN, the evidence predicates represent the observed triple patterns in the RDF data. These predicates are defined over the *uri* and *literal* domains. Partitioning the space of triples provides a simpler, more direct way for writing rule bodies. As such, we use *Triple1* to represent RDF triples that have URIs in the object position, and *Triple2* to represent RDF triples what have literals in the object position. Then, for example, we can use the *Triple2* predicate in Rule R_7 in Figure 2 for the query predicate *AttributeValue* because we expect a literal in the object position. Such partitioning can be extended to include BNodes, although, here we only use *uri* and *literal* domains.

There are seven query predicates that characterize the hypotheses we want to substantiate using the evidence, which represent the constructs in the meta-model in Figure 1. The *Entity*, *EntityType*, and *Attribute* predicates model whether a given URI represents an entity, entity type, or attribute. For example, given the RDF graph shown in Figure 3(b), the following are true:

Entity(dbr:Open_University),
Attribute(foaf:name), and
EntityType(schema:Organization).

Conversely, given the graph shown in Figure 3(a),the following are false:

Entity(dbp:name),
Attribute(dbr:London), and
EntityType(dbr:Universities_UK).

We also define predicates for *Has* and *IsInstanceOf*. *Has* models a relationship between two URIs where the first is an entity type and the second is an attribute of that type. *IsInstanceOf* models a relationship between two URIs where the first is an entity, the second is an entity type, and the first is an instance of the second. Examples from Figure 3 are:

Has(dbo:Organisation,dbp:location), and
IsInstanceOf(dbr:Open_University,dbo:Organisation).

Finally, we use two predicates to interpret values in the object position of RDF triples, viz., *AttributeValue* and *LnkAttributeValue*.

In addition to the described predicates, the MLN contains rules that encode knowledge about the relationships between MLN predicates. Rules R_1, R_8 and R_9 utilize the *rdf:type* construct for the inference of *Entity*, *EntityType*, *IsInstanceOf* and *Has*. Note that R_1 makes a direct inference from the evidence, whereas R_8 and R_9 additionally rely on *EntityType* and *Attribute*.

4.2 Experiments

Dataset

To our knowledge, there is no publicly available standard dataset that allows us to evaluate our proposed approach. In order to learn the weights for the MLN described in Section 4.1 we conducted 10 searches using the Sindice [22] LD search engine. The terms used in these searches are shown in Table 1. The results were pre-processed by removing triples containing *rdf:type* objects that belong to the *yago*[4] and *dbyago*[5] name-spaces. The reason is that such types are used for categorizing resources as opposed to assigning real-world entity types to resources. These types cannot be easily assigned specific attributes. Also triples which contain domain-independent RDF predicates and have literal objects were removed: *dct:abstract*, *rdfs:comment*, *rdfs:label*, *skos:prefLabel*, *skos:altLabel*, *skos:note* and *dce:description*.

[4] http://yago-knowledge.org/resource/
[5] http://dbpedia.org/class/yago/

Predicate	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Average**
Entity	0.41	**0.51**	0.41	**0.61**	**0.90**	**0.568** ± 0.203
EntityType	**0.95**	**0.97**	**0.88**	**0.92**	**0.97**	**0.938** ± 0.038
Attribute	0.25	0.15	0.24	0.14	0.17	0.190 ± 0.051
Has	0.06	0.08	0.04	0.04	0.07	0.058 ± 0.018
InstanceOf	**0.85**	**0.89**	**0.90**	**0.90**	**0.77**	**0.862** ± 0.055

Table 3: AUC PR scores per fold

Domain	**Search Terms**
Cities	Berlin,Manchester
Movies	Godfather,Casablanca
Organizations	Apple Inc.,Microsoft
People	Tim Berners-Lee, Chris Bizer
Collections	Godfather actors, UK universities

Table 1: 10 search terms used in constructing the learning/evaluation dataset

Entity(dbr:Open_University)
Entity(dbr:Open_University)
EntityType(dbo:Organisation)
EntityType(dbo:University)
Has(dbo:University,dbp:postgrad)
Has(dbo:University,dbo:chancellor)
IsInstanceOf(dbr:Open_University,dbo:University)
IsInstanceOf(dbr:Open_University,dbo:Organisation)

Figure 4: Ground truth annotation for the RDF in Figure 3(b)

From each search, the top five results were selected to be annotated with the ground truth. Figure 4 shows an example of the ground truth annotation for the RDF graph shown in Figure 3(b).

Methodology and Results

To learn the weights of the rules in the MLN, we used a 5-fold cross-validation procedure on the annotated dataset. To ensure that the weights are not skewed towards a particular domain of search, we randomized the search results to ensure that every split contained search results from every one of the domains shown in Table 1. Table 2 shows the average weights learned for the MLN rules. A positive weight of a rule is an indication that the rule is supported by the evidence present in the data. The higher the weight, the stronger the evidence for the corresponding rule. One obvious observation is that the weight of R_1 is much higher than the weight of other rules. The reason for this is that *rdf:type* provides the strongest signal that URI represents an entity as opposed to the strength of the signal for the other query predicates in the MLN.

As mentioned in Section 3, the ML inference engine returns the probability that a query atom is true. To evaluate the inference results we measured the area under precision/recall (AUC PR) curves for all query predicates in the MLN model. The precision/recall curve is computed by varying the threshold above which a query atom is predicted to be true. Table 3 shows the AUC PR scores obtained for every query predicate for different test folds. We note that while the standard deviation for *Entity* is rather high, for all other query predicates it is relatively small.

Rule ID	**Average Weight**
1	55.259 ± 2.255
2	1.433 ± 0.151
3	1.793 ± 0.188
4	4.051 ± 0.121
5	4.161 ± 0.111
6	2.903 ± 0.248
7	3.227 ± 0.086
8	1.140 ± 0.358
9	1.229 ± 0.249

Table 2: Average weights learned for each rule

The MLN seems to perform well on *Entity*, *EntityType*, and *IsInstanceOf*; it is able to extract the signal from the data that allows the inference of these predicates. On the other hand, the MLN does not perform well on *Attribute* and *Has*. This could be attributed to inference chains in the body of the rules that define these predicates. In ML, longer chains mean that the inference engine has a larger space to sample from, which reduces the likelihood of finding the correct answer for the query predicate. Note that *Attribute* and *Has* atoms correspond to schema elements with weak signal from the data (e.g., no one-step inference from *rdf:type* as with rule R1). Previous research has shown that schema inference from instance data is challenging [4]. In LD search, this problem is even harder because of the variability in the data resources in terms of the descriptions they use. In Section 5 we discuss proposals for improving the performance of the MLN.

5. CONCLUSIONS AND FUTURE WORK

Search results over the web of data consist of collections of triples from a range of sources, and typically contain RDF resources that describe real-world entities of different types. In addition, search results often contain assertions that provide additional metadata about the entities, which means that the result of a search is a complex data set that may be difficult to interpret automatically.

With a view to managing this complexity, we have investigated the use of Markov Logic to infer, with uncertainty, which triples in a search result represent types, individuals, attributes and attribute values. This we have done by learning the weights of an MLN, where the associated rules express various hypotheses about the relationships between the triples in a search result and the roles the elements in those triples may be able to play in an entity relationship diagram. The reason for targeting an entity relationship diagram is that we would expect to be able to generate intuitive tabular reports capturing features of search results from such a representation.

The initial results might be considered to be somewhat disappointing. Although we have been able to identify entity types and instance-of relationships from the search domain with high confidence, entity instances and attributes are not being identified reliably by our MLN. Although there are different possible reasons for this (e.g. that the rules for identifying such features could be improved upon), our preferred interpretation is that the rules are plausible, it is simply that the evidence to support them in actual search results is relatively weak. In this context, the evidence may be lacking because, for example, different publishers publish the data in different ways, or a significant fraction of the data retrieved is not directly concerned with the structure of the data in the domain. This in turn suggests that the problem of capturing the domain knowledge in a search result, from the contents of that result, is a difficult one.

Given these challenges, what might be the way ahead? The Markov Logic framework is quite a general one, and we selected it for use with this problem in part because this seems to be an evidence-rich problem, in which the results of the search can be combined with additional information to enable well founded inferences to be drawn. We envisage that the following avenues can be pursued:

- *Additional generic integration rules.* To date, the integration rules have focused on the identification of concepts from entity relationship diagrams using the data from the search result. However, it would be possible to write additional generic rules. For example, none of the current rules make use of the search terms, and no attempt is made to identify duplicate triples or entities across resources retrieved by the search. Additional rules that capture such relationships may be useful in distinguishing between the domain knowledge in a result and associated metadata.
- *Domain-specific integration rules.* Successful applications of Markov Logic, for example in entity resolution or knowledge base construction, have often made use of domain-specific rules. As such, although searches are generic, it would be possible to write rules that know about certain domains, and the widely used terminologies in such domains. For example, rules could be written that are informed by common searches in search logs, or that cover terminologies that are widely used in practice [13]. Another possibility here is writing rules that use mappings between instance level or schema level elements produced by exiting tools.
- *Integration of results of other analyses.* The current rules act directly on the search results. However, it would be possible to run additional analyses on these search results, which in turn could be reflected in rules. For example, analyses could carry out ontology alignment between search results, or could cluster triples based on attribute values. The results of such analyses could then be used as evidence predicates, and included in additional generic or domain-specific integration rules.
- *Integration of feedback.* In Sig.ma, users are able to refine the reports produced by ruling in/out specific sources of data. However, in pay-as-you-go data integration, feedback of different forms can be provided, for example on the correctness or relevance of specific results. Such feedback could be used as evidence by an MLN to inform the inference of results for different query predicates.

6. REFERENCES

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[2] Silvana Castano, Alfio Ferrara, Stefano Montanelli, and Gaia Varese. Ontology and instance matching. In Georgios Paliouras, ConstantineD. Spyropoulos, and George Tsatsaronis, editors, *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution*, volume 6050 of *Lecture Notes in Computer Science*, pages 167–195. Springer Berlin Heidelberg, 2011.

[3] Gong Cheng and Yuzhong Qu. Searching linked objects with falcons: Approach, implementation and evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):49–70, 2009.

[4] Klitos Christodoulou, Norman W Paton, and Alvaro AA Fernandes. Structure inference for linked data sources using clustering. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 60–67. ACM, 2013.

[5] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659. ACM, 2004.

[6] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence.* Morgan & Claypool, 2009.

[7] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[8] Alfio Ferraram, Andriy Nikolov, and François Scharffe. Data linking for the semantic web. *Semantic Web: Ontology and Knowledge Base Enabled Tools, Services, and Applications*, page 169, 2013.

[9] Linyun Fu, Haofen Wang, Wei Jin, and Yong Yu. Towards better understanding and utilizing relations in dbpedia. *Web Intelligence and Agent Systems*, 10(3):291–303, 2012.

[10] Kalpa Gunaratna, Krishnaprasad Thirunarayan, Prateek Jain, Amit Sheth, and Sanjaya Wijeratne. A statistical and schema independent approach to identify equivalent properties on linked data. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 33–40. ACM, 2013.

[11] Cornelia Hedeler, Khalid Belhajjame, Norman W Paton, Alessandro Campi, Alvaro AA Fernandes, and Suzanne M Embury. Dataspaces. In *Search Computing*, pages 114–134. Springer, 2010.

[12] Aidan Hogan. *Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora.* PhD thesis, National University of Ireland,

Galway, 2011.

[13] Aidan Hogan, Jürgen Umbrich, Andreas Harth, Richard Cyganiak, Axel Polleres, and Stefan Decker. An empirical survey of linked data conformance. *J. Web Sem.*, 14:14–44, 2012.

[14] Wei Hu, Jianfeng Chen, and Yuzhong Qu. A self-training approach for resolving object coreference on the semantic web. In *Proceedings of the 20th international conference on World wide web*, pages 87–96. ACM, 2011.

[15] Robert Isele and Christian Bizer. Active learning of expressive linkage rules using genetic programming. *Web Semantics: Science, Services and Agents on the World Wide Web*, 23:2–15, 2013.

[16] Robert Isele, Anja Jentzsch, and Christian Bizer. Silk server-adding missing links while consuming linked data. In *COLD*, 2010.

[17] Hassan Khosravi and Bahareh Bina. A survey on statistical relational learning. In *Advances in Artificial Intelligence*, pages 256–268. Springer, 2010.

[18] Axel-Cyrille Ngonga Ngomo and Sören Auer. Limes: a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2312–2317. AAAI Press, 2011.

[19] Mathias Niepert, Jan Noessner, Christian Meilicke, and Heiner Stuckenschmidt. Probabilistic-logical web data integration. In *Reasoning Web. Semantic Technologies for the Web of Data*, pages 504–533. Springer, 2011.

[20] Feng Niu, Ce Zhang, Christopher Ré, and Jude Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 8(3):42–73, 2012.

[21] Jan Noessner, Mathias Niepert, Christian Meilicke, and Heiner Stuckenschmidt. Leveraging terminological structure for object reconciliation. In *The Semantic Web: Research and Applications*, pages 334–348. Springer, 2010.

[22] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *IJMSO*, 3(1):37–52, 2008.

[23] P. Singla and P. Domingos. Entity resolution with markov logic. In *Data Mining, 2006. ICDM '06. Sixth International Conference on*, pages 572–582, Dec 2006.

[24] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig.ma: Live views on the web of data. *J. Web Semantics*, 8(4):355 – 364, 2010.

[25] Johanna Völker and Mathias Niepert. Statistical schema induction. In *The Semantic Web: Research and Applications*, pages 124–138. Springer, 2011.

[26] Ziqi Zhang, Anna Lisa Gentile, Isabelle Augenstein, Eva Blomqvist, and Fabio Ciravegna. Mining equivalent relations from linked data. In *ACL (2)*, pages 289–293, 2013.

[27] Man Zhu, Zhiqiang Gao, Jeff Z Pan, Yuting Zhao, Ying Xu, and Zhibin Quan. Ontology learning from incomplete semantic web data by belnet. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, pages 761–768. IEEE, 2013.

[28] Nansu Zong, Dong-Hyuk Im, Sungkwon Yang, Hyun Namgoon, and Hong-Gee Kim. Dynamic generation of concepts hierarchies for knowledge discovering in bio-medical linked data sets. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, page 12. ACM, 2012.