Leveraging Data and Structure in Ontology Integration

Octavian Udrea
Univ. of Maryland
udrea@cs.umd.edu

Lise Getoor
Univ. of Maryland
getoor@cs.umd.edu

Renée J. Miller
Univ. of Toronto
miller@cs.toronto.edu

ABSTRACT

There is a great deal of research on ontology integration which makes use of rich logical constraints to reason about the structural and logical alignment of ontologies. There is also considerable work on matching data instances from heterogeneous schema or ontologies. However, little work exploits the fact that ontologies include both data and structure. We aim to close this gap by presenting a new algorithm (ILIADS) that tightly integrates both data matching and logical reasoning to achieve better matching of ontologies. We evaluate our algorithm on a set of 30 pairs of OWL Lite ontologies with the schema and data matchings found by human reviewers. We compare against two systems - the ontology matching tool FCA-merge [28] and the schema matching tool COMA++ [1]. ILIADS shows an average improvement of 25% in quality over FCA-merge and a 11% improvement in recall over COMA++.

Categories and Subject Descriptors: I.2.4 Knowledge Representation Formalisms and Methods: Semantic networks, D.2.12 Interoperability: Data mapping

General Terms: Algorithms, Experimentation.

Keywords: Ontology alignment, schema mapping, data integration, logical inference, statistical inference.

1. INTRODUCTION

Ontologies are becoming more and more plentiful. Ontologies model the structure of data (for example, representing sets of classes and their properties or attributes), the semantics of data (in the form of axioms that express constraints such as inheritance relationships, or constraints on properties), and data instances (often called individuals). To integrate ontologies, we must understand the relationship between structures (classes and properties) and data (individuals) in different ontologies. Furthermore, we must be able to use the semantics of the ontology to model these relationships, and create a coherent and consistent integrated ontology.

SIGMOD'07, June 11–14, 2007, Beijing, China.

Informally, the *ontology alignment problem* can be stated as: given two ontologies O_1 and O_2, determine a set of relationships (for example, subsumption and equivalence relationships) between entities (classes, properties, and individuals) in the two ontologies. This in turn relates to the *integration problem*: determine a new, consistent ontology O (often called the *integration witness*) from the union of O_1 and O_2.

The alignment and integration problems for ontologies are closely related to the problems of schema mapping discovery [18] and schema integration [3, 23]. To align and integrate schemas, approaches usually use some form of schema matching [24] to suggest the existence of possible relationships between structural entities (classes/tables and properties/attributes/elements). However, in the case of ontologies, we additionally need to use data matching (to match individuals modeled in both ontologies) [20]. Matches are used to create (logical) mappings that represent the semantics of relationships (for example, subsumption or equivalence) [13]. A final step is to use the mappings to create an integrated schema [3, 23] or ontology [28]. Both mapping creation and integration make use of logical reasoning over the constraints in the schemas or ontologies [22, 12].

In our work, we combine the statistical learning used for matching with the logical inference traditionally used in mapping discovery and integration. Specifically, we combine a flexible similarity matching algorithm with an incremental logical inference algorithm which reasons about the consequences of learned alignment relationships. The similarity matching algorithm uses lexical, structural and extent information; in addition, it uses an estimate of the logical consequences to determine the similarity of entities (classes, properties, and individuals). Like other similarity-based approaches for matching [17, 5], similarity is propagated incrementally; however our approach is based on a clustering algorithm which considers relationships among sets of equivalent entities, rather than individual pairs of entities.

The logical reasoning and statistical reasoning are tightly integrated: a matching step is followed by a constrained set of logical inferences (constrained to keep the computational expense feasible); the logical inferences inform the matching by influencing the matching statistics. In the end, our approach results in an integrated ontology. Because we do not use complete logical inference at each step, we may produce an inconsistent ontology. However, we show that in practice, our efficient, (but incomplete), inference is sufficient to produce a consistent integration in almost all cases.

In this paper, we make the following contributions. First,

we present a novel integration approach that tightly couples statistical and logical inference. We present and evaluate several alternative strategies for selecting matches and for selecting logical inferences, and show how they can be used collectively in integration. Second, we introduce the ILIADS system (*Integrated Learning In Alignment of Data and Schema*) that integrates ontologies represented in OWL Lite. We pay particular attention to developing a system that is flexible enough to produce high accuracy results for real-world heterogeneous ontologies. Finally, we present an extensive evaluation of our algorithm on a set of 30 pairs of OWL Lite ontologies. We compare against two well-known ontology integration systems – FCA-merge [28] and COMA++ [1]. We show that the integration of logical and statistical inference in ILIADS leads to higher accuracy and recall. We also investigate the correlation between the best way to compute similarity and various structural features of real-world ontologies.

2. A BRIEF OVERVIEW OF OWL LITE

We start by giving a short overview of OWL Lite. The full description of the language syntax and semantics is available at `http://www.w3.org/TR/owl-ref/`. The Web Ontology Language (OWL) was designed to offer support for publishing and sharing ontologies on the World Wide Web. OWL is available in three different dialects; here we focus on the simplest dialect OWL Lite. Although it is the least complex of the three, problems in OWL Lite are quite difficult. For instance, the *conjunctive query answering* problem is 2EXPTIME-complete in the worst case. Despite this negative theoretical result, the majority of real world ontologies are tractable – reasoning problems can be solved in polynomial time.

An OWL Lite ontology consists of several sets of entities: *classes*, *properties*, *individuals*, *data values*, *triples* and *axioms*. We will illustrate each on the two example OWL Lite ontologies in Figure 1. The example in Figure 1(a) is a subset of the disease ontology available from `http://diseaseontology.sourceforge.net/` re-written in OWL Lite; the ontology in Figure 1(b) is a subset of an ontology manually extracted from `www.wrongdiagnosis.com`, a website that provides medical-related information to the general public based on a set of templates (i.e., a schema). The two ontologies use part of the same URI namespace, hence some entities with the same node labels (such as *Condition*) are in fact identical in both. This type of reuse is common practice in the set of 30 pairs of ontologies we have examined. To improve the readability of the figures, we omit the URI namespace prefixes.

In OWL Lite, *classes* are used to model sets of entities with common characteristics. For instance, *BacterialInfection* models the set of medical conditions that are caused by bacteria and manifest themselves as infections. Similarly, the *Pathogen* class models the set of organisms that cause diseases. OWL Lite provides a number of ways in which a class can be defined other than by simply declaring it; among these, a class can be defined as an intersection of other classes or based on a *restriction* (e.g., a value or cardinality restriction). We use $\mathcal{C}$ to denote the set of classes in an ontology.

Individuals (or *instances*) are elements of the set modeled by a class. For instance, the fact that *D. Salmon* is an individual of type *Person* is depicted graphically by the edge *(D.Salmon, rdf:type, Person)* in Figure 1(b). rdf:type is a reserved keyword in OWL which denotes that an individual is an instance of given class. Note that in Figure 1, individuals are represented by square nodes and classes by round nodes. The set of instances belonging to a class is called the *extension* of that class. For example, in Figure 1(a), the extension of class *Person* is {*TheodorEsterich, DanielElmerSalmon*}. We use $\mathcal{I}$ to denote the set of individuals in an ontology. For a given class $c \in \mathcal{C}$ use $\epsilon(c)$ to denote the set of individuals in the extension of c.

Data values are values of one of the primitive types allowed by XML Schema[1]. In Figure 1(a), *73,000* and *40,000* are examples of data values of type *unsigned integer*. We use $\mathcal{T}$ to denote the set of all primitive data types (e.g. *xsd:integer*, *xsd:string*, etc.) and $\mathcal{D}$ to denote the set of all possible data values of types in $\mathcal{T}$.

A *property* is used to specify relationships between two individuals or between an individual and a data value. The set of properties in Figure 1 includes among others *discoveredBy*, *averageCases* and *associatedWith*. The edge *(E-ColiPoisoning, riskFactor, Gastroenteritis)* states that the individual *Gastroenteritis* of class *BacterialInfection* is a risk factor of the individual *E-ColiPoisoning*. Similarly, the edge *(E-ColiPoisoning, averageCases, 73,000)* states that there are approximately 73,000 cases of E-Coli food poisoning in the US every year. Note that the meaning of a property may not always be evident from its label (in this case, a more appropriate label would be *annualAverageCasesUS*). Ontology creators generally address this problem by giving comments on what a property or class means. In this paper, do not make use of such metadata. We use $\mathcal{P}$ to denote the set of properties in an ontology. This includes predefined OWL properties such as owl:sameAs, which states the two individuals are semantically identical, and owl:differentFrom which states two individuals are different.

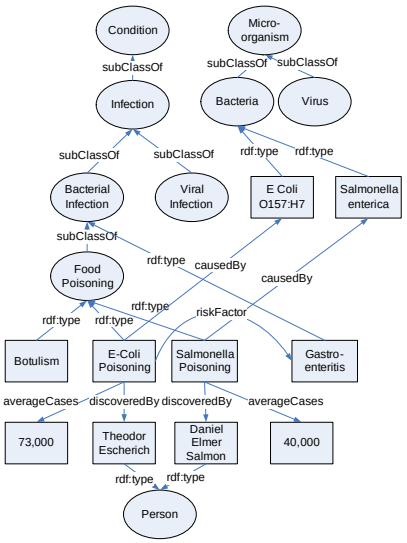
All information in OWL is generally expressed in the form of a *triple*. In this discussion, we limit the meaning of the term *triple* (or *fact*) to include only relationships between an individual and a class, an individual or a data value. Examples of triples include *(Botulism, rdf:type, FoodPoisoning)*, *(E-ColiPoisoning, averageCases, 73000)* and *(E-ColiPosoning, discoveredBy, TheodorEscherich)*, but not *(ViralInfection, rdfs:subClassOf, Infection)*[2]. In this discussion, the latter will be categorized as an *axiom*. We will use $\mathcal{F}$ to denote the set of triples in an ontology. For a given property $p \in \mathcal{P}$, we define the extension of p as $\mu(p) = \{(X,Y) | X \in \mathcal{I}, Y \in \mathcal{I} \cup \mathcal{D} \cup \mathcal{C}, (X,p,Y) \in \mathcal{F}\}$. We will refer to $|\mathcal{F}|$ as the *size of an ontology*.

Axioms provide much of the expressive power of OWL Lite. The axioms for our two ontologies are shown at the bottom of Figure 1. As we mentioned before, the two ontologies share parts of their schema, including the axiom *(discoveredBy, owl:inverseOf, discovered)*. This axiom states that the *discoveredBy* property is an inverse of *discovered* – which means that for each triple *(X, discoveredBy, Y)* in the ontology, we can infer *(Y, discovered, X)*. We present a few types of axioms permitted in OWL Lite (more details are provided in Section 5):

- *(c, rdfs:subClassOf, c')* states that class c is a subclass

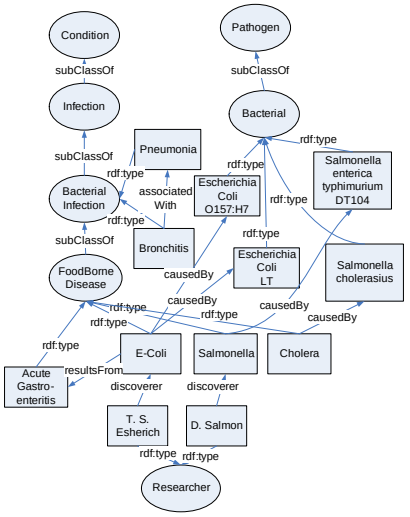
[1] `http://www.w3.org/XML/Schema`.

[2] In Figure 1, we omit the rdfs prefix from rdfs:subClassOf for the sake of readability.



$(discoveredBy, \text{owl:inverseOf}, discoverer)$
$(discoveredBy, \text{owl:type}, \text{owl:FunctionalProperty})$

(a)



$(discoveredBy, \text{owl:inverseOf}, discoverer)$
$(associatedWith, \text{owl:type}, \text{owl:TransitiveProperty})$
$(resultsFrom, \text{rdfs:subPropertyOf}, associatedWith)$

(b)

Figure 1: Two example OWL Lite ontologies

of class c'. From this, we can infer that $\epsilon(c) \subseteq \epsilon(c')$.

- *(p, owl:type, owl:FunctionalProperty)* states that p is a functional property, hence $(X,Y),(X,Z) \in \mu(p) \Rightarrow Y = Z$.
- *(p, owl:inverseOf, p′)* states that p is an inverse of p', hence $(X,Y) \in \mu(p) \Rightarrow (Y,X) \in \mu(p')$ and $(X',Y') \in \mu(p') \Rightarrow (Y',X') \in \mu(p)$.

In this paper, we address the problem of integrating two consistent OWL-Lite ontologies O_1, O_2 into a single, **consistent** output ontology O. Therefore, from the various reasoning problems in OWL-Lite[3], we are particularly interested in the problem of determining whether an ontology is **consistent**. Horrocks et al. [10] have shown that OWL-Lite is equivalent to the logic $\mathcal{SHIF}(\mathrm{D})$ and provided a formal model theoretic semantics for the language. Formally, an OWL-Lite ontology is **consistent** if there exists a satisfying interpretation for the ontology. For instance, an ontology from which we can infer that *(x, owl:differentFrom,y)* and *(x, owl:sameAs, y)* at the same time cannot have a satisfying interpretation. We use the Pellet[27] implementation of the *tableaux algorithm* of Horocks et al.[11] to verify whether the result of integrating the input ontologies O_1, O_2 is consistent.

3. ONTOLOGY INTEGRATION

Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two OWL Lite ontologies. The alignment problem is that of finding a set of axioms and facts $\mathcal{A}^*$ on $\mathcal{O}_1 \cup \mathcal{O}_2$ that link entities in $\mathcal{O}_1$ and $\mathcal{O}_2$. Specifically, we are looking for axioms that specify subsumption and equivalence between classes and properties and for facts that specify equality between individuals. In addition, the integration problem requires that adding the axioms in A^* to $\mathcal{O}_1 \cup \mathcal{O}_2$ produces a consistent ontology.

[3]See `http://www.w3.org/Submission/owl11-tractable/` for a complete list.

EXAMPLE 1 (ALIGNMENT). *For the ontologies depicted in Figure 1(a) and (b), the following set of axioms and facts gives a possible (but clearly incomplete) integration:*
$(FoodPoisoning, \text{rdfs:subClassOf}, FoodBorneDisease)$
$(Bacterial, \text{owl:equivalentClass}, Bacteria)$
$(resultsFrom, \text{rdfs:subPropertyOf}, riskFactor)$
$(E\text{-}ColiPoisoning, \text{owl:sameAs}, E\text{-}Coli)$
$(EscherichiaColiO157:H7, \text{owl:sameAs}, EColiO157:H7)$
$(TheodorEscherich, \text{owl:sameAs}, T.S.Esherich)$

We introduce a powerful framework that combines the flexibility of approximate matching for finding equivalence and subsumption candidates with the power of logical inference to determine the consequences of the new alignments. Like many systems, our matching algorithms use a combination of lexical, structural and extent information. However our algorithm is unique in the tight integration of the statistical matching algorithm with logical inference. This leads to a system which uses the full power of data and schema integration to construct consistent integrated ontologies.

Our *Integrated Learning In Alignment of Data and Schema* (ILIADS) algorithm is based on an interleaving of a hierarchical clustering algorithm with an incremental logical inference algorithm. Clustering entities creates new relationships among the entities; these new relationships may have logical consequences in OWL Lite. For example, consider the union of the two ontologies in Figure 1. Adding the relationship *(EColi-Poisoning, owl:sameAs, E-Coli)*, in conjunction with the existing axioms *(discoveredBy, owl:inverseOf, discoverer)* and *(discoveredBy, owl:type, owl:FunctionalProperty)* logically implies that *(TheodorEscherich, owl:sameAs, T.S. Escherich)*. Furthermore, the results of the logical inference process after adding an integration axiom need to be used to update the similarity score of other relationships. Figure 2 shows an outline of the main ontology integration algorithm. In the algorithm, we repeatedly find candidate clusters of similar entities, add an equivalence or subsumption relationship

Algorithm ILIADS(O_1, O_2)

Input: Consistent ontologies O_1 and O_2.
Output: Alignment A^* of O_1 and O_2 such that the integration of O_1 and O_2 under A^* is consistent.

1: Compute $O \leftarrow O_1 \cup O_2$, $A^* \leftarrow \emptyset$
2: Initialize each cluster c with an $e \in \mathcal{C} \cup \mathcal{P} \cup \mathcal{I}$
3: Merge clusters containing equivalent entities and add appropriate equivalence axioms to $\mathcal{O}$
4: **repeat**
5: Heuristically select a type of clusters (from {classes, instances, properties})
6: Let G be the set of all clusters of that type
7: **for** $(c, c') \in G \times G$ **do**
8: Determine candidate relationship $a_{(c,c')}$
9: Perform incremental inference, $O_{c,c'} \leftarrow IncrementalInference(O, a_{c,c'})$
10: Compute $sim_{inf}(c, c')$ based on $O_{c,c'}$.
11: **end for**
12: Choose $a_{c,c'}$ with the highest $sim_{inf}(c, c')$ score.
13: $O \leftarrow O_{c,c'}$, $A^* \leftarrow A^* \cup \{a_{c,c'}\}$
14: **until** no more candidate clusters
15: **return** A^*

Figure 2: The main ILIADS algorithm

between the sets of entities, determine the logical consequences of the new relationship and update the similarities between remaining clusters. Note that the relationship between candidates for alignment (line 8) is expressed as an OWL axiom (for instance, equivalence between two properties is expressed as (p, owl:equivalentProperty, p')) or as a owl:sameAs fact (only for equivalences between instances) and then added to the ontology. We provide more detail about the process of creating axioms and facts from alignment candidates in Section 4.2. In the following subsections, we give details on the two main components: the statistical matching and the incremental logical inference.

4. CLUSTERING ALGORITHM

Our statistical inference uses a hierarchical agglomerative clustering algorithm to select candidate clusters to link in the ontologies. During this process, we form separate clusterings over the set of classes, over the set of individuals, and over the set of properties. Notably, the clustering process for all entities (classes, individuals and properties), is influenced by the current clustering for other types of entities. For example, in determining whether to merge two sets (clusters) of individuals, we consider not only their properties, but also possibly inferences we can make about those properties based on the current clustering of properties (and similarly for classes).

An important component of any clustering algorithm is the computation of the similarity between clusters. We start by describing the similarity measures for entities and sets of entities in Section 4.1. Then we describe our method for choosing between subsumption and equivalence of candidates clusters in Section 4.2.

4.1 Similarity measures

Deciding on a set of integration axioms and facts means we have to express how "similar" we believe a pair of candidate entities are based on their lexical, semantic and graph-structural information. The similarity score is based on three components: (i) the *lexical score* handles lexical similarities between names (or URIs), (ii) the *structural score* models how similar the neighboring nodes of the two entities are and (iii) the *extensional score* models similarity between the ϵ and μ values of entities being compared. For a pair of entities e_1, e_2 we express the similarity score as a linear combination of these three components:

$$sim(e_1, e_2) = \lambda_x sim_{lex}(e_1, e_2) + \lambda_s sim_{struct}(e_1, e_2) + \lambda_e sim_{ext}(e_1, e_2)$$

We do not assume that the λ parameters should be the same for classes, properties and instances; we will denote the λ parameters for classes, properties and individuals respectively with a c, p, i upper index. This gives us a set of nine parameters for measuring similarity; here, we consider $\lambda \in [0, 1]$. We will discuss how the λ are chosen in Section 7.

4.1.1 Lexical Similarity

There are a variety of string similarity measures that are commonly used [4]; we used Jaro-Winkler[29] because empirically it provides the best answer quality for our dataset. In addition, for low Jaro-Winkler scores, WordNet is used to determine potential synonymity between entity names; we then assign a similarity score based on which synonym set the pair belongs to (synsets are ordered in WordNet by frequency of use). For class and properties, we use the maximum of Jaro-Winkler and WordNet. For instances, WordNet similarity is ineffective, because the names are typically not part of the WordNet lexicon.

4.1.2 Structural Similarity

To compute the structural similarity of two classes c_1, c_2, we consider the Jaccard distance between their sets of neighbors in the rdfs:subClassOf hierarchy. For two sets S_1, S_2, the Jaccard distance is defined as $Jaccard(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$ In our experimental evaluation, we looked at subclasses and superclasses at various distances from c_1, c_2, and obtained the best answer quality for neighborhoods of distance[4] 2. Neighborhoods of distance 1 do not account for potential differences in the level of specificity in the two ontologies, whereas neighborhoods of distance 3 or more contain elements too far removed from the target classes. We also tried using the Jaccard distance on the sets of properties incident to the two classes (for instance, in Figure 1(a), *riskFactor* is incident to *FoodPoisoning*, since it is incident to *E-ColiPoisoning*, an instance of *FoodPoisoning*). This latter measure did not yield a good answer quality during the experimental evaluation.

For properties, the structural similarity is based on their rdfs:subPropertyOf neighborhood. In this case, we found experimentally that axioms about properties played an important part in determining similarity. As described in Section 2, a property can have the features *transitive*, *functional*, *symmetric* or *inverse functional*. Let H_p be the set of features of a property p. We define $sim_{features}(p_1, p_2) = Jaccard(H_{p_1}, H_{p_2})$. We multiply the Jaccard distance between the rdfs:subPropertyOf neighborhoods with $sim_{features}$ to obtain the similarity score between two properties.

Finally, to compute the structural similarity for two instances $e_1, e_2 \in \mathcal{I}$, we compute the Jaccard distance between the sets of pairs (p, Y) such that $(e_i\ p\ Y) \in \mathcal{F}$. Experimentally, we found that using structural similarity, we missed

[4]The maximum number of edges between the target node and a node in the neighborhood.

Table 1: Summary of the component similarity measures

	sim_{lex}	sim_{struct}	sim_{ext}
Classes	Jaro-Winkler/ Wordnet.	Jaccard on rdfs:subClassOf neighbors.	Jaccard on set of instances.
Properties	Jaro-Winkler/ Wordnet.	Jaccard on rdfs:subPropertyOf neighbors multiplied by $sim_{features}$.	Jaccard on set of (X, Y) pairs in property extension.
Instances	Jaro-Winkler.	Jaccard on sets of incident (p, Y) pairs.	0

very few alignments between proper names (see Section 7).

4.1.3 Extensional Similarity

The extensional similarity for classes and properties is also computed using Jaccard similarity. For example, the extensional similarity of two classes c_1 and c_2 is $Jaccard(\epsilon(c_1), \epsilon(c_2))$. The extensional similarity of properties is defined in terms of their μ values. There is no extensional similarity for instances. Table 1 summarizes the component similarity measures used for classes, properties and instances.

4.1.4 Cluster Similarity

Given a similarity score between two entities, it can be extended to clusters of entities C_1 and C_2 in one of the following standard ways:

1. **Single link**: $sim(C_1, C_2) = \min_{e_1 \in C_1, e_2 \in C_2} (sim(e_1, e_2))$.
2. **Complete link**: $sim(C_1, C_2) = \max_{e_1 \in C_1, e_2 \in E_2} (sim(e_1, e_2))$
3. **Average link**: $sim(C_1, C_2) = \frac{\sum_{e_1 \in C_1} \sum_{e_1 \in C_2} sim(e_1, e_2)}{|C_1| \times |C_2|}$

Note that the cluster similarity measures are linear in the product of $|C_1| \cdot |C_2|$. We have found empirically that cluster sizes in our ontology dataset are relatively small, hence the computation of cluster similarity scores does not have a great impact on the running time of the algorithm.

4.2 Determining Cluster Relationships

Whenever we consider a pair of classes or properties for clustering, we need to decide whether they should be in a subsumption or an equivalence relationship (line 8, Figure 2). We use a simple decision procedure that we found effective in practice to determine the relationship. We will exemplify the decision process on classes, but it applies to properties in the same form. In the following, let $c_1, c_2 \in \mathcal{C}$ be two classes. Let $\lambda_r \in [0, 1]$ be an arbitrary, but fixed threshold value.

1. If $\frac{|\epsilon(c_1) - \epsilon(c_2)|}{|\epsilon(c_1)|} < \lambda_r$ and $\frac{|\epsilon(c_2) - \epsilon(c_1)|}{|\epsilon(c_2)|} \geq \lambda_r$, then consider the axiom (c_1, rdfs:subClassOf, c_2).
2. If $\frac{|\epsilon(c_1) - \epsilon(c_2)|}{|\epsilon(c_1)|} \geq \lambda_r$ and $\frac{|\epsilon(c_2) - \epsilon(c_1)|}{|\epsilon(c_2)|} < \lambda_r$, then consider the axiom (c_2, rdfs:subClassOf, c_1).
3. Otherwise consider the axiom (c_1, owl:equivalentClass, c_2).

Intuitively, this set of rules states that if the set of instances of $\epsilon(c_1)$ is "almost" a subset of the set of instances $\epsilon(c_2)$, then we should consider a subsumption relation (c_1, rdfs:subClassOf, c_2). The threshold value λ_r determines how many of the instances of c_1 should also be instances of c_2. In our experiments we have determined that a value $\lambda_r = 0.2$ provides the best answer quality. Note that this method can also be applied to clusters of classes $C \subseteq \mathcal{C}$ by defining $\epsilon(C) = \cup_{c \in C} \epsilon(c)$ and similarly to clusters of properties.

EXAMPLE 2. *Consider the two ontologies in Figure 1 and the candidate pair* (FoodPoisoning, FoodBorneDisease). *Then*

$$\begin{aligned} |\epsilon(FoodPoisoning) - \epsilon(FoodBorneDisease)| &= 1 \\ |\epsilon(FoodBorneDisease) - \epsilon(FoodPoisoning)| &= 2 \\ |\epsilon(FoodPoisoning)| &= 3 \\ |\epsilon(FoodBorneDisease)| &= 4 \end{aligned}$$

If $\lambda_r = .5$ *we would consider the relationship* (FoodPoisoning, rdfs:subClassOf, FoodBorneDisease). *However, if* $\lambda_r = .7$, *we would consider the relationship* (FoodPoisoning, owl:equivalentClass, FoodBorneDisease)

5. INCREMENTAL LOGICAL INFERENCE

In ILIADS, to evaluate whether or not to merge two clusters, we need to test the effect such a merge would have on the alignment. We do this by modeling the merged cluster using a new axiom in the ontology (for example, a new axiom asserting the equivalence of classes). Hence, we need to perform inference, but we also need to control the order in which axioms are applied (something that most out-of-the-box OWL reasoners do not permit).

Therefore, instead of using an existing reasoner, we introduce an incremental logical inference algorithm based on the tableaux method which incrementally goes through the inference process up to a maximum number of steps. We point out that all OWL reasoning algorithms, including the one used by ILIADS are variations of the tableaux method. The two key differences between inference in ILIADS and existing reasoners such as Pellet are that our algorithm (i) only executes a constant number N of inference rules and (ii) it uses heuristics to select from the set of applicable rules a single rule to apply at each step. The analytical tableaux method can be applied to OWL Lite by looking at axioms as inference rules. For example, the owl:FunctionalProperty rule could be written as:

$$\frac{\{(X,p,Y),(X,p,Y'),(p,\text{owl:type},\text{owl:FunctionalProperty})\}}{\{(Y,\text{owl:sameAs},Y')\}}$$

The other inference rules are:

$$\frac{\{(c,\text{rdfs:subClassOf},d),(X,\text{rdf:type},c)\}}{\{(X,\text{rdf:type},d)\}} \qquad \frac{\{(c,\text{owl:equivalentClass},d),(X,\text{rdf:type},c)\}}{\{(X,\text{rdf:type},d)\}}$$

$$\frac{\{(p,\text{owl:equivalentProperty},q),(X,p,Y)\}}{\{(X,q,Y)\}} \qquad \frac{\{(X,p,Y),(p,\text{rdfs:subPropertyOf},q)\}}{\{(X,q,Y)\}}$$

$$\frac{\{(X,p,Y),(Y,p,Z),(p,\text{owl:type},\text{owl:TransitiveProperty})\}}{\{(X,p,Z)\}}$$

$$\frac{\{(X,p,Y),(X',p,Y),(p,\text{owl:type},\text{owl:InverseFunctionalProperty})\}}{\{(X,\text{owl:sameAs},X')\}}$$

$$\frac{\{(X,p,Y),(p,\text{owl:type},\text{owl:SymmetricProperty})\}}{\{(Y,p,X)\}}$$

The algorithm is described in Figure 3. For reasons of space, we omit the set of rules by which we check for incon-

sistencies. A simple example is:

$$\frac{\{(X,\text{owl:sameAs},Y),(X,\text{owl:differentFrom},Y)\}}{\bot}$$

Other possible reasons for inconsistency in OWL Lite include violations of class restrictions and unsatisfiable concepts. Note that we only check for inconsistencies that occur during a limited number of inference steps, hence the approach is not sound.

EXAMPLE 3. *In the ontology in Figure 1(a), from the triple* (Botulism, rdf:type, FoodPoisoning), *by applying the axiom* (FoodPoisoning, rdfs:subClassOf, BacterialInfection) *we can infer that* (Botulism, rdf:type, BacterialInfection). *Similarly, from the triple* (E-Coli, resultsFrom, AcuteGastroenteritis), *and the axiom* (resultsFrom, rdfs:subPropertyOf, associatedWith), *we infer that* (E-Coli, associatedWith, AcuteGastroenteritis). *Note that in both cases above, we increased our knowledge about the class and property extensions ϵ and μ. In the former case, we discovered that* Botulism $\in \epsilon$(BacterialInfection) *and in the latter we discovered that* (E-Coli Poisoning, Gastroenteritis) $\in \mu$(associatedWith).

Algorithm IncrementalInference(O, a)
Input:
O — the ontology we perform inference on.
a — the current candidate relationship (e.g., (C_1, owl:equivalentClass, C_2)).
N — the number of inference steps to take.
Output:
Ontology O' obtained from $O \cup \{a\}$ after N inference steps
1: $O.\mathcal{A} \leftarrow O.\mathcal{A} \cup \{a\}$, $O' \leftarrow O$
2: **for** $i = 1$ to N **do**
3: Heuristically select an axiom $a' \in A$, $a' \neq a$
4: Apply the rule for a' on O'
5: **if** there is an inconsistency **then**
6: **return** FAIL
7: **end if**
8: **end for**
9: **return** O'

Figure 3: The Incremental Logical Inference algorithm

Let O'_i denote the ontology O' at the beginning of step i of the algorithm. For most axioms, each inference step i is linear in the size of O'_i. Also, the size of O'_{i+1} is at most double the size of O'_i. However, in order to apply the rules for owl:TransitiveProperty, owl:FunctionalProperty and owl:InverseFunctionalProperty axioms, we need to iterate through all pairs of triples in $\mathcal{F} \times \mathcal{F}$. Step i of the algorithm is therefore quadratic in the size of O'_i. Assuming that the initial ontology is consistent, the incremental algorithm above is also *complete* for $N \to \infty$. Note that the number of facts in the ontology is increasing monotonically, and the number of facts that can be inferred with the entities in the initial ontology is finite. Hence, for large enough values of N, the $IncrementalInference$ algorithm will eventually reach a fixpoint at step i, when independently of the axiom a' chosen on line 4, $O'_{i+1} = O'_i$.

In conjunction with the main algorithm in Figure 2, we observe the following:

1. $IncrementalInference$ always increases the number of facts in the ontology, but keeps the number of axioms constant.
2. The clustering algorithm:
   (a) Decreases the number of clusters at each iteration.
   (b) Adds axioms or facts to the ontology as the result of merging two clusters.

Since both the number of facts and the number of axioms that can be expressed with the entities and data values in the ontology is finite, the main algorithm in Figure 2 is guaranteed to terminate.

If an inconsistency is found while computing the new values of ϵ, μ, we return FAIL. Note that this approach is not sound, since we do not check consistency for the entire ontology (a very time consuming process). However, in our experimental evaluation the algorithm produced inconsistent ontologies less than .5% of the time.

5.1 Logical Inference Similarity

After computing the results of $IncrementalInference$, the main algorithm in Figure 2 computes the *logical inference similarity* score of the candidate clusters. This value is obtained by multiplying the initial similarity score between the two clusters with a factor that summarizes the effects of the inference process. Let Q be the set of entity pairs that have become equivalent after executing $IncrementalInference$. The inference similarity factor is $f = \prod_{(e,e') \in Q} \frac{sim(e,e')}{1-sim(e,e')}$.

Let c, c' be a pair of clusters with similarity (before inference) $sim(c, c')$. Then the logical inference similarity score for c, c' is defined as $sim_{inf}(c, c') = min(f \cdot sim(c, c'), 1)$. The intuition behind computing this score is the following: if we can logically infer from a candidate relationship $a_{c,c'}$ equivalences that are probable ($\frac{sim(e,e')}{1-sim(e,e')} > 1$), we are more inclined to believe $a_{c,c'}$ is a good alignment. On the other hand, if $a_{c,c'}$ implies equivalences that are unlikely ($\frac{sim(e,e')}{1-sim(e,e')} < 1$), its similarity score will decrease. Note that $f \cdot sim(c, c')$ is not guaranteed to be lower than 1 because f cannot be normalized w.r.t. $sim(c, c')$ without reaching a fixpoint in $IncrementalInference$; to normalize f, we need to know **all** possible consequences of adding $a_{c,c'}$ to the ontology.

EXAMPLE 4 (INFERENCE SIMILARITY). *Consider the two ontologies in Figure 1. Assume that the candidate axiom is* (E-ColiPoisoning, owl:sameAs, E-Coli), *the initial similarity score is .5 and $N = 2$. Also assume that the initial similarity score* sim(TheodorEscherich, T.S.Escherich) = .6. *The axiom selection process will heuristically select an axiom for each of the two steps of inference (the process is detailed in Section 6). For now let's assume that the axiom in the first step is* (discoveredBy, owl:inverseOf, discoverer). *Applying this axiom will add the pair* (E-Coli, discoveredBy, T.S.Escherich) *to μ(discoveredBy) (among others). Note that no new equivalence relations appear in this step, hence we move to the second inference step and the axiom* (discoveredBy, owl:type, owl:FunctionalProperty). *Applying this axiom to* (E-ColiPoisoning, owl:sameAs, E-Coli), (E-ColiPoisoning, discoveredBy, TheodorEscherich) *and* (E-Coli, discoveredBy, T.S.Escherich), *we obtain* (Theodor Escherich, owl:sameAs, T.S.Escherich). *The similarity score for the two* Escherich *nodes is $\frac{.6}{.4} = 1.5$, hence the new score will become $.5 \cdot 1.5 = .75$. The similarity between the two representations of the name of Theodor Escherich strength-*

ens our belief in the equivalence between the two forms of the E-coli food poisoning condition.

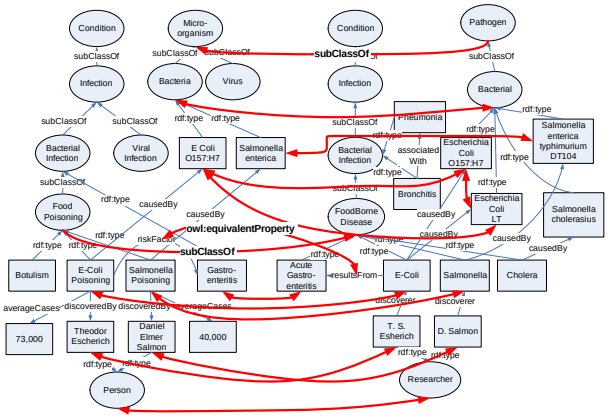


Figure 4: Alignment results for the two example OWL Lite ontologies.

6. HEURISTIC SELECTION POLICIES

The ILIADS algorithm (Figure 2) contains two heuristic steps that we have not yet explained. First, in line 9, when we perform $IncrementalInference$, we are given a new candidate relationship $a_{(c,c')}$. For the inference, we want to select an axiom a', apply N steps of inference and return the updated ontology (called $\mathcal{O}_{(c,c')}$ in Figure 2). Our guiding heuristic principle will be to select an axiom that produces equivalence relationships to be used in the inference similarity score. Second, in the clustering process (lines 5–6 of Figure 2), we select a subset of clusters (all clusters of type class, or all clusters of type individual, or all clusters of type property) from which a candidate pair to be merged will be chosen. We explain these two heuristic steps next.

6.1 Axiom Selection

Selecting an axiom that maximizes the chances of obtaining equivalence relationships requires two steps: (i) selecting a subgroup of candidate axioms A_0 and (ii) selecting a candidate from the group in the previous step. We have used the following rules to select groups of axioms:

(AR1) If the relationship a involves classes C, C', then add to A_0 any axiom that either (i) involves C or C' OR (ii) involves any property p adjacent to an individual $X \in \epsilon(C) \cup \epsilon(C')$. Informally, this rule states that we should select only those axioms that are related to C or C' either directly or through their extensions.

(AR2) If the relationship a involves properties p, p' then add to A_0 any axiom that contains either p or p'.

(AR3) If the relationship a involves individuals X, X', then add to A_0 any axiom the contains: (i) a class C such that $X \in \epsilon(C)$ or $X' \in \epsilon(C)$ OR (ii) a property p that is incident to either X or X'.

To select one axiom from A_0 we have defined the following *axiom selection policies*:

1. **Random**. Simply select a random axiom.
2. **Properties first**. Select at random from axioms involving properties.
3. **Classes first**. Select at random from axioms involving classes.
4. **Transitive/Inverse/Functional first**. Select at random from axioms about functional, transitive and inverse properties. We remind the reader that we interpret a functional property *(p, owl:type, owl:FunctionalProperty)* by $(X, Y), (X, Z) \in \mu(p) \Rightarrow Y = Z$. We choose functional properties first because their direct effect is to "produce" new equivalences between individuals as a logical consequence. Transitive and inverse properties can be used effectively to produce the information necessary in order to apply a functional property axiom.

6.2 Cluster Type Selection

In the main ILIADS algorithm (Figure 2), lines 5–6)) we heuristically select a group of clusters consisting of all class clusters, all instance clusters or all property clusters based on one of the following policies:

1. **Random** Select one of the three groups at random.
2. **Weighted random** Select one of the three groups with probability proportional to the size of each group.
3. **Classes first** Select the classes group until no good candidates (with similarity above λ_t) are found.
4. **Individuals first** Similar to the **classes first**, but selects the group of individuals.
5. **Alternate** Alternate the selection between the three groups.

Note that there is no **Properties first** policy. We determined experimentally that to match a pair of properties p_1, p_2, we first need to have a good alignment of classes and individuals adjacent to p_1, p_2, hence in general leaving property alignments for later phases is a good policy.

EXAMPLE 5 (ALIGNMENT). *Figure 4 contains the alignment of the two ontologies in Figure 1(a) and (b) for* $\lambda_x^c = .2$, $\lambda_x^i = .4$, $\lambda_x^p = .1$, $\lambda_s^c = .5$, $\lambda_s^i = .6$, $\lambda_s^p = .4$, $\lambda_e^c = .3$, $\lambda_e^p = .5$, $\lambda_t = .7$, $\lambda_r = .2$, $N = 5$, *using the* ***Transitive/Inverse/Functional first*** *strategy for selecting inference axioms and the* ***Alternate*** *strategy for selecting cluster groups. Note that the classes* Condition, Infection, BacterialInfection *were common to the two schemas. The bold double-arrow lines point to entities that were clustered under equivalence axioms during the alignment. Note that not all alignments are completely accurate – for instance,* (resultsFrom, owl:equivalentProperty, riskFactor). *However, as we mentioned before, the intended meaning of properties cannot be determined from the ontology alone in most cases. As we will see in our experimental evaluation in Section 7, matching properties is a task of considerable difficulty.*

7. EXPERIMENTAL EVALUATION

We evaluated ILIADS experimentally on 30 pairs of OWL Lite ontologies obtained primarily from ontology libraries[5]

[5]`http://www.fb10.uni-bremen.de/anglistik/langpro/webspace/jb/info-pages/ontology/ontology-root.htm` at the University of Bremen contains a very good index of ontology sites and related projects.

such as www.daml.org, www.schemaweb.info and protege.cim3.net and a few ontology-related project websites such as diseaseontology.sourceforge.net and www.geneontology.org. A few characteristics of the dataset are given in Table 2. Four human reviewers were asked to manually integrate the 30 pairs of ontologies in the following way:

1. At the beginning, each pair of ontologies $(\mathcal{O}_1, \mathcal{O}_2)$ was assigned to a randomly chosen reviewer, who was asked to produce an integration $\mathcal{O}$ of the two ontologies.
2. In the next step, the pair $(\mathcal{O}_1, \mathcal{O}_2)$ and the integration result $\mathcal{O}$ were assigned to a new reviewer who was asked to improve $\mathcal{O}$.
3. The process was repeated until each pair was seen once by every reviewer.

Table 2: Dataset statistics

	# inst.	# cls.	# prop.	# triples	# axioms
Min	45	57	14	194	74
Max	13451	689	97	21345	1235
Average	1984	541	64	1254	789
StDev	4534	149	47	5165	176

The integration provided by the reviewers was used as ground truth when measuring precision and recall. There was not a lot of direct overlap among the ontologies; on average, 3% of the entities in each pair of ontologies had the same name and URI in both ontologies. On average, the ground truth contains 379 axioms, from which approximately 47% are class subsumption, 26% class equivalences, 19% property subsumption and 8% property equivalence axioms; the average number of owl:sameAs facts in the ground truth is 287. One significant issue when computing these measures is comparing equivalence against subsumption axioms. Assume for instance that the ground truth contains *(c, owl:equivalentClass, c′)* for two classes c, c', whereas the algorithm finds that *(c, rdfs:subClassOf, c′)*. In order to handle such cases we split each equivalence axiom into two subsumption axioms. For instance, in the previous example we replace *(c, owl:equivalentClass, c′)* by {*(c, rdfs:subClassOf, c′)*, *(c′, rdfs:subClassOf, c)*}.

The implementation of ILIADS consists of approximately 5200 lines of Java code. We use the Pellet OWL API [27] to parse OWL Lite ontologies and check whether the resulting integration is consistent at the end of each experiment. ILIADS produced inconsistencies in less than .5% of the runs. All experiments were performed on a Pentium IV 3Ghz desktop machine with 1GB of RAM under SuSE Linux 9.3. The running times, as well as precision and recall values are reported as an average over 5 independent runs[6].

We compared our results to those of two leading systems: FCA-merge [28] and COMA++ [1]. FCA-merge (from Formal Concept Analysis) is a human-aided tool for the bottom-up merging of ontologies. The algorithm uses a corpus of natural language documents relevant to both ontologies, from which it extracts two formal contexts indicating which ontology concepts appear in which document. The algorithm then merges the two contexts and builds a formal concept lattice with the same degree of detail as the source ontologies. The concept lattice is then used to derive the merged ontology. For each ontology pair, we automatically collected a corpus of approximately 100 articles from the US and European press on topics present in the two ontologies; this corpus was then used as an input to the algorithm. FCA-merge produces alignment suggestions which are then processed by an ontology engineer. We automatically select the subset of suggestions that produces the highest F1 answer quality. COMA++ is a tool that implements multiple match strategies to align relational schemas, XML and OWL. The tool uses a generic data model that supports schemas written in different languages, as well as a repository of previous match results. Multiple matching strategies are employed, among which fragment-based matching (where we decompose a large problem into smaller matching problems) and reuse-oriented matching (which uses previous match results). COMA++ also provides a comprehensive graphical user interface, which proved invaluable during our evaluation.

[6]Some of axiom and cluster selection policies use randomization.

Table 3: ILIADS **optimal parameter values and ranges**

	λ_x^c	λ_x^i	λ_x^p	λ_s^c	λ_s^i	λ_s^p	λ_e^c	λ_e^p	λ_t	λ_r
Avg.	.2	.4	.1	.5	.6	.4	.3	.5	.7	.2
Min	.15	.4	0	.3	.45	.35	.2	.35	.65	.2
Max	.25	.45	.1	.65	.7	.5	.35	.65	.7	.2

The objectives of our experimental evaluation are threefold. First, we investigate the optimal choice of λ parameters, heuristic selection policies and the number of inference steps that produce the best F1 quality w.r.t. the ground truth. Second, we compare the precision, recall and F1 quality of ILIADS with those of FCA-merge and COMA++ and investigate the extent to which the important gains in F1 quality in ILIADS are due to (a) the tight integration of statistical and logical inference and (b) the interplay between schema and data integration. Third, we investigate the ILIADS λ parameters and how they correlate with the structural properties of the input ontologies.

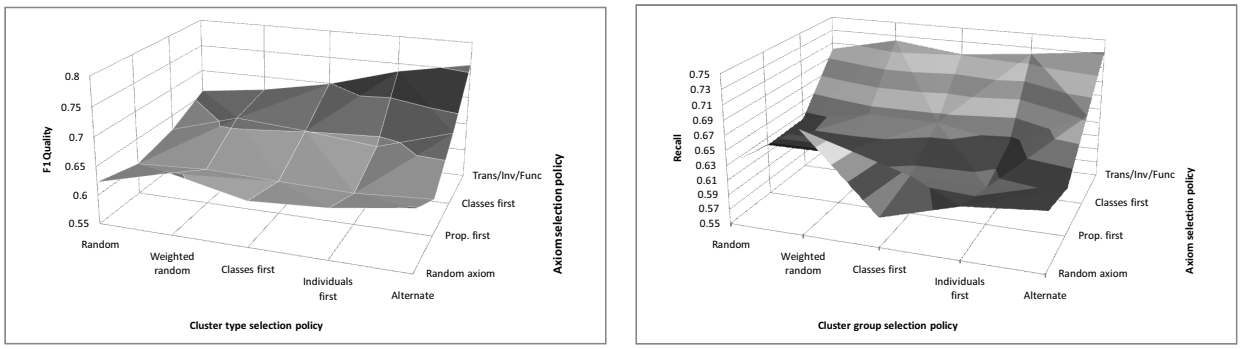
7.1 Optimal ILIADS Parameters

The set of parameters for ILIADS is comprised of:

- The 8 λ parameters for similarity computations.
- The threshold λ_t is the minimum similarity value for which a pair of clusters are considered as candidates for merging.
- The threshold λ_r is used to determine whether to use subsumption or equivalence in a candidate pair.
- The number of inference steps N.
- The *axiom selection policy* and the *cluster group selection policy*.
- The aggregation method used to compute the similarity of two clusters based on the similarities of their elements.

For a pair of ontologies, we denote by A_G^* the ground truth and by A^* the set of integration axioms and facts determined by the algorithm. We compute precision as $P = \frac{|A_G^* \cap A^*|}{|A^*|}$ and recall as $R = \frac{|A_G^* \cap A^*|}{|A_G^*|}$. The F1 quality measure is defined as usual $F1 = \frac{2 \cdot P \cdot R}{P+R}$.

In our first set of experiments, we determine the parameter settings that maximizes the average F1 quality over the



(a) (b)

Figure 5: (a) The variation of recall with the choice of policies; (b) The variation of F1 quality with the choice of policies.

entire dataset. The optimal values of N and the two selection policies turned out to be independent of the specific pair of ontologies. Briefly, we found that values of $N \geq 5$ do not produce statistically significant increases in overall quality. We also found the best cluster group selection policy is to *alternate* between classes and individuals, whereas the best axiom selection policy is *Transitive/Inverse/Functional first*. We discuss these parameters in detail later in this section. In 95% of the runs, the *complete link* method outperformed average link and single link when aggregating cluster similarities. For the rest of the paper, *complete link* will be used as the similarity aggregation method.

We varied the values of the 10 λ parameters in increments of .05 in the range $[0, 1]$. We then identified the configuration of parameters that maximizes the average F1 value over the entire dataset. For each pair of ontologies, we also determined the configuration of λ parameters that maximizes the F1 quality value for that pair. This resulted in a set of 30 values for each λ parameter. Line 1 of Table 3 contains the parameter configuration that maximizes the average F1 value for all the pairs. Lines 2 and 3 of Table 3 contain the minimum and respectively maximum of the set of 30 values for each λ parameter.

Next, we experimented with two different versions of ILIADS. The first (ILIADS *fixed parameters* or ILIADS-FP) uses the λ parameter configuration in line 1 of Table 3. The second version (ILIADS *best parameters* or ILIADS-BP) uses the best λ parameters for each ontology pair. We looked at how the choice of N affects the running time of ILIADS and the answer quality. Since most of the running time is spent on inference, we expected a sharp increase with N. The experiments confirmed that the running time increases almost exponentially for $N > 5$, while F1 quality stabilizes to a a value of approximately .762 for the same values of N. From these two observations, we determined that $N = 5$ is a good compromise between quality and running time.

Finally, we looked at how the axiom selection and cluster group selection policies affect precision, recall and F1 quality for ILIADS-BP. Figure 5(a) and (b) show the variation of recall and quality respectively. We omit the chart for precision, as it is very similar to the one for recall. First, we note that selecting axioms according to the *Transitive/Inverse/Functional first* policy and selecting cluster groups according to the *Alternate* policy outperforms any other combination. Second, rather surprisingly, *Random* and *Weighted Random* policies yield better quality than *Classes first* or *Properties first*. This again confirms that looking at both instances and structure in alternation yields better quality answers.

7.2 Comparison with FCA and COMA++

We compared precision, recall and F1 quality against FCA-merge and COMA++. Figure 6(a) shows the average precision, recall and F1 quality for all four methods. From the experimental data, we see that, on average, the improvement in F1 quality of ILIADS-BP is significant; 6% over COMA++ and 25% for FCA-merge. In addition, ILIADS-BP has an impressive improvement in recall: 11% and 55% for COMA++ and FCA-merge respectively. Figure 6(b) displays a plot of recall on the Y-axis and precision on the X-axis for every pair in the dataset. The figure suggests that FCA-merge generally has high precision (most points are on right side of the plot), but low recall (points are grouped in the bottom half of the plot). COMA++ on the other hand lies somewhere between FCA-merge and the two ILIADS methods.

Furthermore, we have found that ILIADS performs significantly better than average on ontologies which contain a reasonable amount of instance data. For a subset of 21 pairs of ontologies in our dataset for which the ratio of instances to classes is greater than 1.9, the average precision, recall and F1 quality are displayed in Figure 6(c). On this subset of ontology pairs, the improvement in average F1 quality of ILIADS-FP is of 10.3% and 29.3% respectively for COMA++ and FCA-merge, while that of ILIADS-BP is 14% (more than double than the average for the entire dataset) and 33% respectively for COMA++ and FCA-merge. This suggests that when we have enough instance data, the interplay between integrating schema and data in ILIADS yields significantly better F1 quality than existing tools.

To further test this hypothesis and establish a baseline, we removed **all instance data** from all input ontologies. We point out that removing all instances seriously disadvantages ILIADS in two ways: (i) first, there is no alternation between matching schema and data and (ii) the effects of the inference similarity score are greatly reduced[7]. Even under these conditions, the performance of ILIADS

[7]In that more than 75% of the relevant inference similarity scores come from facts inferred about data instances.

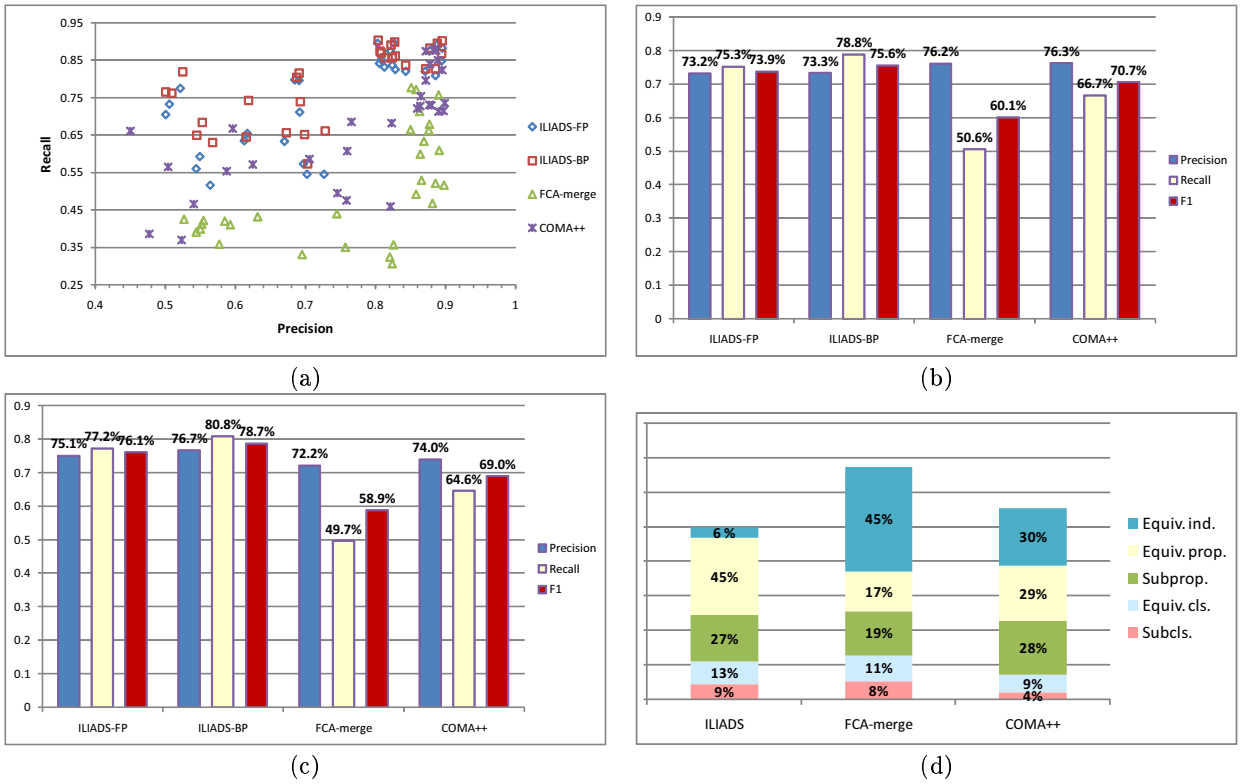


Figure 6: (a) Precision and recall scatter plot for ILIADS, **FCA-merge and COMA++; (b) Comparison of the average precision, recall and F1 quality for** ILIADS, **FCA-merge and COMA++; (c) Comparison of average precision, recall and F1 quality on a dataset of 21 pairs of ontologies; (d) Breakdown of false negatives in** ILIADS, **FCA-merge and COMA++.**

was comparable to FCA-merge and COMA++. The average F1 quality of ILIADS-FP, ILIADS-BP, FCA-merge and COMA++ under these conditions is 66.9%, 68.4%, 60% and 70.5% respectively.

We also measured the running time for the three systems. Overall, all algorithms produced the alignments in reasonable amounts of time. On average, ILIADS-BP took approximately 416 seconds per ontology pair, with a standard deviation of 85.6. FCA-merge was the fastest of the three, with an average of 140 seconds per pair, whereas COMA++ has an average of 218 seconds. For ILIADS-BP, we measured the running time in each step and determined that on average the clustering steps use approximately 19% of the clock cycles, whereas inference uses approximately 80%.

We also looked at the alignment axioms and facts that ILIADS, FCA-merge and COMA++ missed (i.e., false negatives). We broke these down into categories based on the type of axiom or fact (e.g., class and property subsumption and equivalence, etc.) Figure 6(d) presents the false negative breakdown per category; the column heights are proportional to the average number of false negatives for each algorithm. It is interesting to note that the large majority of false negatives in ILIADS come from missing axioms about properties. Such axioms also have a fair share in the false negatives of FCA-merge and COMA++. The explanation for this phenomenon is that the intended semantics of a property is not always evident from the ontology graph alone. For instance, in Section 2 we discussed the intended meaning of the *averageCases* property in Figure 1(a). However, human reviewers also use their background knowledge when determining relationships. We suspect background knowledge is also the reason FCA-merge does a little better on properties than the other two systems. Recall that FCA-merge has access to additional information in the form of a document corpus. Figure 6(d) also suggests that the reason for better F1 quality in ILIADS lies in its ability to handle both ontology structure and instance data. FCA-merge and COMA++ have a large number of false negatives (45% and 30% respectively) due to missing equivalences between individuals.

7.3 Sensitivity of λ Parameter Settings

Next, we looked at the variation of the λ parameters between pairs of ontologies. First, we note that λ_t and λ_r are fairly stable as illustrated in Table 3. The same holds for the three λ_x parameters; the only one with significant variation is λ_x^p, primarily due to difficulties in finding matches between properties (as illustrated in our analysis of false positives).

Since λ_s and λ_e vary significantly with the ontology pair, we looked at structural indicators of the ontologies and how they correlate with the parameters. We found that the correlation coefficients between λ_s^c, λ_s^p and λ_s^i and the average degree of a node[8] in each ontology pair are .453, .311 and .38 respectively. Intuitively, this says that the higher the

[8]Here, we use "degree of a node" to mean the number of edges adjacent to that node.

average degree of a node, the more important structural similarity becomes. We also confirmed that λ_s^c positively correlates to the number of rdfs:subClassOf axioms in the ontology (the coefficient is .61). Informally this means that the richer the class taxonomy becomes, the more important structural similarity is for classes. We also found that for 25 out of the 30 pairs of ontologies, the variations in the λ_s parameters are much smaller ($\lambda_s^c \in [.49.61]$, $\lambda_s^i \in [.56, .63]$ and $\lambda_s^p \in [.39, .45]$). When looking at the remaining 5 pairs (the "outliers") we found that the ontologies in these pairs have a large difference between their average node degrees (higher than 1.5 in all cases), whereas for the 25 pairs, the differences in the average node degrees were below 0.5. To validate this hypothesis, we computed the correlation coefficient between (1) the distance of each λ_s parameter to its overall average and (2) the difference between the average node degrees in the corresponding ontology pair. The coefficients were .416, .61 and .5 for λ_s^c, λ_s^p and λ_s^i respectively. This suggests that the λ_s parameters are generally stable around the average for pairs with structures that were not vastly different.

The extension similarity parameters λ_e^c and λ_e^p correlated positively with the ratio of instances to classes in an ontology with coefficients of .39 and .471 respectively. Importantly, we should note that the variation in these parameters produces significant improvement in the F1 quality only for ontology pairs where the ratio between instances and classes is above 1.9 (the 21 ontology pair subset). We also looked at the correlation between the distance from the average and the differences in instance-to-class ratio in the 21 ontology pairs and found coefficients of .34 and .43 for λ_e^c and λ_e^p respectively. This suggests that the closer the ratio of instances to classes in the two ontologies in the pair, the closer the λ_e parameters are to the average.

8. RELATED WORK

Ontology alignment methods can generally be classified as *local methods* and *global methods*. Local methods are further decomposed into *terminological* (or lexical), *structural*, *semantic* and *extensional*. Many of the terminological approaches use an external lexicon to match terms which are semantically related. Patel et al. [21] developed OntoGenie, a tool that parses web pages to create knowledge instances for a given ontology using WordNet as a bridge. Silva and Rocha [26] use a semantic similarity measure proposed by Resnik [25] to establish correspondences between terms. They use these to transform instances for a source ontology into instances in a target ontology. The similarity measure uses a probabilistic assignment of terms to synonym sets based on an external corpus. ILIADS uses WordNet as well, however the weight given to the information from the lexicon is configurable through the λ_x parameters. As seen in Section 7, in ILIADS lexical similarity accounts for about 20% of the final similarity score.

In the area of structural methods, Li and Clifton [14] present a tool called SEMantic INTegrator (SEMINT) that uses neural networks to assist in finding correspondences between attributes in heterogeneous databases. SEMINT uses both schema and data (like many schema matching tools [24]) to produce matching rules automatically. The schema information used includes data types, length, scale, precision, the existence of keys and other constraints, etc. The data is used to compute statistics (min, max, variance, scale, etc.) to determine data distributions that can be compared. Ehrig and Sure [7] proposed the definition of a set of rules for determining similarity between ontology instances, and were the first to point out that OWL features such as symmetry and restriction of values could be used. ILIADS uses such features to a certain extent to measure structural similarity (e.g., in the computation of $sim_{features}$). However, in our experiments we determine that structural similarity alone does not necessarily lead to an answer with high recall.

When we define our compound similarity measure based on node neighborhoods, the similarity measure is essentially still local, since it considers only the neighborhood of a node and not the entire ontology. Moreover, local similarity measures can be defined in a complex interacting manner, as in the case of ILIADS, which means they cannot be computed in a single step. One can view the integrated incremental inference that we perform as a sort of propagation of similarity through the graph. This helps to relate our approach to other similarity propagation methods. Melnik et al. [17] introduced *similarity flooding*, a generic graph matching algorithm which computes the correspondences of nodes in the graphs as the fixpoint of an operator. Doan et al. [6] present the Glue system, which learns classifiers for classes based on instance data, and finally computes the joint probability distribution of instances. The principle is the same as the extensional approach – two classes are more similar if their instances are the same. The system uses a relaxation labeling process to propagate similarity. ILIADS uses a slightly different similarity propagation method. For instance, assume nodes c, c' are similar enough to be clustered in the current step. Then the similarity between nodes in their neighborhoods will increase, and may now be considered as candidates for merging. The logical inferences made will also result in changes to the similarity. The most successful ontology matching systems, like COMA++ and FCA-merge, do not fall neatly into any of these categories, but rather use a mixture of techniques. Anchor-PROMPT [20] and Chimerae [16] use a variety of lexical and structural based techniques to evaluate the semantic similarity of concepts and allow a human reviewer to make the final decision in difficult cases. IF-map [12] is another system rooted in formal concept analysis that aligns two ontologies based on how they map to a third reference ontology. S-Match [9] is a schema-matching system based on an extensible library of matching generators ranging from lexical methods to SAT solvers. HumMer [19] is a system that performs schema and data fusion for relational databases. Schema matching is performed through duplicate detection, where similar or duplicate data values are used to infer schema matchings. HumMer also performs data fusion by allowing the user to select from a large list of conflict resolution strategies. ILIADS takes a different approach to the integration problem, by interleaving data and schema integration, as well as taking advantage of the richer structure of OWL-Lite ontologies to infer possible matchings. In [2], Bernstein et al. present an interactive schema matching tool that, in addition to structural and lexical properties of the two schemas also uses user action history and existing matches to personalize the process of generating schema matchings. Unlike ILIADS, the system does not yet use data values when generating matches and is built for XML-to-XML mapping, a language that does not provide support for logical inference. Madhavan et al. [15] present an architecture for a schema matching

system that leverages past experience. The authors use a *Mapping Knowledge Base* to learn classifiers for schema elements seen in the past and predict the degree of similarity between schema elements. The approach is complementary to that of ILIADS – the name, instance and structure learning techniques defined could be used to improve the quality of ILIADS once a sufficiently large corpus of ontology pairs is available. OLA [8] is the only system designed specifically for OWL Lite and which uses a global similarity measure for ontology alignment. Local similarity between entities in the ontologies produces a set of equations which are iteratively solved to provide a set of mappings. Unlike ILIADS, OLA does not use the inference capabilities of OWL directly.

9. CONCLUSION

We have presented ILIADS, a novel ontology integration tool which tightly integrates statistical matching with logical inference. This tight integration means that our algorithm can exploit data and structure effectively to produce high quality integration results. We have investigated how the ontological structure itself affects the utility of different inference and matching strategies and how our algorithm can adapt to the characteristics of the ontologies to be merged.

We have validated our results on an extensive collection of real-world ontologies. Our most important findings are that: (i) the number of inferences steps and the heuristic policies are independent of the particular input ontologies, (ii) ILIADS significantly outperforms COMA++ and FCA-merge, especially so for ontologies with a reasonable amount of instance data and (iii) the parameters of ILIADS correlate to structural properties of the ontologies and are stable for ontology pairs that do not have very different structures.

10. ACKNOWLEDGEMENTS

This work is funded in part by NSF grant 0438866, AFOSR grants FA95500510298 and FA95500610405 and ARO grant DAAD190310202, and by NSERC.

11. REFERENCES

[1] D. Aumueller, H. H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with COMA++. In *Proc. SIGMOD*, pages 906–908, 2005.

[2] P. A. Bernstein, S. Melnik, and J. E. Churchill. Incremental schema matching. In *Proc. VLDB*, pages 1167–1170, 2006.

[3] P. Buneman, S. Davidson, and A. Kosky. Theoretical Aspects of Schema Merging. In *EDBT*, pages 152–167, 1992.

[4] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*, 2003.

[5] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Learning to map between ontologies on the semantic web. In *WWW*, pages 662–673, 2002.

[6] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Ontology Matching: A Machine Learning Approach. In *Handbook on Ontologies*, pages 385–404. Springer-Verlag, 2004.

[7] M. Ehrig and S. Staab. QOM - Quick Ontology Mapping. In *ISWC*, pages 683–697, 2004.

[8] J. Euzenat, D. Loup, M. Touzani, and P. Valtchev. Ontology alignment with OLA. In *ISWC EON*, pages 59–68, 2004.

[9] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *Semantic Interoperability and Integration*, number 04391 in Dagstuhl Sem. Proc., 2005.

[10] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[11] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic J. of the IGPL*, 8(3):239–264, 2000.

[12] Y. Kalfoglou and M. Schorlemmer. IF-map: an ontology mapping method based on information flow theory. *J. Data Sem.*, 1(1):98–127, Oct. 2003.

[13] M. Lenzerini. Data Integration: A Theoretical Perspective. In *PODS*, pages 233–246, 2002.

[14] W.-S. Li and C. Clifton. SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data Knowledge Eng.*, 33(1):49–84, 2000.

[15] J. Madhavan, P. A. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. *ICDE*, 0:57–68, 2005.

[16] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *KR*, pages 483–493, 2000.

[17] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. *ICDE*, page 117, 2002.

[18] R. J. Miller, L. M. Haas, and M. Hernández. Schema Mapping as Query Discovery. In *VLDB*, pages 77–88, 2000.

[19] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps: Resolving schema, tuple, and value inconsistencies. *IEEE Data Eng. Bull.*, 29(2):21–31, 2006.

[20] N. F. Noy and M. A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *Int. J. of Hum.-Comp. Stud.*, 59(6):983–1024, 2003.

[21] C. Patel, K. Supekar, and Y. Lee. OntoGenie: Extracting Ontology Instances from WWW. In *Human Language Technology for the Semantic Web and Web Services, ISWC*, 2003.

[22] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating Web Data. In *VLDB*, pages 598–609, 2002.

[23] R. Pottinger and P. A. Bernstein. Merging Models Based on Given Correspondences. In *VLDB*, pages 826–873, 2003.

[24] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *The VLDB Journal*, 10(4):334–350, 2001.

[25] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[26] N. Silva and J. Rocha. Ontology mapping for interoperability in semantic web. In *ICWI*, pages 603–610, 2003.

[27] E. Sirin and B. Parsia. Pellet: An OWL DL Reasoner. In *Description Logics*, volume 104 of *CEUR Work. Proc.*, 2004.

[28] G. Stumme and A. Maedche. FCA-MERGE: Bottom-Up Merging of Ontologies. In *IJCAI*, pages 225–234, 2001.

[29] W. Winkler. Advanced methods for record linkage., 1994. Technical report, Statistical Research Division, Washington, DC: U.S. Bureau of the Census.