One Size Does Not Fit All: Customizing Ontology Alignment Using User Feedback

Songyun Duan, Achille Fokoue, and Kavitha Srinivas

IBM T.J. Watson Research Center, NY, USA
{sduan,achille,ksrinivs}@us.ibm.com

Abstract. A key problem in ontology alignment is that different ontological features (*e.g.,* lexical, structural or semantic) vary widely in their importance for different ontology comparisons. In this paper, we present a set of principled techniques that exploit user feedback to customize the alignment process for a given pair of ontologies. Specifically, we propose an *iterative* supervised-learning approach to (i) determine the weights assigned to each alignment strategy and use these weights to combine them for matching ontology entities; and (ii) determine the degree to which the information from such matches should be propagated to their neighbors along different relationships for *collective* matching. We demonstrate the utility of these techniques with standard benchmark datasets and large, real-world ontologies, showing improvements in F-scores of up to 70% from the weighting mechanism and up to 40% from collective matching, compared to an unweighted linear combination of matching strategies without information propagation.

1 Introduction

Ontology alignment and the related problem of schema matching is a richly studied area [9, 10, 14], with significant advances of alignment techniques in recent years. There are a number of systems that perform pretty well on the ontology alignment evaluation initiative (OAEI) benchmarks (for most recent examples, see Lily [17], ASMOV [8], Anchor-Flood [11], and RiMOM [12]).

A common aspect of most alignment systems is that they combine semantic and lexical features of ontology entities with structural propagation (*e.g.,* as in similarity flooding [13] or in iterative structural propagation of QOM [6]). When such structural propagation is applied, two key assumptions dominate the literature: (i) Structural propagation is beneficial to ontology alignment; and (ii) The alignment results at the last iteration are the best to be produced as the final results. Due to the lack of a principled way to determine the optimal number of iterations, most systems perform structural propagation either to a fixed number of iterations, or until further propagation does not produce additional matchings [6, 7].

Our key observation, based on work with real-world ontologies, is that the importance of any of these features (lexical, semantic or structural) varies widely across different ontology alignments. Furthermore, the degree of structural propagation required for optimal performance also varies widely. More structural propagation does not necessarily lead to better alignment results; in some cases, *any* structural propagation actually *impairs* alignment quality.

P.F. Patel-Schneider et al. (Eds.): ISWC 2010, Part I, LNCS 6496, pp. 177–192, 2010.

More recently, collective matching approaches (*e.g.,* [1]) have been proposed to take structural information into account, in a principled manner, for matching ontology entities. These approaches typically use sophisticated statistical models such as Markov Networks [15] to explicitly represent interdependencies between various matching choices. In a sense, they do not optimize the quality of individual matching decisions (*i.e.,* matching between individual pairs of ontology entities); instead, they optimize the quality of the whole collection of matching decisions. However, a serious drawback with these approaches to ontology alignment using complex models is their high computational cost; thus making such systems hard to use with large, real-world ontologies.

In this paper, we propose a principled and scalable technique to incorporate lexical, semantic and structural features, using *iterative supervised structural propagation*. Our approach relies on customizing two key components of ontology alignment. First, at a lexical level, alignment depends on a number of different alignment strategies (*e.g.,* alignment based on the names of ontology entities as encoded in a URI, or the associated documentation in terms of *rdfs:label*, *rdfs:comment*). For a given pair of ontologies, empirical evaluation may find out that an alignment strategy based on name may be more appropriate than that based on documentation. Our approach uses user feedback to learn the relative importance of these different alignment strategies for a given ontology pair, which is similar to the approach taken by APFEL [7]. Specifically, we use logistic regression [2] to determine the weights assigned to different strategies based on user feedback.

Second, we address the issue of how to systematically propagate lexical-level and user-specified matches along structural relations in the ontology. Here, we diverge from previous iterative structural propagation approaches such as [13] and [6] in that we adopt *iterative supervised learning* to estimate the optimal number of iterations needed for a given ontology pair. Specifically, we use the training phase to observe exactly which iteration yields maximal benefits in alignment, and use this information to determine the stopping condition for structural propagation at test. Our experimental evaluation shows clear advantages of our approach over previous approaches (*e.g.,* APFEL [7]) that do not take user feedback as guidance across iterations during the structural propagation phase.

Our contributions in this paper are as follows:

- We use supervised learning to customize the weights for different alignment strategies for a given ontology pair, and to customize the degree to which those matches at an entity level get propagated to its neighbors for collective matching.
- We demonstrate the effectiveness of this approach on two benchmark datasets, and 6 other large, real-world ontology alignments. The experimental results show good scalability of our approach, and confirm the hypotheses about great variability in features across ontology alignments. Our results also show dramatic improvements in alignment from the weighting (up to 70% increase in F-scores), and collective matching (up to 40% increase in F-scores).
- We demonstrate that incorporating supervision into the process of structural propagation is key to the selection of the relevant features. Weighting features using supervision *after* the process of unsupervised structural propagation yields poor results in some cases.

The rest of the paper is organized as follows. Section 2 gives an overview of the framework for ontology alignment. Section 3 describes the ontological features and similarity metrics. Section 4 presents a supervised-learning technique for similarity aggregation and interpretation. Section 5 presents the technique of iterative supervised structural propagation. Section 6 presents experimental results. Section 7 discusses related work, and Section 8 concludes.

2 Overview of Ontology Alignment

In this section, we briefly introduce important notations, and present the overall structure of our approach to ontology alignment. We use the terms alignment/matching and element/entity interchangeably when there is no confusion.

An ontology $\mathcal{O}$ is represented as a labeled graph $G = (V, E, vlabel, elabel)$. The set of vertices V contains ontology entities such as concepts and properties. Edges in E ($E \subseteq V \times V$) represent structural relationships between entities. The edge labeling function $elabel$, which maps an edge $(v, v') \in E$ to a subset of the set SL of structural labels, which in turn specify the nature of the structural relationships between entities (*e.g.,* subclassOf). Let LL denote the set of lexical labels associated with entities (*e.g.,* name, documentation). Finally, the vertex labeling function, $vlabel : V \times LL \to String$, maps a pair $(e, l) \in V \times LL$ to a string corresponding to the value of the lexical label l (*e.g.,* name) associated with the entity e.

Given two ontologies $\mathcal{O}$ and $\mathcal{O}'$, the ontology alignment problem consists of finding a set of matchings (e, e'), where e and e' are entities in $\mathcal{O}$ and $\mathcal{O}'$, respectively. Additionally, a similarity measure, denoted $\mathtt{sim}_{agg}$, which maps the pair of entities $(e, e') \in \mathcal{O} \times \mathcal{O}'$ to a real number in $[0, 1]$, provides the confidence in a matching. We assume that for any entity in $\mathcal{O}$, there is at most one matching entity in $\mathcal{O}'$.

The alignment approach presented in this paper is similar in its overall structure to the process adopted by many existing matching engines such as [6]:

1. **Generation of Candidate Matchings:** This step includes feature engineering (*i.e.,*, the extraction of the relevant characteristics of ontology entities in both the source and the target ontology) and the selection of candidate matchings (to avoid considering the Cartesian product of entities in the two ontologies).
2. **Similarity Aggregation and Interpretation:** This step computes various similarity metrics on candidate matchings identified in the previous step. Each individual similarity metric is a function of only the features extracted from the two ontology entities being compared. The similarity scores are then aggregated into a single similarity score for each candidate matching. Interpretation is then based on the aggregated similarity scores, and involves a decision about which candidate matchings should be selected as valid matchings — typically using a threshold.
3. **Structural Propagation.** This step propagates matching information along ontology structure, by repeating the previous steps, typically, either to a fixed number of iterations or until no additional matchings are produced.

Our approach significantly differs from previous work in two ways. First, our similarity aggregation step is not based on an unsupervised (thus ad-hoc) weighted combination of similarity scores. We use a fully supervised-learning approach (described in more details in Section 4) to learn, at each iteration, from user feedback an optimal combination of similarity scores. Second, our stopping condition for the structural propagation is more principled. Note that previous work stop propagation based on an arbitrary number of iterations or the absence of additional matchings, which assumes that matching quality monotonically improves over successive iterations (this assumption does not hold in many cases, as shown in the experiment section). We stop iterations when there is no significant improvement in information gain at training, and select only the matchings produced at the iteration where the matching result has the best consistency with user feedback (see Section 5 for more details).

3 Generation of Candidate Matchings

In this section, we describe the features that can be extracted from ontologies, and the lexical similarity metrics we consider in this paper (structural similarities are discussed in Section 5).

3.1 Feature Engineering

In our approach, the feature engineering step is essentially responsible for transforming models in various representations (*e.g.,* XML Schemas, UML models, OWL ontologies, etc) into an ontology $\mathcal{O}$ represented as the labeled graph $G = (V, E, vlabel, elabel)$. Structural features are represented as edge labels.

In this section, we present features extracted from models encoded as OWL ontologies or OBO ontologies.

Lexical features (*i.e.,* elements of the set LL) extracted from ontology entities (concepts or properties) are as follows:

- name, which corresponds to the last segment of the ontology entity's URI (*e.g.,* 'Person' for 'http://www.ibm.com/hr/Person').
- documentation, which consists of the concatenation of the values of rdfs:label, rdfs:comment, obo:def, obo:comment, and obo:synonym.

Structural features (*i.e.,* elements of the set SL) are shown in the first column of Table 1. The second column of Table 1 indicates the condition under which an edge (e_0, e_1) is assigned a given label. Note that, although these structural features do not capture all the structural and semantic constructs of OWL ontologies (*e.g.,* union, disjointWith, complementOf, and nested structures are not currently taken into account), they are sufficient to produce robust structural improvements on the ontologies we tested with (see Section 6 for more details).

3.2 Lexical Similarities and Initial Selection of Candidate Matchings

Similarity Metrics. Various similarity metrics can be employed to compare entities from different perspectives. In an abstract form, a similarity metric is a function that maps a pair of entities to a value between 0 and 1.

$$\mathtt{sim}(e, e') \rightarrow [0, 1] \tag{1}$$

Table 1. Structural Labels

Label	**Label** $\in elabel(e_0, e_1)$ **iff.**
subclassOf	e_0 is a direct subclass of e_1.
superclassOf	e_0 is a direct superclass of e_1.
isRangeOf	The concept e_0 is the range of the property e_1.
isDomainOf	The concept e_0 is the domain of the property e_1.
subPropertyOf	e_0 is a direct subproperty of e_1
superPropertyOf	e_0 is a direct superproperty of e_1
hasRange	The range of the property e_0 is the concept e_1.
hasDomain	The domain of the property e_0 is the concept e_1.
hasExistRestrictionOnProperty	The property e_1 is used to define the concept e_0 in terms of an existential or minimal cardinality restriction (*e.g.,* e_0 is defined as $e_0 \sqsubseteq \exists e_1.C$)
hasForAllRestrictionOnProperty	The property e_1 is used to define the concept e_0 in terms of a universal restriction (*e.g.,* e_0 is defined as $e_0 \sqsubseteq \forall e_1.C$))
hasExistRestrictionOnClass	The concept e_1 is used to define e_0 in terms of an existential or minimal cardinality restriction (*e.g.,* assuming normalization to NNF, $e0$ is defined as $e_0 \sqsubseteq \exists R.e_1$).
hasForAllRestrictionOnClass	The concept e_1 is used to define e_0 in terms of a universal restriction (e.g, assuming normalization to NNF, $e0$ is defined as $e_0 \sqsubseteq \forall R.e_1$).
existRestrictionUsedFor	The concept e_1 is defined as an existential or minimum cardinality restriction using the property e_0 (*e.g.,* if e_1 is defined as $e_1 \sqsubseteq \exists e_0.C$)
forAllRestrictionUsedFor	The concept e_1 is defined as a universal restriction using the property e_0 (*e.g.,* if e_1 is defined as $e_1 \sqsubseteq \forall e_0.C$)

For a given pair of entities (e, e'), multiple similarity metrics can be applied. The similarity metrics are denoted as $\mathtt{sim}_i(e, e')$ $(i = 1, 2, \ldots)$. Note that the similarity metric can be as general as a matching technique.

For lexical similarity, standard similarity metrics exist for strings such as Levenshtein similarity or Jaccard similarity on n-grams. This works fine for lexical features, such as name, whose values are expected to consist of only a few words. However, for lexical features such as documentation, the values may consist of many paragraphs. Therefore, as explained in [3], we cast the problem into a classical information retrieval problem. We transform entities (*e.g.,* concepts and properties) into virtual documents. A virtual document consists of fields corresponding to the two lexical features described in the previous section, namely, name and documentation. These virtual documents are stored and indexed by a high-performance text search engine such as Lucene[1]. A Vector Space Model (VSM) [16] is adopted for comparison: each field F (name or documentation) of a virtual document is represented as a vector in a N_F-dimensional space, with N_F denoting the number of distinct words in field F of all documents.

[1] `http://lucene.apache.org/java/docs/index.html`

Traditional TF-IDF (Term Frequency-Inverse Document Frequency) values are used as the weights of coordinates associated with terms. The lexical similarity on a field $F \in \{\mathsf{name}, \mathsf{documentation}\}$ between two entities e and e' is referred to as $\texttt{sim}_F(e, e')$, and is computed as the *cosine* of the angle formed by their F vectors. We adjust for slight syntactic variations by using a term similarity metric (such as Levenshtein or Jaccard over n-grams) between terms as explained in [3].

Candidate Selection. In the first iteration (*i.e.,* before any structural propagation is performed), we use the text search engine, for each entity e in the source ontology $\mathcal{O}$, to select top-k candidate matchings of e in the target ontology $\mathcal{O}'$, by retrieving the virtual documents representing entities in $\mathcal{O}'$ that match well with e in terms of lexical similarity (*e.g.,* based on Lucene score).

4 Similarity Aggregation and Interpretation

4.1 User Feedback

In this paper, we assume that for any entity in $\mathcal{O}$, there is at most one matching entity in $\mathcal{O}'$. Also, we assume a simple format for user feedback (users specify which pairs of entities should be matched) that is fed to our system through a file of *gold standard* matchings. For a matching (e, e') specified by the user, we will label the matching (e, e') as `true`. For any candidate matching (e, e'') generated in Section 3, where e'' is not equal to e', we label it as `false`. Thus, we generate a set of training tuples in the following form:

$$\begin{aligned} &\langle\ \texttt{sim}_1(e, e'), \ldots, \texttt{sim}_n(e, e'), \texttt{true}\rangle \\ &\langle\ \texttt{sim}_1(e, e''), \ldots, \texttt{sim}_n(e, e''), \texttt{false}\rangle (\forall e'' \neq e') \end{aligned} \tag{2}$$

4.2 Weighted Aggregation

To interpret the matching result, a common practice is to aggregate the similarity metrics with a linear combination and set a threshold to decide which matchings are estimated to be `true`. However, it is well accepted that linearly (unweighted) combining the similarity metrics (or matching strategies) may adversely affect the overall matching quality. With user feedback, we can infer which similarity metrics are more reliable than others, and assign higher weights to the more reliable ones. A natural extension is to get a weighted sum (with the weight vector $\overrightarrow{\omega}$) of the similarity measures and apply a threshold ω_0 to predict whether a matching is `true` or `false`. The prediction is done with a decision boundary $\texttt{f}\ (\overrightarrow{\omega}, \texttt{sim}) = 0$, where the function `f` is defined as follows:

$$\texttt{f}(\overrightarrow{\omega}, \texttt{sim}) = \omega_0 + \omega_1 \times \texttt{sim}_1 + \ldots + \omega_n \times \texttt{sim}_n \tag{3}$$

4.3 Probabilistic Matching

For a candidate matching, the above decision boundary produces a binary value indicating the matching is `true` or `false`. However, it is more important to also produce a probability (between 0 and 1) along with the binary prediction, such that the matching result can be easily incorporated in other matching strategies (we will see such an

Algorithm 1. Learning of Weights for Ontology Matching

```
Input: ontologies $\mathcal{O}$ and $\mathcal{O}'$, gold standard matchings $\texttt{M}$ from user feedback, similarity metrics $\texttt{sim}_i$
Output: a list of matchings, $\langle (e, e'), P((e, e') = \texttt{true}) \rangle$
1. for each matching $\texttt{m} = (e, e')$ in gold standard $\texttt{M}$ do
    (i) Label candidate matchings for $e$: $(e, e')$ as true and $(e, e'')$ ($\forall e'' \neq e'$) as false;
    (ii) Compute the similarities of each candidate matching with given similarity metrics $\texttt{sim}_i$;
    (iii) Generate the training tuples in the way described in Section 4.1;
2. Learn the weights for combining the similarities and the threshold to decide whether a matching should be produced or not;
3. Use the learned weights and the threshold to generate the matching result.
```

example in Section 5). In statistics, the output of the real-valued function f can be mapped to a probability value, using the sigmoid function $P(t) = \frac{1}{1+e^{-t}}$. Specifically, given a candidate matching (e, e') with similarity measures sim, the probability of this matching is true is:

$$P((e, e') = \texttt{true}) = \frac{1}{1 + e^{-f(\overrightarrow{\omega}, \texttt{sim})}} \tag{4}$$

The probability that the matching is false is $P((e, e') = \texttt{false}) = 1 - P((e, e') = \texttt{true})$. The key issue is how to determine the weight vector $\overrightarrow{\omega}$ based on user feedback. Recall that the user feedback can be represented in the form of tuples $\langle \texttt{sim}, \texttt{true}/\texttt{false} \rangle$. The weight vector $\overrightarrow{\omega}$ that maximizes the likelihood of observing these tuples is the one that is most consistent with user feedback. In statistics, $\overrightarrow{\omega}$ can be determined using the MLE (maximum likelihood estimation) technique for logistic regression [2]. Algorithm 1 describes the key steps of the supervised-learning approach to ontology alignment.

5 Iterative Supervised Structural Propagation of User Feedback

In this section, we make the internal linkages of entities within ontologies explicit for learning. Specifically, for a candidate matching (e, e'), we take into account the matching results of e's neighbors in the ontology $\mathcal{O}$ when making the matching decision for (e, e'). The intuition is, for example, the matching of e's subclass with e''s subclass may add evidence that e and e' should be matched.

5.1 Structure-Based Similarity

For a candidate matching (e, e'), we extend the list of similarity metrics introduced in Section 3 with structure-based metrics as follows. Consider a structural label l (*e.g.,* subclassOf) in the ontologies. Suppose there is a set of entities $\texttt{SE}(e, l)$ that are connected to e with the structural label l in $\mathcal{O}$ (*i.e.,* $\texttt{SE}(e, l) = \{x | l \in elabel(e, x)\}$); correspondingly, $\texttt{SE}(e', l)$ for e' in $\mathcal{O}'$. It is important to aggregate the similarity values between the two sets, *i.e.,* $\texttt{SE}(e, l)$ and $\texttt{SE}(e', l)$, and extend the list of similarity metrics for (e, e') with the aggregation metrics. Below we briefly describe two types of aggregation metrics. (We considered other types of aggregation metrics such as min and sum, but empirically observed that max and avg are more effective.)

- $\mathtt{max}(S_1, S_2, \mathtt{sim})$ is the maximum similarity between any pair of entities, from two sets of entities S_1 and S_2 respectively, in the Cartesian product of $S_1 \times S_2$. For instance, S_1 can be $\mathtt{SE}(e, l)$, S_2 can be $\mathtt{SE}(e', l)$, and sim can be a lexical similarity metric, as described in Section 3.
$$\mathtt{max}(S_1, S_2, \mathtt{sim}) = \max_{(e_1,e_2) \in S_1 \times S_2} \mathtt{sim}(e_1, e_2)$$
- $\mathtt{avg}(S_1, S_2, \mathtt{sim})$ is the average similarity of pairs of entities in the Cartesian product $S_1 \times S_2$:
$$\mathtt{avg}(S_1, S_2, \mathtt{sim}) = \frac{\sum_{(e_1,e_2) \in S_1 \times S_2} \mathtt{sim}(e_1, e_2)}{(|S_1| + |S_2|)/2}$$

For a candidate matching (e, e'), we can generate various structure-based similarity metrics based on their sets of neighbors $\mathtt{SE}(e, l)$ and $\mathtt{SE}(e', l)$. Concretely, the structure-based similarity metrics can be:

- $\mathtt{max}(\mathtt{SE}(e, l), \mathtt{SE}(e', l), \mathtt{sim}_{name})$
- $\mathtt{avg}(\mathtt{SE}(e, l), \mathtt{SE}(e', l), \mathtt{sim}_{name})$
- $\mathtt{max}(\mathtt{SE}(e, l), \mathtt{SE}(e', l), \mathtt{sim}_{doc})$
- $\mathtt{avg}(\mathtt{SE}(e, l), \mathtt{SE}(e', l), \mathtt{sim}_{doc})$
- $\mathtt{max}(\mathtt{SE}(e, l), \mathtt{SE}(e', l), \mathtt{sim}_{agg})$
- $\mathtt{avg}(\mathtt{SE}(e, l), \mathtt{SE}(e', l), \mathtt{sim}_{agg})$

In the above metrics, $\mathtt{sim}_{name}$ is lexical similarity on the name field of two entities, $\mathtt{sim}_{doc}$ is the lexical similarity on the documentation/comment field of two entities, and $\mathtt{sim}_{agg}$ can be the aggregated score of similarity metrics in Algorithm 1 (*i.e.,* $\mathtt{sim}_{agg}(e, e') = P((e, e') = \mathtt{true})$).

5.2 Determining the Degree of Structural Propagation

At the bootstrapping step, we generate the aggregated similarity for a candidate matching (e, e') in the following way: If (e, e') is part of the ground truth (*i.e.,* provided by user feedback), its value is 1; otherwise, its value is 0. At the following iterations, we can utilize the matching result from the previous iteration. Note that for the pairs of entities that appear as training tuples (see Formula 3), we replace their matching scores with the ground truth (1 for true, and 0 for false).

The above structural similarity metrics allow the propagation of information conveyed by user feedback along the structure of the two ontologies. We thus extend the initial set of similarity metrics (Section 3) with the six structure-based similarity metrics per relation type. As a result, the number of similarity metrics that can be used for ontology matching is large. Since the amount of user feedback is limited, we adopt dimensionality reduction techniques to avoid the overfitting problem in Section 5.4.

At each iteration, the selection of matching candidates is extended to include pairs of entities having at least one non-zero structural similarity measure. The impact of the neighbor matching scores on the candidate matching in consideration is learned based

Algorithm 2. Iterative Supervised Structural Propagation for Ontology Matching

```
Input: ontologies $\mathcal{O}$ and $\mathcal{O}'$
Output: a list of matchings, $\langle (e, e'), P((e, e') = \texttt{true}) \rangle$
1. Bootstrapping: Generate training tuples with basic similarity metrics and
   structure-based similarity metrics;
2. Learn a weight vector to integrate similarity metrics that maximize the likelihood of
   user feedback being correct;
3. Generate a new list of matchings by combining the similarity metrics using the newly
   learned weight vector;
4. Update the training tuples with the aggregated similarities from Step 3, and add
   candidate matchings whose structure-based similarity measures become nonzero;
5. If it does not meet stopping condition, go to Step 2.
```

on user feedback, as described in Section 4. This process iterates until some stopping condition is satisfied; the following describes a metric to define the stopping condition.

5.3 Determining the Right Number of Iterations

We observe that too many iterations may be detrimental to matching quality (see the experiment section). Therefore, we propose a metric G, which is the *training error*, to decide the optimal number of iterations. G is computed as the absolute difference of the matching result (in the form of $\langle (e, e'), P((e, e') = \texttt{true}) \rangle$) at each iteration with regard to the ground truth (*i.e.,* user feedback).

$$\texttt{G} = \sum_{(e,e') \in \texttt{Ground Truth}} (1 - P((e, e') = \texttt{true})) + \sum_{(e,e') \notin \texttt{Ground Truth}} P((e, e') = \texttt{true})$$

The hypothesis is that the smaller the value of G, the better the matching result. This hypothesis will be verified with experiments in the next section.

5.4 Techniques for Scalability

Dimensionality Reduction. For large ontologies, possibly with many edge-labels, the generated attribute list (of similarity metrics) can be huge. Due to the limited amount of user feedback, it is necessary to reduce the dimensionality of the attribute space, to avoid the well-known overfitting problem. We use a standard unsupervised dimensionality reduction technique, principle component analysis (PCA) [2], to extract the most important dimensions for learning from the originally high dimensional space.

Blocking Unreliable Information Propagation. In Algorithm 2, the number of candidate matchings will monotonically increase after each iteration, since new candidate matchings are generated if their neighbors have *confident* matchings. To avoid propagating noisy information from neighbors, we set a threshold on the matching scores to keep the low-confidence matchings from propagating to neighbors. A side benefit of such blocking is efficiency; the number of tuples in the training data generated based on user feedback will increase slowly, thus saving the time to learn the weight vector (in Section 4.2) for each iteration. Note that if there is no blocking of propagation, the number of tuples in the training data may increase exponentially during iterations.

6 Experimental Evaluation

The focus of our experimental evaluation is to determine whether the great variability in ontology alignments can be reduced by using (i) a supervised-learning technique to customize the weights assigned to lexical features, and (ii) an iterative supervised-learning approach to determine the appropriate degree of structural propagation for each ontology alignment.

6.1 Experimental Setting

We focused on parts of the OAEI benchmark suite that are most suited for evaluating the effects of structural propagation. Test 202 was selected because it modifies the original ontology by obfuscating all names and documentations,and is a test of alignment based on structural similarity. We also selected the anatomy segment of the benchmark because the pair of ontologies in that benchmark encode structural information within an extensive part-of hierarchy. We also added 6 other ontology alignments from BioPortal into the evaluation to ensure that our results generalize well to different types of ontology alignments. Table 2 shows the characteristics of these 6 additional ontology alignments, and the number of matchings manually discovered by domain experts.

To evaluate the effects of training on similarity combination and structural propagation[2], we performed random sampling to split the reference matchings in the following way: we assigned 50% of the matchings to the 'test' group, and from the rest we further sampled 50% of the matchings to create the 'training' group (*i.e.,*, training ratio was 25% of the total number of matchings for the ontology alignment). Note that the actual number of matchings used for learning is small with respect to ontology size. For both training and testing, we varied the number of iterations used for structural propagation to a maximum of about 10 iterations for each ontology alignment.

The experiments were performed on a server with 8 way machine with 4 dual-core Intel Xeon chips at 3.20 GHz, with 20 GB of memory. For all the experiments, we used a maximum Java heap size of 10 GB.

6.2 Evaluation Metric

In our experimental evaluation, we had a complete gold standard for Test 202; for all other ontology alignments, we only had partial reference alignments[3]. We therefore measured `F-scores` in the standard manner only on Test 202. For all other ontology matching tasks, we computed an `F-score` only on the partial alignments available to us, and only considered ontology entities that were in the reference alignments (all other matchings we produced for entities in the source ontology not present in the partial alignment were not taken into account for `precision` or `recall` estimates). We assumed that there is at most one matching entity in the target ontology for each entity in the source ontology.

[2] The threshold we used for blocking unreliable information (Section 5.4) is 0.5.

[3] The lack of complete reference alignments is a frequent problem in real world ontology alignments. The matchings in Bioportal, for example, are almost always partial because the ontologies are large, and cannot be perfectly aligned manually.

Table 2. BioPortal Ontology alignments

Ontology 1	#Classes	Ontology 2	#Classes	#matchings
Mosquito gross anatomy (TGMA)	2,404	Drosophilia gross anatomy (FBbt)	8,742	324
Human devt. anatomy (EHDA)	11,575	Amphibian gross anatomy (AAO)	833	684
BRENDA tisse source (BTO)	4,950	Experimental Factor Ontology (EFO)	2,891	366
Experimental Factor Ontology (EFO)	2,891	Mouse Adult Gross Anatomy (MA)	3,504	212
ABA Adult Mouse Brain (ABA)	915	Mouse Adult Gross Anatomy (MA)	3,504	90
BIRNLex (birnlex)	3,582	UBER anatomy ontology (UBERON)	3,619	744

$$\texttt{precision} = \frac{|\texttt{M} \cap M_{GS}|}{|\texttt{M}|}, \texttt{recall} = \frac{|\texttt{M} \cap M_{GS}|}{|M_{GS}|}$$

$$\texttt{F-score} = \frac{2 \times \texttt{precision} \times \texttt{recall}}{\texttt{precision} + \texttt{recall}}$$

where M is the matchings discovered by our technique and the first ontology entity of each matching appears in the reference alignment M_{GS}.

In the following experimental results, we report F-score at the specific thresholds of (0.7, 0.8, 0.9), which are used to filter out low-confidence matchings, as users typically do not trust matchings with low matching confidence in practice.

6.3 Effect of Learning for Weighted Combination

Given the enormous variability in the importance of lexical and structural features to different ontology alignments, our hypothesis is that there is a principled way to weight these features appropriately using limited user feedback. We begin by examining the effect of learning to combine lexical features. Table 3 reports the F-score from our learning technique for weighted combination, compared with unweighted linear combination of similarity metrics for matching. For some ontology alignments (*e.g.,* BTO - EFO), there is a significant improvement in F-score (from 5% to around 70%); which clearly shows the effect of learning.

6.4 Effect of Iterative Supervised Structural Propagation

Figures 1 - 8 plot the changes in F-score as the structural propagation is iterated (in a supervised fashion), along with the corresponding training errors at iterations. These figures show:

- There is in fact a great deal of variability across ontologies, with lexical matches contributing to accuracy in the range of 10% to well above 90%.

Table 3. Effect of learning on `F-scores` at different thresholds

ontology alignment	Unweighted combination			Weighted combination		
	0.7 (%)	0.8 (%)	0.9 (%)	0.7 (%)	0.8 (%)	0.9 (%)
OAEI Anatomy	93	93	94	93	94	94
TGMA - FBbt	9	7	5	26	21	15
EHDA - AAO	99	99	99	99	99	99
BTO - EFO	5	1	1	74	72	68
EFO - MA	88	90	86	91	90	88
ABA - MA	93	90	85	94	92	90
BIRNLex - UBERON	77	66	46	83	81	73

- Structural propagation shows similar variability in its importance, with it improving accuracy by up to 40% in some cases (*e.g.*,, Figures 1, 3, 5, 8), but as shown in Figure 4, propagation of *any* structure in some ontologies causes a precipitous drop in accuracy by almost 25%, at high confidence thresholds (0.9).
- The number of iterations required to maximize the effects of structural propagation varies widely as well. In some cases (*e.g.,* Figure 1), a greater number of iterations of structural propagation is required, with peak matching quality being reached at about 5 iterations. In other cases (*e.g.*,, Figure 8), just one iteration is sufficient to maximize the benefits of structural propagation.

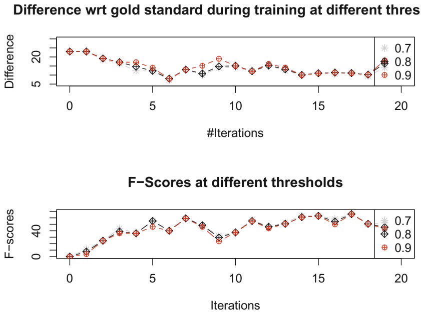

Fig. 1. OAEI 202

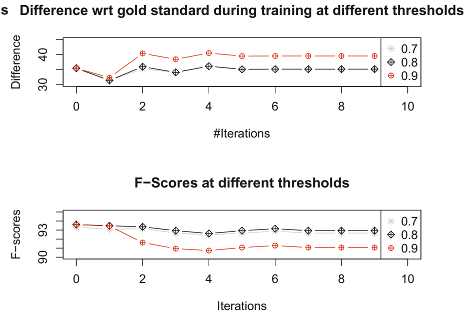

Fig. 2. OAEI Anatomy

Picking the Right Number of Iterations. For structural features, we hypothesized that the training error (*i.e.*,, the absolute difference between the matching results and reference matchings at training) can be used to estimate (i) whether structural propagation is useful, and (ii) to what degree structure needs to be propagated to maximize the overall matching quality. Because training error conceptually reflects *goodness of fit* [2], `F-score` at test is expected to be the best when training error is minimal. The general trend in Figures 1- 8 validated this hypothesis, therefore, we can pick the right number of iterations, in a principled way, to maximize the quality of matchings for a given pair of ontologies.

Comparison with Previous Work. We compare our approach with the technique proposed in [7] by simulating their process of matching in the following steps: (i) perform

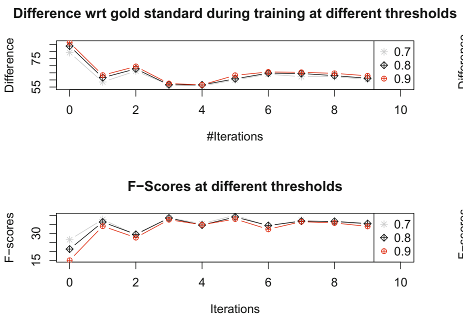

Fig. 3. TGMA-FBbt

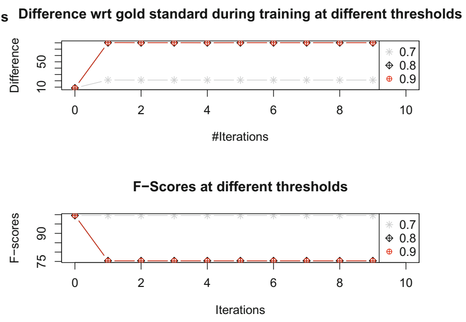

Fig. 4. EHDA-AAO

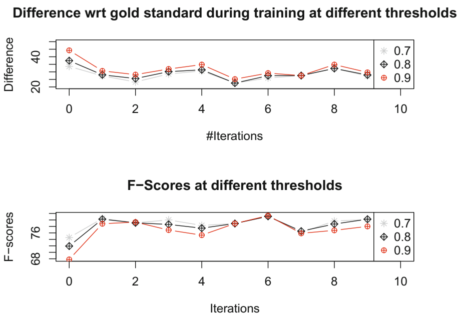

Fig. 5. BTO-EFO

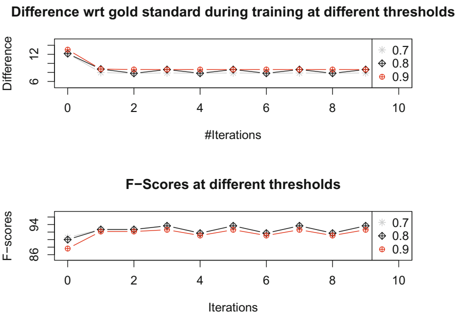

Fig. 6. EFO-MA

iterative unsupervised structural propagation from iterations 1 to 8, and (ii) apply supervised learning to determine weighted combination of both lexical and structural similarity measures returned from the last iteration. The result of this matching approach is shown in Figures 9 and 10. Several points to note here include: (i) The unsupervised structural propagation actually affects the F-score adversely, thus highlighting the importance of supervised propagation; and (ii) At the last iteration (with supervised learning), we get mixed results; in the case of BTO-EFO the F-score at the last iteration improves over the matching results based on purely lexical similarity measures (from 2% to 71%), while in another case structural propagation hurts F-score compared to lexical similarity measures (from 85% to 78%). Note that this is in contrast to our result. For the same two cases, we observed (in Figures 5 and 6) a significant improvement in F-score. Specifically, with our approach, the F-score for BTO-EFO increases from 67% to 81%; and the F-score for EFO-MA increases from 87% to 92%. In any of the two cases, our approach outperforms that of the previous work, due to iterative supervised structural propagation. One lesson we learned here is iterative structural propagation without the guidance of user feedback is not reliable and can be harmful.

For OAEI 202, our best F-score (84%) across all thresholds makes our approach competitive to the top 5 matching engines with best F-score between 80% and 90%.

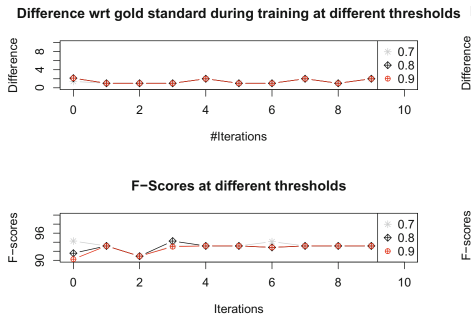

Fig. 7. ABA-MA

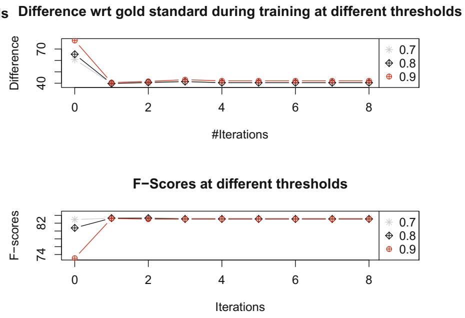

Fig. 8. BIRNLex-UBERON

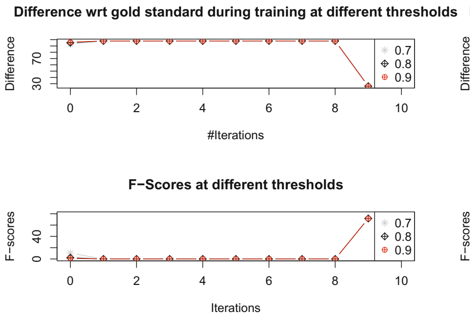

Fig. 9. BTO-EFO

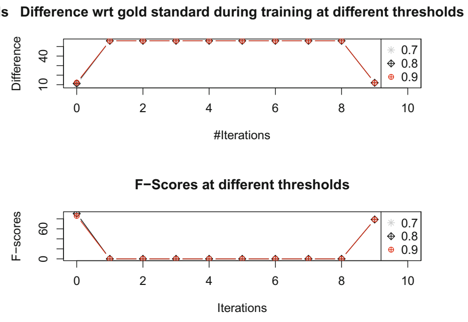

Fig. 10. EFO-MA

6.5 Discussion and Future Work

How much training data do we need to observe the beneficial results reported in this paper? We ran experiments with a smaller training ratio (10%) of the reference matchings, and observed a big variation in `F-scores` for the matching of BTO-EFO. The reason is that the absolute number of matchings (in this case, 36) used for training is too small considering the ontology size (in this case, 4,950). Our hypothesis is that when the sample sizes are too small (relative to the size of the ontology), careful selection of candidate matchings for user feedback is needed to ensure that enough structure is maintained for learning. Better sampling techniques (instead of random sampling) to reduce user feedback is an issue we leave for future work.

Another issue we observed is that for each ontology alignment in Table 2, our approach generates thousands of extra matchings with scores above 0.9, and these are not in the reference matchings. Based on the effectiveness of our technique on the reference alignments, we expect these extra matchings to be valuable to domain users, if only to recommend matchings for user validation.

One final point is about the scalability of our technique of iterative supervised structural propagation. The running time of each iteration was less than 3 minutes. Compared to existing collective matching based on sophisticated statistical models (*e.g.,* [1]), which have issues of scalability, our approach has a clear advantage in performance.

7 Related Work

Our approach, which applies an iterative supervised-learning technique to combine both lexical and structural similarities, can be contrasted with previous work that adopt either (unsupervised) iterative structural propagation technique (*e.g.,* similarity flooding [13] and its variants) or collective matching approaches (*e.g.,* [1]).

Similar to those systems (*e.g.,* [6] [12]) that apply variants of similarity flooding technique [13], our approach also iteratively propagates similarity metrics along ontology structures. However, our approach differs from them in two significant aspects. First, at each iteration, those systems aggregate various similarity metrics in an unsupervised (thus ad-hoc) fashion. In contrast, our approach applies supervised learning to learn from user feedback an optimal combination of both lexical and structural similarity metrics at each iteration; thus the information propagated across iterations is more reliable. Second, unlike those systems that assume matching result at the last iteration is the best (which is not necessarily true), we propose a novel and sound metric to estimate the matching quality at each iteration, based on the consistency of matching result with user feedback. Reference [7] views a matching engine (such as [6]) as a black box that returns its matching results and the similarity measures; it applies a supervised-learning technique to decide the optimal combination of the similarity measures returned by such a matching engine. Unlike our approach, the aggregation step occurring within the black-box matching engine remains unsupervised. As a result, the final structural similarities returned by the black-box engine may be less accurate; their iterative structural propagation misses the guidance from user feedback. Hence, the value of the supervised-learning approach to decide the weights of similarity measures is limited, resulting in sub-optimal matching results (we verified this point in the experiment section).

Recently, collective matching approaches (*e.g.,* [1]) have been proposed to take structural information into matching decisions using sophisticated statistical models. In a nutshell, those approaches use complex statistical models such as Markov Network [15] to explicitly represent interdependencies between matchings of interconnected ontology entities. Our approach is similar to this category of work in the sense that supervised learning techniques are applied to combine lexical and structural similarities in a principled way. However, due to the high computational complexity, in both learning and inference, of the complex statistical models used for encoding structural dependencies, those approaches based on sophisticated statistical models typically scale poorly to large ontologies[4].

Meta-learning (*i.e.,* integration of multiple alignment strategies) has also been implemented by GLUE [4] and other systems (*e.g.,* [5]). GLUE uses a supervised learning approach to build concept classifiers based on the associated instances (our approach does not assume instance information), but the way it combines inputs from various classifiers and performs structural propagation through relaxation labeling is unsupervised. Reference [5] also applies supervised learning to optimize the combination of multiple matching strategies, but it makes matching decisions for each entity independently, thus lacking the favor of collective matching.

[4] Simpler statistical models (*e.g.,* Markov Chain, Linear-chain Conditional Random Field, etc.) with scalable learning and inference algorithms are not sufficiently expressive to faithfully capture the structural dependencies.

8 Conclusion

To address the great variability in the importance of various features across ontology alignments, we have presented a principled and scalable technique to customize ontology alignment for a given pair of ontologies based on user feedback. We have shown how iterative supervised structural propagation, where each step is guided by user input, can optimally incorporate and propagate lexical-level and user-specified matches through the structure of the ontologies. Our experimental evaluation demonstrates the effectiveness of the new approach on both benchmark datasets and large, real-world bio-ontologies.

As future work, we plan to tackle the important, but orthogonal, problem of reducing user feedback by picking the most informative matches through active learning techniques.

References

1. Albagli, S., Ben-Eliyahu-Zohary, R., Shimony, S.E.: Markov network based ontology matching. In: IJCAI 2009 (2009)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2007)
3. Byrne, B., Fokoue, A., Kalyanpur, A., Srinivas, K., Wang, M.: Scalable matching of industry models - a case study. In: OM (2009)
4. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Ontology matching: A machine learning approach. In: Handbook on Ontologies in Information Systems. Springer, Heidelberg (2003)
5. Eckert, K., Meilicke, C., Stuckenschmidt, H.: Improving ontology matching using meta-level learning. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 158–172. Springer, Heidelberg (2009)
6. Ehrig, M., Staab, S.: QOM – quick ontology mapping. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 683–697. Springer, Heidelberg (2004)
7. Ehrig, M., Staab, S., Sure, Y.: Bootstrapping ontology alignment methods with APFEL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 186–200. Springer, Heidelberg (2005)
8. Jean-Mary, Y.R., et al.: ASMOV: Results for OAEI 2009. In: OM (2009)
9. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)
10. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-match: an algorithm and an implementation of semantic matching. In: ESWC (2004)
11. Hanif, M.S., Aono, M.: Anchor-Flood: Results for OAEI 2009. In: OM (2009)
12. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A dynamic multistrategy ontology alignment framework. IEEE Trans. Knowl. Data Eng. (2009)
13. Melnik, S., Garcia-molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm. In: ICDE (2002)
14. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. SIGMOD Rec. (2004)
15. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference (1988)
16. Raghavan, V.V., Wong, S.K.M.: A critical analysis of vector space model for information retrieval. Journal of the American Society for Information Science (1999)
17. Wang, P., Xu, B.: Lily: Ontology alignment results for OAEI 2009. In: OM (2009)