Actively Learning Ontology Matching via User Interaction ⋆

Feng Shi†, Juanzi Li†, Jie Tang†, Guotong Xie♯, and Hanyu Li♯

†Department of Computer Science and Technology
Tsinghua National Laboratory for Information Science and Technology
Tsinghua University, Beijing, 100084. China
{shifeng,ljz,tangjie}@keg.cs.tsinghua.edu.cn
♯IBM China Research Laboratory,Beijing 100094,China
{xieguot,lihanyu}@cn.ibm.com

Abstract. Ontology matching plays a key role for semantic interoperability. Many methods have been proposed for automatically finding the alignment between heterogeneous ontologies. However, in many real-world applications, finding the alignment in a completely automatic way is *highly infeasible*. Ideally, an ontology matching system would have an interactive interface to allow users to provide feedbacks to guide the automatic algorithm. Fundamentally, we need answer the following questions: How can a system perform an efficiently interactive process with the user? How many interactions are sufficient for finding a more accurate matching? To address these questions, we propose an active learning framework for ontology matching, which tries to find the *most informative* candidate matches to query the user. The user's feedbacks are used to: 1) correct the mistake matching and 2) propagate the supervise information to help the entire matching process. Three measures are proposed to estimate the confidence of each matching candidate. A correct propagation algorithm is further proposed to maximize the spread of the user's "guidance". Experimental results on several public data sets show that the proposed approach can significantly improve the matching accuracy (+8.0% better than the baseline methods).

1 Introduction

The growing need of information sharing poses many challenges for semantic integration. Ontology matching, aiming to obtain semantic correspondences between two ontologies, is the key to realize ontology interoperability [10]. Recently, with the success of many online social networks, such as Facebook, MySpace, and Twitter, a large amount of user-defined ontologies are created and published on the Social Web, which makes it much more challenging for the ontology matching problem. At the same time, the Social Web also provides some opportunities (e.g., rich user interactions) to solve the matching problem.

⋆ The work is supported by the Natural Science Foundation of China (No. 60703059), Chinese National Key Foundation Research (No. 2007CB310803), National High-tech R&D Program (No. 2009AA01Z138), and Chinese Young Faculty Research Fund (No. 20070003093).

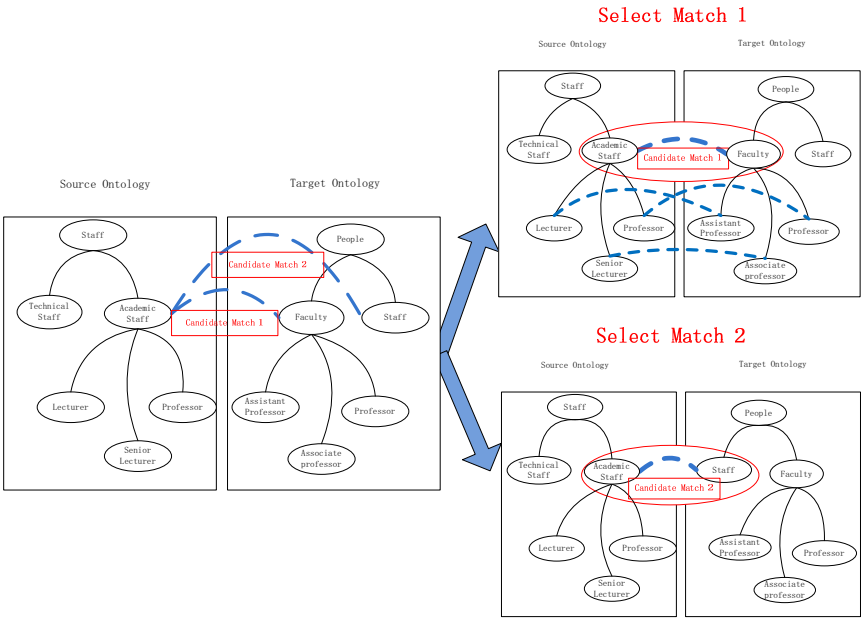

Fig. 1. Example of the candidate match selection. Ideally, we hope to query candidate match 1, instead of match 2. With the user correction, we can propagate the user correction to guide the matching process of other elements, as illustrated in the top-right.

Much efforts has been made for ontology matching. Methods such as Edit Distance [13], KNN [3] and Bayesian similarity [26], have been proposed to calculate the similarity between elements (e.g., concepts). However, most of existing works aim to find the ontology matching in a completely automatic way, although the complete automation is infeasible in many real cases [24]. One promising solution is to involve user interactions into the matching process to improve the quality of matching results [24]. However, regarding the large size of ontologies, random user interactions are really limited for helping ontology matching. Now, the question is: is there any way to minimize the amount of user interactions, while maximize the effect (accuracy improvement) of interactive efforts? Or a more specific question: given a fixed number of user interactions, can a system "actively" query the user so as to maximize spread of the interactions?

It is non-trivial to address the problem. A simple way is to let the user select candidate matches or to select matches with a low confidence (similarity) to query. Such queries can benefit the queried matches, however, it may not be helpful to the other non-queried candidate matches. Our goal is not only to correct the possibly wrong matches through the user interactions, but also to maximize the correction via spread (propagation) of the interactions. Thus, how to design an algorithm to actively select candidate matches to query is a challenging issue.

Motivating Example We use an example to demonstrate the motivation of the work. Figure 1 shows an example of the candidate match selection. The source and target

ontologies are both about academic staff. If we select the candidate match 1 (“Academic Staff”, “Faculty”) to query, with the user correction, we can further infer and correct another potential error match (“Academic Staff”, “Staff”). Moreover, the sub matches of (“Lecturer”, “Assistant Professor”) and (“Senior Lecturer”, “Associate Professor”) would also gain a higher confidence. While if we just select the candidate match 2 (“Academic Staff”, “Staff”), it is very possible to be no improvement.

Our Solution In this paper, we propose a novel problem of *ontology matching with active user interaction*. In particular, we propose an active learning framework for ontology matching, which tries to find the most informative candidate matches to query, and propagate the user correction according to the ontology structure to improve the matching accuracy. We present a simple but effective algorithm to select the threshold with user feedbacks. Three measures to evaluate the confidence of each match. A *correct propagation* algorithm is proposed to spread the user corrections. Experimental results demonstrate that the proposed approach can significantly improve (+8.0%) the matching performance with only a few queries (< 10).

The rest of this paper is organized as follows. Section 2 formalizes the problem. Section 3 describes our active learning framework for ontology matching. Section 4 explains the method of threshold selection, three measures for error matches detection, and the correct propagation algorithm. Section 5 presents the experimental results. Finally, we discuss related work in Section 6 and conclude in Section 7.

2 Problem Formulation

In this section, we first give the definition of ontology and then formalize the problem of ontology matching with active user interaction.

An ontology usually provides a set of vocabularies to describe the information of interest. The major components of an ontology are concepts, relations, instances and axioms [26].

In this paper, our ontology matching mainly focuses on concepts and relations. According to the relations between concepts, an ontology can be viewed as a directed graph, in which vertexes represent concepts and edges represent relations.

Given a source ontology O_S, a target ontology O_D, and an element (a concept or a relation) e_i in O_S, the procedure to find the semantically equivalent element e_j in O_D to e_i is called ontology matching, denoted as M. Formally, for each element (e_i in O_S), the ontology matching M can be represented [26] as:

$$M(e_i, O_S, O_D) = \{e_j\} \tag{1}$$

Furthermore, M could be generalized to elements set matching, that is for a set of elements $\{e_i\}$ in O_S, the ontology matching is defined as:

$$M(\{e_i\}, O_S, O_D) = \{e_j\} \tag{2}$$

If $\{e_i\}$ contains all the elements of O_S, the matching can be predigested as

$$M(O_S, O_D) = \{e_j\} \tag{3}$$

3 An Active Learning Framework for Ontology Matching

We propose an active learning framework for ontology matching. Algorithm 1 gives an overview of the active learning framework. Assume that O_S is the source ontology, O_D is the target ontology. M is a traditional method of ontology matching, L is the set of confirmed matches submitted by users, and N is the iteration number, which is also the number of candidate matches to query.

Algorithm 1. The Active Learning Framework for Ontology Matching

Input:

- the source ontology O_S, the target ontology O_D,
- A traditional method of ontology matching M,
- the confirmed match set L,
- number of matches need to be confirmed N

Initialization:

- apply M to map O_S to O_D, and get the match result R
- initialize L with $\emptyset$

Loop for N iterations:

- let $< (e_S, e_D), ? >$ = SelectQueryMatch();
- query users to confirm the match $< (e_S, e_D), ? >$
- add $< (e_S, e_D), l >$ to L
- improve the matching result R with $< (e_S, e_D), l >$

The basic process of the framework is as follows: first, it applies the traditional method of ontology matching M to map O_S to O_D, and gets the match result R, where multi-method results of different types are usually more useful for the next step. Second, it selects an informative candidate match $< (e_S, e_D), ? >$ to query users for confirmation with the result of the first step and the structure information of the two ontologies O_S and O_D. After the user confirmation, it adds the match $< (e_S, e_D), l >$ to the confirmed match set L, and improves the match result R with the confirmed matches. Then it repeats the second step for N iterations, or until it gets a result good enough.

Algorithm 1 is merely a shell, serving as a framework to many possible instantiations. What separates a successful instantiation from a poor one is the follow two problems:

1. First, how to select the most informative candidate match to query.
2. Second, how to improve the matching result with the confirmed matches.

In the next section, we will give the solutions to these two problems.

4 Match Selection and Correct Propagation

This part introduces our solution to the two core problems of ontology matching with active learning: the candidate match selection and the matching result improvement.

We first present a simple but effective algorithm to select the threshold for ontology matching with user feedback, and then give several measures to detect informative candidate matches to query. In the end of this section, we describe our correct propagation algorithm to improve the matching result with the user confirmed matches.

4.1 Threshold Selection with User Feedback

Most methods of ontology matching find matches through calculating the similarities between elements of source and target ontologies. The similarity can be string similarity, structure similarity, semantic similarity and so on [26]. No matter what kind of similarity is chosen, it needs a threshold to estimate which matches are correct. So it is very important to select a suitable threshold. However, the threshold selection is very difficult, especially when there is no any supervised information.

Through analysis we find the relationship of thresholds and matching results in most cases, as shown in Figure 2 (precisions, recalls and F1-Measures whose definitions are introduced in section 5).

From Figure 2, we can find that the precision curve is an increasing one, while the recall curve is a decreasing one, and the magnitude of change is getting smaller as the threshold getting bigger. So the F1-Measure curve has a maximum value on some threshold, which is our aim.

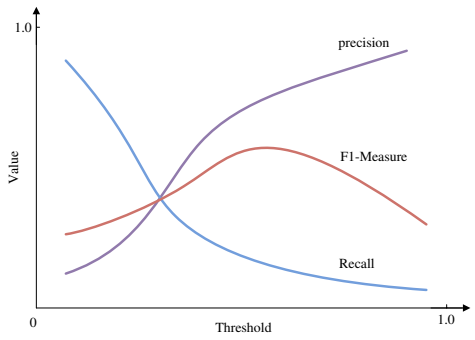

Fig. 2. Relationship of thresholds and matching performance (precision, recall and f1-Measure).

Algorithm 2 shows our algorithm of threshold selection. The input of the algorithm consists of the similarity set S, which contains all the matches and their similarity degree, an update step st for the threshold's updating, and an attenuation factor λ for st, and an initial threshold θ_0. First, the similarity set S needs to be normalized, and all the similarity degrees should be normalized into $[0, 1]$, and then let the threshold θ be the initial value θ_0. Second, it finds the match (e_S, e_D) whose similarity degree is the closest to θ, and let a user check whether the match is correct. If it is correct, the threshold θ increases by st, otherwise θ decreases by st. The second step is an iterative process, and st updates according to the correctness of the selected match each iteration. If the correctness of the selected match is different from last one, the update step st will multiply

the attenuation factor λ. Because the attenuation factor is a decimal in range $(0, 1)$, so after sufficient iterations, the update step st will be small enough so that the threshold θ will stabilize at some value, which is our final threshold.

The algorithm cannot always achieve a good result, but if the value of F1-Measure with the threshold increases first and then decreases, which is typically the case, our algorithm can usually achieve a good value. Moreover, when the data is huge, our algorithm usually can get the result after a few iterations. That is to say the number of iteration will not increase much as the data becomes huge.

Algorithm 2. Threshold Selection

1. **Input:** The similarity set: S, an initial threshold: θ_0, an update step: st, an attenuation factor: λ.
2. **Output:** The threshold of the matching result: θ.
3. Normalize the similarity set S
4. Let θ be θ_0
5. **While** st is big enough
6. let $(e_S, e_D) = \min\{|similarity(e_S, e_D) - \theta|\}$
7. ask users to comfirm the match (e_S, e_D)
8. **if** (e_S, e_D) is correct
9. **if** last match is not correct
10. $st = st * \lambda$
11. **end if**
12. $\theta = \theta - st$
13. **else**
14. **if** last match is correct
15. $st = st * \lambda$
16. **end if**
17. $\theta = \theta + st$
18. **end if**
19. **end while**

4.2 Candidate Match Selection

One of the key points of ontology matching with active learning is to select informative matches to query. The most informative match means the match that can maximize the improvement of the matching performance. If the correctness of a match is found different from the matching result after the user confirmation, we call this match an *error match*. An error match is considered to be informative, because the result can be improved as long as the error is corrected. If the size of the data is small, or the original match result is already very good, this kind of improvement will be limited. If the data size is not small, or the original result is not good, we can also use the information of the *error match* to find other errors to significantly improve the matching result, which will be introduced in the next subsection.

The probability that a match is an error match is measured with *error rate*, and we propose three measures to estimate the error rate of a match as follows. Finally we combine these three measures to find the error matches.

Confidence The first measure we define is called *confidence*. Assume e_S and e_D are elements of the source ontology O_S and the target ontology O_D respectively. f is a similarity computing function of some ontology matching method M, and θ is its threshold. The confidence of M on a match (e_S, e_D) can be defined as follows:

$$Confidence(f(e_S, e_D)) = |\theta - f(e_S, e_D)| \tag{4}$$

The confidence can measure how sure the method M is about the correctness of the match. So the match with least confidence is most possible to be an error match, which is called least confidence selection.

If there are k ontology matching methods of different types: $\{M_1, M_2, ..., M_k\}$, we can extend the least confidence selection as follows:

$$Q = \min\{\sum_{f_i \in \{M_1, M_2, ..., M_k\}} w_i * |\theta_i - f_i(e_S, e_D)|\} \tag{5}$$

In the formula, f_i is one of the similarity computing functions of different ontology matching methods $\{M_1, M_2, ..., M_k\}$, θ_i and w_i are its threshold and weight respectively. Q is the match selecting function. It means that the similarity of a match is closer to the threshold, it is more possible to be an error match.

Similarity Distance The second measure is named *similarity distance*. Assume e_S is an element from the source ontology O_S, and method M maps the element e_S to e_D and $e_D^{'}$, which are two elements from the target ontology O_D. If the difference of $f(e_S, e_D)$ and $f(e_S, e_D^{'})$ is very small, there is very likely to be *error matches* in these two matches. Finally, we select the match that has the minimum difference. Formally, similarity distance is defined as:

$$SD(e_S, e_D) = \min\{|f(e_S, e_D) - f(e_S^{'}, e_D^{'})|\}; \quad (e_S = e_S^{'} \text{ or } e_D = e_D^{'}) \tag{6}$$

The similarity distance is very efficient in a one-to-one matching, in which most methods only select the best one from the similar matches.

Contention Point We define another measure, named, *contention point*, to find mistakes from the contention of different methods. This measure is defined based on the results of the matching results of some other algorithms such as edit distance or vector-space based similarity. The contention point is defined as:

$$ContentionPoint = \{< (e_s, e_D), ? > \in U | \exists i, j \ st. \ R_i(e_S, e_D) \neq R_j(e_S, e_D)\} \tag{7}$$

For a match (e_s, e_D), some of the k methods $\{M_1, M_2, ..., M_k\}$ consider it as matched, while the others consider not. Thus there must be mistakes among these methods, that is to say it is likely to be an error match. The contentious degree of a contention point can be further defined as:

$$Q = \min_{(e_S,e_D)\in ContentionPoint} \{ \max_{f_i\in\{M_1,M_2,...,M_k\}} Confidence(f_i(e_S, e_D)) - \max_{f_j\in\{M_1,M_2,...,M_k\}} Confidence(f_j(e_S, e_D))\}; \quad (8)$$
$$f_i(e_S, e_D) \neq f_j(e_S, e_D)$$

Intuitively, a contention one indicates that for the given match some methods consider it correct and some others consider not. That is to say that the matching algorithms have a maximal disagreement with the similarity confidences. In this case, the final matching result is very likely to be an error match.

4.3 Correct Propagation

When the size of the ontology is huge, correcting the selected error match only is far from sufficient. It is desirable that an algorithm can propagate the supervised information to help to correct the other potential error matches between the two ontologies.

Based on this consideration, we propose a *correct propagation* algorithm, which aims at detecting related error matches to the selected one. Thus when selecting a match to query, we need consider not only the error rate but also the effect of the error match on others, where the effect on others is called *propagation rate*.

Firstly, we introduce the concept of similarity propagation graph, which comes from the algorithm of similarity flooding [18]. A similarity propagation graph is an auxiliary data structure derived from ontologies O_S and O_D. The construction of propagation(PG) abides the principle as follows:

$$((a, b), p, (a_1, b_1)) \in PG(O_S, O_D) \Longleftrightarrow (a, p, a_1) \in O_S \; and \; (b, p, b_1) \in O_D \quad (9)$$

Each node in the propagation graph is an element from $O_S \times O_D$. Such nodes are called map pairs. The intuition behind arcs that connect map pairs is the following. For map pairs (a, b) and (a_1, b_1), if a is similar to b, then probably a_1 is somewhat similar to b_1. Figure 3 gives an example of the propagation graph.

For every edge in the propagation graph, it adds an additional edge going in the opposite direction against the original one. The weights placed on the edges of the propagation graph indicate how well the similarity of a given map pair propagates to its neighbors and back. These so-called propagation coefficients range from 0 to 1 inclusively and can be computed in many different ways.

Our algorithm of *correct propagation* is also based on the propagation graph, but we both consider the negative and active effects of the propagation arcs. According to the character of the propagation graph, for a map pair (a, b) and (a_1, b_1), if a is not matched with b, then probably a_1 is not matched with b_1. With the measurement of error rate,

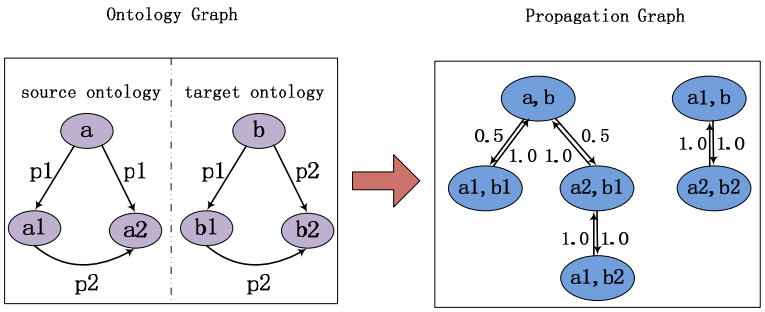

Fig. 3. Example of the similarity propagation graph.

error matches are easier to be detected, and we can correct more error matches related to the confirmed match according to the propagation graph, which is called correct propagation.

Before introducing the propagation, we consider the match selection again. To correct more error matches, we should not only consider the error rate, but also the propagation rate which measures the influence ability of a match. It mainly includes two factors: first, the number of matches that a match can influence. The bigger of the number, the range that the match can influence is wider, accordingly it is possible to correct more error matches. Second, the similarity differences between the match and its related matches. If the similarity difference is big, it is very possible to be error match among the match and its related ones.

Now, our match selection is according to the calculation of both error rate and propagation rate. When the correction (or confirmation) of the selected matches is provided by users, we conduct the correct propagation to update all the matches. Taking Figure 3 as an example, assume that we select the match (a_2, b_1) to query, and it is proved to be an error match. If the match (a_2, b_1) is confirmed to be an error by the user, then the similarities of the matches (a, b) and (a_1, b_2) which are related to the match (a_2, b_1) would be decreased. On the contrary, if the match (a_2, b_1) is confirmed to be correct, then the similarities of (a, b) and (a_1, b_2) should be increased. The update (decrease or increase) should be related to the similarities of the selected match, the error rates of related matches, and the weight of the arcs. Therefore, we can define the following update rules:

$$\begin{aligned} sim(a_i, b_i) = sim(a_i, b_i) + \alpha * w((x, y), (a_i, b_i)) \\ *(1 - sim(x, y)) * (1 - er(a_i, b_i)); \\ (x, p, a_i) \in O_S, (y, p, b_i) \in O_D \end{aligned} \tag{10}$$

$$\begin{aligned} sim(a_i, b_i) = sim(a_i, b_i) - \alpha * w((x, y), (a_i, b_i)) * sim(x, y) * er(a_i, b_i); \\ (x, p, a_i) \in O_S, (y, p, b_i) \in O_D \end{aligned} \tag{11}$$

In the formula, the match (x, y) is the selected error match, and $sim(x, y)$ is its similarity degree. The match (a_i, b_i) is one of the matches related to the match (x, y), and $w((x, y), (a_i, b_i))$ is the weight of their relation, and $er(a_i, b_i)$ stands for the error rate of the match (a_i, b_i), and α is an effect factor which is used to control the rate of the propagation. If the match (x, y) is correct (by the user), the update function uses Formula 10, else it uses Formula 11.

The correct propagation runs in an iterative process. In each iteration, it selects the match for user feedback with the error rate and the propagation rate, and then let users to confirm the selected match. After the confirmation, it updates the similarity degree, error rate and the propagation rate of related matches. Then it repeats this process until convergence (e.g., no any change) or the number of query times reaches a predefined threshold.

5 Experiments

We present details of the experiments in this section.

5.1 Experiment Setup, Data, and Evaluation Methodology

We implement all the algorithms using Java 2 JDK version 1.6.0 environment. All experiments are performed on a PC with AMD Athlon 4000+ dual core CPU (2.10GHz) and 2GB RAM Windows XP Professional edition OS.

Data sets For our experiments of the first two groups, we use the OAEI 2008 30x benchmark [2]. There are four data sets in the group of benchmark 30x, in which each size is no more than 100 concepts and relations. The traditional matching results on these data sets is very high, hence it is very suitable for the first two experiments. For the experiment of the correct propagation, we use part of the OAEI 2005 Directory benchmark [1], which consists of aligning web sites directory (like open directory or Yahoo's) with more than two thousand elementary tests. The reason we select this data set lies in its available ground truth and its low matching accuracy by the traditional methods [16].

Platform We conduct all the experiments on the ontology matching platform RiMOM [26], which is a dynamic multi-strategy ontology alignment framework. With RiMOM, we participated into the campaigns of the Ontology Alignment Evaluation Initiative (OAEI) from 2006 to 2008, and our system is among the top three performers on the benchmark data sets.

Performance Metrics We use precision, recall, F1-Measure to measure the performance of the matching result. They are defined next.

Precision: It is the percentage of the correct discovered matches in all discovered matches.

Recall: It is the percentage of the correct discovered matches in all correct matches.

F1-Measure: F1-Measure considers the overall result of precision and recall.

$$F1 - Measure = 2(Precison * Recall)/(Precision + Recall) \quad (12)$$

5.2 Threshold Selection

We first analyze the performance of our approach for threshold selection. Figure 4 shows the results on the OAEI 2008 benchmark 301 [2], and the matching method is a combination of KNN [3], Edit Distance [13] and the method using the thesaurus WordNet [4].

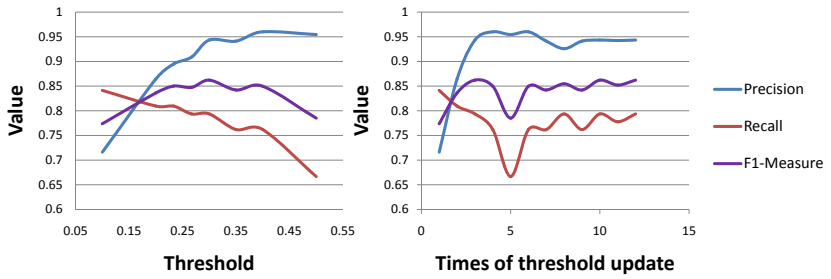

Fig. 4. Performance of threshold selection on OAEI 2008 benchmark 301.

The left one in Figure 4 shows the relationship between thresholds and performances of matching results (precision, recall and F1-Measure), and we can see it is consistent with our point introduced in section 4 except a few dithering points. The right one in Figure 4 present the result of our approach.

5.3 Measurements of Error Match Selection

In this subsection, we evaluate the effectiveness of the different strategies for error match selection: confidence, similarity distance and contention point.

Figure 5 is an experiment on the OAEI 2008 benchmark 304. From the precision figure (left), we note that the measurement combined least confidence and similarity distance performs much better than others. But after about 10 matches confirmed, it is hard to keep improving the matching accuracy. The reason is that the size of the ontology is small, and the original matching accuracy is already high.

Figure 6 is another experiment on the OAEI 2008 benchmark 301. The results are very similar to Figure 5. From the recall figure (right) we note that it improves the recall slightly. While the recall figure (right) of Figure 5 has no improvement. The reason why the recall has little improvement is that the thresholds chosen for the original matching results are very low, and almost all the matches with similarity lower than the threshold are not correct ones. Our approach can only correct the errors. Thus a draft conclusion

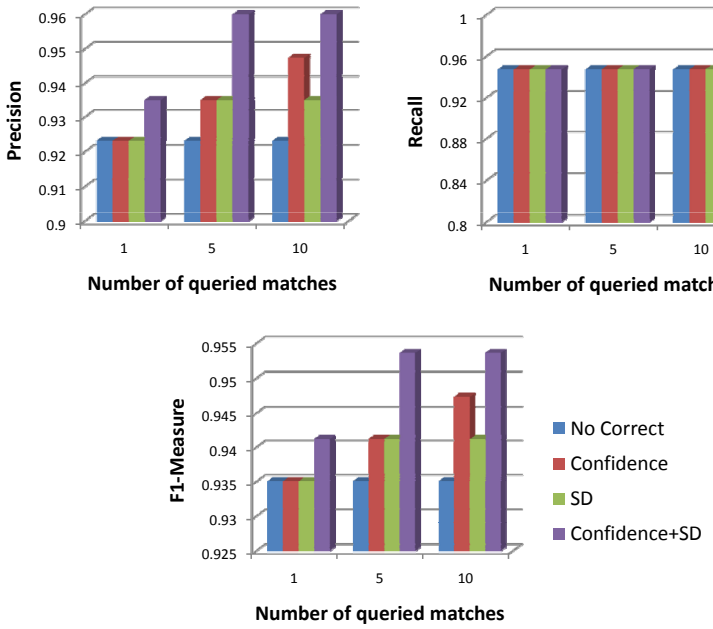

Fig. 5. Performance of matching after correcting error matches on OAEI 2008 benchmark 304.

is that if there are no error matches below the threshold, the approach cannot improve the recall value.

Figure 7 is an experimental result on the OAEI 2008 benchmark 302, which is the best result of all the four benchmarks. From the figure we note that the measurement combined with least confidence, similarity distance and contention point improves fastest, but these measurements themselves improve slightly. This confirms us that combining these three measurements is useful for ontology matching.

5.4 Correct Propagation

Figure 8 is an experiment on the approach of correct propagation with the OAEI 2005 Directory benchmark [1]. From the precision figure (left) we note that the result of correct propagation is much better than the approach of just correcting error matches. This implies that after propagation, more error matches are corrected with the selected one. Sometimes, the selected match is not an error match, so the approach of correcting error matches has no improvement, but the approach of correct propagation has. From the F1-Measure figure (below), it is not surprising that the approach of correct propagation grows faster than the others. Moreover, we find that the curve is steeper at the beginning. The reason is that the first few matches have bigger propagation rate, which means it can help to find more error matches.

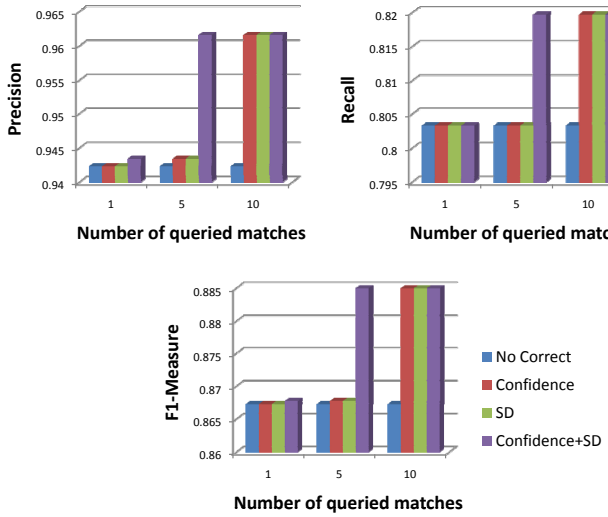

Fig. 6. Performance of matching after correcting error matches on OAEI 2008 benchmark 301.

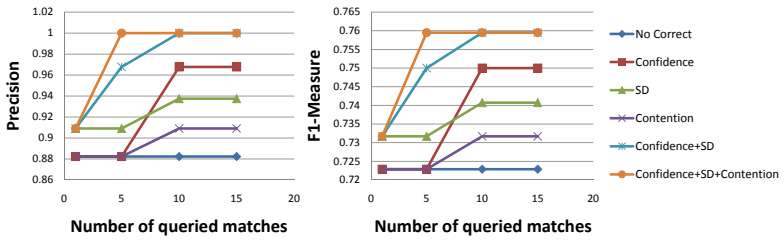

Fig. 7. Performance of matching after correcting error matches on OAEI 2008 benchmark 302.

5.5 Summary

We summarize the experimental results as follows.

- First, in most cases our method of threshold selection can efficiently find a good threshold after a few queries.
- Second, all the three measures for match selection can help find the error matches, which are helpful to improve the matching result.
- Third, our approach of correct propagation can further improve the matching result. The improvement is more significant at the beginning than later. This also satisfies the limit of user feedback, that is also the reason we can improve the matching result greatly via only a few queries.

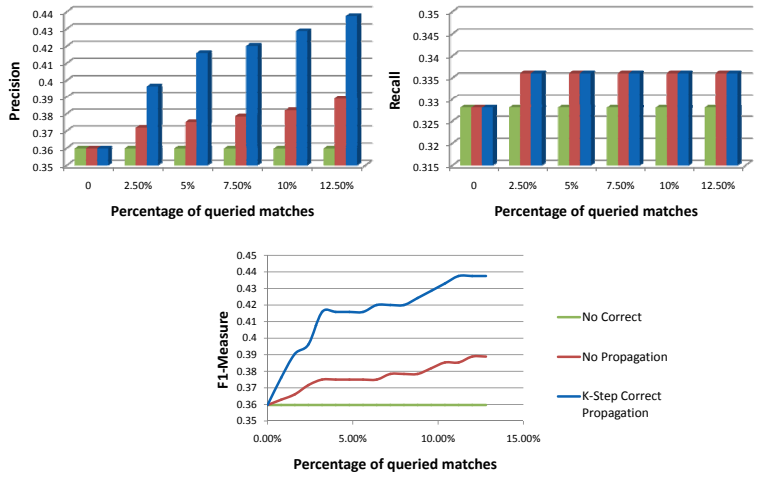

Fig. 8. Performance of matching after correct propagation on OAEI 2005 Directory.

6 Related Work

6.1 Ontology Matching

Many works have addressed ontology matching in the context of ontology design and integration [6][17][19][21]. Some of them use the names, labels or comments of elements in the ontologies to suggest the semantic correspondences. [7] gives a detailed compare of various string-based matching techniques, including edit-distance [13] and token-based functions, e.g., Jaccard similarity [25] and TF/IDF [22]. Many works do not deal with explicit notions of similarity. They use a variety of heuristics to match ontology elements [17][19].

Some other works consider the structure information of ontologies. [15] uses the cardinalities of properties to match concepts. The method of similarity flooding is also an example using structure information [18]. Another type of method utilizes the background knowledge to improve the performance of ontology matching. For example, [4] proposes a similarity calculation method by using thesaurus WordNet. [12] presents a novel approximate method to discover the matches between concepts in directory ontology hierarchies. It utilizes information from Google search engine to define the approximate matches between concepts. [5] makes semantic mappings more amenable to matching through revising the mediated schema. Other methods based on instances of ontologies [28] or reasoning [27] also achieve good results.

6.2 Active Learning

Active learning can be viewed as a natural development from the earlier work on optimum experimental design [11]. This method is widely used in the domain of machine learning. [23] introduces an active-learning based approach to entity resolution that

requests user feedback to help train classifiers. Selective supervision [14] combines decision theory with active learning. It uses a value of information approach for selecting unclassified cases for labeling. Co-Testing [20] is an active learning technique for multi-view learning tasks.

There are many works addressing ontology matching with user interaction, like GLUE [8], APFELi [9], [28], etc. Nevertheless, the annotation step is time-consuming and expensive, and users are usually not patient enough to label thousands of concept pairs for the relevance feedback. So our approach takes the concept of active learning to alleviate the burden of confirming large amounts of candidate matches, and the measurements we propose are based on the features of ontology. Our approach of correct propagation uses the propagation graph as the approach of similarity flooding [18]. The difference is that we propagate the similarity partly and purposely, and do not do it up and down. Our approach is more focused and more efficient than similarity flooding.

7 Conclusion and Future Work

In this paper we propose an active learning framework for ontology matching. But the framework is just a shell, and what separates a successful instantiation from a poor one is the selection of matches to query and the approach to improve the traditional matching result with the confirmed matches by users. We present a series of measurements to detect the error match. Furthermore we propose an approach named correct propagation to improve the matching result with the confirmed error matches. We also present a simple but effective method of selecting the threshold with user feedback, which is also helpful for the error match selection. Experimental results clearly demonstrate the effectiveness of the approaches.

As the future work, one interesting direction is to explore other types of user feedbacks. In our experiments, we take the standard boolean answer as the user feedback. However, in some cases users cannot give a simple correct or not answer for the queried match, especially when the ontologies are defined for some special domain. One solution is to select matches that the users are familiar with for confirmation, or translate the match into a question that the users can easily answer. Another interesting topic is how to reduce the negative effect of user mistakes.

References

1. Oaei 2005 directory download site. In *http://oaei.ontologymatching.org/2005/*.
2. Ontology alignment evaluation initiative. In *http://oaei.ontologymatching.org/*.
3. T. Baily and A. K. Jain. A note on distance-weighted k-nearest neighbor rules. *IEEE Transaction System Man Cybern*, 8(4):311–313, 1978.
4. A. Budanitsky and G. Hirst. Evaluating wordnet based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.
5. X. Chai, M. Sayyadian, A. Doan, A. Rosenthal, and L. Seligman. Analyzing and revising mediated schemas to improve their matchability. In *Proceedings of the VLDB Conference 2008*, 2008.
6. H. Chalupsky. Ontomorph: A translation system for symbolic knowledge. In *In Principles of Knowledge Representation and Reasoning*, pages 471–482, 2000.

7. W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 73–78, 2003.
8. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. *Ontology matching: A machine learning approach*. Springer-Velag, 2003.
9. M. Ehrig, S. Staab, and Y. Sure. Bootstrapping ontology alignment methods with apfel. In *Proceedings of the 4th International Semantic Web Conference*, pages 186–200, 2005.
10. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2007.
11. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, 1972.
12. R. Gligorov, Z. Aleksovski, W. Kate, and F. Harmelen. Using google distance to weight approximate ontology matches. In *Proceedings of the 16th International World Wide Web Conference (WWW)*, pages 767–776, 2007.
13. D. Gusfield. *Algorithms on strings, trees, and sequences*. Cambridge University Press, 1997.
14. A. Kapoor, E. Horvitz, and S. Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *Proceedings of the IJCAI Conference 2007*, pages 877–882, 2007.
15. M. Lee, L. Yang, W. Hsu, and X. Yang. Xclust: Clustering xml schemas for effective integration. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM)*, pages 292–299, 2002.
16. Y. Li, Q. Zhong, J. Li, and J. Tang. Result of ontology alignment with rimom at oaei07. pages –1–1. OM, 2007.
17. D. McGuinness, R. Fikes, J. Rice, and S. Wilder. The chimaera ontology environment. In *Proceedings of the 17th National Conference on Articial Intelligence (AAAI)*, pages 1123–1124, 2000.
18. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of 18th International Conference of Data Engineering (ICDE)*, pages 117–128, 2002.
19. P. Mitra, G. Wiederhold, and J. Jannink. Semi-automatic integration of knowledge sources. In *Proceedings of the 2nd International Conference On Information FUSION*, 1999.
20. I. Muslea. Active learning with multiple views. *PhD thesis, Department of Computer Science, University of Southern California*, 2002.
21. N. Noy and M. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the National Conference on Articial Intelligence (AAAI)*, pages 450–455, 2000.
22. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, pages 513–523, 1988.
23. S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–278, 2002.
24. P. Shvaiko and J. Euzenat. Ten challenges for ontology matching. In *On the Move to Meaningful Internet Systems (OTM)*, 2008.
25. P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. 2005.
26. J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang. Using bayesian decision for ontology mapping. *Web Semantics*, 4(4):243–262, 2006.
27. O. Udrea, L. Getoor, and R. J. Miller. Leveraging data and structure in ontology integration. In *Proceedings of the 26th International Conference on Management of Data (SIGMOD'07)*, pages 449–460, 2007.
28. S. Wang, G. Englebienne, and S. Schlobach. Learning concept mappings from instance similarity. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2008)*, pages 339–355, 2008.