http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet

Asian Network for Scientific Information

308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 8 (3): 378-382, 2009
ISSN 1812-5638

A Hybrid Path Matching Algorithm For XML Schemas

[1]A. Rajesh and [2]S.K. Srivatsa
[1]Dr. MGR Educational and Research Institute, Maduravoyil, Chennai, Tamil Nadu, India
[2]St. Joseph's College of Engineering, Chennai, India

Abstract: The approach proposed in this study uses a simple path matching algorithm to perform the structural matching. The novelty in this approach is that the path matching algorithm considers only the paths to leaf nodes in the schema trees for the matching process there by eliminating the need for repeatedly parsing the elements of the schema tree as in the other approaches. This greatly reduces the time required to identify the matches. And the paths are treated as a set of strings comprising of the labels of the nodes in the path. Treating the paths as set of strings greatly simplifies the matching process as the same approaches used in the linguistic matching process can be used in the path matching process.

Key words: XML schema integration, schema integration, XML schema matching, schema matching, hybrid schema matching, semantic matching

INTRODUCTION

Schema matching is a vital and critical step in many application domains such as semantic web, integration of web oriented data, e-commerce, schema evolution and migration, application evolution, data warehousing, database design, XML message mapping, XML-relational data mapping and schema integration. A schema matching process takes two schemas as input and returns a set of matching element pairs which are semantically related to one another between the two schemas (Madhavan *et al*., 2001; Melnik *et al*., 2002; Do *et al*., 2002; Bernstein *et al*., 2004; Aumueller *et al*., 2005).

Schema matching is still at large a manual process and hence, is highly time consuming, error-prone and expensive. Hence, automation of this process aids in a faster, less labor intensive integration approach which is crucial for large scale data integration systems. We have concentrated in this study on matching XML schemas, due to the popularity of XML model, the large proliferation of XML documents on-line and the emergence of XML as a defacto standard for sharing data among sources. Though there are several methods proposed in the literature for the schema matching process, there is still a large space for improvement in terms of accuracy and reduction in complexity.

The matching process in general comprises of linguistic matching phase, a structural matching phase (Madhavan *et al*., 2001; Do *et al*., 2002; Aumueller *et al*., 2005) and in some approaches a filtering phase to extract the matching element pairs from the two schemas (Melnik *et al*., 2002). The linguistic matching process identifies linguistic similarity between the elements that are similar in meaning. The structural matching process identifies similarity between elements based on their structure i.e., based on the number of similar attributes or sub elements they have.

In this study, a novel method for identifying element similarities using a simple path matching algorithm has been proposed.

A brief survey of some of the most successful schema matching approaches found in the literature will help in understanding the current direction of research in the schema matching field. All these approaches in general compute the similarity between elements of the two schemas as a value in the interval 0 and 1, where, a value of 1 means the elements are identical and 0 means completely dissimilar.

Cupid (Madhavan *et al*., 2001) proposes a hybrid schema matching algorithm to compute the similarity between the schema elements. The matching algorithm operates on schema trees. It first computes the linguistic similarity between the schema elements based on morphological normalization, categorization, string-based techniques and a thesauri look-up. Then it computes structural similarity between the schema elements. It computes the final similarity coefficient combining the weighted linguistic similarity measure and weighted structural similarity measure. The chosen matching pairs are those whose similarity coefficients are greater than a given threshold.

SF (Melnik *et al*., 2002) proposes a structural algorithm to match diverse data structures like data schemas, data instances, or a combination of both. It first

Corresponding Author: A. Rajesh, Dr. MGR Educational and Research Institute, Chennai, India

converts the schemas into directed labeled graphs. It then uses an iterative fix point computation to find the similarities between the schema elements. The similarity computation relies on the intuition that elements are similar if their adjacent elements are similar. It also proposes several filters for extracting the best matching pairs from the resultant candidate matches returned by the algorithm.

LSD (Doan *et al*., 2001) uses machine learning techniques to semi automatically compute semantic mapping between schemas. It uses a set of learners like Whirl Learner, Naïve Bayesian learner, Name Matcher using TF/IDF similarity measure and also domain specific learners like County-Name Recognizer to learn the similarity patterns between the schema elements. The predictions of the individual learners are then combined using a meta-learner. The technique is extensible i.e., new learners can be added to it.

COMA (Do *et al*., 2002) proposes a flexible composite match approach to combine different match algorithms. It uses multiple schema level matchers exploiting linguistic, data-type, structural information and previous matches to perform the matching process. The novelty in present approach is that, we use a simple path matching algorithm in conjunction with the regular match approaches, there by improving the accuracy of the matching process.

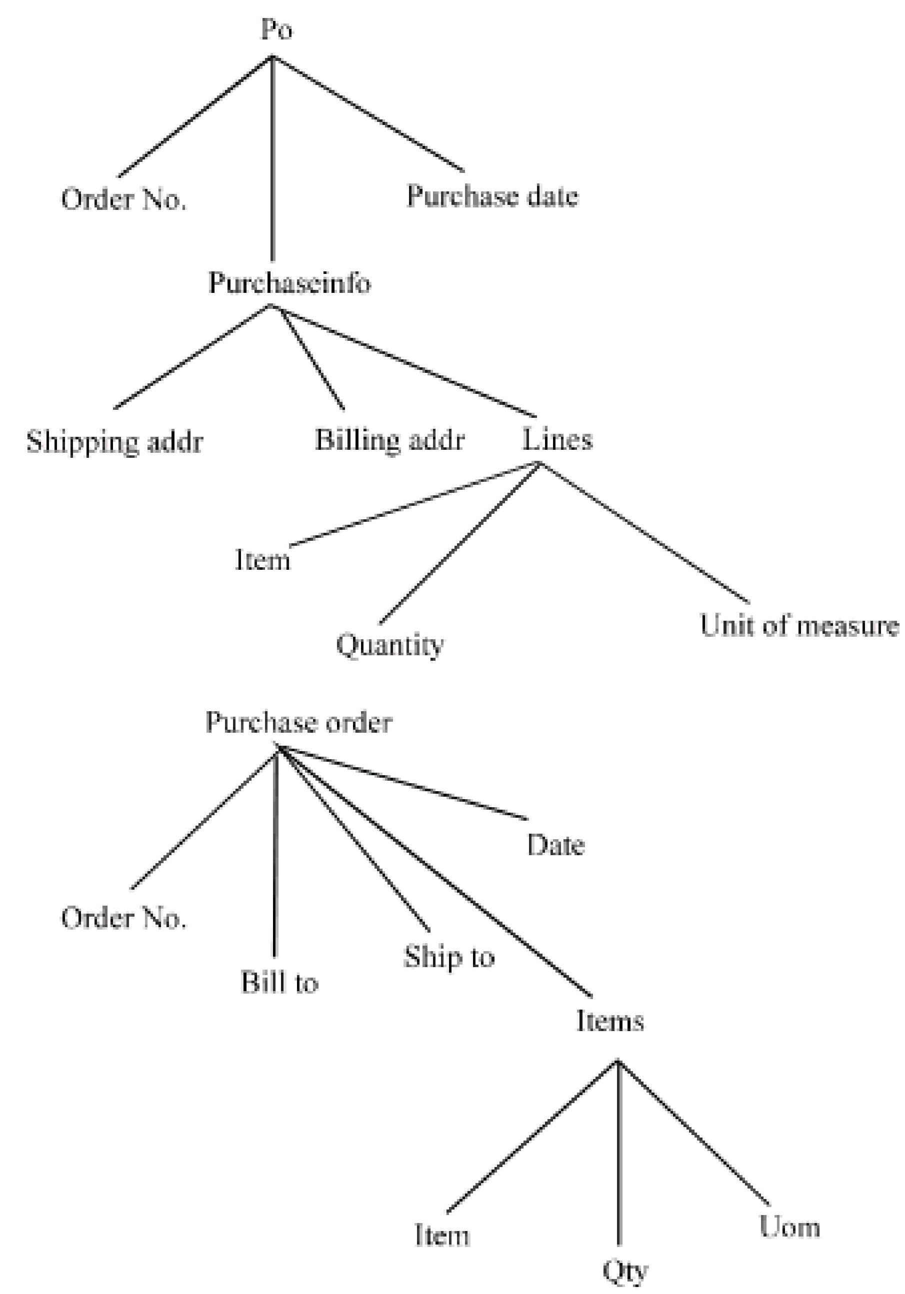

Fig. 1: Po, purchase order schema variant 1

A HYBRID PATH MATCHING ALGORITHM

The focus of the proposed approach is in the development of a simple path matching algorithm which is used for identifying candidate match pairs between the schema elements. The algorithm takes two schemas in the form of schema trees (Fig. 1) as input. First, it extracts the paths from the root of the schema tree to its various leaf elements. The algorithm treats each path in the schema tree as a vector of strings. It then matches each path of the source schema tree with the paths of the target schema tree and returns a path similarity matrix between the source and target schema tree paths.

The similarity between any two path is a value in the interval 0 and 1, where, 0 means that the paths are totally dissimilar and 1 means the paths are totally identical. The path similarity, pathsimmat is computed based on the number of similar elements between the paths. The element similarity simmat, is the summation of linguistic similarity measure LingSim computed using linguistic similarity computation techniques like-levenshtein distance, affix matching anddomain specific dictionaries and structural similarity measure StructSim, computed using a simple structural similarity computation technique-based on the number of linguistically similar children each pair of matched elements have.

The element pairs whose linguistic similarity measure is greater than a predefined threshold lcutoff, are considered to be similar. The element similarity measure, simmat thus computed is again a value in the interval 0 and 1, where, 0 means the elements are dissimilar and 1 means the elements are identical. Then for each path from source schema tree the path with the maximum similarity measure from the target schema tree is identified and if this value is greater than a predefined threshold, scutoff the paths are considered as similar. The complete algorithm is sown in Code 1.

```
Function HPathSim(stree1,Stree2) returns a path
similarity matrix
path1list=extractpaths(stree1)
path2list=extractpaths(stree2)
for i=1 to |path1list|{
for j=1 to |path2list|{
pathsimmat[i,j]=PathSim(path1[i],path2[j])
}
}
return pathsimmat.

Function PathSim(p1,p2)returns a simval between 0
and 1
for i=1 to |P1|{
for j=1 to |P2|{
simmat[i,j]=LingSim(p1[i],P2[j])+StructSim(p1[i],
p2[j])
```

```
}
}
rsum=0
for i=1 to |P1|{
max=0
for j=1 to |P2|{
if max> simmat[i,j]
max=simmat[i,j]
}
rsum=rsum+max
}
csum=0
for j=1 to |P2|{
max=0
for i=1 to |P1|{
if max> simmat[j,i]
max=simmat [j,i]
}
csum=csum+max
}
pathsim=rsum+csum/|P1|+|P2|
return pathsim
```

Code 1: PathSim algorithm

Once similar paths are identified, the element pairs from the matching path whose similarity measure simmat is greater than a given threshold, cutoff value is chosen as the candidate matching pairs.

EXPERIMENTATION

In order to evaluate the performance of this algorithm a prototype implementing this algorithm was built. The performance of the algorithm has been shown in terms of precision and recall, a common measure proposed in the literature (Do *et al*., 2002) used to test these types of systems. Precision and Recall are defined as:

Precision = Ratio of the correct matches identified to total number of matches returned

Recall = Ratio of the correct matches identified to total number of manually determined matches

The data sources were chosen from diverse domains so that the performance is not domain specific. The characteristics of data sources are shown in Table 1.

First the optimum values for the tunable parameters lcutoff, scutoff and cutoff value are obtained. This is done by gradually increasing the values of lcutoff and scutoff from 0 to 1 and trying to find the precision and recall of the algorithm for those values. From the experimentation it was found that for lcutoff = 0.4 and scutoff = 0.1 optimum precision and recall values resulted. The results are shown in Fig. 2. Similarly by varying the value of cutoff value from 0 to 1 for the optimum values of lcutoff and scutoff determined earlier, it was found that for cutoff value = 0.5 the algorithm gave optimum precision and recall values. The result of this experimentation is shown in Fig. 3.

Table 1: Characteristics of the data sources used in the experimentation

XML schemas	No. of elements	Max. depth	No. of leaves
Po, purchase order variant 1	10×9	4×3	7×7
Po, purchase order variant 2	13×15	4×4	8×8
Course schemas variant 1	14×16	4×4	10×12
Course schemas variant 2	14×20	4×5	10×15
University schemas	8×9	3×3	5×6
Statistic schemas	14×14	2×3	10×9
Supplier schemas	17×43	5×2	10×34

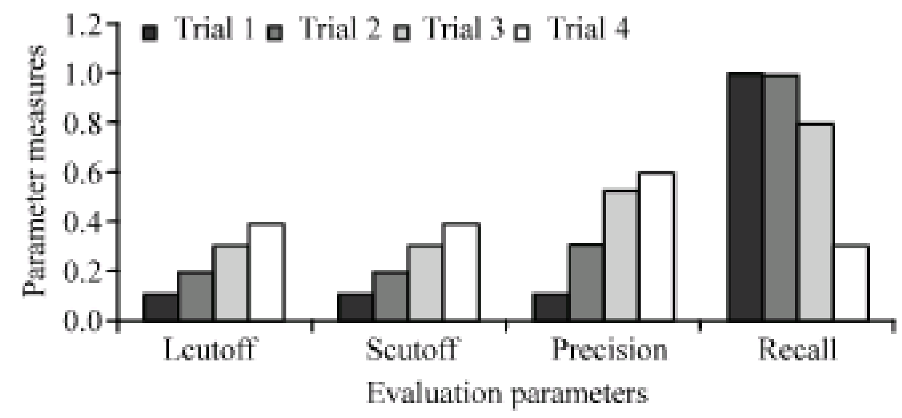

Fig. 2: Experimentation results of precision and recall for various values of lcutoff and scutoff

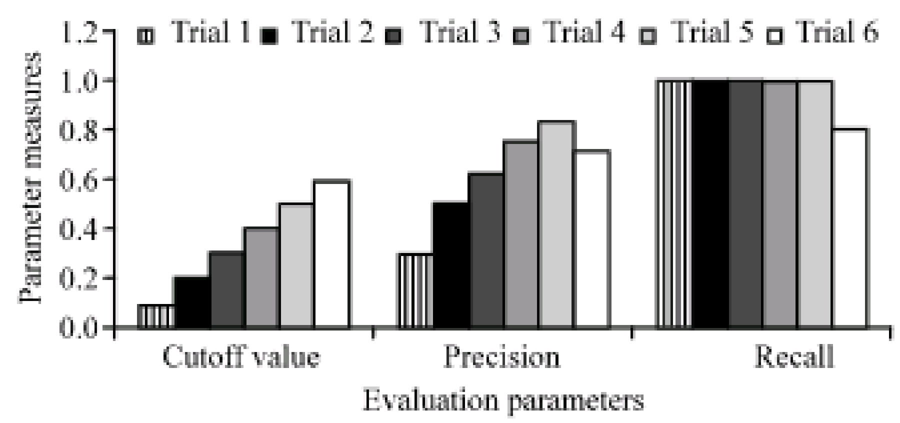

Fig. 3: Experimentation results for various of cutoff value

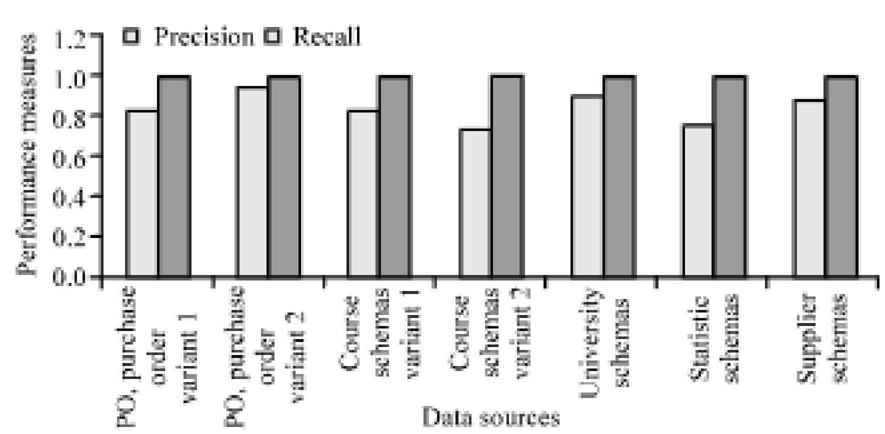

Fig. 4: Performance analysis of the algorithm for the different data sources

Then the performance of the algorithm was evaluated for optimum values of the tunable parameters lcutoff, scutoff and cutoff value that was determined in the earlier experimentation. The results of this experimentation are shown in Fig. 4. It can be seen that the precision and recall of the algorithm ranged between 0.75 to 1.0 which was quite impressive given the performances of similar approaches in the literature (Do *et al*., 2002; Doan *et al*., 2001; Madhavan *et al*., 2001). The snapshots of the prototype that we used for experimentation purpose are shown in Fig. 5 and 6.

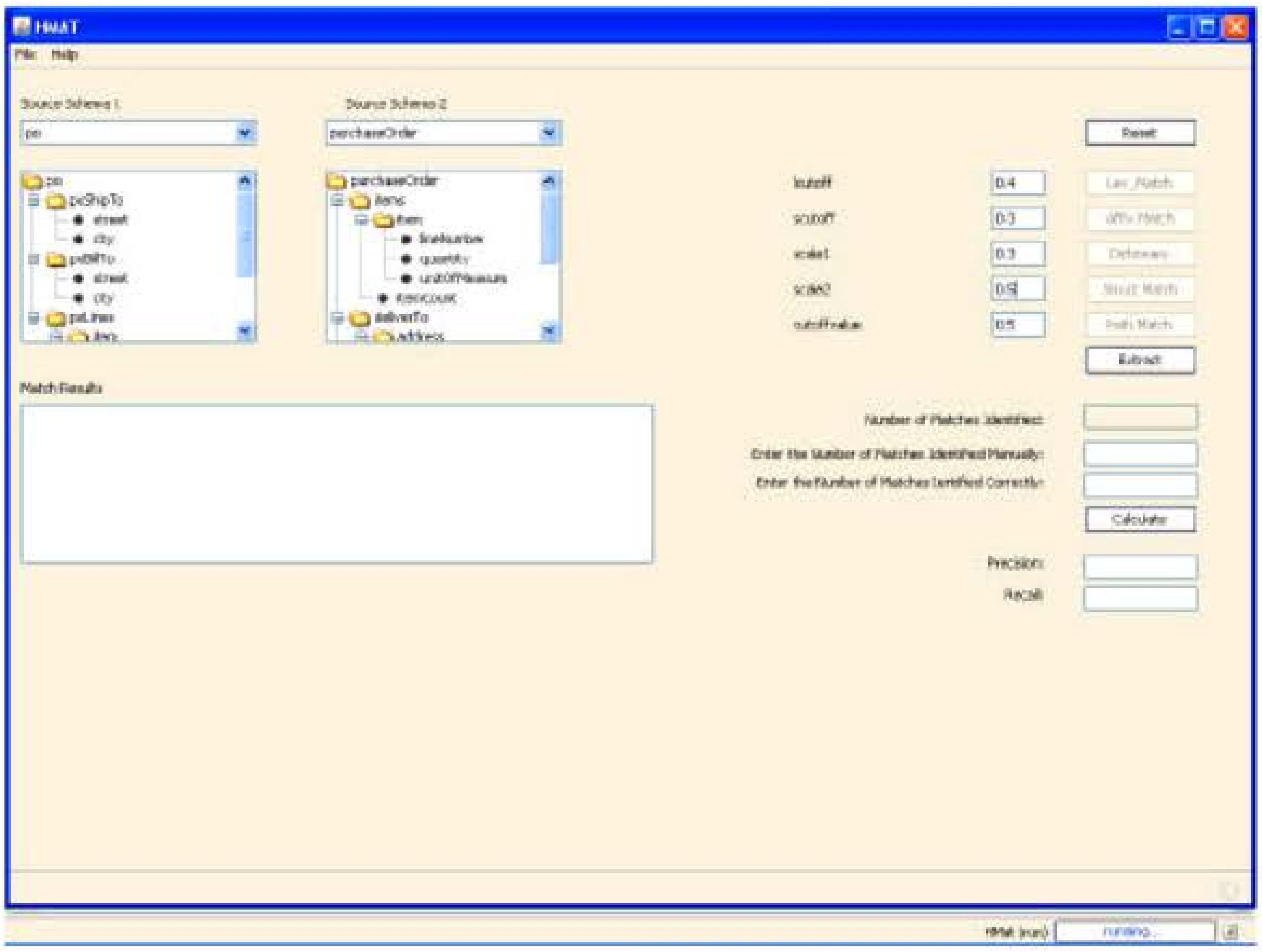

Fig. 5: Selection of schemas for matching purpose and setting up the various parameters for operation

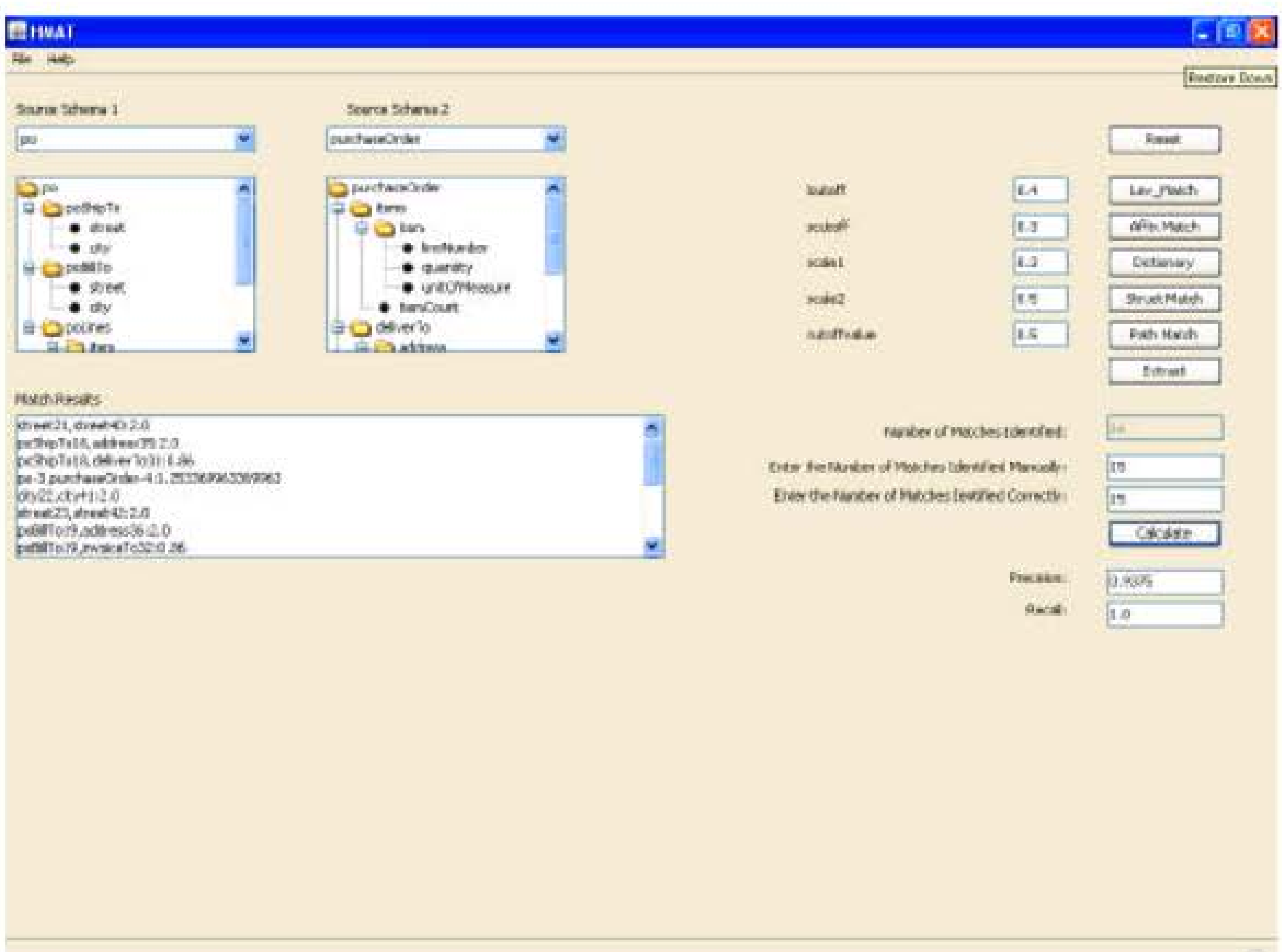

Fig. 6: Match results for the two selected schemas

CONCLUSION AND FUTURE WORK

The algorithm is extremely successful in identifying one to one correspondences between schema elements. Future directions are working on ways to upgrade the algorithm to identify other type of correspondences such as one to many, many to one and many to many correspondences. And working on ways to use user feedback and earlier match results to further enhance the performance of the algorithm.

REFERENCES

Aumueller, D., H.H. Do, S. Massmann and E. Rahm, 2005. Schema and ontology matching with COMA^{++}. Proceedings of ACM SIGMOD International Conference on Management of Data, Jun. 14-16, Baltimore, Maryland, USA., pp: 906-908.

Bernstein, P.A., S. Melnik, M. Petropolous and C. Quix, 2004. Industrial strength schema matching. SIGMOD Rec., 33: 38-43.

Do, H.H. and E. Rahm, 2002. COMA-a system for flexible combination of schema matching approach. Proceedings of 28th VLDB Conference, Aug. 20-23, Hong Kong, China, pp: 610-620.

Doan, A.H., P. Domingoas and A.Y. Halevy, 2001. Reconciling schemas of disparate data sources: A machine-learning approach. ACM SIGMOD Record, 30: 509-520.

Madhavan, J., P.A. Bernstein and E. Rahm, 2001. Generic schema matching with cupid. Proceedings of the 27th International Conference on very Large Data Bases, Sept. 11-14, Roma, Italy, pp: 49-58.

Melnik, S., H. Garcia Molinia and E. Rahm, 2002. Similarity flooding: A versatile graph matching algorithm. Proceedings of ICDE, Feb. 26-Mar. 1, San Jose CA, pp: 117-128.