# Web interface query integration in holistic matching approach using clustering algorithms

**Fatemeh Salahshour**[1], **Reza Tavoli**[2]

[1] **Department of computer,**
**Pooyandegane Danesh University Of Chaloos, Iran**

[2] **Department of electrical and computer,**
**Qazvin Islamic Azad University (QIAU) Qazvin, Iran**

**Abstract**

Information integration is an interesting research topic that has recently attracted the attention of researchers. Schema matching is One of the critical issues in the implementation of the plan integrate information. A large number of information sources available on the web, but their contents can be accessed through the query interfaces. As an important step to integrate data sources, we focus on the integration of query interfaces. Holistic schema matching are implemented to deal with the challenges of large-scale schemas. More specifically in the article focus on the integration of semantic mapping fields with the holistic schema matching. In this paper, clustering-based method for matching query interfaces are provided. This method is based on two general observations. First, because of the use of the dictionary is easier to find features that are semantically related to the same query interfaces. Secondly, related features identified in the interfaces can help to find related features in other interfaces. The proposed method, divided into 3 main steps, the first step of pre-processing dataset and automatic manufacturing of dictionary, second step is Completion of first dictionary by user, and the third step, uploading datasets and secondary dictionary and applying proposed algorithms on them. K-Means clustering algorithm was used to apply the proposed method. Algorithm K-Means is very useful for the matching of large-scale. And the results, shows the impact of the proposed methodology and the accuracy and performance of the proposed method. The method has a high accuracy in the range of returns. This method handles simple and complex matching.

***Keywords:*** *Information integration, query interfaces, holistic schema matching, clustering, web interface.*

## 1. Introduction

With the growth of heterogeneous data sources on the web, it is clear that integrating these resources has become a major problem. It is observed that a large number of data sources are hidden behind query interfaces [1], [2]. Related features finding process between schemas refers to schema matching [3].

Integration web query interfaces are divided into two phases [4]. Firstly, semantic mappings of fields on all specified interfaces; and the second step of integration interfaces based on the mapping specified in the first step. Clearly, precision mapping fields as output of the first step, is extremely important in successful integration. In this article we focus on the first step.

Although schema matching studied extensively [6], [8], [12], [13], [14], [15], integration of web query interfaces have attracted the most attention [4], [8]. The most problematic element in the integration process is the difference in naming and dissimilarity of data structures. This inequality has challenged integration process [16]. the process of schema matching has become an important aspect in various applications, such as query processing, e-business, data warehousing and Semantic Web [17] existing solutions can be grouped into two main types: pair-wise matching [5], [9], [10], a holistic matching [4], [6], [7], [8], [11]. Integration process In small-scale by identifying features concerned pair-wise and because of its limitations, these methods do not produce high-quality results [18]. On the other hand the holistic matching, adjust several schemas to find relevant features at the same time. Although the holistic matching benefits from many data, but also suffers from noisy data [3].

Clustering is the process of grouping data based on their similarity [19]. The problem of holistic schema matching, combined with the complexity of the problem is exponential, and clustering used as a central method for matching large-scale schemas and improve matching efficiency [20]. Recently, a number of methods used based clustering [3], [4], [7]. Methods based on clustering use K-means algorithm or agglomerative hierarchical algorithms [21] to improve the efficiency of their holistic matching.

In this paper, clustering-based method for matching query interfaces are provided. This method is based on two general observations. First, because of the use of the dictionary is easier to find features that are semantically related to the same query interfaces. Secondly, related

features identified in the interface can help to find related features in other interfaces. The proposed method, is divided into three main steps. The first step, pre-processing datasets and automatic manufacturing of dictionary, second step is completion of first dictionary by user, and the third step, uploading datasets and secondary dictionary and applying proposed algorithms on them.
The rest of the article is as follows. Section 2 shows difference of the method with relevant work done in this area. Section 3 describes our proposed approach and 3-1, 3-2, 3- 3 sections expand our proposed method. Section 4 describes the dataset used to test the proposed method. Section 4.3 reports the results of experiments. Section 5 concludes the paper.

## 2. Related Work

Although much work has been done in the area of schema matching, most work has focused on the pair-wise schema matching [5], [9], [10]. Recently holistic schema matching [4], [6], [7], [8], has gained greater attention because it affects explore information in text and its scalability [20].
The method proposed by Pei and his colleagues uses only the k-mean clustering [20]. This method adjust characteristics all at once depending on the similarity measures that are used by the k-mean algorithm. This method consists of three clustering steps: clustering schemas, the clustering fields in clusters of similar schemas, and clustering fields of all the different clustering schemas. Unfortunately, the third step has not been clearly demonstrated, and only use the name, type of data, and label to measure the similarity between characteristics in various schemes.
Wu and colleagues [7] offered an interactive cluster-based approach for the matching of query interface, which records hierarchical nature of the interfaces. In this way agglomerative hierarchical clustering algorithm is used. Similarly of pair-wise matching which re-use of existing mapping, this method also uses bridging effect.
Alofairiand Ahmad in [19] provided an effective way to holistic schema matching to reduce the search space. They used the method of k-mean clustering techniques and agglomerative clustering algorithm. Combination of the two sequences, produce a state of strong clusters and improve the clustering process efficiency. To achieve the main goal, this method consists of two preprocessing and clustering steps. The results of this study show that their proposed matching method reduces the search space using integrating techniques to clusters, which immediately classifies relevant fields in the Similar Clusters. The results prove that this method is effective and promising in holistic schema matching and leads to better results than proposed method by Pei and colleagues [3] and Wu and colleagues [7].
He and colleagues [4] suggested a method based on weight and two-step clustering for finding related fields that is similar to the method proposed by pei and colleagues [3]. Wise performance depends on the quality of input data and structure of synonymous semantic relationships.
Madhavan and colleagues [11] have proposed a method based on the Corpus which uses a series of initiatives as a training set to improve future adjustments. The method learns from previous results for benchmarking initiatives and to this uses algorithms with Supervisor.

## 3. The proposed approach

The proposed method, divided into 3 main steps, the first step, pre-processing of datasets and automatic manufacturing of dictionary, second step is Completion of first dictionary by user, and the third step, uploading datasets and secondary dictionary and applying proposed algorithms on them. Figure 1 show block diagram of our approach.

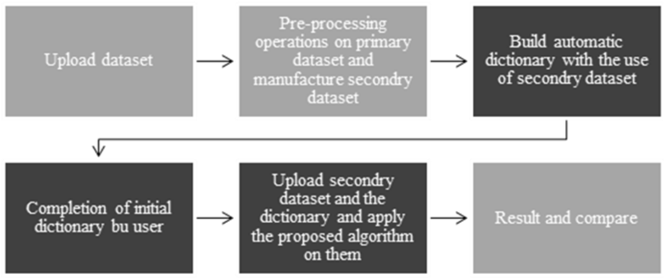
Upload dataset
Pre-processing operations on primary dataset and manufacture secondry dataset
Build automatic dictionary with the use of secondry dataset
Completion of initial dictionary bu user
Upload secondry dataset and the dictionary and apply the proposed algorithm on them
Result and compare

Fig. 1 Proposed approach block diagram.

### 3.1 Pre-processing of datasets and dictionary automated manufacturing

Our main goal at this step apply pre-processing operations on datasets and is making the dictionary. According to Figure 2, this involves two main steps:

**First step:** pre-processing operations on primary dataset and manufacture secondary dataset
In this step, we can apply the initial dataset as input and pre-processing operations on it. For pre-processing operations requires survey all existing schemas in the dataset and fields in each dataset. In pre-processing step, following operations carried out.
1) Converting all files to lowercase.
2) Filtering dataset - remove prepositions, delete

and edit additional characters.

3) Remove the distance on both sides of the field.
4) Remove empty array of dataset.
5) Save the results - dataset after applying pre-processing operations - on txt files.

Table 1 shows dataset before pre-processing, consider the following dataset after pre-processing operations applying the changes. In table 2 you can see clearly pre-processing impact on dataset, which has optimized it.

```
Input: (D)
D: the input dataset
Output: Dataset after preprocessing and automatic dictionary
Steps:
  Step 1: Preprocessing
    For each S_i in D Do: /* separated by enter */
      For each a in S_i Do: /* separated by ^ */
        Change capital dataset to small
        Filtering dataset
          Remove and edit preposition(or ,of,# ,$) Remove and
          edit extra characters ("-", "/", ",")
        Remove spaces from two side of fields
        Delete the empty arrays from dataset
        Store the result in a txt file /* keeping separate schemas
        by enter & fields by ^ */
  Step 2: Automatic dictionary
    Remove digits, Remove and edit preposition (with, the, in, do,
    by, on, for, if, so, to)
    Vacuum ()
    Takes an input array (R) and returns a new array without
    duplicate values
    Store the result in a txt file /* keeping separate schemas by
    enter & fields by ^ */
```

Fig. 2 Pseudo-code for the first step

Table 1: dataset before pre-processing

| *Id* | *Schema* |
|---|---|
| 1 | From ^ To ^ Adults ^ Children ^ Class of service ^ Number of connections |
| 2 | From ^ To ^ Depart ^ Returns ^ Adult ^ Cabin |
| 3 | Leaving from ^ Destination ^ # of adults |
| 4 | Departing from ^ Going to ^ Depart ^ Returns |
| 5 | From city ^ To city ^ Departure date ^ Return date ^ Adult(s)(over 12) ^ Children(under 12) ^ Infant(s)(under 2) |
| 6 | From ^ To ^ Flight departure date ^ Adults ^ children ^ Booking class |
| 7 | Leaving from (city or airport) ^ Going to (city or airport) ^ Departing on ^ Returning on ^ Adults ^ Children ^ Infants ^ Flight class |
| 8 | From ^ To ^ Depart ^ Return ^ Adults 12-64 ^ Children ^ Seniors ^ Class ^ With a maximum of connections |

Table 1 shows dataset before pre-processing, consider the following dataset after pre-processing operations applying the changes. In table 2 you can see clearly pre-processing impact on dataset, which has optimized it.

TABLE 2: DATASET AFTER PRE-PROCESSING

| *Id* | *Schema* |
|---|---|
| 1 | from ^ to ^ adults ^ children ^ class of service ^ number of connections |
| 2 | from ^ to ^ depart ^ returns ^ adult ^ cabin |
| 3 | leaving from ^ destination ^ number of adults |
| 4 | departing from ^ going to ^ depart ^ returns |
| 5 | from city ^ to city ^ departure date ^ return date ^ adult ^ children ^ infant |
| 6 | from ^ to ^ flight departure date ^ adults ^ children ^ booking class |
| 7 | leaving from ^ going to ^ departing on ^ returning on ^ adults ^ children ^ infants ^ flight class |
| 8 | from ^ to ^ depart ^ return ^ adults 12 64 ^ children ^ seniors ^ class ^ with a maximum of connections |

**Second step:** Build automatic dictionary with the use of secondary dataset

The dictionary automatically build of the dataset extracted from the first step. We need to do the following steps to build the dictionary:

1) Remove numbers, delete and edit prepositions.
2) Run vacuum function - in section 3.1.1 we have explained this function in details.
3) We receive the Output of vacuum function and we'll remove duplicate values in the dictionary.
4) Saving the Results - the first dictionary - in txt file.

### 3.1.1 Vacuum function

As shown in Figure 3, this function takes dataset obtained in the first step of figure 2, as its input, and produces automatic dictionary. These function steps are as follows.

For vacuum operation, all existing schemas in the dataset and fields in each schemas should be studied.

We choose the first field in the first schema as subject-a.

Then we compare a with all existing fields in the schemas that are called $S_{ij}$ as well as the fields in which a is located in and are called $S_{ij}$. And using the cosine similarity measure, we calculate the similarity between each of $S_{ij}$ with a. And an array of -P- values returns cosine similarity of $S_{ij}$ and a (these values are between 0-1).

Then the index with the highest value in p is highlighted – index value is shown by *b*.

If there was no similarity between a and $S_{ij}$, namely $b$=0, it returns unIndex value. Otherwise, if the value of b was greater than the threshold value $\tau$, we put the index b in L and remove the index from $S_i$.

Finally the dictionary will be printed in the output.

Because after the operation Vacuum may find some $S_{ij}$ which are not belong to any array and they don't enter into the dictionary, we will repeat Vacuum operation. This operation is as follows: this time we calculate cosine similarity between $S_{ij}$ and $R_{ij}$. This work will continue until the remaining $S_{ij}$ in datasets enter into the house which they belong to.

## 3.2 Completion of the initial dictionary by User

Built-in dictionary in some dataset is complete, but some of them need to be modified. To edit a built-in dictionary we need human resources.

```
Input: (D)
  D: the input dataset  /* dataset after preprocessing */
Output: automatic dictionary
Steps:
For each $S_i \in$ D Do:
  For each $S_{ij} \in S_i$ Do: /* choosing a as */
    Choose a as subjects
    For each $S_i \in$ D Do:
      For each $S_{ij} \in S_i$
        Start to get similarity between a and $S_{ij}$ by using Cos
        similarity function
        /* return an array that filled by Cos similarity values
        know as P- array filled by {0-1} */
      Highlighting the index that has MAX value in P /* max
      index value know as b */
      If no similarity between a and $S_{ij}$
        Return unIndex  /* value =0& b =unIndex */
      else if b >$\tau$
        L = index
        Delete index from $S_i$
      R = L /* R is an 2D array */
Return R
```

Fig. 3 Vacuum function

## 3.3 Upload Secondary datasets and the dictionary

Our main goal at this step cluster schema is available in the dataset. The collection of secondary data-collection after the preprocessing, and secondary dictionary after dictionary edited by the user as input and clustering operations apply on it.

In this step, first we ready datasets to perform clustering. To prepare the dataset 1) we remove distances on both sides of the fields 2) All fields are converted to lowercase 3) we choose interface schema which is shown byM. This schema has the greatest number of fields in all schemas and requires that all schemas to be matched. 4) We synchronize dictionary index and the interface.

Clustering requires all schemas to carry out operations in datasets and survey each available fields in each schemas. As shown in Figure 4, the clustering is done as follows:

1) Using the cosine similarity measure, we calculate the similarity between each $S_{ij}$ and $W_i$ and an array of -*P*- values returns $W_i$ and $S_{ij}$ cosine similarity (these values are between 0-1).
2) Then index with the highest value in *P* is highlighted and the value index is shown by *b*.
   1. If there is no similarity between $W_i$ and $S_{ij}$, namely *b* smaller than the threshold value $\tau$, the new cluster is created and assign $S_{ij}$ field as the centroid of the cluster.
   2. Otherwise, if the value of *b* greater than the threshold value $\tau$:
      i. If another field from the $S_i$ schema was dedicated $C_i$, we will allocate the field $S_{ij}$ into $C_i$ and with a comma (,) we connect $C_i$ and $S_{ij}$.
      ii. Otherwise we assign $S_{ij}$ field to $C_i$.
3) Finally *N* cluster is produced and printed output.

```
Input: (W, D)
  D: the input dataset
  W: the input dictionary
  C: cluster
Output: N clusters
Steps:
  Remove spaces from two side of fields
  Change capital dataset to small
  Select mediated schema /*known as M*/
  Select the attributes of chosen mediated schema as first centroids.
  Integrate index of W & C
  For each $S_i \in D$ Do:
    For each $S_{ij} \in S_i$ Do:
      Start to get similarity between $W_i$ and $S_{ij}$ by using Cos
      similarity function
      /* return an array that filled by Cos similarity values know
      as P- array filled by {0-1} */
      Highlighting the index that has MAX value in P /*max
      index value know as b*/
      If $b < \tau$ Then
        Create a new cluster and assign attribute $S_{ij}$ as the
        centroid of this cluster
      else /* 1:m */
        If an attribute was assigned to $C_i$ from current $S_i$ then
          Assign attribute $S_{ij}$ to $C_i$ and merge $S_{ij}$ and $C_{ij}$ via ,
        Else
          Assign attribute $S_{ij}$ to $C_i$
  Return N clusters
```

Fig. 4 Pseudo-code for the third step

## 4. Experiment

In this part we describe done experiments to evaluate the holistic schema matching based on clustering method and we show as well as the impact of user interaction. Finally we review the results before and after user's interaction.

### 4.1 Dataset

The dataset used in this paper, are the dataset used in [7], [9], [19], [20] and [22] that were named the dataset number 1, 2 and 3.

Dataset interfaces query ICQ[1] used in [7], [19], [20], that from now on is called dataset number 1, includes four different domain: Airfare, Automobile, Books and Jobs. Each dataset contains 20 schemas from the query interface. Each schema includes fields of that schema.

BAMM query dataset interfaces[2] used in [2] and [9], that from now on is called dataset number 2, contains four different domains: Automobiles, Books, Movies and Music. Roughly every domain includes 50 query interface schema.

TEL-8 query interfaces dataset[3] used in [22], which from now on is called the dataset number 3. This dataset has examined eight different domains: Airfares, Automobiles, Books, CarRentals, Hotels, Jobs, Movies and MusicRecords. Roughly per domain includes 80 interface schema.

### 4.2 Performance measure

To review the effectiveness of proposed clustering method, popular performance measures in schema matching and information retrieval such as precision, recall and F-measure [4],[5],[8] were used.

**Precision:** is the fraction of retrieved documents that are relevant to the query.

$$Precision = \frac{|\{releventDocuments\} \cap \{retrievedDocuments\}|}{\{retrievedDocuments\}} \quad (1)$$

**Recall:** is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$Recall = \frac{|\{releventDocuments\} \cap \{retrievedDocuments\}|}{\{releventDocuments\}} \quad (2)$$

**F-measure:** combines precision and recall is the harmonic mean of precision and recall.

$$F-measure = 2.\frac{precision.recall}{precision + recall} \quad (3)$$

### 4.3 Results

In this section we describe the experimental results of mentioned three datasets. In each dataset we study two contributions of this paper. First we evaluate the results obtained before user interactions and using the dictionary generated by the system. Secondly, the results obtained after user interactivity and completion of dictionary by the user. Table 3, 4 and 5 show the results of our approach before and after user interaction.

[1] http://metaquerier.cs.uiuc.edu/repository/datasets/icq/index.html
[2] http://metaquerier.cs.uiuc.edu/repository/datasets/bamm/
[3] http://metaquerier.cs.uiuc.edu/repository/datasets/tel-8/

TABLE 3: RESULT DATASET NUMBER ONE

| Dataset | Before user interaction | | | After user interaction | | |
|---|---|---|---|---|---|---|
| | ***P*** | ***R*** | ***F*** | ***P*** | ***R*** | ***F*** |
| Airfare | 88% | 96% | 92% | 100% | 100% | 100% |
| Auto | 97% | 93% | 95% | 100% | 100% | 100% |
| Books | 88% | 78% | 83% | 100% | 100% | 100% |
| Job | 96% | 100% | 98% | 100% | 100% | 100% |

TABLE 4: RESULT DATASET NUMBER TWO

| Dataset | Before user interaction | | | After user interaction | | |
|---|---|---|---|---|---|---|
| | ***P*** | ***R*** | ***F*** | ***P*** | ***R*** | ***F*** |
| Airfares | 86% | 85% | 86% | 100% | 100% | 100% |
| Auto | 85% | 83% | 84% | 100% | 100% | 100% |
| Books | 83% | 82% | 82% | 100% | 100% | 100% |
| Car | 76% | 86% | 81% | 100% | 100% | 100% |
| Hotel | 80% | 92% | 85% | 100% | 100% | 100% |
| Jobs | 83% | 89% | 86% | 100% | 100% | 100% |
| Movie | 84% | 84% | 84% | 100% | 100% | 100% |
| Music | 84% | 89% | 86% | 100% | 100% | 100% |

TABLE 5: RESULT DATASET NUMBER TWO

| Dataset | Before user interaction | | | After user interaction | | |
|---|---|---|---|---|---|---|
| | ***P*** | ***R*** | ***F*** | ***P*** | ***R*** | ***F*** |
| Auto | 100% | 100% | 100% | 100% | 100% | 100% |
| Books | 100% | 100% | 100% | 100% | 100% | 100% |
| movies | 100% | 100% | 100% | 100% | 100% | 100% |
| Music | 100% | 100% | 100% | 100% | 100% | 100% |

## 4.4 Reference System

Finally, we studied and compared the proposed available method of clustering [7], [19], [20], which use the dataset number one and the same evaluation measures (precision, recall and F-measure) in this study.

The method proposed by Pei and his colleagues uses only the k-mean clustering [20]. This method adjust characteristics all at once depending on the similarity measures that are used by the k-mean algorithm. This method consists of three clustering steps: clustering schemas, the clustering fields in clusters of similar schemas, and clustering fields of all the different clustering schemas. Unfortunately, the third step has not been clearly demonstrated, and only use the name, type of data, and label to measure the similarity between characteristics in various schemes.

Wu and colleagues [7] offered an interactive cluster-based approach for the matching of query interface, which records hierarchical nature of the interfaces. In this way agglomerative hierarchical clustering algorithm is used. Similarly of pair-wise matching which re-use of existing mapping, this method also uses bridging effect.

Alofairiand Ahmad in [19] provided an effective way to holistic schema matching to reduce the search space. They used the method of k-mean clustering techniques and agglomerative clustering algorithm. Combination of the two sequences, produce a state of strong clusters and improve the clustering process efficiency. To achieve the main goal, this method consists of two preprocessing and clustering steps. The results of this study show that their proposed matching method reduces the search space using integrating techniques to clusters, which immediately classifies relevant fields in the Similar Clusters. The results prove that this method is effective and promising in holistic schema matching and leads to better results than proposed method by Pei and colleagues [3] and Wu and colleagues [7].

## 4.5 Compare results

In our experiments, we assume that the domain values are not available. Figure 5-8 shows the comparison of three measure in four domains. Three measure of our approach is based on the number of estimated clusters that obtained after user interaction. The results in Figure 5 show that the proposed method has the greatest amount of precision, recall and F-measure.

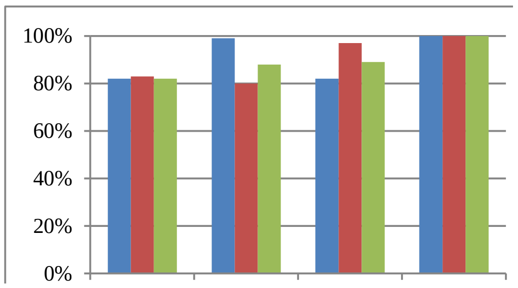

Fig. 5 compare results with Airfair dataset

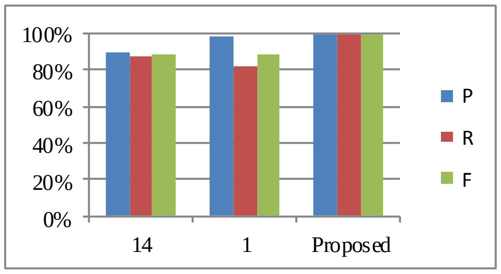

Fig. 6 compare results with Auto dataset

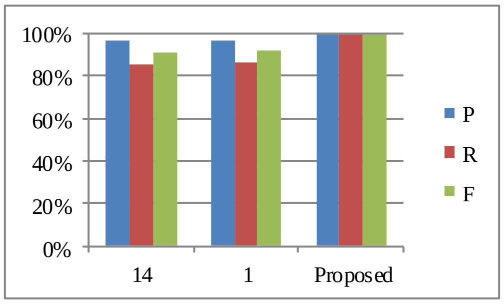

Fig. 7 compare results with Books dataset

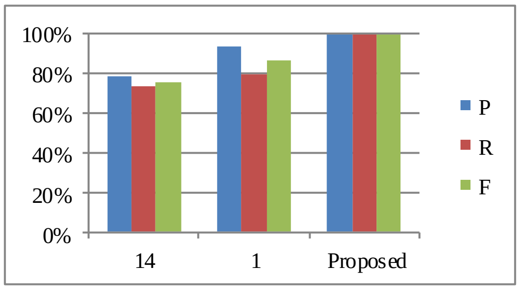

Fig. 8 compare results with Books dataset

## 5. Summary and Conclusion

To holistic schema matching, the majority of matching systems, use of available techniques in clustering. The main objective of using clustering techniques is to provide method that improve results and performance of matching process.

As we had explained we used dictionary in our method to build clusters. According to the techniques used in the construction of the dictionary, and according to the results shown in section 3.1, this method had a good performance. To be more carefully clusters we used user interactions to edit dictionary and apply the necessary reforms. Our proposed approach handles both simple and complex matching.

Unlike previous methods that they had regarded only simulative fields, our proposed method did not rely only on physical similarity and visual and semantic similarities were chosen together and this led to better output.

We offered a method with high degree of accuracy in different domains and returns the best results. Experiment results in Section 3 also shows the accuracy and performance of our method. In addition, the famous K-Means algorithm was used. Algorithm K-Means, is a clustering algorithm with a linear time complexity and is very useful for the matching in large-scale.

## References

[1] M. Bergman, The Deep Web: Surfacing the hidden value, BrightPlanet.com, 2001.
[2] S. Lawrence, C. L. Giles, “Accessibility of information on the Web,” in Nature, vol. 11, 1999, pp. 32-39.
[3] J. Pei, D. Hong, D. Bell, “A Novel Clustering-Based Approach to Schema Matching. In Proceedings of the 4th International Conference on Advances in Information Systems (ADVIS),” 2006, pp. 60-69.
[4] H. He, W. Meng, C. YU, Z. Wu, “WISE-Integrator: An Automatic Integrator of Web Search Interfaces for E-Commerce,” in Proc. 29th VLDB Conf. Berlin, Germany, 2004, pp. 256-273.
[5] S. Melnik, H. Garcia-Molina, E. Rahm, “Similarity flooding: A versatile graph matching algorithm and its application to schema matching,” in Proc. 18th International Conference Data (ICDE '02), 2002, pp. 117-128.
[6] B. He, K. C. Chang, J. Han, “Discovering Complex Matchings across Web Query Interfaces: A Correlation Mining Approach,” In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD’04). 2004, pp. 148-157.
[7] W. Wu, C. Yu, A. Doan, W. Meng, “An Interactive Clustering based Approach to Integrating Source Query Interfaces on the Deep Web,” In Proceedings of the 2004 ACM SIGMOD International conference on Management of Data (SIGMOD’04). 2004, pp. 95–106.
[8] H. Do, E. Rahm, “COMA - A system for flexible combination of schema matching approaches,” In Proceedings of the Very Large Data Bases Conference (VLDB). 2002, pp. 610-621.
[9] B. He, K. C. Chang, “Statistical schema matching across web query interfaces,” In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD’03). 2003, pp. 217–228.
[10] E. Rahm , Ph. A. Bernstein, “A survey of approaches to automatic schema matching,” VLDB Journal: Very Large Data Bases,. 2001. vol. 10, pp. 334–350.
[11] J. Madhavan, Ph. A. Bernstein, A. Doan, A. Halevy, “Corpus-based Schema Matching,” In Proceedings of International Conference on Data Engineering (ICDE’05). 2005, pp. 57-68.
[12] A. Doan, P. Domingos , A. Halevy, “Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach,” In Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD’01). vol. 30, 2001, pp. 509-520.
[13] L. Haas, “Beauty and the Beast: The Theory and Practice of Information Integration,” In Proceedings of the 11th

international conference on Database Theory (ICDT'07). 2007, pp. 28-43.
[14] W. Li, C. Clifton, “Semint: A tool for identifying attribute correspondence in heterogeneous databases using neural networks,” Journal of Data & Knowledge Engineering. vol. 33, 2000, pp. 49-84.
[15] J. Madhavan , Ph. A. Bernstein, E. Rahm, “Generic schema matching with Cupid,” In Proceedings of the 27th International Conference on Very Large Data (VLDB '01). 2001, pp. 49-58.
[16] O. Unal, H. Afsarmanesh, “Semiautomatedschema integration withSASMINT,” Journal of Knowledge and Information Systems. vol. 23, 2010, pp. 99-128.
[17] H. Do, E. Rahm, “Matching large schemas: Approaches and evaluation,” Journal of Information Systems. vol. 32, 2007, pp. 857–885.
[18] Y. Qian, H. Zhang, J. Song, Z. Liu, “A New Complex Schema Matching System,” In Proceedings of International Conference on Innovative Computing and Communication and Asia-Pacific Conference on Information Technology and Ocean Engineering CICC-ITOE. 2010, pp. 195-198.
[19] A. A. Alofairi, K. Ahmad, “an integrated clustering method for holistic schema matching,” Journal of Theoretical and Applied Information Technology. vol. 68, 2014. pp. 294-301.
[20] S. S. Aïcha, N. Benharkat, Y. Amghar, “Towards a more scalable schema matching: a novel approach,” International Journal of Distributed Systems and Technologies (IJDST). vol. 1, 2010, pp. 17-39.
[21] L. Kaufman, P. J. Rousseeuw, “Finding Groups in Data: An Introduction to Cluster Analysis,” John Wiley & Sons. New York, 1990.
[22] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, P. Domingos, “iMAP: Discovering Complex Semantic Matches between Database Schemas,” In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD’04). 2004, pp. 383-394.

**Fatemeh Salahshour** received her B.s in information technology engineering from Iran University of Tabari, Babol, Iran. She received her M.s information technology engineering from Pooyandegan danesh, Chalus, Iran. Her research interests Include web minig and data mining.

**Reza Tavoli** received his B.s in software engineering from Iran University of Science & Technology, Behshahr, Iran. He received his M.s software engineering from Islamic Azad University, science & Research Branch, Tehran, Iran. Currently, He is pursuing PhD in Software engineering at Islamic Azad University, Qazvin Branch, and Qazvin, Iran. His research interests Include document image retrieval and data mining.