CASM: Coherent Automated Schema Matcher

Rudraneel Chakraborty, Faiyaz Ahmed, and Shazzad Hosain

Abstract. Schema matching has been one of the basic tasks in almost every data intensive distributed applications such as enterprize information integration, collaborating web services, web catalogue integration, and schema based point to point database systems and so on. Typical schema matchers perform manually and use a set of matching algorithms with a composition function by using them in an arbitrary manner which results in wasteful computations and needs manual specification for different domains. Recently, there has been some schema matching strategy proposed with partial or full automation. Such a schema matching strategy is OntoMatch. In this paper, we propose an element level automated linguistic based schema matching strategy motivated by the concept of OntoMatch, with more powerful matching algorithms and definite property construction for matcher selection that produces better output. Experimental result is also provided to support the claim of the improvement.

1 Introduction

As the Internet becomes a vast repository of information, often it requires integrating many different sources to answer a single query. For example, let one likes to attend the DEIT 2011 conference at Bali, Indonesia, he likes to stay in a hotel nearby the conference venue and the hotel should be nearby of a public transport facility so that he could visit tourist places easily. To get all the necessary information often he requires visiting several Web sites such as airline ticket reservation sites, hotel reservation sites, tourism sites etc. and gleaning the information to get the correct picture. To address the problem it requires autonomous integration of different sites and the research community has investigated several approaches in this direction.

Rudraneel Chakraborty · Faiyaz Ahmed · Shazzad Hosain
Department of EECS, North South University, Dhaka, Bangladesh
e-mail: {rudraneel,faiyaz}@eecs.northsouth.edu,
shazzad@northsouth.edu

F.L. Gaol (Ed.): Recent Progress in DEIT, Vol. 1, LNEE 156, pp. 219–224.
springerlink.com

One of the approaches is to model the Web as relations [7] so that semantic heterogeneity can be resolved through automatic object identification and consolidation process. However, the key problem of this approach is to find a good schema matching technique, often known as *schema matcher*, so that it can match two different terms from two different sources automatically.

Schema matching is the solution of finding semantic relationship between elements of two schemas. Manually performing schema matchers usually use a set of simple matching algorithms such as, edit distance matcher [10], synonym matcher [3] etc, then combine the individual matching scores using some function such as average, weighted average and so on. This approach has its inherent limitations. For example, let us take two terms *movie* and *cinema*. While the synonym matcher would give a score of 1 in the scale of 0 to 1, which reflects a perfect match; using another matcher, let's say edit-distance matcher would give a score of 0.3. If we take the average of these two scores, it will be 0.65, and if we take 0.8 as a threshold, then the matcher would output a mismatch. So the problem is to decide which set of matchers in which order should we consider to determine the final score of the match?

A number of approaches have been proposed that use machine learning techniques, data mining techniques and so on. However, very recently Bhattacharjee et. al. [5] proposed a new way to decide different matching algorithms from a set of matching algorithms. He proposed OntoMatch [5], a property based matcher, that cleverly decides a subset of matchers from a set of known matchers to increase the quality of match. Though OntoMatch performs well to Cupid [8], COMA [6] and so on, it has several limitations. It uses edit-distance matcher for string similarity, while Jaro-Winkler [11] and Monge-Elkan [9] perform better then Edit-Distance-Matcher. Also there are few limitations in matcher selection approach, which we will describe in next section. In our research we addressed these limitations of OntoMatch and developed a new property based schema matcher called CASM that outperforms OntoMatch [5] and other matchers such as COMA [6].

The paper is organized as the following. Section 2 provides the related works, section 3 gives problem formulation, section 4 describes matcher characterization, section 5 gives CASM implementation, section 6 gives comparative analysis and finally section 7 gives conclusion and future works.

2 Related Works

In the literature, we found three types of matching algorithms: instance based matching, representation based or schema matching and usage based or ontology matching. We are only interested in schema matching approaches that only consider schema information, not the instance data. While most of matchers have emerged from the context of a specific application, a few approaches such as COMA [6], Cupid [8], and OntoMatch [5], try to address the schema matching problem in a generic way that is suitable for different applications and schema languages. Cupid [8] is a linguistic based matcher. Systems like COMA [6] uses a set of matchers and

combines the match result with similarity combination function. OntoMatch [5] also uses a set of matchers but selects those based on some properties of the matchers. In our research we also select matchers from a set of matchers much like OntoMatch do.

3 Problem Formulation

Before defining the problem, let us formally define the following few concepts.

Definition 3.1 (Schema and Term). A schema S of a database is a list of attributes $\mathscr{A}$, where each $a \in \mathscr{A}$ has its associated domain $d \in \mathscr{D}$. A term t is either a or d.

Definition 3.2 (Term Similarity). Let ψ' be a similarity function[1] such that for any two $t_1, t_2 \in \mathscr{A} \vee \mathscr{D}$, $\psi'(t_1,t_2) \in [0,1]$, where 0 indicates complete dissimilarity and 1 means two terms are identical. Let ψ be a function and ε be a threshold such that $\psi(\varepsilon, \psi'(t_1,t_2)) \in \{0,1\}$, i.e., $\psi(\varepsilon, \psi'(t_1,t_2)) = 0$ if $\psi'(t_1,t_2) < \varepsilon$, $\psi(\varepsilon, \psi'(t_1,t_2)) = 1$, otherwise, and we say t_1 and t_2 are similar, denoted $t_1 \sim t_2$. From here on, we write $\psi(t_1,t_2)$ as a short hand for $\psi(\varepsilon, \psi'(t_1,t_2))$ unless specified otherwise.

Definition 3.3 (Mapping). Given two schemas S_1 and S_2 the mapping is a list of attribute pairs i.e. $\{< a_1,b_1 >, < a_2,b_2 >, \ldots\}$, where $a_i \in S_1$, $b_i \in S_2$ and $a_i \sim b_i$

Definition 3.4 (Term Matcher). Given two terms t_1, t_2 a term matcher μ returns whether the two terms match or not, based on the values of similarity functions.

Definition 3.5 (Distance Matrix). Let S_1, S_2 be two schemas, m is the number of attributes in S_1, n is the number of attributes in S_2, then a distance matrix is a two dimensional array $dm[m][n]$ that stores the value of $\psi'(t_i,t_j)$, where $1 \leq i \leq m$ and $1 \leq j \leq n$.

3.1 Schema Matching Problem

Given two schemas S_1 and S_2 the schema matching problem is to find all the correct mappings among the terms of the two schemas. Usually the linguistic matchers use more than one term matcher from a set of term matchers $\{\mu_1, \mu_2, \ldots, \mu_n\}$ to determine the ultimate mapping. In our research effort we focus on selecting the best term matcher(s) based on some properties of the matchers for the schema matching problem. To limit our problem, we choose the following five term matchers, of which we apply one or more term matchers to find the final matching value.

- **Jaro-Winkler:** The Jaro-Winkler [4] distance metric is a measure of similarity which is basically a variant of the Jaro distance metric algorithm. For two terms t_1 and t_2, the Jaro metric calculates the similarity by identifying the number and the placement of the common characters between the two terms.

[1] Such as string edit distance [10], thesaurus at WordNet [3], etc. or a combination of such functions.

- **Abbreviation Matcher (AB):** For two terms t_1 and t_2, the abbreviation matcher returns value 1 if abbreviation of t_1 is t_2 or abbreviation of t_1 is abbreviation of t_2 or vice versa. Otherwise it returns 0.
- **Monger-Elkan:** Monger-Elkan [9] matcher is an affine (assigns unique cost to each edit operation) variant of Smith-Waterman distance metric. It has unified cost parameters that produces similarity values between 0 and 1.
- **Sound-Matcher:** For two terms t_1 and t_2, the sound-matcher returns 1 if they sounds similar otherwise returns 0. The matcher is based on the Soundex [12] algorithm, which is a phonetic algorithm that indexes names by their sounds when pronounced in English.
- **Synonym-Matcher:** Given two terms t_1 and t_2, the synonym matcher returns 1 if the terms are synonymous to each other, else it returns 0.

4 Matcher Characterization

The basic goal here is to identify the best matcher μ for the term t_1,t_2 from a set of available distinct matchers $\mu_1, \mu_2, \ldots, \mu_n$. To get the best applicable matcher, we set some properties to matchers. First, the abbreviation matcher, AB matcher (μ_1), is used based on the property that one term is much shorter than the second term. If the AB matcher provides a score less than the threshold or AB matcher is not applicable, i.e. the terms are of similar length, then it applies Jaro-Winkler-Monge-Elkan matcher, JWME matcher (μ_2). The JWME matcher applies two matchers Jaro-Winkler [4] and Monge-Elkan matcher [9] and chooses one that gives the best score. If JWME matcher does not provide score greater than the threshold, it applies synonym matcher, SM matcher (μ_3) and sound matcher, SD matcher (μ_4), if required. Finally, if no satisfactory match result is found, it returns the best match value produced so far.

5 CASM Implementation

To verify the concept, we have implemented CASM and used the data available in the UIUC web integration repository [2]. The UIUC repository contains XML schemas, while the CASM works on relational schemas. So we first parse the XML schema to convert it into relational schema, then apply the CASM algorithm. The overall mapping process is divided into three steps:

1. Parse the XML schema using external XML–parser such as JDOM [1].
2. Compute the distance matrix *dm*, definition 3.5, using the CASM algorithm given below.
3. Obtain a final mapping based on threshold, matching score and distance matrix by applying the *stable marriage* algorithm.

Based on the properties the algorithm first sorts the term matchers. The order is $\mu_1, \ldots, \mu_4$ in this case as stated in section 4. The algorithm then applies the matchers

Algorithm 1. Algorithm to select the best applicable matcher

Require: $S_1, S_2, bestScore = 0, dm[m][n] = 0$, where $m = |S_1|, n = |S_2|$
  sort the matchers $\mu_i \in \mathscr{M}$
  for each term $t_i \in S_1$ **do**
    for each term $t_j \in S_2$ **do**
      $bestScore = 0$
      for each matcher $\mu_i \in \mathscr{M}$ **do**
        $score = \mu_i(t_i, t_j)$
        if $score >= \tau$ **then**
          bestScore = score
          break
        end if
        if $score >= bestScore$ **then**
          $bestScore = score$
        end if
      end for
      $dm[i][j] = bestScore$
    end for
  end for

successively, stops when it gets a match or assigns the best score among all the scores that are less than the threshold.

6 Comparative Analysis

The performance of CASM was compared using BAMM extracted query schemas available in the UIUC web integration repository [2]. We took three types of schemas: Automobiles, MusicRecords and Jobs. Each category contains 40 schemas on an average. We chose one schema and compared it with all others schemas of the same category. The schemas have 3 to 20 attributes and we chose the average one with 12 attributes to compare it with other schemas of the same category. Figure 1 shows the comparison of CASM with two other matchers.

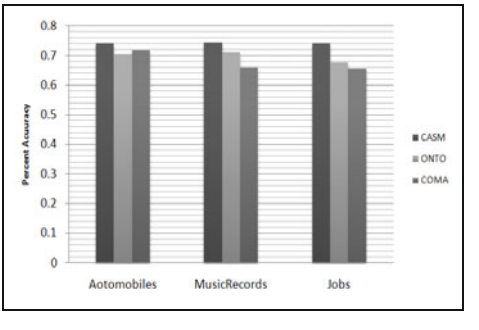

Fig. 1 CASM is compared with two other matchers OntoMatch and COMA. It uses the area under the precision recall curve (AUPRC), the same technique used by OntoMatch, to measure performance. CASM performs better than both of the matchers in all three data sets.

7 Conclusion and Future Works

In this paper we presented a new approach towards ordering term matchers in an efficient way. This new approach selects one/more term matchers from a pool of matchers for automatic schema matching. The matcher CASM performed better than OntoMatch and COMA. We provided the experimental result to support the claim. Also the CASM's selection algorithm showed that only the appropriate matcher(s) was(were) selected thus avoiding wasteful computation.

In future, we like to modify the algorithm to include more term matchers and to use more data sets to increase the confidence of the result. However, the more important future direction would be improve CASM to handle XML schemas.

References

1. JDOM, `http://www.jdom.org`
2. UIUC Web Repository, `http://metaquerier.cs.uiuc.edu/repository/datasets/bamm/index.html`
3. WordNet, `http://wordnet.princeton.edu/`
4. Jaro, M.A.: Probabilistic linkage of large public health data files. Stat Med. 14(5-7), 491–498 (1999)
5. Bhattacharjee, A., Jamil, H.M.: OntoMatch: A monotonically improving schema matching system for autonomous data integration. In: IEEE IRI, Las Vegas, NV (2009)
6. Do, H.H., Rahm, E.: COMA - A system for flexible combination of schema matching approaches. In: VLDB, Hong Kong, China, pp. 610–621 (2002)
7. Hosain, S., Jmail, H.: An algebraic language for semantic data integration on the hidden web. In: Proceedings of the 3rd IEEE ICSC 2009, Berkeley, California, pp. 237–244 (2009)
8. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with Cupid. In: VLDB, Italy, pp. 49–58 (2001)
9. Monge, A., Elkan, C.: The field matching problem: Algorithms and applications. In: 2nd International Conference on KDDM, pp. 267–270 (1996)
10. Ristad, E.S., Yianilos, P.N.: Learning string edit distance. IEEE Transactions on Pattern Recognition and Machine Intelligence 20(5), 522–532 (1998)
11. Winkler, W.E.: The state of record linkage and current research problems. Tech. rep., Statistical Research Division, U.S. Census Bureau (1999)
12. Zobel, J.: Phonetic string matching: Lessons from information retrieval. In: SIGIR 1996, pp. 166–172 (1996)