Information Systems 35 (2010) 140–148

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

# An evidential approach to query interface matching on the deep Web

Jun Hong *, Zhongtian He, David A. Bell

*School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT7 1NN, United Kingdom*

ARTICLE INFO

*Keywords:*
Query interface matching
Schema matching
Deep Web data integration
Evidential reasoning
Dempster–Shafer theory of evidence

ABSTRACT

Matching query interfaces is a crucial step in data integration across multiple Web databases. The problem is closely related to schema matching that typically exploits different features of schemas. Relying on a particular feature of schemas is not sufficient. We propose an evidential approach to combining multiple matchers using Dempster–Shafer theory of evidence. First, our approach views the match results of an individual matcher as a source of evidence that provides a level of confidence on the validity of each candidate attribute correspondence. Second, it combines multiple sources of evidence to get a combined mass function that represents the overall level of confidence, taking into account the match results of different matchers. Our combination mechanism does not require the use of weighing parameters, hence no setting and tuning of them is needed. Third, it selects the top $k$ attribute correspondences of each source attribute from the target schema based on the combined mass function. Finally it uses some heuristics to resolve any conflicts between the attribute correspondences of different source attributes. Our experimental results show that our approach is highly accurate and effective.



## 1. Introduction

Web databases are now pervasive, which can be accessed via their query interfaces (usually HTML query forms) only. Query forms provide a natural way for the user to make queries to the underlying databases without using a particular query language. On receiving form-based queries, these databases return query results encoded in HTML, which are then displayed to the user.

Many E-commerce sites are supported by Web databases. In a specific domain (e.g. flight booking, book sales), there are many database-driven Web sites that sell similar products or services. It is a daunting task for the user to visit numerous Web sites individually to search for and compare services or products. Web data integration aims to provide integrated access to a multitude of Web databases, where users need to fill in only a uniform query form, and on receiving a user query the system will automatically make connections to different sites, fill in the local query forms on these sites, submit these forms, combine the query results, and return the combined results to the user.

Matching query interfaces is a crucial step in Web data integration, which finds attribute correspondences between two query interfaces. For example, Fig. 1 shows two query interfaces on the Web, where we can find attribute correspondences such as those indicated by different shapes or underlined.

The problem is closely related to schema matching that takes two schemas as input and produces a set of attribute correspondences between them [1]. Schema matching has been extensively studied (e.g. [1–8]). These approaches exploit different features of schemas, including structural and linguistic features and data types, etc. to match attributes between schemas. Schema matching is inherently uncertain due to lack of complete knowledge about schemas. Relying on a single feature of schemas is not sufficient and the match results of individual matchers are often inaccurate and uncertain. Approaches have been

* Corresponding author.
*E-mail addresses:* j.hong@qub.ac.uk (J. Hong), zhe01@qub.ac.uk (Z. He), da.bell@qub.ac.uk (D.A. Bell).

0306-4379/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2009.08.004

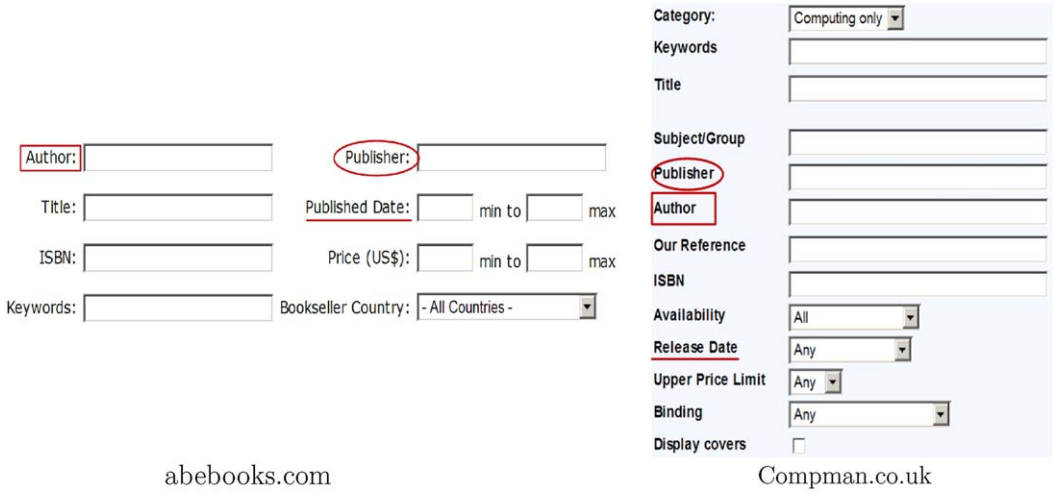


**Fig. 1.** Two Web query interfaces in the book domain.

proposed to combine multiple matchers taking into account different features of schemas. A common approach is to apply different weight coefficients to the match results of individual matchers reflecting their different levels of importance, and their weighted values are then added together as the combined match results. However, these weight coefficients are often manually set for a particular domain in a trial and error manner.

There are several differences between query interfaces and traditional database schemas. First, database schemas are designed internally for database developers. As a consequence, the attributes of the schemas may be named in a highly inconsistent manner, imposing many difficulties in schema matching. In contrast, query interfaces are designed for normal users and are likely more meaningful and consistent. For example, labels on query forms are usually words or phrases, whereas attribute names of database schemas are often abbreviations. Second, database schema matching has mainly focused on schema-level matching while instance-level matching has not been done extensively. This is often due to the unavailability of data instances and the assumption that the same domains of values have been used across different schemas. Whereas in query interfaces, the user is likely to be given ranges of values to choose from and as these values are designed for human use they are also likely to be more meaningful and consistent. As data instances are pervasive, semantic heterogeneity of data instances between query interfaces has to be addressed.

The availability of rich linguistic and semantic features in both attributes and data instances of query interfaces motivates us to explore multiple sources of evidence from these linguistic and semantic features. We propose an evidential approach to combining the match results of multiple matchers using Dempster–Shafer theory of evidence. First, this approach views the match results of an individual matcher as a source of evidence that provides some degree of belief on the validity of each candidate attribute correspondence. Second, it combines multiple sources of evidence to get a combined mass function that represents the overall level of confidence, taking into account the match results of different matchers. Our combination mechanism does not require the use of weighing parameters, hence no setting and tuning of them is needed. Third, it selects the top $k$ attribute correspondences of each source attribute from the target schema based on the combined mass function. Finally it uses some heuristics to resolve any conflicts between the attribute correspondences of different source attributes. Our experimental results show that our approach is highly accurate and effective.

## 2. Dempster–Shafer (DS) theory of evidence

Dempster–Shafer theory of evidence, sometimes called evidential reasoning [9] or belief function theory, is a mechanism formalized by Shafer [10] for representing and reasoning with uncertain, imprecise and incomplete information. The theory represents a set of propositional hypotheses by a frame of discernment.

**Definition 1** (*Frames of discernment*). A frame of discernment (or simply a frame), usually denoted as $\Theta$, contains mutually exclusive and exhaustive propositional hypotheses, one and only one of which is true.

**Example 1.** A patient has been observed having two symptoms: 'coughing' and 'sniveling' and only three types of illness could have caused these symptoms: 'flu' ($F$), 'cold' ($C$) and 'pneumonia' ($P$). We can use frame $\Theta = \{F, C, P\}$ to represent these types of illness.

There are three important functions in DS theory: the basic probability assignment (*bpa* or $m$), the belief function (*bel*), and the plausibility function (*pl*).

**Definition 2** (*Basic probability assignment (mass function)*). A function, $m : 2^{\Theta} \rightarrow [0, 1]$, is called a basic probability assignment on a frame $\Theta$ if it satisfies the following two

conditions:

$$m(\phi) = 0 \tag{1}$$

$$\sum_{A \subseteq \Theta} m(A) = 1 \tag{2}$$

where $\phi$ is an empty set and $A$ is any subset of $\Theta$.

The basic probability assignment (mass function thereafter) is a primitive function. Given a frame, $\Theta$, for each source of evidence, a mass function assigns a mass to every subset of $\Theta$, which represents the degree of belief that one of the hypotheses in the subset is true, given the source of evidence. In Example 1, when the patient has been observed having the symptom 'coughing', the degree of belief that the patient has 'flu' or 'cold' is 0.6 and the degree of belief that the patient has 'pneumonia' is 0.4. We then have a mass function: $m_1(\{C,F\}) = 0.6$ and $m_1(\{P\}) = 0.4$. Similarly, with the symptom of 'sniveling', we have another mass function: $m_2(\{F\}) = 0.7$, $m_2(\{C\}) = 0.2$ and $m_2(\{P\}) = 0.1$.

From the mass function for a source of evidence, the belief function (*bel*) and the plausibility function (*pl*) can be defined, which represent the upper and lower bounds of an interval for every subset $A$ of $\Theta$, which contains the precise probability $p(A)$ that one of the hypotheses in $A$ is true, given the source of evidence, i.e. $bel(A) \le p(A) \le pl(A)$.

**Definition 3** (*Belief function (bel)*). For every subset $A$ of $\Theta$, $bel(A)$ is defined as the sum of the masses assigned to every subset $B$ of $A$:

$$bel(A) = \sum_{B \subseteq A} m(B) \tag{3}$$

**Definition 4** (*Plausibility function (pl)*). For every subset $A$ of $\Theta$, $pl(A)$ is defined as the sum of the masses assigned to every subset $B$ of $\Theta$ that intersects $A$:

$$pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \tag{4}$$

The belief function and the plausibility function are related to each other as follows:

$$pl(A) = 1 - bel(\overline{A}) \tag{5}$$

It follows from Eqs. (3)–(5) that from one of the three functions $m$, *bel* and *pl* the other two functions can be derived.

Given two mass functions $m_1$ and $m_2$, DS theory also provides Dempster's combination rule for combining them, which is defined as follows:

$$m(C) = \frac{\sum_{A \cap B = C} m_1(A) m_2(B)}{1 - \sum_{A \cap B = \phi} m_1(A) m_2(B)} \tag{6}$$

In Example 1, we combine two mass functions, $m_1$ and $m_2$, to get a combined mass function: $m(C) = 0.207$, $m(F) = 0.724$ and $m(P) = 0.069$. From this mass function, the corresponding belief and plausibility functions can also be derived. When all masses have been assigned to subsets with a single element only, *bel*, *pl* and $m$ are all the same. Since $bel(A) \le p(A) \le pl(A)$ for every subset $A$ of $\Theta$, we have a probability distribution, $p(C) = 0.207$, $p(F) = 0.724$ and $p(P) = 0.069$. Therefore given the two symptoms the patient has, it is more likely that he is having 'flu'.

## 3. Combining multiple matchers using DS theory

Based on DS theory, we propose an evidential approach to combining multiple matchers that exploit different features of schemas. Given a source schema and a target schema, for every source attribute, each target attribute is one of its candidate correspondences. An individual matcher provides a different measure on the validity of each candidate correspondence of the source attribute. Applying this measure to all the candidate correspondences provides a source of evidence on the validity of each candidate correspondence. Based on this source of evidence, we can generate a mass function that assigns a mass to every subset of the given frame, reflecting the degree of belief that the valid correspondence of the source attribute belongs to the subset.

A set of different matchers provide multiple measures and applying these measures to all the candidate correspondences provides multiple sources of evidence on the validity of each candidate correspondence, based on which we can generate multiple mass functions. These mass functions can then be combined using Dempster's combination rule to calculate the overall degree of belief on the validity of each candidate attribute correspondence, reflecting the match results of different matchers. The combined mass function can then be further used for deciding on the top $k$ attribute correspondences of each source attribute.

### 3.1. Individual matchers

We use four individual matchers. The first three matchers are based on different linguistic features of attribute names and the last matcher uses the data types of attributes.

#### 3.1.1. Semantic based similarity

We measure semantic similarity between two words based on their relationships on a general linguistic ontology, WordNet.[1] WordNet [11] is a lexical database developed by Princeton University, which provides taxonomic hierarchies of natural language concepts. WordNet contains nouns, verbs, adjectives and adverbs only, which are grouped into synonym sets (synsets) representing concepts. If a word is polysemous (i.e. having multiple senses), the word may be contained in multiple synsets each of which represents one sense of the word. Synsets (concepts) are related to each other by different types of semantic relationship. The two types of semantic relationship used in WordNet are Hyponym/Hypernym (i.e. Is-A) and Meronym/Holonym (i.e. Part-Of). We use only the Hyponym/Hypernym relationship in WordNet, which together with synsets form the Is-A hierarchy of WordNet. A fragment of the Is-A hierarchy in WordNet is shown in Fig. 2.

[1] http://wordnet.princeton.edu/

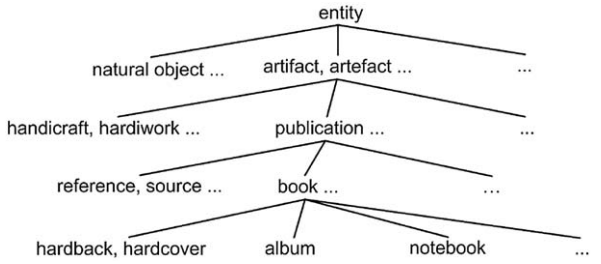


**Fig. 2.** A fragment of the Is-a hierarchy in WordNet.

We use the traditional edge counting approach [12] to measure word similarity. Given two words $w_1$ and $w_2$, the edge counting method finds a path between two synsets that contain the two words, respectively, in WordNet. A path between two synsets is one that consists of a series of synsets and Hyponym/Hypernym links, in which there is only one synset that subsumes the two given synsets. If a word is polysemous (i.e. having multiple senses) and appears in multiple synsets, only the shortest path, among all the possible paths between every pair of synsets (i.e. senses) containing $w_1$ and $w_2$, respectively, is used in calculating semantic similarity between $w_1$ and $w_2$.

We define semantic similarity between two different words, $w_1$ and $w_2$, as $Sim_{se}(w_1, w_2) = 1/(1 + L)$, where $L$ is the length of the shortest path in WordNet between $w_1$ and $w_2$, if both $w_1$ and $w_2$ can be found in WordNet. Otherwise, $Sim_{se}(w_1, w_2) = 0$. For two same words, $w_1$ and $w_2$, we define semantic similarity between them as $Sim_{se}(w_1, w_2) = 1$. For example, as shown in Fig. 2, the length of the shortest path in WordNet between words 'reference' and 'book' is 2 and the semantic similarity between them is 0.33.

#### 3.1.2. Edit distance based similarity

Edit distance (Levenshtein distance) between two strings is measured by the number of edit operations, including insertions, deletions and substitutions, necessary to transform one string into another [13]. We define the edit distance based similarity between two words, $w_1$ and $w_2$, as follows:

$$Sim_{ed}(w_1, w_2) = \frac{1}{1 + ed(s_1, s_2)} \tag{7}$$

where $s_1$ and $s_2$ are two strings in $w_1$ and $w_2$, respectively, and $ed(s_1, s_2)$ is the edit distance between $s_1$ and $s_2$. For example, as shown in Fig. 1, the edit distance between 'Author' on abebooks.com and 'Category' on Compman.co.uk is 5 and the edit distance based similarity between them is 0.167.

#### 3.1.3. Jaro distance based similarity

The Jaro distance based string similarity between two strings is defined based on the number and order of the common characters between them. Given two strings $s = a_1, \ldots, a_k$ and $t = b_1, \ldots, b_l$, a character $a_i$ in $s$ is common with $t$, if there is a $b_j = a_i$ in $t$ such that $i - H \le j \le i + H$, where $H = \min(|s|, |t|)/2$. Let $s' = a'_1, \ldots, a'_k$, be the characters in $s$ which are common with $t$ (in the same order they appear in $s$) and $t' = b'_1, \ldots, b'_l$ similarly. We define a *transposition* for $s'$ and $t'$ to be a position $i$ such that $a'_i \neq b'_i$. Let $T_{s',t'}$ be half the number of transpositions for $s'$ and $t'$. The Jaro distance based string similarity between $s$ and $t$ is defined as follows [14]:

$$Jaro(s, t) = \frac{1}{3} \cdot \left( \frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|} \right) \tag{8}$$

The Jaro distance based similarity $Sim_{ja}(w_1, w_2)$ between two words, $w_1$ and $w_2$, is defined as the Jaro distance based string similarity $Jaro(s_1, s_2)$ between two strings $s_1$ and $s_2$ in $w_1$ and $w_2$. For example, as shown in Fig. 1, the Jaro distance based similarity between 'Author' on abebooks.com and 'Category' on Compman.co.uk is 0.72.

#### 3.1.4. Similarity between attribute names

Assume that two attribute names, $A_1$ and $A_2$, contain two sets of words, $A_1 = \{w_1, w_2, \ldots, w_m\}$ and $A_2 = \{w'_1, w'_2, \ldots, w'_n\}$. For each word, $w_i$ for $i = 1, 2, \ldots m$, in $A_1$, we calculate its similarity with every word in $A_2$ and find the maximum similarity value $v_i$. We then get a similarity value set for $A_1$: $Sim_1 = \{v_1, v_2, \ldots, v_m\}$. Similarly, we get a similarity value set for $A_2$: $Sim_2 = \{v'_1, v'_2, \ldots, v'_n\}$. We calculate similarity between two attribute names $A_1$ and $A_2$ as follows:

$$Sim(A_1, A_2) = \frac{\sum_{i=1}^{m} v_i + \sum_{i=1}^{n} v'_i}{m + n} \tag{9}$$

where $m$ is the number of words in $A_1$, $n$ is the number of words in $A_2$.

#### 3.1.5. Data types of attribute names

We define that two data types are compatible if one subsumes another (i.e. is-a relationship). Fig. 3 shows all the data types we use and their is-a relationships. The data type of an attribute with an input box is 'any' since the input box can accept data of different types such as 'string' or 'integer'. The other data type of an attribute is recognized on the basis of the set of labels and form elements that form the attribute.

### 3.2. Representing possible matches of source attributes in DS

Assume that we have a source schema, $S = \{a_1, a_2, \ldots, a_m\}$, where $a_i$, for $i = 1, 2, \ldots, m$, is a source attribute, and a target schema, $T = \{b_1, b_2, \ldots, b_n\}$, where $b_j$, for $j = 1, 2, \ldots, n$, is a target attribute. For each source attribute, $a_i$, we have a set of candidate correspondences in the target schema $\{\langle a_i, b_1 \rangle, \langle a_i, b_2 \rangle, \ldots, \langle a_i, b_n \rangle\}$. It is also possible that $a_i$ may have no correspondence in the target schema

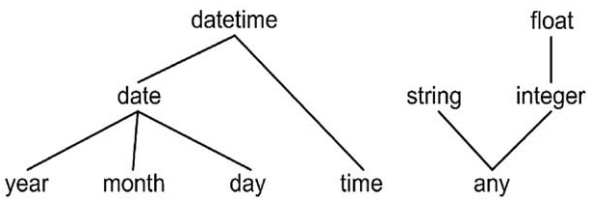


**Fig. 3.** A hierarchy of data types.

at all. We represent all possible matches of $a_i$ in $T$ using a frame of discernment, $\Theta = \{\langle a_i, b_1\rangle, \langle a_i, b_2\rangle, \ldots, \langle a_i, b_n\rangle, \langle a_i, null\rangle\}$, where $\langle a_i, b_j\rangle$, for $j = 1, 2, \ldots, n$, represent all possible candidate correspondences and $\langle a_i, null\rangle$ represents that there is no correspondence of $a_i$ in the target schema.

**Example 2.** As shown in Fig. 1, for 'Author' on abebooks.com, a frame of discernment for it is defined as follows:

$\Theta = \{\langle Author, Category\rangle, \langle Author, Keywords\rangle, \langle Author, Title\rangle, \langle Author, Subject\rangle, \langle Author, Publisher\rangle, \langle Author, Author\rangle, \langle Author, OurReference\rangle, \langle Author, ISBN\rangle, \langle Author, Availability\rangle, \langle Author, ReleasedDate\rangle, \langle Author, UpperPriceLimit\rangle, \langle Author, Binding\rangle, \langle Author, null\rangle\}$

### 3.2.1. Generating subsets of indistinguishable attribute correspondences

Given frame $\Theta = \{\langle a_i, b_1\rangle, \langle a_i, b_2\rangle, \ldots, \langle a_i, b_n\rangle, \langle a_i, null\rangle\}$ for a source attribute $a_i$, an individual matcher may not be able to distinguish two attribute correspondences in $\Theta$ from each other in terms of level of confidence on the validity of the attribute correspondences. For example, if the data type of the source attribute is compatible with the data types of two target attributes, the data type matcher cannot distinguish the two attribute correspondences from each other. The data type matcher cannot distinguish an attribute correspondence, in which the data type of the source attribute is compatible with the one of the target attributes, from $\langle a_i, null\rangle$ either. A string similarity based matcher cannot distinguish two strings 'abc' and 'bcd' from each other when string 'bc' is matched to each of them. We therefore cluster attribute correspondences in $\Theta$ into subsets of indistinguishable attribute correspondences. We do not have any further knowledge on the subset of indistinguishable attribute correspondences to be able to tell which attribute correspondence is more valid than the others. What we have is some degree of belief that one of the attribute correspondences in the subset is valid, according to a particular matcher.

In Example 2, based on edit distance based similarity, the frame of discernment for 'Author' on abebooks.com is divided into the following indistinguishable subsets:

$\{\langle Author, Category\rangle, \langle Author, Title\rangle, \langle Author, OurReference\rangle\}$

$\{\langle Author, Keywords\rangle, \langle Author, Subject\rangle, \langle Author, Publisher\rangle, \langle Author, ISBN\rangle\}$

$\{\langle Author, Author\rangle\}$

$\{\langle Author, Availability\rangle\}$

$\{\langle Author, ReleasedDate\rangle\}$

$\{\langle Author, UpperPriceLimit\rangle\}$

$\{\langle Author, Binding\rangle\}$

$\{\langle Author, null\rangle\}$

### 3.2.2. Generating mass distributions on subsets of indistinguishable attribute correspondences

Given a similarity based matcher, for each indistinguishable subset of attribute correspondences, we have a similarity value for each correspondence in the set, which represents how well the two attributes in the correspondence match according to the measure used by the matcher. Suppose we have subset $A = \{\langle a_i, b_{i1}\rangle, \langle a_i, b_{i2}\rangle, \ldots, \langle a_i, b_{il}\rangle\}$, a mass assigned to $A$ is calculated based on the similarity values for all the attribute correspondences in $A$ as follows:

$$m'(A) = 1 - \prod_{j=1}^{l}(1 - Sim(a_i, b_{ij})) \tag{10}$$

where $Sim(a_i, b_{ij})$ is similarity value for the correspondence $\langle a_i, b_{ij}\rangle$ in $A$.

Since we do not have a similarity value for $\langle a_i, null\rangle$ by the matcher, we form a special singleton subset, $\{\langle a_i, null\rangle\}$. The mass assigned to the subset is calculated as follows:

$$m'(\{\langle a_i, null\rangle\}) = \prod_{j=1}^{n}(1 - Sim(a_i, b_j)) \tag{11}$$

The mass assigned to $\{\langle a_i, null\rangle\}$, therefore, represents the degree of belief that none of the target attributes is the attribute correspondence of source attribute, $a_i$.

We scale the mass distribution, $m'$, so that the sum of all masses assigned to every indistinguishable subset equals to 1:

$$m(A) = \frac{m'(A)}{\sum_{B \subseteq \Theta} m'(B)} \tag{12}$$

where $A$ and $B$ are indistinguishable subsets of $\Theta$.

Therefore these indistinguishable subsets of $\Theta$ are the focal elements of $\Theta$. The mass assigned to any other subset of $\Theta$ is 0.

In Example 2, based on edit distance based similarity, for 'Author' on abebooks.com, we have a mass function defined as follows:

$m_{ed}(\{\langle Author, Category\rangle, \langle Author, Title\rangle, \langle Author, OurReference\rangle\}) = 0.175$

$m_{ed}(\{\langle Author, Keywords\rangle, \langle Author, Subject\rangle, \langle Author, Publisher\rangle, \langle Author, ISBN\rangle\}) = 0.191$

$m_{ed}(\{\langle Author, Author\rangle\}) = 0.414$

$m_{ed}(\{\langle Author, Availability\rangle\}) = 0.035$

$m_{ed}(\{\langle Author, ReleasedDate\rangle\}) = 0.063$

$m_{ed}(\{\langle Author, UpperPriceLimit\rangle\}) = 0.071$

$m_{ed}(\{\langle Author, Binding\rangle\}) = 0.052$

$m_{ed}(\{\langle Author, null\rangle\}) = 0.000$

For the data type based matcher, we create two subsets of indistinguishable attribute correspondences, $A$ and $A'$. $A$ contains both attribute correspondences in which the data type of the source attribute is compatible with the one of the target attribute and $\langle a_i, null\rangle$ where $a_i$ is the source attribute. $A'$ contains attribute correspondences in which the data type of the source attribute is not compatible with the one of the target attribute. We have a mass function defined as: $m_{dd}(A) = 1$ and $m_{dd}(A') = 0$.

### 3.3. Combining mass functions from multiple matchers

We now have a mass function by each of the individual matchers, which assigns a mass to every indistinguishable subset of $\Theta$. Using Dempster's combination rule, we can take into account different sources of evidence witnessed by different matchers by combining the appropriate mass functions by these matchers.

In Example 2, for 'Author' on abebooks.com, we have a combined mass function defined as follows:

$m(\{\langle Author, OurReference\rangle\}) = 0.020$
$m(\{\langle Author, UpperPriceLimit\rangle\}) = 0.003$
$m(\{\langle Author, Publisher\rangle\}) = 0.048$
$m(\{\langle Author, Author\rangle\}) = 0.929$

### 3.4. Selecting top k attribute correspondences

This section shows how to select the top $k$ attribute correspondences of each source attribute. The mass function produced in the previous section is used to derive both the *bel* and *pl* functions, which are further used to select the top $k$ attribute correspondences of each source attribute. When all masses have been assigned to subsets with a single element only, *bel*, *pl* and *m* are all the same. In this case, we choose those attribute correspondences with top $k$ masses as the top $k$ attribute correspondences.

In Example 2, for 'Author' on abebooks.com, we have the top three attribute correspondences as follows:

$m(\{\langle Author, Author\rangle\}) = 0.929$
$m(\{\langle Author, Publisher\rangle\}) = 0.048$
$m(\{\langle Author, OurReference\rangle\}) = 0.020$

## 4. Resolving conflicts between attribute correspondences

We have now the top $k$ attribute correspondences of each source attribute, which have been selected for an individual source attribute only. There might be conflicts between attribute correspondences of two source attributes (i.e. the best correspondences of two different source attributes are the same target attribute). To resolve any conflicts, the attribute correspondences of source attributes are collectively selected to maximize the sum of all the masses on the attribute correspondence of every source attribute. The algorithm is given in Algorithm 1.

**Algorithm 1.** Resolving conflicts

**Input:** A set of all the possible combinations of attribute correspondences for each source attribute $\Omega = \{C | C = \{\langle a_1, b'_1\rangle, \langle a_2, b'_2\rangle, \ldots, \langle a_m, b'_m\rangle\}\}$, where $\langle a_i, b'_i\rangle \in \{\langle a_i, b_{i1}\rangle, \langle a_i, b_{i2}\rangle, \ldots, \langle a_i, b_{ik}\rangle\}$ (the top $k$ correspondences of $a_i$)
**Output:** A collection of attribute correspondences with the maximum sum of the mass values of the correspondences for every source attribute

```
1:  Set Max to 0, Best to ∅.
2:  for each C∈ Ω do
3:      Set Sum to Σ_{i=1}^{m} m(⟨a_i, b'_i⟩),where m(⟨a_i, b'_i⟩) is the mass
    function value of ⟨a_i, b'_i⟩
4:      if Sum > Max then
5:          Set Max to Sum, Best to C;
6:  return Best
```

**Example 3.** For every attribute on www.esearchhome.com, as shown in Fig. 4, we have their top three attribute correspondences on www.esearchhome.com as follows:

$m(\{\langle State, SelectaState\rangle\}) = 0.921$
$m(\{\langle State, SelectaCity\rangle\}) = 0.071$
$m(\{\langle State, Null\rangle\}) = 0.008$

$m(\{\langle County, SelectaCity\rangle\}) = 0.286$
$m(\{\langle County, Null\rangle\}) = 0.190$
$m(\{\langle County, MinimumNumberofBedrooms\rangle\}) = 0.167$

$m(\{\langle City/Cities, SelectaCity\rangle\}) = 0.920$
$m(\{\langle City/Cities, SelectaState\rangle\}) = 0.069$
$m(\{\langle City/Cities, Null\rangle\}) = 0.011$

$m(\{\langle PropertyType, SelectaCity\rangle\}) = 0.467$
$m(\{\langle PropertyType, SelectaState\rangle\}) = 0.466$
$m(\{\langle PropertyType, Null\rangle\}) = 0.067$

$m(\{\langle Bedrooms, MinimumNumberofBedrooms\rangle\}) = 0.824$
$m(\{\langle Bedrooms, MinimumNumberofBathrooms\rangle\}) = 0.100$
$m(\{\langle Bedrooms, Null\rangle\}) = 0.041$

$m(\{\langle Bathrooms, MinimumNumberofBathrooms\rangle\}) = 0.830$
$m(\{\langle Bathrooms, MinimumNumberofBedrooms\rangle\}) = 0.101$
$m(\{\langle Bathrooms, PriceRange\rangle\}) = 0.049$

$m(\{\langle Price, PriceRange\rangle\}) = 0.945$
$m(\{\langle Price, MinimumNumberofBedrooms\rangle\}) = 0.027$


**Fig. 4.** Two Web query interfaces in the real estate domain.

$m(\{\langle Price, MinimumNumberofBathrooms\rangle\}) = 0.026$

$m(\{\langle Size, PriceRange\rangle\}) = 0.387$
$m(\{\langle Size, MinimumNumberofBedrooms\rangle\}) = 0.333$
$m(\{\langle Size, MinimumNumberofBathrooms\rangle\}) = 0.248$

There are conflicts among 'County', 'City' and 'Property Type' and also between 'Price' and 'Size'. Using Algorithm 1, we get the following attribute correspondences that have the maximum sum of mass values:

$\langle State, SelectaState\rangle$
$\langle City/Cities, SelectaCity\rangle$
$\langle Bedrooms, MinimumNumberofBedrooms\rangle$
$\langle Bathrooms, MinimumNumberofBathrooms\rangle$
$\langle Price, PriceRange\rangle$
$\langle County, Null\rangle$
$\langle PropertyType, Null\rangle$
$\langle Size, Null\rangle$

In Example 2, for every attribute on abebooks.com, as shown in Fig. 1, we have their top three attribute correspondences as follows:

$m(\{\langle Author, Author\rangle\}) = 0.929$
$m(\{\langle Author, Publisher\rangle\}) = 0.048$
$m(\{\langle Author, OurReference\rangle\}) = 0.020$

$m(\{\langle Title, Title\rangle\}) = 0.948$
$m(\{\langle Title, Binding\rangle\}) = 0.027$
$m(\{\langle Title, Category, \rangle\}) = 0.014$

$m(\{\langle Publisher, Publisher\rangle\}) = 0.937$
$m(\{\langle Publisher, Category\rangle\}) = 0.041$
$m(\{\langle Publisher, Subject\rangle\}) = 0.012$

$m(\{\langle Keywords, Keywords\rangle\}) = 1.000$

$m(\{\langle ISBN, ISBN\rangle\}) = 1.000$

$m(\{\langle PublishedDate, ReleaseDate\rangle\}) = 1.000$

$m(\{\langle BooksellerCountry, OurReference\rangle\}) = 0.224$
$m(\{\langle BooksellerCountry, Category\rangle\}) = 0.196$
$m(\{\langle BooksellerCountry, Subject\rangle\}) = 0.153$

$m(\{\langle Price, UpperPriceLimit\rangle\}) = 0.532$
$m(\{\langle Price, Subject\rangle\}) = 0.352$
$m(\{\langle Price, Title\rangle\}) = 0.087$

No source attributes have the same target attribute and hence are in conflict. Using Algorithm 1, we get the following attribute correspondences that have the maximum sum of mass values:

$\langle Author, Author\rangle$
$\langle Title, Title\rangle$
$\langle Publisher, Publisher\rangle$
$\langle Keywords, Keywords\rangle$
$\langle ISBN, ISBN\rangle$
$\langle PublishedDate, ReleaseDate\rangle$
$\langle BooksellerCountry, OurReference\rangle$
$\langle Price, UpperPriceLimit\rangle$

## 5. Experimental results

We use a set of 88 query interfaces selected from the ICQ Query Interfaces data set at UIUC, which contains manually extracted schemas of interfaces in five different domains: airfares, automobiles, books, jobs, and real estates, which involve 1:1 matching only (as we have focused on 1:1 matching in this paper). In each domain we create a uniform query interface as the source interface, and take each of the query interfaces in the domain as a target interface.

We use three performance metrics: precision (P), recall (R), and F-measure (F). Precision is the percentage of correct matches over all the matches by a matcher. Recall is the percentage of correct matches by a matcher over all the matches by domain experts. F-measure is the incorporation of precision and recall as follows: $F = 2PR/(P+R)$.

First, in each domain we use three individual matchers: edit distance, Jaro distance and semantic similarity (the data type matcher cannot be used alone), and compare them with our new approach that combines four individual matchers: edit distance, Jaro distance, semantic similarity and data type. Given a source attribute $a_i$, for each individual matcher, we treat $\langle a_i, null\rangle$, i.e. 'no match', as a possible answer. We also have every candidate attribute correspondence, $\langle a_i, b_j\rangle$, as a possible answer. We therefore have a set of possible answers, $\theta = \{\langle a_i, b_1\rangle, \langle a_i, b_2\rangle, \ldots, \langle a_i, b_n\rangle, \langle a_i, null\rangle\}$. One and only one of these possible answers is true. Therefore, in our experiments, the number of matches by a matcher equals to the number of matches identified by experts, and precision, recall and F-measure turn out to be the same. We use precision only.

For every $\langle a_i, b_j\rangle$, for $j = 1, 2, \ldots, n$, we have a similarity value by the matcher. Since we do not have a similarity value for $\langle a_i, null\rangle$, a value is calculated as follows:

$$s(\langle a_i, null\rangle) = \prod_{j=1}^{n}(1 - Sim(a_i, b_j))$$

In our experiments, we do not need a threshold to decide whether there is a match. If similarity values for every candidate attribute correspondence are all very low, a value calculated for $\langle a_i, null\rangle$, i.e. 'no match', will be high enough so that $\langle a_i, null\rangle$ will be selected as the top answer.

As shown in Table 1, our matcher gets much higher precision than the individual matchers.

The following is one of the cases in the Jobs domain used in the experiments, in which we have selected 'www.alljobsearch.com' and 'www.AfterCollege.com' as the source and target interfaces. 'alljobsearch' contains the following attributes: 'Category', 'Keyword', 'Country', 'State/Province', 'City', 'Job Type' and 'Post Date'. 'AfterCollege' contains the following attributes: 'Job type', 'Industry', 'Type of work', 'Location' and 'Keyword'. In this case, the results by each of the individual matchers and our combined matcher show how our matcher has recognized all correct attribute correspondences while individual matchers have failed to do so.

**Table 1**
Precisions of individual matchers and our combined matcher.

| | Edit distance (%) | Jaro distance (%) | Semantic similarity (%) | Ours (%) |
|---|---|---|---|---|
| Airfares | 83.3 | 56.8 | 86.4 | 92.0 |
| Autos | 84.4 | 48.1 | 93.1 | 96.3 |
| Books | 87.0 | 48.8 | 92.0 | 94.4 |
| Jobs | 68.5 | 50.0 | 71.0 | 91.9 |
| Estates | 86.8 | 52.9 | 81.6 | 93.8 |
| Average | 82.1 | 51.3 | 84.8 | 93.7 |

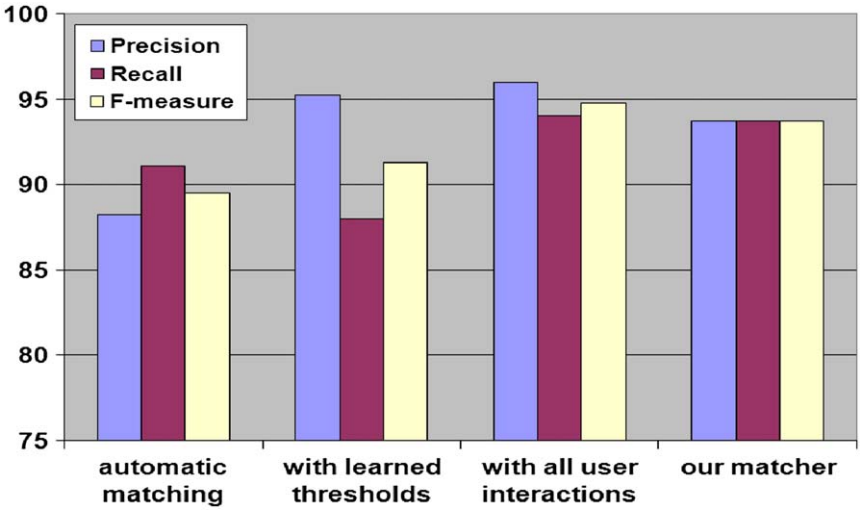


**Fig. 5.** Precisions, recalls and F-measures of different matchers.

Jaro distance-based matcher:

⟨*Keyword*, *Keyword*⟩
⟨*Country*, *Industry*⟩
⟨*City*, *Location*⟩
⟨*JobType*, *Jobtype*⟩
⟨*Category*, *Null*⟩
⟨*State*/*Province*, *Null*⟩
⟨*PostDate*, *Typeofwork*⟩

in which attribute correspondence ⟨*Country*, *Industry*⟩ is wrong.

Edit distance-based matcher:

⟨*Keyword*, *Keyword*⟩
⟨*JobType*, *Jobtype*⟩
⟨*Category*, *Null*⟩
⟨*Country*, *Null*⟩
⟨*State*/*Province*, *Null*⟩
⟨*City*, *Null*⟩
⟨*PostDate*, *Null*⟩

in which attribute correspondences ⟨*Category*, *Null*⟩ and ⟨*City*, *Null*⟩ are both wrong.

Semantic-based matcher:

⟨*Keyword*, *Keyword*⟩
⟨*JobType*, *Jobtype*⟩
⟨*Category*, *Null*⟩
⟨*Country*, *Null*⟩
⟨*State*/*Province*, *Null*⟩
⟨*City*, *Null*⟩
⟨*PostDate*, *Null*⟩

in which attribute correspondences ⟨*Category*, *Null*⟩ and ⟨*City*, *Null*⟩ are both wrong.

Our combined matcher:

⟨*Category*, *Industry*⟩

⟨*Keyword*, *Keyword*⟩
⟨*City*, *Location*⟩
⟨*JobType*, *Jobtype*⟩
⟨*Country*, *Null*⟩
⟨*State*/*Province*, *Null*⟩
⟨*PostDate*, *Null*⟩

in which all seven attribute correspondences are correct.

Second, we compare our results with the work in [8], which uses a similar data set for their experiments, covering the same five domains as ours. However, they also handle $1:m$ matching. In their experiments, a $1:m$ match is counted as $m$ 1:1 matches. They did three experiments. The first is on automatic field matching which uses weight coefficients to combine multiple matchers and the clustering thresholds for two fields to be matched are set to zero for all domains. The second uses thresholds learned by user interactions. The last also uses user interactions for resolving uncertainties in match results. As shown in Fig. 5, without using learned

thresholds, the results of our approach are better. When the learned thresholds are used, their precision is better than ours, but we have higher recall and F-measure. Finally, when user interactions are also used to resolve uncertainties in match results, their results are better than ours. Our approach is effective and accurate for automatic schema matching across query interfaces without automated learning and user interaction.

## 6. Related work

Cupid [6] exploits linguistic and structural similarity between elements and uses a weighted formula to combine these two similarities together. However, weights have to be manually generated and are domain dependent.

COMA [2] allows users to tailor match strategies by selecting a combination of match algorithms for a given problem, including Max, Min, Average and Weighted strategies. It also allows users to provide feedback for improving match results. These strategies are effective in some situations while sometimes they cannot combine results effectively, and choosing strategies by users involves human efforts.

In [8], weight coefficients are also used to combine multiple matchers, which are set to some domain-independent empirical values. However, clustering is used to find attribute correspondences across multiple interfaces, in which thresholds are required for merging clusters. These thresholds need to be either manually set or learned from user interactions and are domain dependent. So this approach also involves human effort.

Some approaches [4,5] use attribute distribution rather than linguistic or domain information. Superior to other schema matching approaches, these approaches can discover synonyms by analyzing attribute distributions in the given schemas. However, they work well only when a large training data set is available, but this is not always the case.

## 7. Conclusions

We proposed a new approach to combining multiple matchers using DS theory and presented an algorithm for resolving conflicts among the correspondences of different source attributes. Applying different matchers to a set of candidate correspondences provides different sources of evidence, and mass distributions are defined on the basis of the match results from these matchers. We use Dempster's combination rule to combine these mass distributions, and choose the top $k$ correspondences of each source attribute. Conflicts between the correspondences of different source attributes are finally resolved. We implemented a prototype and tested it using a large data set that contains real-world query interfaces in five different domains. The experimental results demonstrate the feasibility and accuracy of our approach.

## Acknowledgments

The authors wish to thank Phil Bernstein for his helpful suggestions and comments on the research presented in the paper and the earlier version of the paper.

## Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at 10.1016/j.is.2009.08.004.

## References

[1] E. Rahm, P. Bernstein, A survey of approaches to automatic schema matching, VLDB Journal 10 (4) (2001) 334–350.
[2] H. Do, E. Rahm, COMA—a system for flexible combination of schema matching approaches, in: VLDB'02, 2002, pp. 610–621.
[3] A. Doan, P. Domingos, A. Halevy, Reconciling schemas of disparate data sources: a machine-learning approach, in: SIGMOD'01, 2001, pp. 509–520.
[4] B. He, K. Chang, J. Han, Discovering complex matchings across web query interfaces: a correlation mining approach, in: KDD'04, 2004, pp. 148–157.
[5] B. He, K.C.-C. Chang, Statistical schema matching across web query interfaces, in: SIGMOD Conference, San Diego, CA, 2003, pp. 217–228.
[6] J. Madhavan, P. Bernstein, E. Rahm, Generic schema matching with cupid, in: VLDB'01, 2001, pp. 49–58.
[7] J. Wang, J.-R. Wen, F. Lochovsky, W.-Y. Ma, Instance-based schema matching for web databases by domain-specific query probing, in: Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004, pp. 408–419.
[8] W. Wu, C. Yu, A. Doan, W. Meng, An interactive clustering based approach to integrating source query interfaces on the deep web, in: SIGMOD Conference, Paris, France, 2004, pp. 95–106.
[9] J. Lowrance, T. Garvey, Evidential reasoning: an developing concept, in: ICCS'81, 1981, pp. 6–9.
[10] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, New Jersey, 1976.
[11] G.A. Miller, WordNet: a lexical databases for English, Communications of ACM 38 (11) (1995) 39–41.
[12] R. Rada, H. Mili, E. Bichnell, M. Blettner, Development and application of a metric on semantic nets, IEEE Transactions on Systems, Man and Cybernetics 9 (1) (1989) 17–30.
[13] P. Hall, G. Dowling, Approximate string matching, Computing Surveys 1 (1980) 381–402.
[14] W. Cohen, P. Ravikumar, S. Fienberg, A Comparison of String Distance Metrics for Name-matching Tasks, 2003, pp. 73–78.