Probabilistic Schema Covering

Nguyen Thanh Toan[1], Phan Thanh Cong[1], Duong Chi Thang[2], Nguyen Quoc Viet Hung[3], and Bela Stantic[3]

[1] Back Khoa University, Ho Chi Minh, Vietnam
[2] Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland
[3] Griffith University, Gold Coast, Australia

Abstract. Schema covering is the process of representing large and complex schemas by easily comprehensible common objects. This task is done by identifying a set of common concepts from a repository called concept repository and generating a cover to describe the schema by the concepts. Traditional schema covering approach has two shortcomings: it does not model the uncertainty in the covering process, and it requires user to state an ambiguity constraint which is hard to define. We remedy this problem by incorporating probabilistic model into schema covering to generate probabilistic schema cover. The integrated probabilities not only enhance the coverage of cover results but also eliminate the need of defining the ambiguity parameter. Both probabilistic schema covering and traditional schema covering run on top of a concept repository. Experiments on real-datasets show the competitive performance of our approach.

Keywords: Schema matching, Schema covering, Probabilistic models

1 Introduction

An important part in business nowadays is how to cooperate with little effort. As new organizations arise, they need to integrate with existing organizations. However, new organizations also create their own schemas to describe their business logics, which makes cooperation harder. Since each organization has its own schema, the process of cooperation becomes painful and tiresome. In order to support cooperation, they need to match a schema of an organization to another. Schema matching is the process of finding correspondences between attributes of schemas [1,2]. It is used extensively in many fields [3,4,5], especially data integration [6,7]. In this paper, we study a new approach to schema matching by representing the schemas in high level of abstraction.

Schema matching traditionally performs matching on attribute-level to create attribute correspondences. This process is ineffective considering a large schema with thousands of attributes. Moreover, users tend to think schemas in terms of business object level when designing schema mappings. Therefore, describing the schemas at low-level structure such as attribute makes the manual matching process error-prone. This matching process would be easier if we could represent schemas in a higher level of abstraction.

Since schemas are used to capture everyday business activities and some of these business activities are the same among organizations, these schemas may contain many common parts. These common parts represent business objects that are comprised in the schemas which are called concepts. Some common concepts are “Address”, which describes the location of an entity, or “Contact”, which provides information about a person or an organization. Based on this observation, the process of describing schemas in terms of concepts can be made possible and it is called schema covering. Schema covering is a novel approach which has been studied carefully in [8,9].

In [8], the schema cover found by schema covering must satisfy a pre-defined ambiguity constraint which limits the number of times a schema attribute can be covered. However, this ambiguity constraint is hard to define since it must be stated beforehand and for each attribute in the schema. Traditional schema covering approach has another shortcoming that it does not support modeling uncertainty arisen in the covering process. For example, given a concept “InvoiceToAddres”, we are not sure that it represents subschema “ShipToAddress” or “BillToAddress” in the schema. Schema covering would cover both subschemas by the “InvoiceToAddress” concept as the subschemas and the concept all refer to the real-life concept “Address”. As a result, these problems lead to the employment of probability to express uncertainty. We propose incorporating probabilistic model into schema covering to introduce *probabilistic schema covering*. This allows us to solve the above shortcomings through user feedback. Without probabilities, the user feedback on schema covers would be hard.

In short, our goal is to create a new schema covering mechanism that supports user feedback and does not require a user-defined ambiguity constraint by incorporating probabilistic model. The paper is organized as follows. In §2, we model and formulate the problem of probabilistic schema cover. In §3, we present the probabilistic schema covering framework. In §4, we run various experiments on probabilistic schema covering. Finally, we provide some related work in §5 before §6 concludes the paper.

2 Model and Problem Statement

2.1 Background

Let schema $s = \{a_1, a_2, \ldots, a_n\}$ be a finite set of attributes. Attributes can be both simple or compound and compound attributes should not necessarily be disjoint. Attributes in a “Purchase Order” schema might be *item*, *unitPrice*, *firstName*, etc. A compound attribute might be *name*, combining two other attributes – *firstName*, and *lastName*. Defining a schema as a set of attributes is general as it can represent both relational and hierarchial schema.

Let s and s' be schemas with n and n' attributes, respectively. Let $S = s \times s'$ be the set of all possible *attribute correspondences* between s and s'. S is a set of attribute pairs (e.g., $(item, line)$). Each attribute correspondence (a pair of attributes) is associated with a confidence value $m_{i,j}(s, s')$ which represents the similarity between the i-th attribute of s and the j-th attribute of s' [10,11]. $m_{i,j}$ is defined to be a real number in $[0, 1]$ since it tends to express the probability of an attribute being substituted by the other. These confidence values are generated by schema matchers which are matching tools performing schema matching between two schemas. Some notable schema matchers are AMC [12], COMA++ [13].

A concept c is also a set of attributes: $c = a_1, a_2, ..., a_m$ where a_i is an attribute. A concept and a schema is basically the same as they are both sets of attributes. However, a concept is more meaningful as it describes a business object and it also has a smaller size. Concepts have relations between them called *micromappings*. Each micromapping is actually a set of attribute correspondences.

We also define the counterpart of concepts in the schema which are subschemas. A subschema t is also a set of attributes and it is a subset of schema s. Each concept and its subschema has an alignment score $f(t, c)$ which describes the similarity between them.

2.2 Schema Covering Framework

In general, the schema covering framework mentioned in [8] takes a schema and a prebuilt concept repository as input. The concept repository is a corpus of predefined concepts, which is built before-hand [8]. Schema covering finds the cover of a schema over two steps: schema decomposition and schema covering.

Schema decomposition. In the first step, given a schema s and a concept repository, the schema is decomposed into smaller parts based on the concepts in the repository. More specifically, we find the corresponding subschema t of each concept in the repository by matching the schema with the concept. The corresponding attributes of the concept attributes are grouped as the attributes of the subschema. Each concept and subschema forms a $\langle subschema, concept\rangle$ pair that will be used in the schema covering step. Finally, the alignment score of a pair $f(t, c)$ is calculated based on its correspondences where $\langle t, c\rangle$ is a $\langle subschema, concept\rangle$ pair and f is a pre-defined function. The output of this decomposition step is a set of $\langle subschema, concept\rangle$ pairs with an alignment score attached to each pair.

Schema covering. From the set of pairs after decomposition, we select the pairs that best cover the schema. The selected pairs, which are called a schema cover, must satisfy many constraints depending on the schema covering approach. The formal definition of a schema cover is as follows:

Definition 1. *Given a set of subschemas T_s of schema s, a set C of concepts, we define a set of valid matchings between subschemas and concepts:*

$$E(T_s, C) = \{(t, c)|t \in T_s, c \in C\}$$

where (t, c) is a set of attribute correspondences between subschema t and concept c. A cover of s by C, $v_{s,C} \subseteq E(T_s, C)$ is a subset of valid matchings between T_s and C.

The schema cover found by traditional schema covering approach must satisfy an ambiguity constraint which limits the number of times a schema attribute can be covered. Therefore, traditional schema covering approach is also called ambiguity-based schema covering, which is discussed in [8]. Having described the traditional schema covering approach, we can turn to the problem we want to solve.

2.3 Problem Statement

In order to incorporate probabilistic model into schema covering, we need to find a mechanism to integrate probabilistic model into schema covering.

Formally, our problem takes a set of $\langle subschema, concept \rangle$ pairs, $E(T_s, C) = \{(t, c) | t \in T_s, c \in C\}$, as input where T_s is a set of sub-schemas and C is a set of concepts in the repository. Each pair is attached with an alignment score $f(t, c)$ where f is a user-defined function. These pairs together with their alignment scores are taken from the decomposition result. In other words, we do not focus on the decomposition step and we assume that the decomposition result is available before.

In this problem, we want to compute a probabilistic schema cover. It is a set of possible covers v_i and each cover is associated with a probability $Pr(v_i)$. The formal definition for probabilistic schema cover is described as follows.

Problem 1 (Probabilistic Schema Cover). *Let E be a set of $\langle subschema, concept \rangle$ pairs. The probabilistic schema cover built from E is a set $V = \{(v_1, Pr(v_1)), \ldots, (v_n, Pr(v_n))\}$ such that*

- *For each $i \in [1, n]$, v_i is a cover and for every $i, j \in [1, n]$, $i \neq j \Rightarrow v_i \neq v_j$*
- *$Pr(v_i) \in [0, 1]$ and $\sum_{i=1}^{n} Pr(v_i) = 1$*

The running example below illustrates a probabilistic schema cover.

Example 1. *Assuming we have found two pairs $\langle t_1, c_1 \rangle$ and $\langle t_2, c_2 \rangle$, their alignment scores are respectively 0.8 and 0.3. From this set of pairs, we can generate a set of possible covers namely $\{v_1, v_2, v_3, v_4\} = \{\{\langle t_1, c_1 \rangle, \langle t_2, c_2 \rangle\}, \{\langle t_1, c_1 \rangle\}, \{\langle t_2, c_2 \rangle\}, \emptyset\}$ Each cover is attached with a probability, for example, $(Pr(v_1), Pr(v_2), Pr(v_3), Pr(v_4))$ $= (0.24, 0.56, 0.06, 0.14)$. Together, they form a probabilistic schema covering.*

3 Probabilistic schema covering

In this section, we discuss the approach to build a probabilistic schema cover. We first describe what properties a good cover must have. After that, we discuss the probabilistic schema covering framework. In the framework, we show how to generate a probabilistic cover from a set of pairs after decomposition.

3.1 Cover characteristics

A schema covering approach must satisfy two fundamental properties to ensure a good cover result.

- *Overlapping of subschemas:* Schema covering must accept the overlapping of subschemas. Since an attribute meaning is ambiguous, it is unreasonable to enforce "one attribute - one concept". For example, attribute "name" could be a person's name or a company's name depending on the context. We may need two concepts "Person" or "Company" to cover this attribute. In addition, as the concepts generating these subschemas are also overlapping, which leads to the overlapping of subschemas.
- Although schema covering can add concepts that cover the same parts of the schema, schema covering should not add bogus concepts into the final cover. A bogus concept is one that is not related to the schema as illustrated in Fig. 1. There are two cases that the concept is not related to the schema: it covers an insignificant part of the schema or it covers the schema incorrectly. For example, the concept "Organization" in Fig. 1 is a bogus concept as it covers only two attributes but one of them is incorrect.

These properties are the guideline to create a good cover for a schema covering approach.

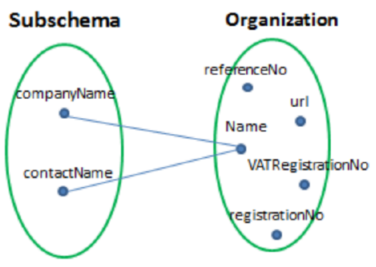

Fig. 1. Organization is a bogus concept

3.2 Framework

The probabilistic schema covering framework has three steps as described in Fig. 2. It takes a set of pairs after decomposition E as input and return a probabilistic schema cover containing a set of covers with probabilities attached to each of them. There are three steps in the framework: generating possible covers, assigning probabilities and consolidating probabilistic cover. These steps are explained shortly below to get the overview of the framework.

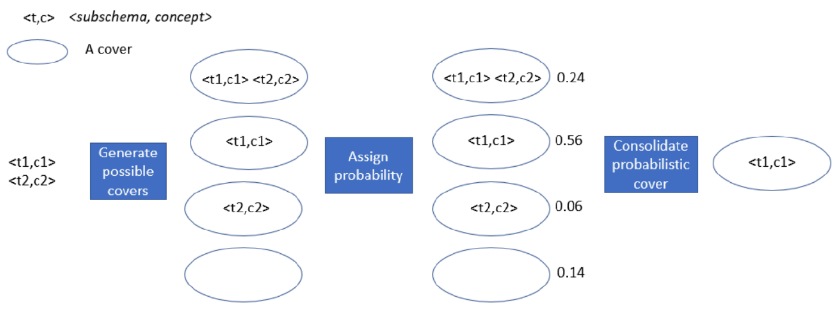

Fig. 2. The probabilistic schema covering framework

One of the fundamental requirement for the probabilities in the probabilistic cover is that they must be added up to 1. Therefore, we generate all the subsets of the decomposition result pairs E in the first step. For each subset of E, we create a candidate cover then it is assigned a probability. Since the number of possible covers is enormous, we present a technique to decrease the computational space.

After generating all possible cover, each cover is assigned a probability in the next step. The assigned probabilities must comply to a consistency constraint as described in §3.4. The constraint is introduced to guarantee the assigned probability to be consistent with the assigning cover.

After running probabilistic schema covering, we have generated a set of covers with probabilities. In the last step, we need to consolidate it from a set of covers to one cover (e.g. by selecting one with highest probability).

3.3 Generate all possible covers

From a set of pairs $E = \{(t, c)|t \in T_s, c \in C\}$ found after decomposing the schema, we generate all its subsets $\Omega = \{v_i|v_i \subset E\}$. Generating its subsets using all the pairs would lead to computational explosion since the size of Ω, $|\Omega| = 2^{|E|}$, is large. Therefore, we need some methods to reduce the computational space.

We introduce the alignment score threshold λ and the error window ϵ to decrease the size of the computational space. Using the threshold λ and the error window ϵ, we define two sets of pairs E_c and E_u:

- Certain set $E_c = \{(t,c) \in E | f(t,c) \geq \lambda + \epsilon\}$
- Uncertain set $E_u = \{(t,c) \in E | f(t,c) < \lambda + \epsilon \wedge f(t,c) \geq \lambda - \epsilon\}$

By setting the alignment score threshold λ, we want to focus only on the promising pairs. Pairs with alignment scores higher than the threshold are more likely to be correct. On the other hand, the error window value ϵ represents pairs that we are unsure if they are correct or not. That means we need to assign probabilities to only these pairs in E_u to express uncertainty.

From the uncertain set of pairs E_u, we generate the possible covers $\Omega_u = v_i^* | v_i^* \subset E_u$. Therefore, the number of possible covers $|\Omega_u|$ is $2^{|E_u|}$. Since $2^{|E_u|} \ll 2^{|E|}$, we have reduced the computational space significantly. Finally, the probabilistic schema cover for E is computed based on Ω_u as follows: $\Omega = \{v_i | v_i = v_i^* \cup E_c, v_i^* \in \Omega_u\}$ and $Pr(v_i) = Pr(v_i^*)$.

In Alg. 1, we describe how to generate the probabilistic cover for the set of pairs E. In Line 1-7, we divide the decomposition result into two sets. In Line 8, we generate subsets of the uncertain pairs and then assign probability to each of them in Line 11. Finally, we combine each set of pairs after assigning probability with the certain pairs to generate a cover of the probabilistic cover in Lines 10-13. The probability assignment step is discussed next.

Algorithm 1: Generating probabilistic cover

```
   input : $P$                                                   ▷ a set of pairs
           $\lambda$                                             ▷ an alignment score threshold
           $\epsilon$                                            ▷ an error window
   output: A set of covers with probability $V$
1  $C = \emptyset$                                               ▷ A set of certain pairs
2  $U = \emptyset$                         ▷ A set of uncertain pairs for pair $p \in P$ do
3      if $p.alignmentScore \geq \lambda + \epsilon$ then
4          $C.add(p)$
5      else if $p.alignmentScore < \lambda + \epsilon$ and $p.alignmentScore \geq \lambda - \epsilon$ then
6          $U.add(p)$
7  $S = \emptyset$                                               ▷ sets of uncertain pairs
8  $S = generateSubsets(U)$
9  $assignProbability(S)$
10 for set $s \in S$ do
11     $c.add(C)$
12     $c.add(s)$
13     $V.add(c)$
14 return $V$
```

3.4 Assign probability to each cover

After the first step, we have generated a set of possible covers Ω_u from the uncertain set of pairs E_u. In this step, we assign probability to each cover $v_i^* \in \Omega_u$.

Consistency constraint Despite the fact that alignment scores express how similar between the subschemas and the concepts, they do not tell us which concept a subschema should align to. For example, although a subschema “ShipToAddress” is more similar to the concept “DeliverToAddress” than to “BillToAddress”, it is still reasonable to align “ShipToAddress” to “BillToAddress” in a cover. The reason is that some people use their delivery address also as a billing address. Therefore, we could find probabilistic schema covers that still make sense according to a set of $\langle subschema, concept\rangle$ pairs. In other words, there could be sets of probabilistic schema covers that are consistent with a set of $\langle subschema, concept\rangle$ pairs. As a result, we can specify a consistency constraint to define which probabilistic covers are consistent with a set of pairs.

Definition 2. *A probabilistic cover V is consistent with a pair (t, c) if the sum of probabilities of all covers that contain (t, c) equals the alignment score $f(t, c)$. A probabilistic cover V is consistent with a pair (t, c) if*

$$\sum_{(t,c)\in v_i} Pr(v_i) = f(t, c)$$

A probabilistic cover V is consistent with a set of pairs M if it is consistent with each pair in M.

This constraint is introduced to ensure that a cover containing a pair with low alignment score has low probability. Since a pair with low alignment score is more likely to be incorrect, the cover in which it participates is also less likely to be correct. We illustrate this observation in Example 2.

Example 2. *Applying the consistency constraint into the covers in Example 1, we have the following equations:*

$$Pr(v_1) + Pr(v_2) = 0.8 \quad (1)$$
$$Pr(v_1) + Pr(v_3) = 0.3 \quad (2)$$

Since $Pr(v_1) \geq 0$, $Pr(v_3) \geq 0$ and due to Eq. 1, $Pr(v_1)$, $Pr(v_3)$ must be less than 0.3. However,the probability for cover v_2 which does not contain pair $\langle t_2, c_2\rangle$ is higher as $Pr(v_1) < 0.3$ then $Pr(v_2)$ must be higher than 0.5 according to Eq. 2. Therefore, the consistency constraint ensures that covers containing pairs with low alignment scores: v_1, v_3 have low probabilities and vice versa.

However, given a set of pairs E_u, there are various probabilistic schema covers that are consistent with it. In Example 2, $(Pr(v_1), Pr(v_2), Pr(v_3), Pr(v_4)) = (0.1, 0.7, 0.2, 0)$ or $(Pr(v_1), Pr(v_2), Pr(v_3), Pr(v_4)) = (0.24, 0.56, 0.06, 0.14)$ are the possible solutions. In the following, we describe how to select a probabilistic schema cover from a set of probabilistic covers that are consistent with a set of pairs.

Entropy maximization Among the probabilistic covers that are consistent with a set of pairs, we select one that maximizes the entropy of its probabilities. Maximizing entropy is reasonable that given a set of candidate covers, we are not sure which one is correct.

This uncertainty leads to the observation that these covers should be treated unbiasedly. Maximizing entropy is not a new method since it has been used in various fields such as natural language processing [14,15].

The probability assignment problem can now be reformulated to a constraint optimization problem (OPT). That is, we need to assign the probabilities to the covers in a probabilistic cover such that both the consistency constraint is satisfied and the entropy is maximized. The optimization problem is described as follows.

Definition 3. *Let $Pr(v_1), \dots, Pr(v_n)$ be the probabilities of cover $v_1, \dots, v_n$ respectively. $Pr(v_i)$ is found by solving the following OPT problem:*

maximize $\sum_{i=1}^{n} -Pr(v_i) \log Pr(v_i)$, subject to:

1. $\forall i \in [1,n], 0 \leq Pr(v_i) \leq 1$
2. $\sum_{i=1..n} Pr(v_i) = 1$
3. $\forall (t,c) \in E_u : \sum_{j \in [1,n]:(t,c) \in v_j} Pr(v_j) = f(t,c)$

Probabilities for the covers in the running example can be found by solving the following problem:

Example 3. *maximize $-Pr(v_1)\log Pr(v_1) - Pr(v_2)\log Pr(v_2) - Pr(v_3)\log Pr(v_3) - Pr(v_4)\log Pr(v_4)$, subject to:*

1. $Pr(v_1) + Pr(v_2) + Pr(v_3) + Pr(v_4) = 1$
2. $Pr(v_1) + Pr(v_2) = 0.8$ *(consistency constraint)*
3. $Pr(v_1) + Pr(v_3) = 0.3$ *(consistency constraint)*
4. $\forall i \in [1,n], 0 \leq Pr(v_i) \leq 1$

This problem can be solved by using an OPT solver such as Knitro [16]. Solving this problem, we get the following solution $(Pr(v_1), Pr(v_2), Pr(v_3), Pr(v_4)) = (0.24, 0.56, 0.06, 0.14)$. We can generate many other solutions if we don't maximize the entropy such as $(Pr(v_1), Pr(v_2),$ $Pr(v_3), Pr(v_4))$ = $(0.1, 0.7, 0.2, 0)$. However, the latter solution is biased as it favors pair (t_2, c_2) over (t_1, c_1). In addition, this solution implies that the pairs (t_1, c_1) and (t_2, c_2) are dependent.

Observing the above example, it may seem counter-intuitive to consider an empty cover as a possible solution. However, if all the pairs have low alignment score, it is rational to regard the empty cover as a possible solution as other non-empty covers could be incorrect.

Probabilistic schema covering complies with the properties we described in §3.1. Since bogus concepts tend to have low alignment score, they are only contained in covers with low probabilities. However, users might focus on the top-k covers with high probabilities where these bogus concepts are not available. In other words, probabilistic schema covering satisfies the first property of generating good covers. Moreover, since the pairs in each cover of the probabilistic schema cover is a subset of the pairs from the decomposition step, these pairs may overlap. This means probabilistic schema covering satisfies the second property of good cover generation.

4 Experiments

We now describe experiments that validate the performance of the probabilistic schema covering and its application. The main goal of these experiments is to examine the quality of the cover obtained from probabilistic schema covering.

4.1 Experimental setup

Dataset We start by introducing the dataset being used for evaluation in this document. In fact, finding an appropriate dataset is a non-trivial task as the collected schemas must be relevant and belong to a same domain. We have collected 5 schemas from the Purchase Order domain. Their statistics are described in Table 1. From these schemas, we also create the golden mappings between them manually. The number of goldenmappings between pairs of schemas is described in Table 2.

Table 1. Statistics of the five schemas

	Apertum	CIDX	Excel	Noris	Paragon
#Nodes	140	40	54	65	77
#Internal Nodes	25/115	7/33	12/42	11/54	12/65
Depth	4	3	3	3	5

Table 2. #Golden mappings between schemas

	Apertum	CIDX	Excel	Noris	Paragon
Apertum		54	79	85	66
CIDX	54		65	32	49
Excel	79	65		50	60
Noris	85	32	50		45
Paragon	66	49	60	45	

Concept repository In order to facilitate the schema covering process, a concept repository must be built first. As it is hard to find a concept repository that contains concepts that are relevant to the above schemas, we create the concepts from the schemas based on their structure.

From three schemas Apertum, Paragon and CIDX, we create the concepts using the leaf-only strategy. If an attribute contains leaf attributes, we create a new concept. The concept name is the attribute's name and the concept attributes are the leaf sub-attributes of it. After breaking the schemas into concepts, we connect these concepts using the available golden correspondences (i.e. correspondences present in the ground truth). For each source and target attribute of a golden correspondence, we find the corresponding source and target attributes of the concepts. Next, we create correspondences between these concept attributes by COMA++ [13] with default parameters. We apply this process to generate all correspondences between the concept. Moreover, as we want the concepts to have various sizes, we create smaller concepts in the repository from the existing ones. For a set of concepts that are connected to each other, we create a new concept from the intersections of them. Next, mappings from these new concepts to their parents and parents' neighbors are made. Eventually, we have constructed a concept repository that contains concepts of various sizes and correct mappings between them. The statistics of the concept repository are: 45 concepts, 50 micromappings, 220 attributes, 5.089 attributes per concept in average, 1.11 micromappings per concept in average.

Metrics Let R be the set of correct correspondences found manually. Let F denote the set of correspondences that we generate (or we consider them to be correct). Let $I = R \cap F$ denote the actual correct correspondences in F. In order to evaluate the result, we use two typical metrics: precision, which is $|I|/|F|$ and recall, which is $|I|/|R|$. A high value of both precision and recall is desired. As it is hard to find the correct cover from such a large concept repository, we take a different approach to calculate the precision and recall. For each subschema and concept pair, we calculate its precision and recall value then we take the average to get the precision, recall of the whole cover.

4.2 Effects of score threshold and error window on cover result

In this experiment, we want to find the cover of schema Excel by the concept repository. We vary the threshold, error window to see their effects on the final cover. In this experiment, to consolidate cover, we select the cover with the highest probability.

The result is shown in Fig. 3. In general, precision and recall are high, both of them are higher than 60%. This means that we can find a good cover. Intuitively, the precision and recall increase when the threshold are higher. This is reasonable that the higher the threshold is, we only consider the more likely-correct pairs.

Moreover, when analyzing the raw data, we observe that with some epsilon values, the precision and recall values among the epsilon values are the same. Therefore, in Fig. 3, we only show the distinct data series. This means probabilistic covering is not sensitive to the epsilon value. When the epsilon value changes slightly, the precision and recall change a little about 10%. Therefore, it is easier for user to select the epsilon value without worrying decreasing the cover quality.

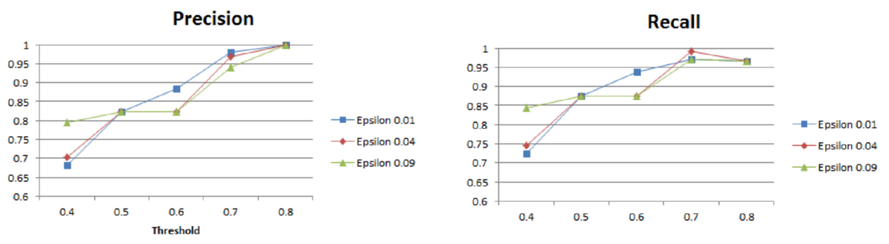

Fig. 3. Precision and recall of the cover of schema Excel

4.3 A comparison with ambiguity-based schema covering

In this experiment, we compare probabilistic schema covering with the ambiguity-based schema covering approach mentioned in [8]. Ambiguity-based schema covering is a covering method based on the ambiguity constraint on attributes. It takes a set of $\langle subschema, concept\rangle$ pairs and an ambiguity threshold as input. This ambiguity threshold defines the maximal times an attribute can be covered. While limiting the number of times an attribute can be covered, ambiguity-based covering tries to find the cover with the maximum total alignment score. It returns only one cover that satisfies two above conditions. In this experiment, we set the ambiguity threshold be 2.

For the probabilistic schema covering approach, we take the cover with the highest probability to compare while vary the threshold and the epsilon value. Since probabilistic schema covering relies on the alignment score threshold, we also add an alignment score threshold to ambiguity-based covering to make the two approaches comparable.

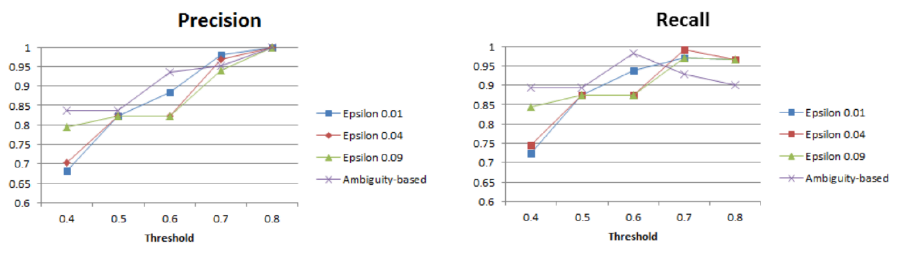

Fig. 4. A comparison with ambiguity-based covering

Fig. 4 shows the comparison of two approaches on the precision and recall value. With a low threshold, ambiguity-based covering has higher precision and recall. However, as we analyze the cover chosen by ambiguity-based covering, we found that this cover contains no pair that has alignment score lower than 0.5. On the other hand, probabilistic covering also consider various pairs with low alignment score that results in lower precision and recall. When the threshold increases, precision of ambiguity-based and probabilistic covering increase also. This is reasonable as we increase the threshold, we only take into account the promising pairs. Since these pairs contain less incorrect correspondences, the precision value must be high.

On the other hand, the higher the threshold, the lower the recall of ambiguity-based covering. As we increase the threshold, we leave out some correct pairs that have low alignment scores. Leaving out these correct pairs makes the recall value of ambiguity-based covering decrease. Although ambiguity- based covering has low recall value, probabilistic covering has high recall value in this case since it accepts overlapping of pairs. For example, the final cover generated by probabilistic covering when the threshold is 0.8 contains pairs of concept “Address Apertum”, “Address Noris” and “Address Paragon”. Whereas, the ambiguity-based cover contains only two pairs “Address Apertum” and “Address Noris” since the ambiguity threshold is only two. The pair “Address Paragon” is correct that adding it to the final cover would increase the recall value. This is the reason why ambiguity-based covering has a lower recall value than probabilistic covering when the threshold is high.

5 Related Work

Schema matching has been studied extensively for more than 25 years [17,18] that various schema matching techniques have been invented. Years of studies have shown the need to automate the matching process since matching schema manually is time consuming and error prone. Therefore, much efforts have been done towards automatic schema matchers that many automatic schema matchers have been proposed such as AMC [12], COMA++ [19] or OpenII [20]. These schema matchers support users with various schema matching techniques and some of them even implement reuse of matching results. For state-of-the-arts schema matching techniques and the comparison of schema matchers, one can consult the surveys at [17,21]. The closest approach to our work is the schema covering technique discussed in [8]. In this work, they have described the general framework for schema covering and showed that schema covering is possible. However, their approach required user to define an ambiguity constraint before starting the covering process, which is unrealistic. Our probability assignment approach for probabilistic schema covering basically follows the approach mentioned in [22,23]. In this work, they have described a systematic way to assign probabilities to schema mappings to generate probabilistic schema mappings. Since there are various analogies between two approaches such as a cover, a pair and its alignment score are similar to a schema mapping, a correspondence and its confidence value, their probability assignment approach is applied in our work.

6 Conclusions

This paper describes a novel approach to schema covering in order to mitigate uncertainty and improve covering results: probabilistic schema covering. In order to propose this approach, we have solved the problem of finding a mechanism to integrate probabilistic model into schema covering In order to generate a probabilistic schema cover, we first construct its possible set of covers and then we assign probability to each cover. The assigned probabilities must satisfy a consistency constraint and their entropy must also be maximized. Throughout the experiments, we have shown that probabilistic schema covering is a robust approach and competitive to traditional schema covering approach.

References

1. Hung, N.Q.V., Luong, X.H., Miklós, Z., Quan, T.T., Aberer, K.: Collaborative schema matching reconciliation. In: CoopIS. (2013) 222–240
2. Hung, N.Q.V., Tam, N.T., Chau, V.T., Wijaya, T.K., Miklós, Z., Aberer, K., Gal, A., Weidlich, M.: SMART: A tool for analyzing and reconciling schema matching networks. In: ICDE. (2015) 1488–1491
3. Hung, N.Q.V., Tam, N.T., Miklós, Z., Aberer, K.: On leveraging crowdsourcing techniques for schema matching networks. In: DASFAA. (2013) 139–154
4. Hung, N.Q.V., Luong, X.H., Miklós, Z., Quan, T.T., Aberer, K.: An MAS negotiation support tool for schema matching. In: AAMAS. (2013) 1391–1392
5. Hung, N.Q.V., Tam, N.T., Miklós, Z., Aberer, K.: Reconciling schema matching networks through crowdsourcing. EAI (2014) e2
6. NGUYEN, Q.V.H.: Reconciling Schema Matching Networks. PhD thesis, Ecole Polytechnique Federale de Lausanne (2014)
7. Gal, A., Sagi, T., Weidlich, M., Levy, E., Shafran, V., Miklós, Z., Hung, N.Q.V.: Making sense of top-k matchings: A unified match graph for schema matching. In: IIWeb. (2012) 6
8. Saha, B., Stanoi, I., Clarkson, K.L.: Schema covering: a step towards enabling reuse in information integration. In: ICDE. (2010) 285–296
9. Gal, A., Katz, M., Sagi, T., Weidlich, M., Aberer, K., Hung, N.Q.V., Miklós, Z., Levy, E., Shafran, V.: Completeness and ambiguity of schema cover. In: CoopIS. (2013) 241–258
10. Hung, N.Q.V., Wijaya, T.K., Miklós, Z., Aberer, K., Levy, E., Shafran, V., Gal, A., Weidlich, M.: Minimizing human effort in reconciling match networks. In: ER. (2013) 212–226

11. Hung, N.Q.V., Tam, N.T., Miklós, Z., Aberer, K., Gal, A., Weidlich, M.: Pay-as-you-go reconciliation in schema matching networks. In: ICDE. (2014) 220–231
12. Peukert, E., Eberius, J., Rahm, E.: Amc-a framework for modelling and comparing matching systems as matching processes. In: Data Engineering (ICDE), 2011 IEEE 27th International Conference on, IEEE (2011) 1304–1307
13. Arnold, P., Rahm, E.: Enriching ontology mappings with semantic relations. Data & Knowledge Engineering **93** (2014) 1–18
14. Berger, A.L., Pietra, V.J.D., Pietra, S.A.D.: A maximum entropy approach to natural language processing. Computational linguistics **22**(1) (1996) 39–71
15. Della Pietra, S., Della Pietra, V., Lafferty, J.: Inducing features of random fields. IEEE transactions on pattern analysis and machine intelligence **19**(4) (1997) 380–393
16. Byrd, R.H., Nocedal, J., Waltz, R.A.: Knitro: An integrated package for nonlinear optimization. In: Large-scale nonlinear optimization. Springer (2006) 35–59
17. Bernstein, P.A., Madhavan, J., Rahm, E.: Generic schema matching, ten years later. Proceedings of the VLDB Endowment **4**(11) (2011) 695–701
18. Rahm, E.: The case for holistic data integration. In: East European Conference on Advances in Databases and Information Systems, Springer (2016) 11–27
19. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with coma++. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, Acm (2005) 906–908
20. Seligman, L., Mork, P., Halevy, A., Smith, K., Carey, M.J., Chen, K., Wolf, C., Madhavan, J., Kannan, A., Burdick, D.: Openii: an open source information integration toolkit. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, ACM (2010) 1057–1060
21. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. the VLDB Journal **10**(4) (2001) 334–350
22. Das Sarma, A., Dong, X., Halevy, A.: Bootstrapping pay-as-you-go data integration systems. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM (2008) 861–874
23. Nguyen, Q.V.H., Nguyen, T.T., Miklos, Z., Aberer, K., Gal, A., Weidlich, M.: Pay-as-you-go reconciliation in schema matching networks. In: Data Engineering (ICDE), 2014 IEEE 30th International Conference on, IEEE (2014) 220–231