Ontology-centric Source Selection for Meta-querier Customization

Xiao Li, Randy Chow
Department of CISE, University of Florida
Gainesville, FL, 32611, USA
{xl1, chow}@cise.ufl.edu

Abstract

With an increasing number of semi-structured data sources, meta-queriers are introduced to facilitate effective information retrieval from multiple data sources that are accessible through query forms. However, a one-size-fits-all meta-querier cannot cater for various individual needs. In meta-querier customization, source selection is arguably one of the most critical problems. This paper proposes a capability-based source selection to meet user needs in terms of query capabilities. The major challenges include modeling, understanding and matching of the user needs and source capabilities. Our solution is based on a light-weight ontology, M-Ontology, which is generated from a number of verified mappings between heterogeneous query forms of the data sources. With the assistance of the concepts and relations in M-Ontology, user demands and source capabilities are modeled as concept sets, identified through query-form annotation, and matched by an additive utility function. The experiments on real-world data illustrate the potential of this ontology-centric method.

1 Introduction

As an increasing number of semi-structured data sources are available online through HTML query forms [4][7][17], integration of data sources is desirable for improving the efficiency of information retrieval. Meta-queriers are virtual data integration systems that shield users from data heterogeneity and source location. They provide the user with a uniform query form (a.k.a. *global form*) for simultaneously accessing a set of disparate data sources in the same domain. The user does not need to input repetitive information to each source query form (a.k.a. *local form*). Based on the *mappings* between global and local forms, user queries over the global form are respectively reformulated to the queries in terms of the local forms, and then the query results from data sources are presented to the user in an integrated format.

There is a wide divergence in the data sources (e.g., in terms of query capabilities, content qualities and site credibility). A one-size-fits-all meta-querier cannot cater for individual needs [28], even in the same application domain. User-driven selection on data sources is a convenient and straightforward method to customize meta-queriers. From the viewpoints of the user, the contents of global query forms and the selection of data sources are the most critical (or perhaps the only) factors that influence the contents retrieved from the meta-queriers. First, source selection determines the content coverage of meta-queriers. The meta-queriers are virtual data integration systems [13] that do not physically store any information. That is, the returned contents are completely determined by the underlying data sources. Second, modifying the controls in global forms is the only way for the user to express their demands on the results. All the returned contents should conform to the user constraints set by modifications of the *controls* (e.g., clicking radio buttons, dropping down menus, and entering texts). In a sense, the contents of global forms are also decided by the selection of data sources, since the global forms should only consists of the functionalities that are supported by every underlying data sources; otherwise, the results might include some/many records that violate the original user-specified conditions. Therefore, source selection is arguably one of the most critical problems in meta-querier customization.

This paper investigates how to exploit the query capabilities [14][6][25] of semi-structured sources for achieving more accurate selection. We propose an ontology-centric approach to source selection. A domain-based ontology (referred to as *M-Ontology*) was designed for meta-querier customization. In meta-querier customization, various customized meta-queriers are constructed, and thus a potentially large number of mappings between global and local forms need to be stored, managed and discovered. These unordered mappings are organized based on their semantics to form the concepts and relations of M-Ontology. This paper utilizes these concepts and relations to model source capabilities and user demands. Our major contributions can be summarized into the following three aspects:

• *Capability modeling and capture*: Without adequately accurate understanding of source capabilities, the selection in the previous research [10][21][16] is normally coarse-grained and unable to distinguish the functionality difference of the sources. In this paper, we view M-Ontology as a domain-specific capability repository. For each integrated data source, its capabilities can be automatically identified through the association of its query form with the concepts in M-Ontology.

• *Demand modeling and elicitation*: Modeling of user demands is still an open problem in the selection of semi-structured sources. We model the demands by a preference vector, in which each entry corresponds to a concept in M-Ontology. A semi-automatic solution to demand elicitation is also proposed through semantic annotation on the query forms of user-preferred data sources.

• *Demand matching*: The desirability of a specific data source for a particular user needs is quantified by a matching value. The value calculation is treated as a multi-criteria decision making problem. Each criterion corresponds to the desirability of a specific capability. An additive utility function is proposed to combine all the criteria, each of which is calculated on the basis of the similarity of the user's preference vector with the source's capability set.

In the remainder of this paper, we first review the related work on source selection in distributed information retrieval systems. Section 3 introduces the algorithm for capturing user demands and discovering the appropriate resources for the identified user demands. Section 4 details the experimental results on real-world data. Finally, we conclude with directions for future research.

2 Related Work

To distinguish our work from the current solutions to source selection, we discuss the solutions based on source types and selection mechanisms:

• Unstructured and (semi-)structured data sources.

In the field of distributed information retrieval systems, the prior researches on source selection mainly focus on the selection of sources containing unstructured data (a.k.a., texts). To acquire the contents of data sources, randomly generated queries are sent to obtain the sample texts. Through these fetched samples, the data sources can be represented as a single big document (such as in CORI[5], CVV[26] and KL[24]) or a set of big documents (such as, in ReDDE[20], CRCS[19] and SUSHI[22]).

With the rapid growth of the deep Web, (semi-)structured information sources have been experiencing a remarkable increase. Normally, the query interfaces of structured information sources are more complex than those of text sources. First, the query-based sampling becomes impractical since it is difficult to retrieve the sample documents through randomly generated queries. The automatically generated queriers usually cannot satisfy the hidden constraints on the inputs to the query forms. Second, the topics of sources can be directly acquired through the semantics analysis on the query forms. More complex query forms often contain more semantics at the same time. Several researches [10][21][16] have been conducted on *cross-domain source selection*. They cluster structured Web sources based on the query-form similarity so that each cluster corresponds to a single domain. However, the domain-oriented clustering is a coarse classification without considering the capability distinction.

• Selection-per-query and selection-per-engine

Source selection is one of the major research issues in meta-querier customization. The selection can be made when users issue a query (called *selection-per-query*) or when the meta-queriers are constructed (called *selection-per-engine*).

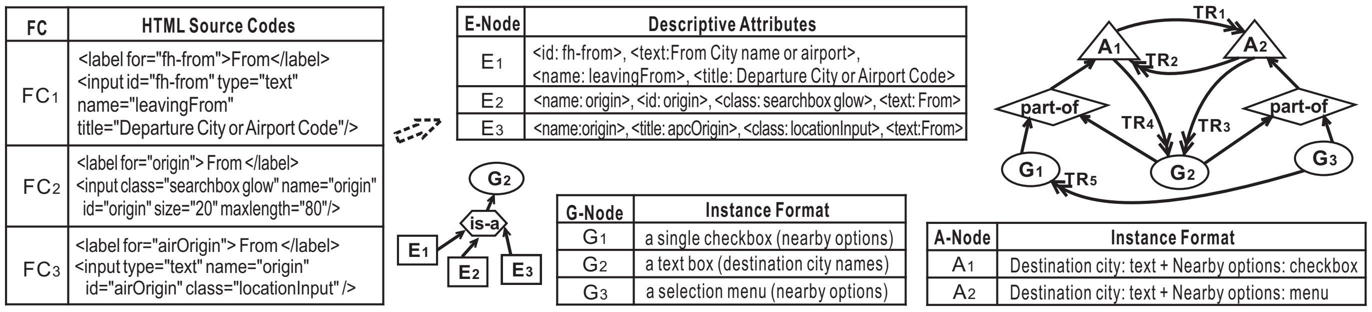
In some domains, the query forms are considerably simple. For example, most news search engines only include a single keyword box and a click button. Thus, the construction of a global query form is not difficult to integrate all the forms in the same domain. In this context, the data sources can be selected based on the user inputs to the global interface (i.e., selection-per-query).

However, in the other domains, it is not practical to build such a single interface (or even a few) to encompass all the functionalities provided by the query interfaces in the same domain. For instance, in the air-ticket booking domain, we can observe that various meta-queriers provide different query interfaces with different functionalities. Most differences are caused by the selection of sources in their construction (i.e., selection-per-engine).

Our work proposes a selection-per-engine strategy in the customization of meta-queriers. Users are allowed to input their preferred data sources. To better reuse the pre-integrated data sources, we provide a capability-based source selection algorithm to recommend the users their potentially desired data sources. Complimentary to the query-based solutions, our approach is based on the query capabilities of data sources, instead of the sampled source contents. Unlike the prior work on query-form clustering, our approach is able to distinguish the query capabilities of the data sources whose query forms have been clustered in the same domain.

3 Capability-based Recommendation

Capability-based source selection is a content-based recommendation problem [3]. In our capability-based recommendation, users can declare their needs of the capabilities by inputting the domains (DM_I) and their preferred data sources (DS_I). Based on the user needs, our system recommends the users a ranked list of the pre-integrated sources (DS_O) from a repository (SR) of data sources. Such a capability-based recommendation problem can be simplified to a *utility maximization* problem. The effectiveness of such a solution relies on the correct understanding of the user demands and data sources. Section 3.1 first explains our ontology-centric model for source capabilities and user preferences, and then Section 3.2 presents the corresponding source-selection algorithms.



FC	HTML Source Codes
FC1	<label for="fh-from">From</label> <input id="fh-from" type="text" name="leavingFrom" title="Departure City or Airport Code"/>
FC2	<label for="origin"> From </label> <input class="searchbox glow" name="origin" id="origin" size="20" maxlength="80"/>
FC3	<label for="airOrigin"> From </label> <input type="text" name="origin" id="airOrigin" class="locationInput" />

E-Node	Descriptive Attributes
E1	<id: fh-from>, <text:From City name or airport>, <name: leavingFrom>, <title: Departure City or Airport Code>
E2	<name: origin>, <id: origin>, <class: searchbox glow>, <text: From>
E3	<name:origin>, <title: apcOrigin>, <class: locationInput>, <text:From>

G-Node	Instance Format
G1	a single checkbox (nearby options)
G2	a text box (destination city names)
G3	a selection menu (nearby options)

A-Node	Instance Format
A1	Destination city: text + Nearby options: checkbox
A2	Destination city: text + Nearby options: menu

Figure 1. A fragment of M-Ontology for air-ticket booking

3.1 Modeling

In the context of meta-querier customization, query capabilities refer to the abstract abilities of data sources to retrieve information. The information of these sources can be accessed through their associated query forms. Generally, the contents of query forms decide the possible queries that can be posed by users, and thus they also dictate the capabilities of the meta-queriers and data sources. Understanding the query forms is a cornerstone of the capability-based recommendation. This paper proposes an ontology-centric approach to represent their capabilities by analyzing and understanding the query forms.

Each query form can be regarded as a set of query conditions [27][11], referred to as *functional components*. For HTML forms, each component consists of a control and its associated attributes. The attributes include control type, name/label, descriptive text, instances, data domain, default value, scale/unit (e.g., kg, million, dollar), and data/value types (e.g., date type, time format, char type, etc.).

An individual component or an ordered component list can represent a specific query capability [14][6][25] that a resource possesses. However, the information carried in individual components is very limited. It does not contain the context or knowledge concerning the textual description. For example, Fig.1 shows three functional components, FC_1, FC_2 and FC_3, all of which are extracted from the real-world query forms. Each represents the departure city for booking air-tickets. However, it is difficult for machines to find the capability equivalence. Such naming conflict is very common among the query forms, which are normally created and maintained by different companies and organizations.

M-Ontology: Methodologies of ontology are commonly used to address such representation heterogeneity. Ontologies store well-defined concepts and relations including context knowledge. If query forms are properly annotated using concepts in the same ontology, machines can understand their semantics. In our previous research [15], we designed a domain-specific and light-weight ontologies (named *M-Ontology*). M-Ontology is designed for storage, management and discovery of mappings that are employed to translate query inputs among query forms. We proposed a semantics-based approach to model and organize these mappings, which can be numerous in the context of meta-querier customization due to various user needs and data sources. Each mapping corresponds to a tuple $\langle List_{FC1}, List_{FC2}, Exp\rangle$, where $List_{FC1}$ and $List_{FC2}$ are respectively two ordered lists of functional components, whose potential user inputs can be transformed through the rule Exp. In the real-world scenarios, for a specific mapping, $List_{FC1}$, $List_{FC2}$ or both might include more than a functional component. In M-Ontology, these components (as a whole) are viewed as a single concept, each of which also corresponds to a single concept. Each mapping can be regarded as an instance of a relation between two concepts. We proposed a semi-automatic solution [15] to M-Ontology construction by incremental insertion of the mappings.

As a language-independent ontology, M-Ontology is modeled by a directed acyclic graph, where nodes are the concepts and edges are the relations. In the following, we briefly introduce the nodes/edges and their correspondences with query capabilities.

(1) *E-Nodes, Is-a Edges and G-Nodes*. Each E-Node encapsulates a functional component in a specific query form. For example, as shown in Fig.1, E_1 corresponds to FC_1. Through generalization of a set of E-Nodes (e.g., E_1, E_2 and E_3) that have the same semantics and instance formats, an Is-a Edge can formulate a new concept node, called a G-Node (e.g., G_2). In essence, each G-Node corresponds to an abstract query capability. For describing the semantics of such a query capability, a representative object ro is automatically abstracted from the associated attributes of the inclusive functional components as follows: i) two bags of descriptive words $Set\langle t_i^{DA}\rangle$, $Set\langle t_i^{IST}\rangle$ are generated respectively from the descriptive attributes and instances of all human-verified E-Nodes in gn, which are normalized using NLP techniques [9] such as tokenization, stop-word removal and stemming; ii) a set of descriptive labels $Set\langle t_i^{DL}\rangle$ is determined by selecting the terms with the top-k frequency weight from $Set\langle t_i^{DA}\rangle$. The descriptive attributes and labels can be manually modified by humans. Finally, we generate the representative object ro with a tuple $\langle Set\langle t_i^{DA}\rangle, Set\langle t_i^{DL}\rangle, Set\langle t_i^{INS}\rangle\rangle$.

(2) *A-Nodes and Part-of Edges*. Each A-Node is formed through aggregating a list of G-Nodes. A Part-of Edge is used to represent such an aggregation relation by linking the A-Node (e.g., A_1) to its inclusive G-Nodes (G_1 and G_2). The

A-Node also denotes an abstract query capability, whose semantics can be approximated to a list of representative objects $List_{ro}$.

(3) *T-Edges*. A T-Edge corresponds to a transformation relation between two concept nodes (i.e., G/A-Nodes) which can be used to fetch similar contents. For example, in Fig.1, TR_1 is a T-Edge representing a transformation relation from A_1 to A_2. The transformation relation contains a specific rule to convert the instances of the connected concept nodes. Since their instances are convertible, all these connected nodes represent similar query capabilities. In a sense, M-Ontology is a domain-specific query capability repository, where each connected G/A-Node sub-graph generally corresponds to an abstracted query capability.

Capability modeling: By using the proposed M-Ontology, we model the capabilities of data sources based on their own query forms. Each data source has its own query form, whose inclusive functional components can be clustered into a set of G-Nodes in M-Ontology. Let M-Ontology includes a set of G-Nodes denoted by $GN = \{N_1, N_2, ..., N_n\}$, which are numbered from 1 to n based on their creation time. Since each G-Node denotes an abstract capability in a specific domain, we use a subset of GN (called Capability Set CS) to represent the capabilities that a data source is able to provide. That is, CS can be denoted by a G-Node set $\{N_i | N_i \in GN\}$.

To obtain such a capability set, we need to seek correct G-Nodes to annotate the functional components in the corresponding query form. More precisely, the process of capability capture can be viewed as *schema annotation*. In the context of meta-querier customization, it is straightforward to capture the capabilities of the pre-integrated data sources from their query forms. M-Ontology functions as a mapping repository, and thus the mappings linked to these query forms should have been inserted into M-Ontology. That means, all the functional components have been clustered into the corresponding G-Nodes. These G-Nodes are the components of the corresponding capability sets. The detailed algorithms are presented in the Section 3.2.

Preference modeling: The widely used preference model is based on keywords. However, in the real-world scenarios, a few keywords are often unable to represent the exact semantics. For the purpose of accurately understanding user demands, our solution relies on the mapping-generated M-Ontology. In our solution, a specific user need is modeled by a vector PV, called a *preference vector*. Given that M-Ontology contains G-Nodes $\{N_1, N_2, ..., N_n\}$, the vector PV has n corresponding entries. The i^{th} entry of vector PV, denoted by $PV(i)$, is a preference value that indicates how the user prefers this capability in the target.

3.2 Demand Capture and Matching

Following the proposed capability model and preference model, this section presents an ontology-centric algorithm for source selection. As illustrated in Fig.2, the whole procedure consists of six phases. Based on user selection (i.e., DM_I) from a list of data domains that exist in the system, *Ontology Selection* chooses an appropriate M-Ontology MO for understanding the user-preferred data sources (i.e., DS_I). Only for the data sources that have not been integrated into any meta-querier, *Q-Form Normalization* is invoked to unify their query form representation. By analyzing the normalized query forms, *Q-Form Analysis* can output the capability set CS for each data source. From the analyzed query forms, *Demand Identification* constructs a preference vector PV, while users are able to correct the preference values. *Demand Matching* generates a ranked list of resources for user selection. After user selection, if necessary, *Annotation Verification* verifies the correctness of analysis on query forms of un-integrated data sources.

The details are described as follows.

• **Ontology Selection** is to choose an appropriate M-Ontology MO. In different query forms, there might exist many functional components with the highly similar representation, but, in fact, their semantics are completely different due to the different context. For example, the word "keywords", which appears in the different domain, has various meanings. In job seeking, it represents job titles or fields, but it also might denote the song name in the music search. Thus, our designed M-Ontology is domain-specific. This assumption is reasonable for the customization of meta-queriers, which combine the data sources in the same domain.

• **Query-form Normalization** is to unify the representation of the query form for each data source in DS_I that is not pre-integrated into our system. The data sources can be categorized as two types, pre-integrated and new data sources. Since the pre-integrated data sources have been analyzed, the results can be directly reused without repetitive normalization. This phase only processes the query forms $qform_I$ that are not included in M-Ontology. For each query form, its inclusive functional components are associated with some descriptive attributes and a set of potential instances (if they exist). We first perform natural language processing techniques on the descriptive attributes and instances in the following orders: tokenization, stop-word removal and stemming. Then, we treat the normalized texts as two unordered bags of words, $Set\langle w_i^{DA}\rangle$ and $Set\langle w_i^{INS}\rangle$. These two bags of words constitute a word-bag pair, which can indicate the semantic characteristics of this component. Fi-

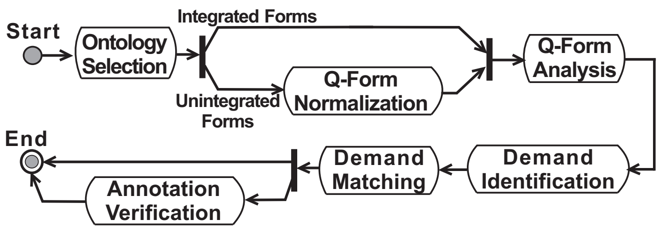


Figure 2. The ontology-centric algorithm for source selection.

nally, each query form corresponds to a set of word-bag pairs, $Set\langle(Set\langle w_i^{DA}\rangle, Set\langle w_i^{INS}\rangle)\rangle$

• **Query-form Analysis** is to understand the query forms by analyzing the semantics of each inclusive functional component fc. Our solution is to annotate each fc by semantics-equivalent concepts (i.e., G-Nodes from the M-Ontology MO). The involved G-Nodes constitute the capability set CS of the corresponding data source. G-Node searching can be divided into two separate types: a) For the pre-integrated data sources, their query forms have been included in the M-Ontology, and thus each functional component of these forms should have been clustered to a certain G-Node. That is, such an association can be directly reused. b) For the new data sources, each functional component in their query forms is normalized to two word bags $Set\langle w_i^{DA}\rangle$ and $Set\langle w_i^{INS}\rangle$. The suitability of a G-Node gn can be measured by the constraint equivalence and semantic similarity between fc and the corresponding representative object ro of gn. The semantic similarity can be calculated as follows.

$$\lambda_1 \sum_{i=0}^{n_1}\sum_{j=0}^{m_1} sim(w_i^{DA}, t_j^{DA}) + \lambda_2 \sum_{i=0}^{n_1}\sum_{j=0}^{m_2} sim(w_i^{DA}, t_j^{DL}) + \lambda_3 \sum_{i=0}^{n_3}\sum_{j=0}^{m_3} sim(w_i^{INS}, t_j^{INS})$$

where, λ_i is a scale factor, two word sets $Set\langle w_i^{DA}\rangle$ and $Set\langle w_i^{INS}\rangle$ are the semantic characteristics of fc, and a tuple $\langle Set\langle t_i^{DA}\rangle, Set\langle t_i^{DL}\rangle, Set\langle t_i^{INS}\rangle\rangle$ is the representative object ro. The function sim is to determine the semantic similarity between two terms (respectively from fc and ro). Our implementation relies on the WordNet-synonyms distance, a linguistic-based matcher.

• **Demand Identification** constructs a preference vector PV based on the selected data sources and user interaction. The whole procedure is composed of two steps:

1) *Automatic discovery*: Each G-Node corresponds to an entry $PV(i)$ whose value indicates the preference degree against a specific capability. In M-Ontology, G-Nodes that are connected via a single or multiple T-Edges and Part-of Edges whose directions are ignored constitute a maximal connected G-Node sub-graph. Such a sub-graph corresponds to an abstracted query capability. In the sub-graph, the preference values of these G-Nodes are correlated. Since the conversion through T-Edges and Part-of Edges might lose the semantics, the distance between two G-Nodes i and j indicates their dissimilarity. Here, the distance refers to the minimum hop number from one node to another. The value $1/(1 + distance(i, j))$ is used to represent the potential difference between their capabilities. Assuming that $GSet_M$ is a multiset that contains the G-Nodes encapsulating the functional components of data source in DS_I. It might contain duplicates. The occurrence number of a G-Node in $GSet_M$ is equal to the appearance frequencies of its encapsulated functional components in DS_I. Preference vectors can be decided by two modes: combination and accumulation.

- The *combination mode* is preferred when users want to find the data sources containing all the capabilities (with the same desirability) that are supported by the user-inputted sources DS_I. The values can be obtained through the following procedure: First, all the entries whose G-Nodes are in $GSet_M$ are set to one, i.e., $PV(i)$ = 1 if $i \in GSet_M$; otherwise, the values are initialized to zero. Second, for the G-Nodes that are not in $GSet_M$, their values are calculated based on the distance to the nearest node in $GSet_M$. The procedure can be represented as eq. (1).

$$PV(i) = \begin{cases} 1 & i \in GSet_M \\ \frac{1}{\min\limits_{j \in GSet_M}\{distance(i,j)+1\}} & i \notin GSet_M \end{cases} \quad (1)$$

- The *accumulation mode* assumes the most critical capabilities that user desired are the ones that appear most frequently in the user-inputted sources. For the entries whose G-Nodes are in $GSet_M$, their values of $PV(i)$ are equal to the multiplicity (i.e., occurrence number) of their corresponding G-Nodes in $GSet_M$. For the other entries, their values are accumulated based on the distances to the nodes in $GSet_M$ and their preference values, as shown in eq. (2).

$$PV(i) = \begin{cases} multiplicity(i) & i \in GSet_M \\ \sum\limits_{j \in GSet_M} \frac{PV(j)}{distance(i,j)+1} & i \notin GSet_M \end{cases} \quad (2)$$

2) *Manual correction* (optional): Automatic decision of preference values might not accurately demonstrate the user demands, optional manual correction is necessary to correct the values by the users themselves. However, it often becomes inappropriate or impractical to present the whole preference vector, especially when the vector is very long. In our design, the users are able to modify the values that are not equal to zero. This feature-oriented mechanism is complementary to the initial user inputs (including the preferred sources DS_I and domain DM_I).

• **Demand Matching**: This phase is to generate a ranked list of data sources that best match the identified demands. The ranking of the list is through comparing the numeric values of the utility function u that demonstrates the desirability of a data source for a specific user need. More exactly, in the application domain DM_I, a resource $R1 \in DM_I$ is preferred to a resource $R2 \in DM_I$ if and only if the expected utility of $R1$ is greater than the expected utility of $R2$: $\forall R1, \forall R2, R1 \succsim R2 \Leftrightarrow u(R1) \geq u(R2)$. Such a rational preference relation $\succsim$ is transitive, reflexive and complete.

The preference ranking is a typical multi-criteria decision making problem. In meta-querier customization, each criterion corresponds to a query capability identified in preference vectors (i.e., a G-Node). To make the ranking outcomes manageable by users, we assume *additive independence* exists among the maximal connected subgraphs, which is a normal assumption [12]. The utility of a resource R can be approximated by using an *additive value function* that breaks one n-criteria function into n individual one-criterion functions. Such an approximation not only simplifies the automatic adjustment and manual correction, but also performs well, even if the assumption does not strictly hold [18]. We construct an additive utility function u to aggregate the utility $cu(R[N_i])$ of each individual capability N_i provided by the resource R. The utility $cu(R[N_i])$ is 1 when the capability set CS of R contains N_i; otherwise it is zero. The additive weight of N_i is decided by its preference value $PV(i)$, which are generated from user inputs. For the nodes in any maximal connected subgraph (sg), the sum of their utility values should be less than or equal to δ. δ is 1, if the combination mode is used. δ is set equal to $\sum_{N_i \in V(sg) \cap GSet_M} PV(i)$, if the accumulation mode is used. Let *MCSG* be a set of maximal connected subgraphs, the weighted utility function can be represented as follows,

$$u(R[N_1, N_2, ..., N_n]) = \sum_{i=0}^{n}(PV(i) \times cu(R[N_i])) \qquad (3)$$

subject to the following constraint,

$$\forall sg \in MCSG, \sum_{N_i \in V(sg)} (PV(i) \times cu(R[N_i])) \leq \delta$$

• **Annotation Verification** (after run-time): This phase is to verify whether the new data sources are annotated by the accurate G-Nodes. For those functional components that cannot match with any concept node, the manual annotation is invoked to update and maintain M-Ontology (e.g., by inserting new mappings among the related query forms). As an optional phase, the manual verification can be conducted after the source selection. Although the new data sources might not be supported immediately, the verified annotation can be reused for the future recommendation.

4 Experiments

A prototype of M-Ontology and the related algorithms has been implemented on an open-source mapping management system, Alignment Server[8]. This system provides some basic functionalities on mapping management and discovery.

To show the effectiveness of our approach in real-world scenarios, we design and conduct two sets of experiments in the domain of air-ticket booking. Since the quality of ranking is subjective, it is hard to measure its correctness. Given that our primary goal is to find suitable data sources from a source repository SR to satisfy user demands (their preferred data sources DS_I) on capabilities, the focus of our experiments is to evaluate whether our approach can correctly identify capability matches between user inputs DS_I and the data sources in SR. Specifically, the experiments are designed to evaluate the effectiveness of capability matching, which is the most critical factor that affect the recommendation performance.

Experiment setup: From the UIUC web integration repository[2], we collect 38 query forms for air-ticket booking after eliminating the inactive webpages. First, we manually extract all the query forms from the webpages. They are expressed in Web Ontology Language (OWL) and follow a query-form ontology that was designed based on the HTML specification [1]. Second, we manually classify the functional components of these forms to generate 54 maximal connected G/A-Node sub-graphs, based on their capabilities. Each functional component is associated with a G-Node and a G/A-Subgraph. The manual classification is utilized in the initial construction of M-Ontology and the final evaluation of our algorithm.

Experiment scenarios: Assume that users input three preferred data sources DS_I, n of which are not in the repository SR. Our experiments will examine how well the algorithm can correctly find the data sources with the desired capabilities (that are possessed by the sources in DS_I) from SR.

The first experiment is for our proposed solution (referred to as *MOM*). Except these n non-inclusive data sources in DS_I, all the remaining sources are used to construct a domain-based M-Ontology (for construction details, see our prior work [15]). We assume users can correctly choose an appropriate domain DM_I for their queries. That is, an appropriate M-Ontology *MO* is chosen. With assistance of *MO*, the query forms in DS_I are normalized and analyzed to identify the user demands.

The second experiment evaluates the performance of a reference solution that is a classical nearest neighbor method (referred to as *NNM*). To find the capability correspondences, it compares the query forms in DS_I with the form of each source in SR. For the performance comparison, we use the same algorithms of query-form normalization and similarity calculation in both *NNM* and *MOM*.

To evaluate the performance, we use the following three measures: *precision* measures the proportion of the identified capabilities that are actually desired by users; *recall* measures the proportion of the desired and identified capabilities out of all the desired and identifiable capabilities; *f-measure* is the weighted harmonic mean of precision and recall.

Experiment results: The performance values per n in the Table 1 and 2 are calculated by the average of 100 samples. All the samples are randomly generated. The first and second set of experiments share the same samples.

Table 1. Capability-based matching by MOM

Unintegrated sources (n)	precision	recall	f-measure
3 of 3	90.1%	86.6%	88.3%
2 of 3	92.8%	90.6%	91.7%
1 of 3	96.5%	95.8%	96.1%

The first set of experiment results in Table 1 show promising evidence of effectiveness of *MOM*. Even if all the data sources in user inputs (DS_I) have not been integrated by any existing meta-querier, the f-measure rate also reaches 88%. If only one data source is unintegrated, the capabilities of almost all the data sources can be correctly and completely identified. That indicates a high possibility that our solution can make an accurate recommendation.

Table 2. Capability-based matching by NNM

Unintegrated sources (n)	precision	recall	f-measure
3 of 3	76.0%	35.4%	48.3%
2 of 3	83.9%	56.8%	67.7%
1 of 3	92.0%	78.5%	84.7%

The second set of experiment results are shown in Table 2. Clearly, our method *MOM* outperforms the reference method *NNM* by a large factor, especially when the user-preferred sources have not been integrated. The major reason is the name ambiguity in HTML codes so that it is difficult to find the capability similarity between two individual data sources. Different from *NNM*, our method *MOM* has a better performance by utilizing some regular patterns that are learned from the integrated data sources.

5 Conclusions and Future Work

Meta-querier customization is desired to meet various user needs. In our customization strategy, we design a capability-based solution to source selection based on user inputs. The core is a light-weight ontology, which is generated from a number of existing mappings among query forms. It is viewed as a repository of capabilities. The user demands and source capabilities are modeled, identified and matched based on this ontology. Initial experiments show the potential of this ontology-centric method. Interesting directions for future work include:

1) To reduce the user interaction, ontology/domain selection should be automated by calculating the similarity between user inputs and ontology contents.

2) To implement a practical source selection, the other properties (e.g., popularity, stability and credibility) should be included in the preference model.

3) To support the implicit user preference, the source selection algorithm should be integrated with the classical recommendation algorithms [3][23], e.g., content and collaborative filtering algorithms.

References

[1] HTML 4.01 specification: Forms. http://www.w3.org/TR/html4/interact/forms.html, 1999.

[2] The UIUC Web integration repository. http://metaquerier.cs.uiuc.edu/repository, 2003.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[4] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. Uncovering the relational Web. In *WebDB*, 2008.

[5] J. P. Callan. Document filtering with inference networks. In *SIGIR*, pages 262–269, 1996.

[6] K. C.-C. Chang, H. Garcia-Molina, and A. Paepcke. Boolean query mapping across heterogeneous information sources. *IEEE Trans. Knowl. Data Eng.*, 8(4):515–521, 1996.

[7] K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the Web: Observations and implications. *SIGMOD Record*, 33(3):61–70, 2004.

[8] J. Euzenat. An api for ontology alignment. In *ISWC*, pages 698–712, 2004.

[9] D. A. Grossman and O. Frieder. *Information Retrieval: Algorithms and Heuristics*. The Kluwer International Series of Information Retrieval. Springer, second edition, 2004.

[10] B. He, T. Tao, and K. C.-C. Chang. Organizing structured web sources by query schemas: a clustering approach. In *CIKM*, pages 22–31, 2004.

[11] J. Hong, Z. He, and D. A. Bell. Extracting Web query interfaces based on form structures and semantic similarity. In *ICDE*, pages 1259–1262, 2009.

[12] R. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[13] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[14] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. *VLDB*, pages 251–262, 1996.

[15] X. Li and R. Chow. An ontology-based mapping repository for meta-querier customization. In *SEKE*, pages 325–330, 2010.

[16] Y. Lu, H. He, Q. Peng, W. Meng, and C. T. Yu. Clustering e-commerce search engines based on their search interface pages using wise-cluster. *Data Knowl. Eng.*, 59(2):231–246, 2006.

[17] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can only afford to pay as you go. In *CIDR*, pages 342–350, 2007.

[18] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, chapter 16.4, pages 622–626. Prentice Hall, 3 edition, 2009.

[19] M. Shokouhi. Central-rank-based collection selection in uncooperative distributed information retrieval. In *ECIR*, pages 160–172, 2007.

[20] L. Si and J. P. Callan. Relevant document distribution estimation method for resource selection. In *SIGIR*, pages 298–305, 2003.

[21] W. Su, J. Wang, and F. H. Lochovsky. Automatic hierarchical classification of structured deep web databases. In *WISE*, pages 210–221, 2006.

[22] P. Thomas and M. Shokouhi. Sushi: scoring scaled samples for server selection. In *SIGIR*, pages 419–426, 2009.

[23] G. Uchyigit and M. Y. Ma, editors. *Personalization Techniques and Recommender Systems*, volume 70. World Scientific Publishing, April 2008.

[24] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *SIGIR*, pages 254–261, 1999.

[25] R. Yerneni, C. Li, H. Garcia-Molina, and J. D. Ullman. Computing capabilities of mediators. In *SIGMOD Conference*, pages 443–454, 1999.

[26] B. Yuwono and D. L. Lee. Server ranking for distributed text retrieval systems on the internet. In *DASFAA*, pages 41–50, 1997.

[27] Z. Zhang, B. He, and K. C.-C. Chang. Understanding Web query interfaces: Best-effort parsing with hidden syntax. In *SIGMOD Conference*, pages 107–118, 2004.

[28] P. Ziegler, K. R. Dittrich, and E. Hunt. A call for personal semantic data integration. In *ICDE Workshops*, pages 250–253, 2008.