COVER SHEET

Cooperative Strategy for Web Data Mining and Cleaning

Yuefeng Li[1], Chengqi Zhang[2], and Shichao Zhang[2]

[1]School of Software Engineering and Data Communications
Queensland University of Technology, Brisbane QLD 4001 Australia

[2]Faculty of Information Technology, University of Technology, Sydney
PO BOX 123, Broadway Sydney NSW 2007 Australia

y2.li@qut.edu.au; {chengqi, zhangsc}@it.uts.edu.au

Abstract. While the Internet and World Wide Web have put a huge volume of low-quality information at the easy access of an information gathering system, filtering out irrelevant information has become a big challenge. In this paper, a Web data mining and cleaning strategy for information gathering is proposed. A data mining model is firstly presented for the data that come from multiple agents. Using the model, a data cleaning algorithm is then presented to eliminate irrelevant data. To evaluate the data cleaning strategy, an interpretation is given for the mining model according to evidence theory. An experiment is also conducted to evaluate the strategy using Web data. The experimental results have shown that the proposed strategy is efficient and promising.

Keywords: Web mining and cleaning, Knowledge discovery, Information gathering

1 Introduction

Modern life depends heavily on certain essential networks, like investments, marketing, commerce, stock jobbing, communication, information gathering, television, and telephone. The Internet is well on its way to becoming the next network staple of modern life, and for good reason. Accordingly, huge volumes of data have being put on the Internet day by day. There also has been a dramatic growth in the number of publicly accessible data sets (documents) on the Internet, and all indications suggest that this growth will continue in the years to come. Information search engines, such as "Google", "Yahoo", "Alta Vista" and "Excite" offer efficient and low-cost ways of collecting relevant information. This has opened the opportunity for users to benefit from the available information. For example, a researcher can benefit from the Internet to gather, analyze, distribute, and share lately publications relevant to his/her interesting areas.

Usually, users retrieve Web data by keyword searching, which is an intuitive form of accessing data on the Web. However, these search strategies present an important challenge: keyword searching return many thousands, even millions of results in response to a user query. A great much information returned is irrelevant to the user query.

To address this challenge, the notion of information gathering (IG) has been proposed in [12] [13] [14] [16] [17]. One of the fundamental issues regarding the efficiency of information gathering is "overload" that means a lot of retrieved documents (or URLs) are not what users need. We hope to eliminate most of "dirty data", the irrelevant information, from the retrieved documents.

The main reason of arising much "dirty data" for IG systems is that much irrelevant information is extracted when responding a user query. For example, we may use a pattern (e.g., a set of keywords, or a weight vector) to represent a document. When an IG systems use this sort of representation to retrieve documents, there are many irrelevant text documents that can match the pattern. The IG systems cannot distinguish the relevant and irrelevant information in a cluster (a class) generated by a single representation.

One solution for this problem is information filtering which uses a big training set to divide documents into many rather smaller classes [24] [20]. In this paper, we present another solution. We use multiple representations of documents to find rather smaller clusters. The new solution is reasonable if the IG systems work in multiple agent environments, where documents are distributed in many collections (databases), and each agent uses several collections.

We will fuse the retrieved information from the different resources. This problem is described as the "query optimization", or referred to "collection fusion" in database systems community (see [10] [29]). To address this problem, we will give a new perspective in this paper. We first decide what sort of knowledge can be discovered from retrieved documents, then a cooperative approach is presented to resort the retrieved documents.

This paper is organized as follows. We first illustrate our motivation for Web-based information gathering systems in Section 2. In Section 3, we discuss the representation of retrieved documents. In Section 4, a model for knowledge discovery is presented. Using the model, we also give a cooperative algorithm to eliminate irrelevant data from multiple resources. In Section 5, to judge the results, we also present an interpretation for our approach according to evidence theory. The performance is also made by using the traditional methods precision and recall. Lastly in Section 6 we discuss our contributions and the further work.

2 Web-Based Information Gathering

Information-gathering systems might help humans to gather the right information for their needs from the Web. The idea that we trust an information gathering system is probably the first to occur to us when we use it to gather information from the Web. We may ask some questions of the designer; such as does the system understand our requests? Can we trust the results it provides? In other words, does the information gathering system use the correct knowledge?

Due to historical reasons, documents in a collection are frequently represented through a set of index terms or keywords. Such keywords might be extracted directly from the text of the document or might be specified by a human subject

(as frequently done in IR arena) [1]. These "representative" keywords (or term sets) provide a logical view or representation of the document, whether they are derived automatically or generated by a specialist [16].

The classic models in IR or information filtering (IF) consider that each document is described by a vector of terms, where terms are called representative keywords. The semantics of a vector helps to remember the document's main themes. If we consider that not all terms are equally useful for describing the document contents, we can assign numerical weights to each term to distinct terms. For this consideration, each document could be described by a vector of numerical weights of terms.

Let t_k be a term, d_i be a document, and $w_{ik} \geq 0$ be a weight associated with the pair (d_i, t_k). This weight quantifies the importance of the terms for describing the document semantic contents. Term weights can be calculated in many different ways. One of the most effective term-weight techniques is called $tf * idf$ (term frequency times inverse document frequency) technique (see Section 2.1.2). To apply this technique, a table is generated off-line containing total frequencies of all terms in a thesaurus, using a sufficiently representative collection of documents as a training set. In the on-line document stream, another table is generated containing the frequencies of all unique terms found in newly arrived documents. Based on the values in the two tables, the following equation is used to derive appropriate weights for terms in each document:

$$w_{ik} \;=\; t_{ik} \times log(N/n_k)$$

where t_{ik} is the number of occurrences of term t_k in document d_i; $log(N/n_k)$ is the inverse document frequency of term t_k in the training set; N is the total number of documents in the training set; and n_k is the number of documents in the training set that contain the given term t_k.

To easy see our motivation, we assume $log(N/n_k) = 1$ for the following example, where the key words are "Java", "program", and "comput", and the 6 retrieved documents are:

d_1 : "Java beans for real programmer."
d_2 : "Java for C C++ programmer."
d_3 : "Java networking."
d_4 : "Java and computing programming."
d_5 : "Computing programming for computer science."
d_6 : "Java architecture Indonesia pictorial works."

In this example, $\mathcal{D} = \{d_1, \ldots, d_6\}$. If $w_{ik} = t_{ik}$ then these documents could be represented as vectors in Table 3.1.

With closer inspection of the table, we can see that the vector representations $d_1 = d_2$, and thus we have the set $E_1 = \{d_1, d_2\}$ which is called an elementary set (or a class). Similar elementary sets can be derived from the table, and exist as follows:

$$E_2 = \{d_3, d_6\},\ E_3 = \{d_4\},\ E_4 = \{d_5\}$$

Table 1. Example: Document Space

$\mathcal{D}$	$w_{i,Java}$	$w_{i,program}$	$w_{i,comput}$
d_1	1	1	0
d_2	1	1	0
d_3	1	0	0
d_4	1	1	1
d_5	0	1	2
d_6	1	0	0

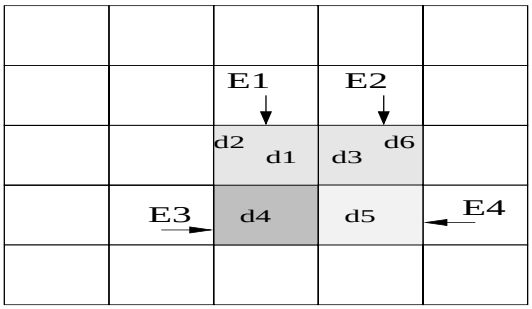

Fig. 1. Large elementary sets

It is easier to answer which of the above documents are relevant by the user. In this example, we assume the relevant documents are d_1, d_3, and d_4.

Figure 1 gives us a graphical view of the document space. The shaded 4 squares are the above elementary sets. The other squares are the potential elementary sets if we consider more documents. From this example, we can understand that the documents are divided into some equivalence classes using one representation of documents. The problem is that some classes include both relevant documents and irrelevant documents (e.g., E_1 and E_2), and the system has not further knowledge to distinguish the relevant and irrelevant documents in an elementary set.

We argue there are two ways to solve this problem. The first one is using a large training set, and the second one is using multiple representations of documents. The former methods can be used in the case there are a lot of feedback from users, and the latter method can be used when the IG system works in a multiple agent environment.

Figure 2 shows the rather smaller elementary sets if we use multiple representations of documents to classify the document space (e.g., E_1 is divided into 2 classes and E_2 is also divided into 2 classes). It is possible to distinguish the relevant documents and irrelevant documents in a big elementary set if we use multiple representations of documents because even using the same IR model (e.g., the $tf * idf$ technique) the representations of documents may be different

if using different training sets.

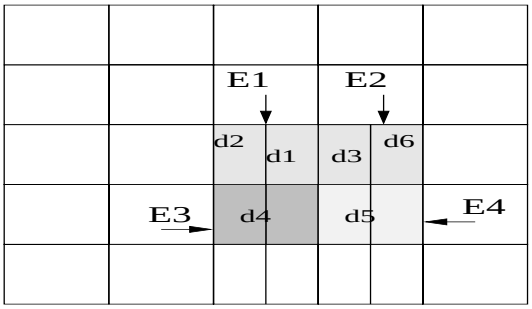

Fig. 2. Small elementary sets on a document space

3 Representation of Web Data

A difficult problem arises when we use multiple representations of documents for IG because collections (databases) may be used by more than one agent. Figure 3 shows an example for this case. The initial example is designed in [10]. In the example shown in Figure 3, databases D_1, D_2, and D_3 are served by agent B, databases D_2, D_3, and D_4 are served by agent C. We assume that the return documents retrieved by agent B belong to D_1, D_2, and D_3; and the return documents retrieved by C belong to D_2, and D_4. The problem for the subject agent A is to evaluate all of the return documents (sometimes called retrieved documents).

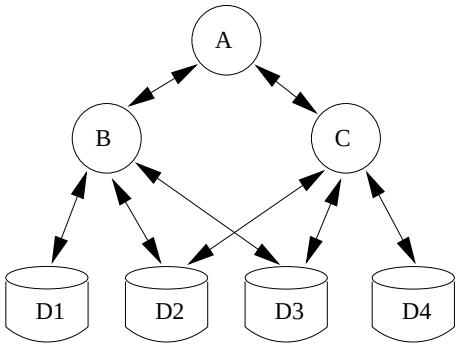

Fig. 3. The Problem for Collection Fusion

A clue for solving this problem is given by Fuhr [10]. The subject agent A can ask agent B to revise its retrieved documents in case D_2 as well as D_3

are ignored. After that the problem will become the case where different agents access disjoint sets of databases. To use this approach for Web-based IG systems, we have to give a negotiation method to select observer agents which are willing to revise their retrieved documents. The process of negotiation will take more time. Another drawback is that this approach cannot use multiple opinions for the same question (one kind of cooperation in multi-agent environments [7]).

We can use a list of facts to illustrate what the subject agent has obtained, where each fact consists of attributes. Table 2 illustrates such facts for the problem shown in Figure 3. We use three levels (H, M, L) to describe our beliefs about the relevance if we assume that the two agents have the same performance. Here, the "H" means that two agents both admit the document relevant (the second row); the "M" means that one agent admits the document relevant, and the other has no idea because it does not use the database (the first row and the last row); and the "L" means that one agent admits the document relevant, but the other denies it.

Table 2. Web Data Example

Doc	*Agent B*		*Agent C*		*Relevant Level*
	Served	Retrieved	Served	Retrieved	
$d_1 \in D_1$	Yes	$\surd$	No	$\times$	M
$d_2 \in D_2$	Yes	$\surd$	Yes	$\surd$	H
$d_3 \in D_2$	Yes	$\surd$	Yes	$\times$	L
$d_4 \in D_2$	Yes	$\times$	Yes	$\surd$	L
$d_5 \in D_3$	Yes	$\surd$	Yes	$\times$	L
$d_6 \in D_4$	No	$\times$	Yes	$\surd$	M

Considering many agents (rather than 3) which may have different performance in a Web-based IG system, the problem looks so complex, and the evaluation for this problem is not easy. The following section, however, will provide a cooperative approach for this problem.

4 The Cooperative Approach

It is difficult to compare the similarity values (the weights generated by IR models), because agents may use different term index vocabularies. For this consideration, the subject agent asks each agent not only returns the similarity values (weights) for the return documents, but also makes a binary decision as to whether the documents should be relevant. This requirement is similar with the requirement for the task of filtering within TREC (the Text REtrieval Conference, see http://trec.nist.gov/cfp.html). In this section we first show what kind of knowledge can be discovered, then give an algorithm for the data cleaning.

4.1 Web Data Mining

In the following, we assume a basic setting as follows. A subject agent submits a query to each agent of Θ. In response, some agents of Θ may produce results for the query. We assume the results include collection names that they have served, and pairs of retrieved documents and collection names, where the first component in a pair is a document, and the second component in the pair is a collection name that the document belongs to. We use Θ_t to denote the set of agents which have returned a result at time t. Let $\mathcal{D}_t = \{d_{t_1}, d_{t_2}, \ldots, d_{t_n}\}$ be the set of all retrieved documents at time t, and $\mathcal{C}^t = \{\mathcal{C}_1^t, \mathcal{C}_2^t, \ldots, \mathcal{C}_m^t\}$ be the set of all collections mentioned by agents at time t.

Based on the above assumption, we can get a partition, a family $\mathcal{D}_{t_i}$ $(i = 1, \ldots, m)$, of $\mathcal{D}_t$ such that

- $\mathcal{D}_t$ is the union of the sets $\mathcal{D}_{t_i}$ $(i = 1, \ldots, m)$,
- each pair $\mathcal{D}_{t_i}$, $\mathcal{D}_{t_j}$ $(i \neq j)$ is disjoint.
- documents in the same part $\mathcal{D}_{t_i}$ are all come from the same collection $\mathcal{C}_i^t$.

For example, in Figure 3 and Table 2, if agent B and C both return the results at time t, then we have:

$$\Theta_t = \{B, C\}$$
$$\mathcal{D}_t = \{d_1, d_2, \ldots, d_6\}$$
$$\mathcal{C}^t = \{D_1, D_2, D_3, D_4\}$$

So we can get a partition of $\mathcal{D}_t$, and the parts in the family are $\{d_1\}$, $\{d_2, d_3, d_4\}$, $\{d_5\}$, and $\{d_6\}$.

The knowledge we believe for each document is based on the possible degree of support from other agents. At time t, we can get a mapping

$$\Gamma_t : \ \Theta_t \rightarrow 2^{\mathcal{D}_t}$$

such that $\Gamma_t(\theta)$ is the set of retrieved documents provided by agent θ for each $\theta \in \Theta_t$, where $2^{\mathcal{D}_t}$ is the power set of $\mathcal{D}_t$. For example, considering Table 2, we have

$$\Gamma_t(B) = \{d_1, d_2, d_3, d_5\}$$
$$\Gamma_t(C) = \{d_2, d_4, d_6\}$$

A rule from agent B perspective is that " it believes that every document in the set $\Gamma_t(B)$ is relevant". The subject agent, however, knows only some documents in the set are the real relevant documents because of the problem of *overload*. At this situation, the subject agent might guess (analyze) the possible degree of relevance to each retrieved document based on its trustworthiness to other agents. The trustworthiness to an agent is based on its previous achievement. We use a precision function to represent an agent's previous achievement. For each $B \in \Theta_t$, its precision function $Precision_B$ is the fraction of the retrieved documents which is relevant. For example, if the set of retrieved documents provided by B is $\Gamma_t(B)$, and R_B is the subset of relevant documents appeared in $\Gamma_t(B)$, then we have

$$Precision_B = \frac{\mid R_B \mid}{\mid \Gamma_t(B) \mid}$$

For example, if we assume that $Precision_B(4) = 0.55$, and $Precision_C(3) = 0.60$, the subject agent A could believe that each document in $\Gamma_t(B)$ can get support degree 0.55 from B, and each document in $\Gamma_t(C)$ can get support degree 0.60 from C.

Obviously the documents in database D_3 cannot get any degree of support from C, because agent C believes they are all irrelevant. For the documents in database D_1, however, the subject agent cannot simply assure "they cannot get any degree of support from C", because D_1 is not served by C. At this case, the subject agent should consider the potential support from C.

According to the above analysis, we can use a random set (Pr, ξ) to represent what we have found from these facts, where Pr is a probability on Θ_t, and ξ is a mapping such that

$$\xi : \Theta_t \to 2^{\mathcal{C}_t \times 2^{\mathcal{D}_t}}$$

For example, from Table 2 we have

$$\xi(B) = \{(D_1, \{d_1\}), (D_2, \{d_2, d_3\}), (D_3, \{d_5\})\}$$

and

$$\xi(C) = \{(D_2, \{d_2, d_4\}), (D_3, \emptyset), (D_4, \{d_6\})\}$$

The subject agent can obtains the following decision rules:

- if agent B obtains a set $\Gamma_t(B)$ at time t then d_1 in D_1 and d_2 and d_3 in D_2 and d_5 in D_3 get $Pr(B)$ support degree from B;
- if agent C obtains a set $\Gamma_t(C)$ at time t then d_2 and d_4 in D_1 and d_6 in D_4 get $Pr(C)$ support degree from C, and documents in D_3 get zero support degree from C.

In Section 5, we will interpret the random sets and show how it is used in the data cleaning algorithm.

4.2 Data Cleaning Algorithm

The algorithm in Table 3 describes the details for data cleaning. It evaluates the support degree and the potential support degree, and resort the retrieved documents.

In this algorithm, the inputs are the set of cooperating agents at time t (Θ_t), the set of retrieved documents at time t ($\mathcal{D}_t = \bigcup_{i=1,\ldots,m} \mathcal{D}_{t_i}$, $\mid \mathcal{D}_t \mid = n$), and the associated collections $\mathcal{C}_1^t, \mathcal{C}_2^t, \ldots, \mathcal{C}_m^t$. We expect the outputs are the relevant degree function R, and the sequence of the retrieved documents.

In this algorithm, a multi-key in step 4 for a document d consists of two keys. The first key is $R(d)$, the second key is $\sum_{\theta \in \Theta_t} weight_\theta(d)$, where $weight_\theta(d)$ is the normalization of d's weight that agent θ provides, it can be computed as follows

$$weight_\theta(d_i) = \frac{1}{\sum_{j=1,\ldots,|\Gamma_t(\theta)|} w_j} w_i,$$

Table 3. The Cleaning Algorithm

1. for $i = 1$ to n //*initial*
   $R(d_i) = 0$;
2. for each $\theta \in \Theta_t$ //*support evaluation*
   for each $d \in \Gamma_t(\theta)$
   $R(d) = R(d) + Precision_\theta(\mid \Gamma_t(\theta) \mid)$;
3. for $j = 1$ to m //*potential support*
   for each $\theta \in \Theta_t$
   if ($\mathcal{C}_j^t$ is not served by θ)
   for each $d \in \mathcal{D}_{t_j}$
   $R(d) = R(d) + Precision_\theta(\mid \mathcal{D}_t \mid)$;
4. Resort $\mathcal{D}_t$ based on multi-keys;

if agent θ provides a set of retrieved documents $\{d_1, \ldots, d_{|\Gamma_t(\theta)|}\}$ with the corresponding weights $\{w_1, \ldots, w_{|\Gamma_t(\theta)|}\}$. The documents will be divided into some classes firstly based on the first key, in which every document has the same R. In each class the documents are then ordered by the second key.

For example, by using the example showed in Figure 3, we have the input data $\Theta_t = \{B, C\}$, $\mathcal{D}_t = \{d_1, d_2, \ldots, d_6\}$, $\mathcal{D}_{t_1} = \{d_1\}$, $\mathcal{D}_{t_2} = \{d_2, d_3, d_4\}$, $\mathcal{D}_{t_3} = \{d_5\}$, $\mathcal{D}_{t_4} = \{d_6\}$, $\mathcal{C}_1^t = D_1$, $\mathcal{C}_2^t = D_2$, $\mathcal{C}_3^t = D_3$, and $\mathcal{C}_4^t = D_4$. In step 1, the algorithm initializes the relevant degree function R as 0. In step 2, the algorithm gives the support degree to each document based on the $Precision$ functions. In step 3 the algorithm considers the potential support degrees for some documents that some agents do not serve.

Table 4 shows an example for the outputs of the first three steps according to the above inputs, where we assume $Precision_B(4) = 0.55$, and $Precision_C(3) = 0.60$, In this table, the second column is the support degree for individual document, and the third column is the total relevant degree (including both the support degree and the potential support degree) for individual document.

Table 4. Example for the Evaluation Algorithm

Step	$R(1)$	$R(2)$	$R(3)$	$R(4)$	$R(5)$	$R(6)$
1	0	0	0	0	0	0
2	0.55	1.15	0.55	0.60	0.55	0.60
3	0.95	1.15	0.55	0.60	0.55	1.00

If the precision functions are decrease functions (typically a precision curve is

a decrease function [11]), the above algorithm would reveal the following general rules for the classification of a retrieved document:

- The possible degree of relevance is high if more than one agent retrieves it.
- The possible degree of relevance is low if some agents do not retrieve it
- The possible degree of relevance is medium if some agents retrieve it but some agents do not use the collection which contains it.

5 Algorithm Analysis

It is obvious that the cleaning algorithm has a polynomial time complexity. In the following we will interpret the meaning of the random sets presented in Section 4.1. We also give a real example to show the performance of this cooperative cleaning algorithm.

5.1 Interpretations of Random Sets

From Table 2, we might classify the documents into three classes:

- Class $\{d_2\}$, two agents both admit document d_2 relevant.
- Class $\{d_1, d_6\}$, one agent admits documents relevant, and the other has no idea because it does not use the databases that the documents belong to.
- Class $\{d_3, d_4, d_5\}$, one agent admits documents relevant, but the other denies them.

Based on the random set (Pr, ξ), agent B will provide a set of retrieved documents (e.g., $\{d_1, d_2, d_3, d_5\}$), and the names of databases that it does not serve (e.g., D_4). The set of retrieved documents (represented by map Γ_t) means that we know there are some relevant documents in the set, but we cannot be sure which documents are the relevant documents. These information can be represented as a pair $< Pr, \sigma >$, such that Pr is the probability function on Θ_t, and σ from $2^{\mathcal{D}_t}$ to 2^{Θ_t} is a basic set assignment [30] [18].

We can use the function of $Precision$ to describe Pr. For example, we may have the following probability function

$$\begin{aligned} Pr(\{B\}) &= \frac{0.55\times|\Gamma_t(B)|}{0.55\times|\Gamma_t(B)|+0.6\times|\Gamma_t(C)|} \\ &= \frac{0.55\times 4}{0.55\times 4+0.6\times 3} = 0.55 \\ Pr(\{C\}) &= \frac{0.6\times|\Gamma_t(C)|}{0.55\times|\Gamma_t(B)|+0.6\times|\Gamma_t(C)|} \\ &= \frac{0.6\times 3}{0.55\times 4+0.6\times 3} = 0.45. \end{aligned}$$

We also get a basic set assignment, which satisfies

$$\sigma(S) = \{\theta \in \Theta_t \mid \Gamma_t(\theta) = S\} \tag{1}$$

for all $S \subseteq \mathcal{D}_t$.

For example, considering the example in Figure 1 and Table 2, we have

$$\begin{aligned}\sigma(\{d_1, d_2, d_3, d_5\}) &= \{B\}\\ \sigma(\{d_2, d_4, d_6\}) &= \{C\}\\ \sigma(S) &= \emptyset, \text{ for any other subset } S.\end{aligned}$$

By fusing Pr and σ, we can obtain a mass function [27] [11] $m_t : 2^{\mathcal{D}_t} \rightarrow [0,1]$, which satisfies

$$m_t(S) = Pr^t(\{\theta \mid \theta \in \Theta_t,\ \Gamma_t(\theta) = S\}) \tag{2}$$

for every $S \subseteq \mathcal{D}_t$. For example, from the above we have

$$\begin{aligned}&m_t(\{d_1, d_2, d_3, d_5\}) = 0.55\\ &m_t(\{d_2, d_4, d_6\}) = 0.45\\ &m_t(S) = 0, \text{ for any other subset } S.\end{aligned}$$

At the decision level, we could use the *pignistic* probability decision approach [28]. This approach will generate a probability function $pignistic_t : \mathcal{D}_t \rightarrow [0,1]$, which satisfies

$$pignistic_t(\{d\}) = \sum_{S \subseteq \mathcal{D}_t,\ d \in S} \frac{m_t(S)}{\mid S \mid}. \tag{3}$$

For example, by using the above mass function m_t, we have

$$\begin{aligned}&pignistic_t(\{d_1\}) = 0.1375\\ &pignistic_t(\{d_2\}) = 0.2875\\ &pignistic_t(\{d_3\}) = 0.1375\\ &pignistic_t(\{d_4\}) = 0.15\\ &pignistic_t(\{d_5\}) = 0.1375\\ &pignistic_t(\{d_6\}) = 0.15\end{aligned}$$

To compare this result with the second row of Table 4, we find that *pignistic* probability function is just the normalization of the second row - the value while the relevant degree function R just gets support degree in step 2. In fact, from equation (2) and (3), we have

$$\begin{aligned}pignistic_t(\{d\}) &= \textstyle\sum_{S \subseteq \mathcal{D}_t,\ d \in S} \frac{m_t(S)}{|S|}\\ &= \textstyle\sum_{S \subseteq \mathcal{D}_t,\ d \in S} \frac{Pr(\{\theta \mid \theta \in \Theta_t,\ \Gamma_t(\theta) = S\})}{|S|}\\ &= \textstyle\sum_{\theta \in \Theta_t,\ d \in \Gamma_t(\theta)} \frac{Pr(\theta)}{|\Gamma_t(\theta)|}\end{aligned}$$

From this equation we can easily understand that the support evaluation in the evaluation algorithm (see Table 3) is just the fusion result while the subject agent only considers the contributions of other agents to retrieved documents.

The above interpretation does not consider the potential support. In order to interpret the random set (Pr, ξ) completely, we view each agent as two sides of one coin: one side is for the support degree and another is for the potential support degree. For example, the set of cooperated agents at time t is

$\{B', C', B'', C''\}$ not $\{B, C\}$, and a mapping $\Gamma_{\mathcal{A}}$ from $\{B', C', B'', C''\}$ to $2^{\mathcal{D}_t}$ is defined as follows:

$$\Gamma_{\mathcal{A}}(B') = \{d_1, d_2, d_3, d_5\}$$
$$\Gamma_{\mathcal{A}}(C') = \{d_2, d_4, d_6\}$$
$$\Gamma_{\mathcal{A}}(B'') = \{d_6\}$$
$$\Gamma_{\mathcal{A}}(C'') = \{d_1\}$$

where $\Gamma_{\mathcal{A}}(B')$ and $\Gamma_{\mathcal{A}}(C')$ are the sets of retrieved documents from B and C respectively (the support information). $\Gamma_{\mathcal{A}}(B'')$ is a subset of $\Gamma_{\mathcal{A}}(C')$, in which the retrieved documents' databases (e.g., D_4, the potential support information) do not served by B. $\Gamma_{\mathcal{A}}(C'')$ is a subset of $\Gamma_{\mathcal{A}}(B')$, in which the retrieved documents' databases (e.g., D_1, the potential support information) do not served by C.

We also use the function of *Precision* to describe Pr. For the potential supports, at moment we consider the minimum precisions, that means we will use the size of the set of all retrieved documents (e.g., 6) to get the corresponding precisions. Similar to equations (1), (2) and (3), the subject agent can obtain a mass function m_t and a *pignistic* probability function. We can also verify that the new *pignistic* probability function is the normalization of the third row of Table 4 – the final value of the relevant degree function R in the fusion algorithm of Table 3.

5.2 Performance of the Algorithm

The resource for this trial is "http://employment.news.com.au/". From this resource we down-loaded 653 jobs as a collection, which are classified as "Programmer" (by using the inner search engine) at time Tuesday December 14 11:25:01 EST 1999 (see [20]). The query is described as the follows:

< *top* >
< *title* > Java, C++ Programmer
< /*title* >
< *desc* > Description:
What job information is available for programmers, in which programming with Java, C++, and Oracle under Unix is preferred?
< /*desc* >
< /*top* >

The purpose of this trial is to show the efficiency of the cooperative data cleaning algorithm.

The subject agent (BROKER) sends this query to three agents: PIRM-based agent, RSBM-based agent, and SDSM-based agent. At fusion time, each agent provides a set of retrieved documents, in which each document has a collection name, document name and weight (similar value).

The PIRM-based agent uses the probabilistic IR model (see [20]). According to the user information need, it uses a query Q = {programmer, Java, Unix,

C++, Oracle} to get the set of possible relevant documents. PIRM-based agent provides 130 documents to reply the broker's request. The RSBM-based agent uses the rough set based IF model (see [20]). It first classifies the new documents into some categories, then selects the relevant documents from the categories based on the user information need. It provides 38 documents to reply this request.

The last set of retrieved documents comes from SDSM-based agent. This agent uses Dempster-Shafer index model to describe the user information need. We let the keyword-list K be {programmer, Java, Unix, C++, Oracle}, and the set U include the 36 relevant documents (feedback documents). SDSM-based agent first decides a mass function m_R on K, which satisfies:

$$m_R(\{programmer\}) = 0.385174$$
$$m_R(\{Java\}) = 0.304158$$
$$m_R(\{Unix\}) = 0.135181$$
$$m_R(\{C++\} = 0.100460$$
$$m_R(\{Oracle\} = 0.074998$$

It then selects the threshold

$$threshold = min_{d \in U}\{bel_{m_R}(C_d)\} = 0.595354,$$

where C_d is the corresponding class of relevant document d. For every downloaded job d_j, it uses the following formula to rank the jobs:

$$bel_{m_R}(C_{d_j}) = \sum_{term \in K, term \in d_j} m_R(\{term\}).$$

By using this threshold (0.595354), it returns 122 documents to reply the broker's request.

The broker, however, does not have any idea about which agents are better. So it assumes that every agent has the same trustworthiness (that means a same precision curve[1]). By using the cooperative fusion algorithm the subject agent divides the retrieved documents into 3 classes, the first class has 38 documents with the support from every agent (3 agents); the second class has 84 documents with the support from 2 agents, the third class has 8 documents with the support from one agent.

By resorting the 130=38+84+8 documents based on the cooperative fusion algorithm, the subject agent can select the top part documents to answer its users based on how many documents users want. The result of the experiment on precision and recall are displayed in Figure 2, where the document cutoff is 15.

To compare with PIRM-based agent and SDSM-based agent, BROKER can get a very high precision. To compare with RSBM-based agent, BROKER can repick up the 3 relevant documents that were filtered out by RSBM-based agent.

[1] The experiment result is not effected by any precision curve if it is decrement function.

In summary, this experiment shows that the fusion result is better than any single agent.

We also can conclude that our fusion result is better than most collection fusion methods' because our fusion result is enhanced rather than only approximated to the effectiveness of searching the entire set of documents as a single collection by any agent (the goal of collection fusion in distributed IR [29]).

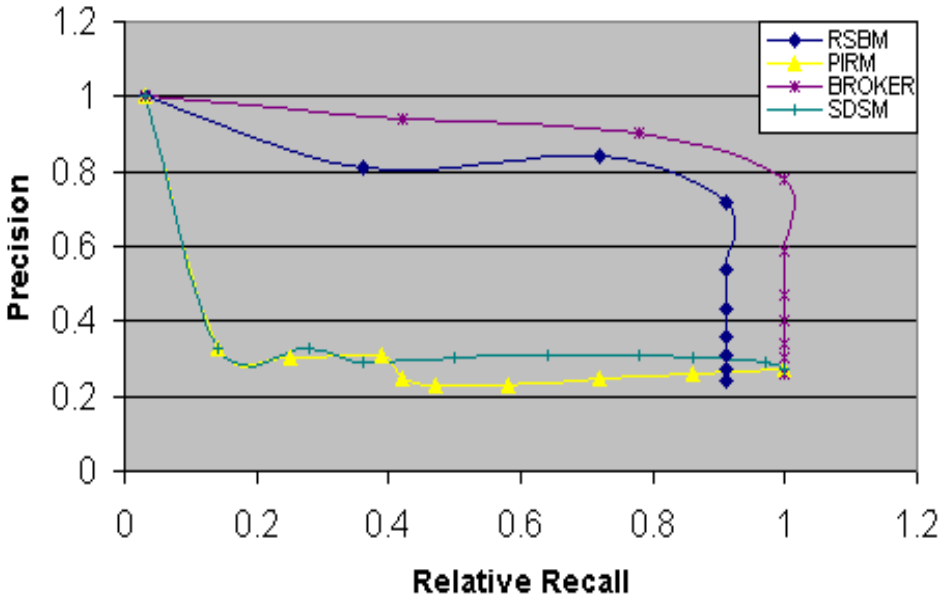

Fig. 4. Fusion Precision/Recall

6 Related Works

With respect to collection fusion, the problem was first identified in the paper [29]. This approach is to approximate the relevant document distribution over the collections by learning them from the results of past queries. Another approach [3] uses an inference network to rank not only documents at the provider sites. The collection fusion problem can also be solved by either using globally valid document frequencies (if the same cosine-based ranking method is used at each site) or by re-ranking selected (retrieved) documents at the broker site [23].

The highly heterogeneous collection (especially on the WWW) is an important feature for the modern IR. In this context "the possibility of using different IR models and index vocabularies at the different sites" is an important prop-

erty. According to the probability ranking principle, [2] presents a fusion model to handle heterogeneity among collections.

Metacrawler [8] is an agent that operates at a higher abstraction level by utilizing eight existing WWW index and search engines. Metacrawler is an example of an agent that does not index the documents itself, but provides a common interface to a number of search engines. This research is similar to the research in distributed IR, but the difference is that Metacrawler only knows the eight search engines very well, it dose not have any knowledge about the collections.

In contrast to other agent-based systems, BIG [13] performs information fusion. It retrieves documents, extracts attributes from the documents, converts unstructured text to structured data, and integrates the data. The last step is similar to solution synthesis in distributed expert systems [31]. Different from this kind of fusion, in this thesis we talk about the information fusion in case of unstructured text rather than structured information (data).

Data mining, which is referred to as knowledge discovery in database is a process of nontrivial extraction of implicit, previously unknown and potentially useful information (patterns) from data in databases [4] [6] [32]. To discover the potential useful knowledge, several typical approached have been presented. They are mining association rules, data classification and clustering, and data generalization and summarization. Data clustering has been studied in information retrieval for many years [1] [11]. The similar technique has already been used in machine leaning, data mining, and Web mining [26]. Different from the above methods, Our approach finds random sets not a set of weight vectors or rules.

7 Summary

As mentioned previously, one of the difficult problems for information gathering from the Web is overload. In this paper, a cooperative strategy and methodology for solving the problem is presented. To assess the results, the analysis for this cooperative cleaning algorithm is also presented based on evidence theory and a real example. We have shown that the cooperative approach can efficiently eliminate dirt data by re-sorting the retrieved documents. The main contributions are as follows:

(1) A Web mining model is presented to discover random sets from multiple representations of retrieved documents. The random set can be interpreted as Dempster-Shafer mass functions. Different to other mining methods, this model finds random sets rather than sets of weight vectors or rules.

(2) A data cleaning algorithm is presented to eliminate irrelevant data from multiple resources, which uses the random sets to resort the retrieved documents.

In addition, the research result can be used for collection fusion in distributed IR systems. It can extend the capability of the traditional collection fusion in distributed IR.

One of further work for this research is to create a model to obtain "potential support". Another is to discuss the relationship between a random set and a set

of weight vectors.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley, 1999.

[2] C. Baumgarten, A probabilistic solution to the selection and fusion problem in distributed information retrieval, *in: Proceedings of SIGIR'99*, 1999, 246-253.

[3] J. P. Callan, Z. Lu and W. B. Croft, Searching distributed collections with inference networks, *in Proceedings of SIGIR'95*, 1995, 21-29.

[4] M.-S. Chen, J. Han, and P. S. Yu, Data mining: an overview from a database perspective, *IEEE Transactions on Knowledge and Data Engineering*, 1996, **8(6):**866-883.

[5] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu, Learning Bayesian networks from data: an information-theory based approach, *Artificial Intelligence*, 2002, **137:** 43-90.

[6] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthrusamy, eds., *Advances in knowledge discovery and data mining*, Menlo Park, California : AAAI Press/ The MIT Press, 1996.

[7] E. H. Durfee, Distributed problem solving and planning, *in: Multiagent systems: a modern approach to distributed artificial intelligence, edited by G. Weiss, The MIT Press, Cambridge, Massachusetts, London, England*, 1999, 121-164.

[8] O. Etzioni, Results from using the metacrawler, *in Proceedings of 4th WWW Conference*, F. Varela and P. Bourgine (Eds.), MIT Press, Cambridge, MA, 1995.

[9] O. Etzioni and D. Weld, A softbotbased interface to the Internet, *Communications of the ACM*, 1994, **37(7):** *72-76.*

[10] N. Fuhr, A decision-theoretic approach to database selection in networked IR, *ACM Transactions on Information Systems*, 1999, **17(3):** 229-249.

[11] D. A. Grossman and O. Frieder, *Information retrieval algorithms and heuristics*, Kluwer Academic Publishers, Boston, 1998.

[11] J. W. Guan and D. A. Bell, *Evidence theory and its applications*, Volume 1, Studies in Computer Science and Artificial Intelligence 7, Elsevier, North-Holland, 1991.

[12] N. R. Jennings, K. Sycara and M. Wooldridge, A Roadmap of agent research and development, *Autonomous Agents and Multi-Agent Systems*, 1998, **1(1):** 7-38.

[13] V. Lesser et al., BIG: an agent for resource-bounded information gathering and decision making, *Artificial Intelligence*, 2000, **118:** 197-244.

[14] V. Lesser and S. Zilberstein, Intelligent information gathering for decision models, *Computer Science Technical Report TR-96-35*, University of Massachusetts at Amherst, 1996.

[15] A. Y. Levy and D. S. Weld, Intelligent Internet systems, *Artificial Intelligence*, 2000, **118:** 1-14.

[16] Y. Li, *Modelling intelligent agents for Web-based information gathering*, PhD Thesis, Deakin University, 2000.

[17] Y. Li, Information fusion for intelligent agent-based information gathering, *in: Web Intelligence: Research and Development (First Asia-Pacific Conference, WI 2001, Japan)*, LNAI 2198, Springer, 433-437.

[18] Y. Li and C. Zhang, A method for combining interval structures, *in: Proceedings of 7th International Conference on Intelligence Systems*, Paris France, 1998, 9-13.

[19] Y. Li and C. Zhang, Perceiving environments for Intelligent Agents, *in: Proceedings of 6th Pacific Rim International Conference on Artificial Intelligence*, Melbourne, Lecture Notes in Artificial Intelligence 1886, Springer-Verlag, 2000, 297-307.
[20] Y. Li, C. Zhang, and J. R. Swan, An information filtering model on the Web and its application in JobAgent, *Knowledge-based Systems*, 2000, **13(5):** 285-296.
[21] P. Maes, Agents that reduce work and information overload, *Communications of the ACM*, 1994, **37(7)**.
[22] T Mitchell el at., Experience with a learning personal assistant, *Communications of the ACM*, 1994, **37(7)**.
[23] W. Meng, K.Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe, Determining text databases to search in the Internet, *in Proceedings of 24th VLDB Conference*, 1998. Extended version.
[24] J. Mostafa, W. Lam and M. Palakal, A multilevel approach to intelligent information filtering: model, system, and evaluation, *ACM Transactions on Information Systems*, 1997, **15(4):** 368-399.
[25] A. Moukas and P. Maes, Amalthaea: an evolving multi-agent information filtering and discovery system for the WWW, *Autonomous Agents and Multi-Agent Systems*, 1998, **1(1):** 59-88.
[26] T. A. Runkler and J. C. Bezdek, Web mining with relational clustering, to appear in *International Journal of Approximate Reasoning.*
[27] G. Shafer, *A mathematical theory of evidence*, Princeton University Press, Princeton, NJ, 1976.
[28] P. Smets and R. Kennes, The transferable belief model, *Artificial Intelligence*, 1994, **66:** 191-234.
[29] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird, The collection fusion problem, *in Proceedings of TREC-3*, 1995, 95-104.
[30] S. K. M. Wong, L.S. Wang and Y.Y. Yao, Interval structure: a framework for representing uncertain information, *in: Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, California, 1992, 336-343.
[31] C. Zhang, Cooperation under uncertainty in distributed expert systems, *Artificial Intelligence*, 1992, **56(1):** 21-69.
[32] C. Zhang and S. Zhang, *Association Rules Mining: Models and Algorithms*, Springer-Verlag Publishers in Lecture Notes on Computer Science, Volume 2307, 2002.

This article was processed using the LaTeX macro package with LLNCS style