Web Intelligence and Agent Systems: An International Journal 0 (2010) 1–22
DOI 10.3233/WIA-2010-0000
IOS Press



# Application of rough ensemble classifier to web services categorization and focused crawling

Suman Saha *, C.A. Murthy and Sankar K. Pal
*Center for Soft Computing Research, Indian Statistical Institute*
*E-mail: {ssaha_r,murthy,sankar}@isical.ac.in*

**Abstract.** This paper discusses the applications of rough ensemble classifier [27] in two emerging problems of web mining, the categorization of web services and the topic specific web crawling. Both applications, discussed here, consist of two major steps: (1) split of feature space based on internal tag structure of web services and hypertext to represent in a tensor space model, and (2) combining classifications obtained on different tensor components using rough ensemble classifier. In the first application we have discussed the classification of web services. Two step improvement on the existing classification results of web services has been shown here. In the first step we achieve better classification results over existing, by using tensor space model. In the second step further improvement of the results has been obtained by using Rough set based ensemble classifier. In the second application we have discussed the focused crawling using rough ensemble prediction. Our experiment regarding this application has provided better Harvest rate and better Target recall for focused crawling.

Keywords: Rough ensemble classifier, web service categorization, WSDL tag structure, focused crawling, URL prediction

## 1. Introduction

Classification of web services and focused crawling are two emerging problems of web mining research. The classification tasks related to these problems are consists of special challenges, due to the semi structure nature of the data. A simple classifier is not directly applicable to these problems unless some special changes are made to capture the available information, and this special design play a crucial role in the final classification results. In our previous article [27], rough ensemble classifier has been designed for text classification task. The rough ensemble classifier has been designed in such a way that it can be use to classify the semi structured web data with all the available information of the data captured in the final results of classification. In this article two application of rough ensemble classifier has been discussed in details.

With the expected growth of the number of Web services available on the web, the need for mechanisms that enable the automatic categorization to organize this vast amount of data, becomes important. A major limitation of the Web services technology is that finding and composing services requires manual effort. This becomes a serious burden with the increasing number of Web services. Describing and organizing this vast amount of resources is essential for realizing the web as an effective information resource. Web Service classification has become an important tool for helping discovery and integration process to organize this vast amount of data. For instance, for categorization in the UDDI (Universal Description Discovery and Integration) registry, one needs to divide the publicly available Web Services into a number of categories for the users to limit the search scope. Moreover, Web Services classification helps the developer to build integrated Web Services. Discovery and integration of web services are becoming an im-

*Corresponding author. E-mail: saha.suman@gmail.com.

portant area of research, some useful articles in this regard are [5,8,19] and [21] to name a few. On the other hand, discovery of useful resources from WWW is an important problem. The visible Web consisting of billions of pages offers a challenging useful resource discovery problem. Even with increasing hardware and bandwidth resources at their disposal, search engines cannot keep up with the growth of the Web [18]. The retrieval challenge is further compounded by the fact that Web pages also change frequently. Thus, despite the attempts of search engines to index the whole Web, it is expected that the subspace eluding indexing will continue to grow. Therefore, collecting domain-specific documents from the Web has been considered one of the most important strategies to take benefit from the large amount of resources. Since late 1990s, there has been much research on different tools to build domain-specific Web collections and currently the most popular and widely-used tool is focused crawler [3,22].

Traditionally, Web Service classification is performed manually by domain experts. However, human classification is unlikely to keep pace with the rate of growth of the number of Web Services. Hence, as the web continues to increase, the importance of automatic Web Service classification becomes necessary. The information available to categorization algorithms comes from two sources. First, the algorithms use the web service description in the WSDL (Web Service Definition Language) format (Fig. 1), which is always available to determine a service's category. Second, in some cases, additional descriptive text is available, such as from a UDDI entry. An example of Address-Lookup web service and a part of its associated WSDL file is shown in Fig. 1.

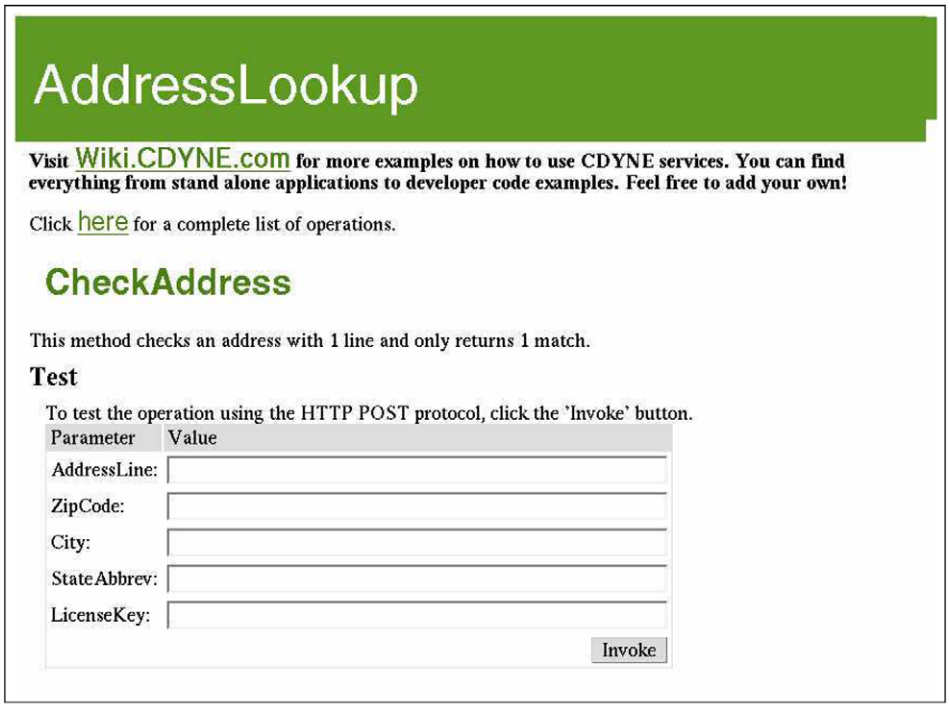
The problem of the automatic classification of Web services has been addressed in the literature with the help of two main approaches, (a) text classification approach [14] and (b) semantic similarity based classification approach [20]. Text classification is a long-standing problem, most solutions to this problem are based on term frequency analysis [2,12]. These approaches are insufficient in the web service context because text documentations for web-service operations are highly compact, and preprocessing techniques for HTML documents are not adequate to preprocess WSDL documents.

Work in the area of semantic similarity based classification approach has developed several methods that try to capture clues about the semantics similarity, and suggests classification based on them [11,16]. Such methods include linguistic analysis, structural analysis, use of domain knowledge and previous classification experience [25]. But these methods suffer from lack of annotation which is a manual process.

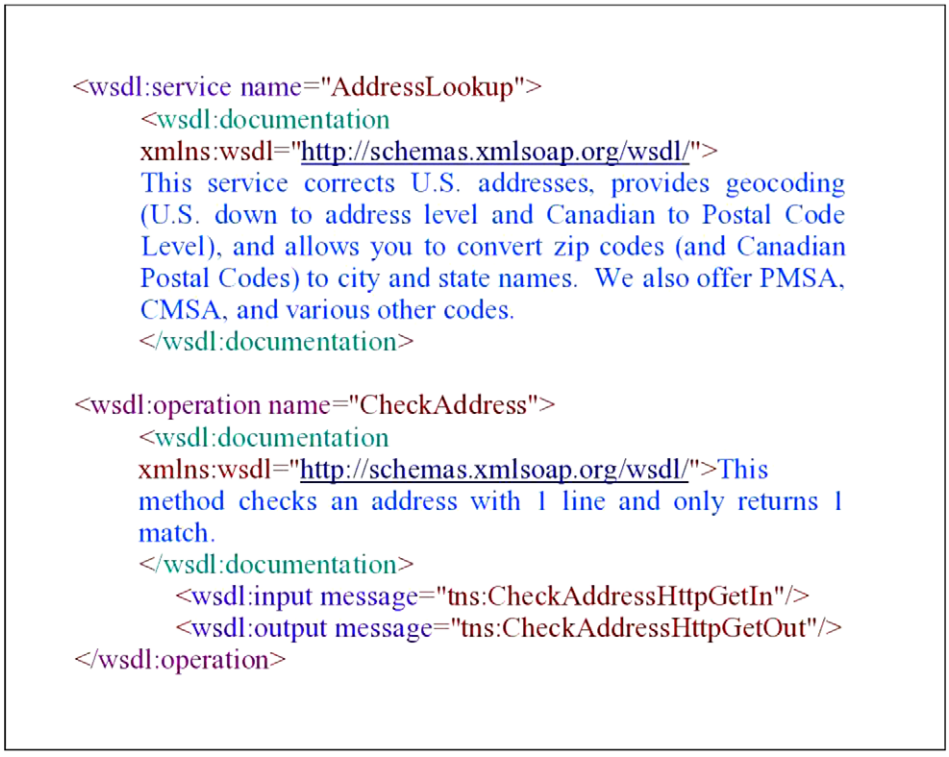
We treat the determination of a web services category as a tag based text classification problem, where the text comes from different tags of the WSDL file and from UDDI text. Unlike standard texts, WSDL descriptions are highly structured. In this article tensor space model is used to captures the information from internal structure of WSDL documents along with the corresponding text content and rough ensemble classifier is used to combine information of the individual tensor components for providing final classification results. Our experiments demonstrate that splitting the feature set based on structure improves the performance of a learning classifier. By combining different classifiers it is possible to improve the performance even further.

A focused crawler based on a hypertext classifier was developed by Chakrabarti et al. The basic idea of the crawler was to classify crawled pages with categories in a topic taxonomy. To begin, the crawler requires a topic taxonomy such as Yahoo. Focused crawlers are programs designed to selectively retrieve Web pages relevant to a specific domain for the use of domain-specific search engines and digital libraries, exploiting the graph structure of the Web to move from page to page [9,10,17]. Unlike the simple crawlers behind most general search engines which collect any reachable Web pages in breadth-first order, focused crawlers try to 'predict' whether or not a target URL is pointing to a relevant and high-quality Web page before actually fetching the page [6]. There has been much research on algorithms designed to determine the quality of Web pages. Basic architecture of a crawler and a focused crawler is given in Fig. 2.

A frontier is the to-do list of a crawler that contains the URLs of unvisited pages. The frontier is implemented as a priority queue. It may be a dynamic array that is always kept sorted by the estimated score of unvisited URLs. At each step, the best URL is picked from the head of the queue. Once the corresponding page is fetched, the URLs are extracted from it and scored based on some heuristic. They are then added to the frontier in such a manner that the order of the priority queue is maintained. Topic-driven crawlers are more specialized in certain topics and rely on different types of approaches to keep the crawling scope within the desired domain. This approaches use different type of features extracted from already crawled




(a) AddressLookup service




(b) WSDL file for AddressLookup service

Fig. 1. An example of (a) AddressLookup service and (b) a part of WSDL file for AddressLookup service.

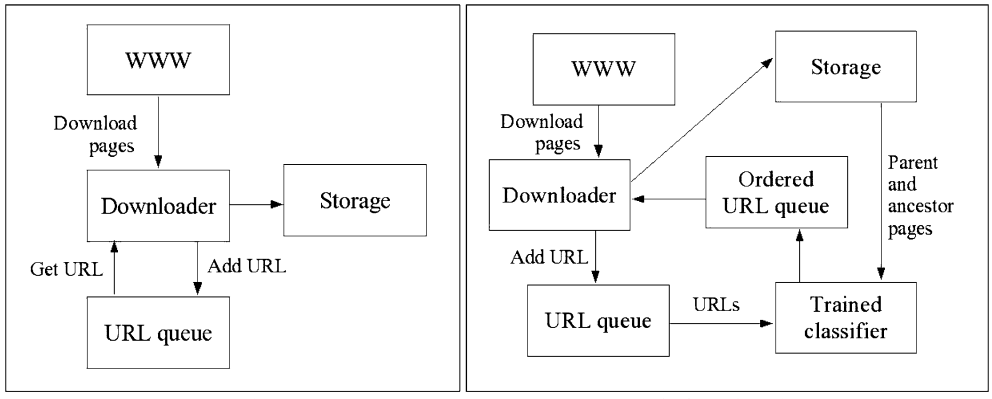



Fig. 2. Basic architecture of a (a) crawler and a (b) focused crawler.

pages. Most commonly used features are URL of the unvisited page, anchor text of the unvisited page, incoming links to unvisited page, outgoing links of parent page, category of parent page and categories of ancestor pages. The different types of features extracted for focused crawling are represented in a tensor space model, where different types of features are indexed in different tensor components. A classifier trained on a tensor component can be used to predict the category of unvisited web page. Here we have used a combination of all available predictions using rough set based ensemble classifier [27]. This method extracts the rules to decide the category of the unvisited URL. Each rules extracted here is associated with a priority value. Relevant web page decided by the high priority rule will be crawled next. In this article a combined method based on rough set theory has been applied. It combines the available predictions using rough decision rules and can build much larger domain specific collections.

In order to realize the specified objectives, Sections 2 and 3 presents tensor space model and rough set based ensemble classifier respectively. Section 4 and 6 covers the application of rough ensemble classifier for web service classification and focused crawling respectively. Finally, the experimental results are reported in Sections 8 and 9.

## 2. Tensor space model

Tensors provide a natural and concise mathematical framework for formulating and solving problems in high dimensional space analysis [2]. An n-order tensor in m-dimensional space is a mathematical object that has $n$ indices and $mn$ components and obeys certain transformation rules. Each index of a tensor ranges over the number of dimensions of space. Tensors are generalizations of scalars (0-order, which have no indices), vectors (1-order, which have a single index), and matrices (2-order, which have two indices) to an arbitrary number of indices.

Document indexing and representation has been a fundamental problem in information retrieval for many years. Most of previous works are based on the Vector Space Model (VSM). The documents are represented as vectors, and each word corresponds to a dimension. In this section, we introduce a new Tensor Space Model (TSM) for web services representation (Fig. 3). In Tensor Space Model, a web service is represented as a tensor. Each element in the tensor corresponds to a feature (word in our case). The tensor space model is based on different types of features extracted from the WSDL document and UDDI description. It offers a potent mathematical framework for analyzing the internal markup structure of WSDL documents along with text content. The TSM for web services consists of a rank two tensor, where first rank represents the types of features considered and the second rank represents the terms of corresponding types extracted from the WSDL and UDDI description. For each type of feature an individual tensor component is constructed. A tensor component is a vector, which represents the terms of particular type corresponding to the component. The tensor space model captures the structural representation of web services.

### *2.0.1. Mathematical formulations of TSM*

Let $W$ be a web service. Let $S$ be a set representing $W$. Let, $S^u = \{e_1^u, e_2^u, \ldots, e_{n_1}^u\}$ be the set corresponding to features of UDDI description, $S^{sn} = \{e_1^{sn}, e_2^{sn}, \ldots, e_{n_2}^{sn}\}$ be the set corresponding to the features of service name, $S^{sd} = \{e_1^{sd}, e_2^{sd}, \ldots, e_{n_3}^{sd}\}$

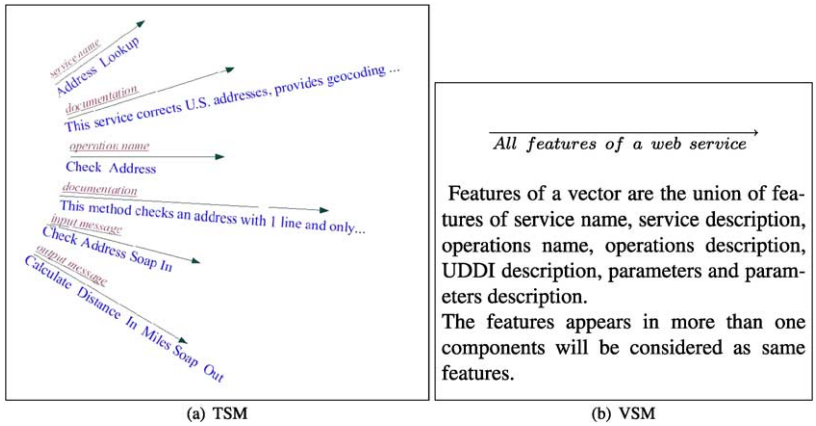



Fig. 3. Web services representation using (a) tensor space model and (b) vector space model.

be the set corresponding to the features of service description, $S^{on} = \{e_1^{on}, e_2^{on}, \ldots, e_{n_4}^{on}\}$ be the set corresponding to the features of operation names, $S^{od} = \{e_1^{od}, e_2^{od}, \ldots, e_{n_5}^{od}\}$ be the set corresponding to the features of operation descriptions, $S^{pn} = \{e_1^{pn}, e_2^{pn}, \ldots, e_{n_6}^{pn}\}$ be the set corresponding to the features of parameter names and $S^{pd} = \{e_1^{pd}, e_2^{pd}, \ldots, e_{n_7}^{pd}\}$ be the set corresponding to the features of parameter descriptions. Clearly $S = S^u \cup S^{sn} \cup S^{sd} \cup S^{on} \cup S^{od} \cup S^{pn} \cup S^{pd}$. Let $S_1$ be the set of features which presents in more than one components. So, $S_1 = \cup_{(x,y \in F) \& x \neq y} S^x \cap S^y$, where, $F = \{u, sn, sd, on, od, pn, pd\}$. Let, $s$ be an element of $S_1$. That is $s$ has occurred in more than one components of a web service. For each appearance of $s$ in different components $s$ may have different significant regarding the categorization of the web services. Now the multiple appearance of $s$ is ignored in $S$, as it is a set of union of the sets corresponding to the components of web services.

In the vector space model, vectors are constructed on $S$, that is, occurrence of $s \in S'$ in different components is ignored. In some advanced vector space model elements of different components are tagged, that is $S' = \phi$. Let $|.|$ denote cardinality of a set. Number of features exist in the different components varies highly. For example, $|S^u| << |S^c|$. In this representation, importance of the elements corresponding to the components with low cardinality, is ignored during magnitude normalization.

In the tensor space model the features corresponding to different components of web services are represented in different components of a tensor. Let $\mathcal{T}$ be the tensor space corresponding to a collection of web services. Each member $T$ of $\mathcal{T}$ is of the form $T = T_{xi}$ where, $x \in F$ and $1 \leqslant i \leqslant |S^x|$, i.e. the value of $T$ at $(x, i)$ is $e_i^x$. Note that $i$ depends on $x$, so it is not just a matrix.

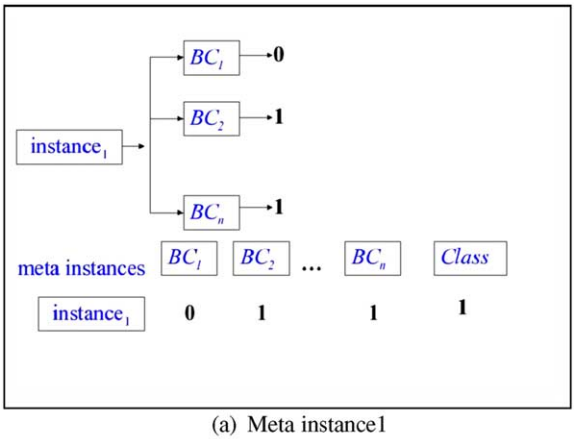
### 2.0.2. Similarity measures on TSM

Cosine similarity is a measure of similarity between two vectors of $n$ dimensions by finding the angle between them, often used to compare documents in text mining. Given two vectors of attributes, $A$ and $B$, the cosine similarity, $sim(A, B) = \frac{|A.B|}{|A|.|B|}$ where the word vectors $A$ and $B$ are represented after removing stop words and stemming. For text matching, the attribute vectors $A$ and $B$ are usually the tf-idf vectors of the documents. The resulting similarity will yield the value of 0 meaning, the vectors are independent, and 1 meaning, the vectors are same, with in-between values indicating intermediate similarities.

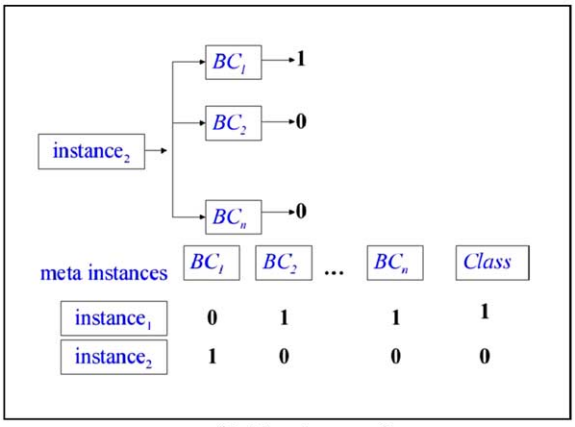
Let $\mathcal{T}$ be the tensor space corresponding to a collection of web services. Each member $T$ of $\mathcal{T}$ is of the form $T = T_{rs}$ where $r$ ranges on the types of features considered and $s$ ranges on number of terms extracted of particular types. The tensor similarity between two tensor $T_i$ and $T_j$ of $\mathcal{T}$ is defined as $sim(T_i, T_j) = \sum_r sim(T_i r, T_j r)$, where $sim(T_i r, T_j r)$ is the similarity between $r^{th}$ component of $T_i$ and $T_j$. Now, for each $r$ the $r^{th}$ component of a tensor $T_i$ is a vector. So, $sim(T_i r, T_j r)$ is basically the similarity between two vectors. Note that, here cosine similarity is considered as vector similarity measure.

### 2.0.3. Computational complexity on TSM

Let $n$ be the total number of features of a collection of web services. Let $n_1, n_2, \ldots, n_r$ be the number of features associated with the $1^{st}, 2^{nd}, \ldots, r^{th}$ components of the tensor respectively. From the definition of TSM we obtain $\sum_{i=1}^{r} n_i = n$. Let $m$ be the num-




(a) Meta instance1

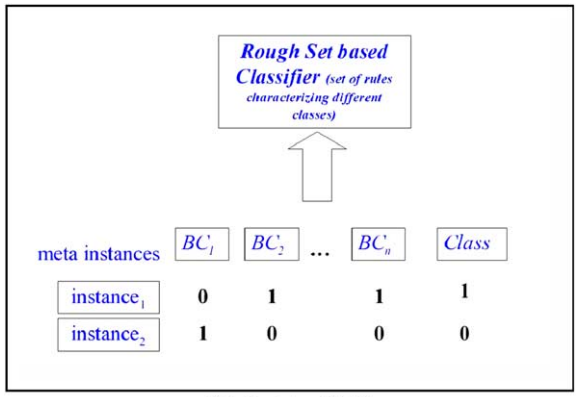



(b) Meta instance2

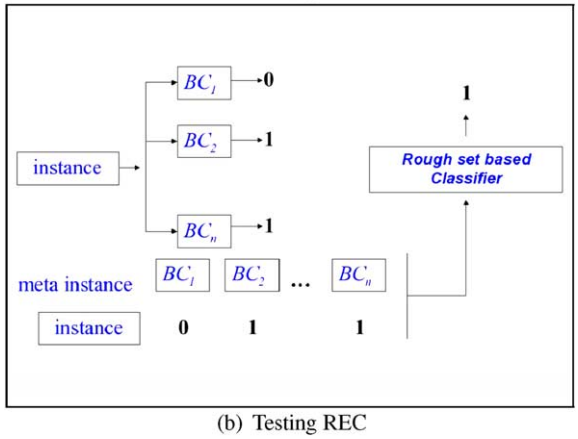
Fig. 4. Construction of decision table from the output of base classifiers. In sub figures (a) and (b) it have been shown that data instances are classified by base classifiers (denoted as BC) and their outputs along with the actual class have been considered to construct decision table.

ber of documents. The complexity of an algorithm, $\mathcal{A}$ constructed on VSM can be expressed as $f(m, n, \alpha)$, where $\alpha$ is corresponding to specific parameters of $\mathcal{A}$. The expression of complexity $f(m, n, \alpha)$ is written as: $O(m^i n^j \alpha^k)$. The complexity of the same algorithm, $\mathcal{A}$ constructed on TSM can be written as: $O(m^i n_t^j \alpha^k)$, where $n_t = max_{s=1}^{r}\{n_1, n_2, \ldots, n_r\}$. Since, $n_t < n$, we can write $(n_t)^j \leqslant n^j$. Hence, $O(m^i n_t^j \alpha^k) \leqslant O(m^i n^j \alpha^k)$. Hence the following theorem can be stated.

**Theorem 1.** *Computational complexity of an algorithm performing on tensor space model using tensor similarity measure as distance is at most the computational complexity of the same algorithm performing on vector space model using vector similarity measure as distance.*




(a) Training REC




(b) Testing REC

Fig. 5. Graphical representation of REC have been explained here. In sub figures (a) training of REC has been shown. Finally in sub figure (b) meta classifier has been used to obtain output.

## 3. Rough ensemble classifier

Our approach named REC is designed to extract decision rules from trained classifier ensembles that perform classification tasks [27]. REC utilizes trained ensembles to generate a number of instances consisting of prediction of individual classifiers as conditional attribute values and actual classes as decision attribute values. Then a decision table is constructed using all the instances with one instance in each row (Fig. 4).

Once the decision table is constructed, rough set attribute reduction is performed to determine core and minimal reducts [26]. The classifiers corresponding to a minimal reduct are then taken to form classifier ensemble for REC classification system. From the minimal reduct, the decision rules are computed by finding mapping between decision attribute and conditional attributes. These decision rules obtained by rough set technique are then used to perform classification tasks (Fig. 5).

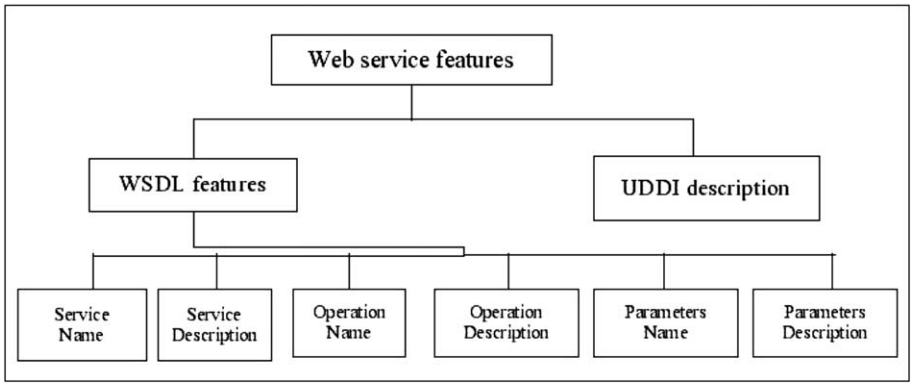



Fig. 6. Features of a web service.

### 3.1. Mathematical formulations of REC

In the problem of classification we train a learning algorithm and validate the trained algorithm. This task is performed, using some test-train split on a given categorized dataset. In the notion of rough set, let $U$ be the given categorized dataset and $P = \{C_1, C_2, \ldots, C_k\}$ where $C_i \neq \phi$ for $i = 1, 2, 3, \ldots, k$, $\cup_{i=1}^{k} C_i = U$ and $C_i \cap C_j = \phi$ for $i \neq j$ and $i, j = 1, 2, 3, \ldots, k$ be a partition on $U$ which determines given categories of $U$. Output of a classifier determines a new partition on $U$. This new partition is close to the given one with respect to some measure. In rough set terminology each class of the given partition is a given concept about dataset and output of classifiers determines new concepts about same dataset. Now given concepts can be expressed approximately by upper and lower approximation constructed by generated concepts [27]. Following theorems exists regarding performance of REC.

**Theorem 2.** *Rough set based combination is an optimal classifier combination technique* [27].

**Theorem 3.** *The performance of the rough set based ensemble classifier is at least same as every one of its constituent single classifiers* [27].

## 4. Classification of web services using rough ensemble classifier on tensor space model

### 4.1. Features of web services

A web service is typically published by registering its WSDL file and a brief description in UDDI registries. The WSDL file describes the functionalities of the web service and a text description in the UDDI registry describes the web service in words. Different types of features are extracted from WSDL file and UDDI text for categorization. These features are described below (Fig. 6).

1) **UDDI text:** Text description in the UDDI registry is a text file. The unique terms found in the text are features.

2) **Service name:** A service name consists of few number of terms i.e., features. These features are very informative for categorization purpose.

3) **Service description:** Service description occurs in the documentation tag below the service name in the WSDL file. This is a small text content. The unique terms found inside this text are features.

4) **Operation name:** Each operation of the web service are determined by its name. Operation names are found in the WSDL file. all the unique terms of operation names provide several features. These features are more meaningful towards the functionality of web services.

5) **Operation description:** Operation descriptions occur in the documentation tag below the corresponding operation name in the WSDL file. These are small text contents. The unique terms found inside this texts are features.

6) **Parameter name:** Each parameter of the web service are determined by its name. Parameter names are found in the WSDL file. all the unique terms of parameter names provide several features. These features are often found to be informative for categorization of web services.

7) **Parameter description:** Parameter descriptions occur in the documentation tag below the corresponding parameter name in the WSDL file. These are small text contents. The unique terms found inside this texts are features.

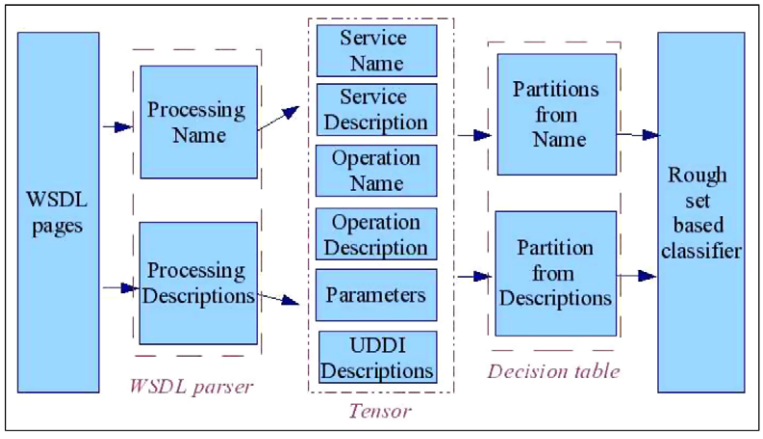



Fig. 7. Block diagram of proposed method.

## 5. Rough ensemble classification of web services

In this article tag based tensor space model is used for the representation of web service documents and Rough ensemble approach for its classification. Splits of the features has been performed based on tag set existing in the WSDL documents corresponding to web services. Tensor space model has been used to represent the services according to tag structure. Base level classification has been performed on individual tensor components. Finally combined classification has been obtained by using rough set based ensemble classifier.

### 5.1. Algorithmic steps

- **Preprocessing of web services:** We parse the port types, operations and messages from the WSDL and extract names as well as comments from various "documentation" tags. We do not extract standard XML Schema data types like string or integer, or information about the service provider. The extracted terms are stemmed with Porter's algorithm (Porter 1980), and a stop-word list is used to discard low-information terms.
- **Tensor space model:** We assume the tensor space model for tag based representation of web services in our classification task. First we select a set of relevant tags from a WSDL document. For each tag an individual tensor component is constructed. One more tensor component is constructed for UDDI description. A tensor component is a vector, which represents the terms found in the text under a particular tag or in UDDI text.
- **Base level classifications on TSM:** We now describe how we generate partitions for each one of the components of the tensor using classifiers.
  - **Partitions from components corresponding to names:** We consider the terms in a name as a bag of words. We have constructed three different bags from service name, operation names and input/output parameter names respectively and constructed three tensor components from each of this bags. Classification algorithm is applied on these tensor components after preprocessing. We obtain three different partitions from three different tensor components corresponding to names of service, operations and parameters.
  - **Partitions from components corresponding to description:** To obtain the partitions from descriptions corresponding to services, operations and parameters, we consider the documentation as a bag of words. Word stemming and stopword removal have been performed to preprocess the data. Classification algorithm is applied on the preprocessed bags to obtain partitions from the tensor components corresponding to service description, operation description and parameter description.
- **Final classification:** A decision table is constructed using the output of base level classifications. Instances of the decision table are found by utilizing trained ensembles to generate a number of instances consisting of prediction of individual classifiers associated with each tensor component as condition attribute values and the known actual class as decision attribute value. Then a decision table is constructed with one instance in each row. Once the decision table is constructed, rough set based attribute reduction is performed to deter-

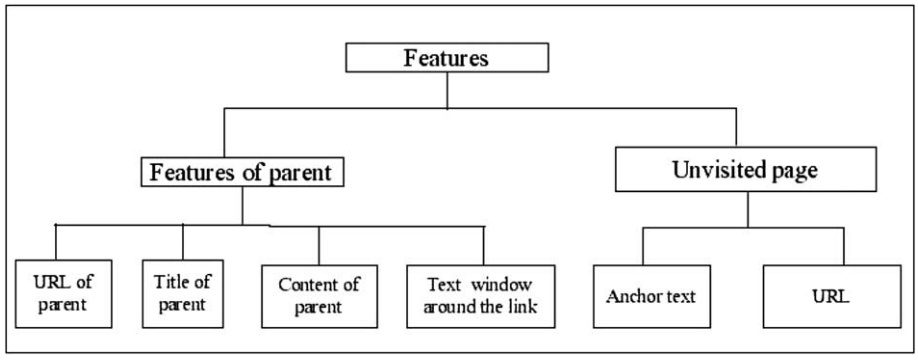



Fig. 8. Features used to predict unvisited pages.

mine core and minimal reduct. From the minimal reduct, the decision rules are computed by finding mapping between decision attribute and condition attributes. These decision rules obtained by rough set technique are then used to perform the final classification task.

## 6. Focused crawling using rough ensemble prediction on tensor space model

### *6.1. Hypertext features used to predict unvisited pages*

Focused crawlers rely on different types of approaches to keep the crawling scope within the desired domain. This approaches use different type of features extracted from already crawled pages. Most commonly used features are URL of the unvisited page, anchor text of the unvisited page, incoming links to unvisited page, outgoing links of parent page, category of parent page and categories of ancestor pages. These features are described below (Fig. 8).

1. URL of the unvisited page: Uniform resource locators (URLs), which mark the address of a resource on the World Wide Web, can be used to predict the category of the resource. A URL is first divided to yield a baseline segmentation into its components as given by the URI protocol (e.g., scheme :// host / path elements / document . extension), and further segmented wherever one or more non-alphanumeric characters appear (e.g., faculty-info → faculty info).
2. Anchor text of the unvisited page: Anchor text usually gives the user relevant descriptive or contextual information about the content of the link's destination. Thus it can be used to predict the category of the target page. Anchor text can provide a good source of information about a target page because it represents how people linking to the page actually describe it. Several studies have tried to use either the anchor text or the text near it to predict a target page's content.
3. Neighborhood features: Category of the already classified neighboring pages can be used to determine the categories of unvisited web pages. Chakrabarti et al. have studied the use of citations in the classification of IBM patents where the citations between them were considered as 'hyperlinks' and the categories were defined in a topical hierarchy. Recent research showed that breadth-first search could be also used to build domain-specific collections. The assumption here is that if the starting URLs are relevant to the target domain, it is likely that pages in the next level are also relevant to the target domain. Results from previous studies have shown that simple crawlers that fetch pages in a breadth-first order could generate domain-specific collections with reasonable quality. However, the size of collections built by such simple crawlers cannot be large because after a large number of Web pages are fetched, breadth-first search starts to lose its focus and introduces a lot of noise into the final collection.

### *6.2. URL ordering in the focused crawling frontier*

The goal in designing a focused crawler is to visits topic relevant pages, for some definition of topic relevancy. Of course, the crawler will only have available crawled pages, so based on these it will have to guess what are the topic relevant pages to fetch next. A crawler keeps a queue of URLs it has seen during a crawl, and must select from this queue the next

URL to visit. The ordering metric $O$ is used by the crawler for this selection, i.e., it selects the URL $u$ such that $O(u)$ has the highest relevancy among all URLs in the queue. The $O$ metric can only use information available in the crawled pages. The $O$ metric should be designed with an importance metric in mind. For instance, if we are searching for the pages of a particular domain, it makes sense to use features of URL to design ordering metric. Based on the feature set used, these kind of ordering algorithms can be categorized into many types. If we are interested for particular content then content-based analysis algorithms should be preferred, which apply indexing techniques for text analysis and keyword extraction to help determine whether a page's content is relevant to the target content. Assigning a higher weight to words and phrases in the title or headings is also standard information-retrieval practice that algorithm can apply based on appropriate HTML tags. The anchor text often contains useful information about a page. Several studies have tried to use either the anchor text or the text near it to predict a target page's content. Note that, most of the ordering algorithms are based on the assumption that the author of a Web page A, who places a link to Web page B, believes that B is relevant to A.

### 6.3. *Different types of focused crawling based on different types of predictions*

Focused crawlers rely on different types of algorithms, based of different types of features, to keep the crawling scope within the desired domain. Different types of algorithms, based of different types of features, to predict the category of unvisited pages are stated below.

1) Prediction algorithm based on URL of unvisited page:

- Preprocessing:
  - A URL is first divided to yield a baseline segmentation into its components as given by the URI protocol (e.g., scheme :// host / path elements / document . extension), and further segmented wherever one or more non-alphanumeric characters appear.
  - These segmented substrings are treated as words. All these words found in a URL will be represented as a vector. A vector space corresponding to URLs of the unvisited pages is constructed using these vectors.
- Classification:
  - First a vector space corresponding to URLs of training samples is constructed.
  - A base line classifier (naive bayes) is trained on these training vectors.
  - During the crawling process the URL of unvisited page is tested by the classifier to decide the category of the page.
  - Relevant URL is added to the crawling frontier.

2) Prediction algorithm based on anchor text:

- Preprocessing:
  - Anchor text is a small text content. The text is stemmed using Porter's stemming algorithm and stop words are removed.
  - Unique words present in the anchor text are represented as a vector. A vector space corresponding to the anchor texts is constructed using these vectors.
- Classification:
  - First a vector space corresponding to anchor texts of training samples is constructed.
  - A base line classifier (naive bayes) is trained on these training vectors.
  - During the crawling process the anchor text of unvisited page is tested by the classifier to decide the category of the page.
  - URL of the relevant anchor text is added to the crawling frontier.

3) Prediction algorithm based on URL of parent page:

- Preprocessing:
  - A URL is first divided to yield a baseline segmentation into its components as given by the URI protocol (e.g., scheme :// host / path elements / document . extension), and further segmented wherever one or more non-alphanumeric characters appear.
  - These segmented substrings are treated as words. All these words found in a URL will be represented as a vector. A vector space corresponding to URLs of the parent pages is constructed using these vectors.
- Classification:
  - First a vector space corresponding to URLs of training samples is constructed.

- A base line classifier (naive bayes) is trained on these training vectors.
- During the crawling process the URL of parent page is tested by the classifier to decide the category of the page.
- If URL of the parent page is relevant, then corresponding unvisited URL is added to the crawling frontier.

4) Prediction algorithm based on title of parent page:

– Preprocessing:

- Title is a small text content. The text is stemmed using Porter's stemming algorithm and stop words are removed.
- Unique words present in the title are represented as a vector. A vector space corresponding to the title of the parent page is constructed using these vectors.

– Classification:

- First a vector space corresponding to title of training samples is constructed.
- A base line classifier (naive bayes) is trained on these training vectors.
- During the crawling process the title of the parent page is tested by the classifier to decide the category of the page.
- If title of the parent page is relevant, then corresponding unvisited URL is added to the crawling frontier.

5) Prediction algorithm based on text content of parent page:

– Preprocessing:

- The text is stemmed using Porter's stemming algorithm and stop words are removed.
- Unique words present in the text are represented as a vector. A vector space corresponding to the text contents of the parent page is constructed using these vectors.

– Classification:

- First a vector space corresponding to text content of the of training samples is constructed.
- A base line classifier (naive bayes) is trained on these training vectors.
- During the crawling process the text contents of the parent page is tested by the classifier to decide the category of the page.
- If text content of the parent page is relevant, then corresponding unvisited URL is added to the crawling frontier.

6) Prediction algorithm based on text window around the URL:

– Preprocessing:

- The text in the window is stemmed using Porter's stemming algorithm and stop words are removed.
- Unique words present in the text window are represented as a vector. A vector space corresponding to the text contents of the parent page is constructed using these vectors.

– Classification:

- First a vector space corresponding to text window of training samples is constructed.
- A base line classifier (naive bayes) is trained on these training vectors.
- During the crawling process the text window around the URL is tested by the classifier to decide the category.
- If text window around the URL is relevant, then the URL is added to the crawling frontier.

## 7. Rough ensemble prediction for focused crawling

In this article we use a novel technique for combining the predictions about the category of unvisited page. Category of unvisited page can be predict using different ways. These predictions are generally made using different types of features available in the crawled pages. Features are extracted from URL, anchor text, and features of neighborhood pages. Predictions from these different types of features are used in the literature to guide a web crawler for topic specific web resource discovery. A combination of all these available predictions to decide the next URL to be crawled has been studied in this article (Fig. 9). Here rough set has been used to combine the predictions made by individual classifiers. Rough set based attribute reduction has been performed to remove redundant predictions. Rough set based decision rules has been used to decide the category of the unvisited web page. Rules obtained in this method are ranked according to their certainty score, which assigned with each rules. URLs corresponding to the most certain rule will be crawled first. In case of many URLs associated with same rule first obtained URL will be selected first.

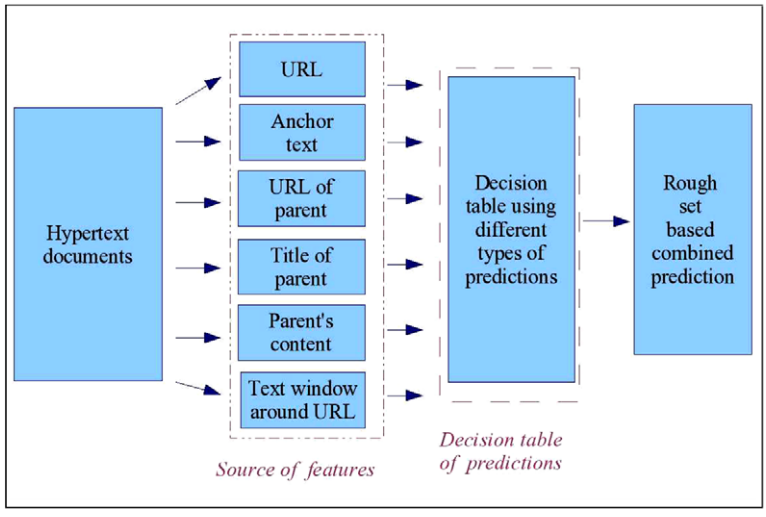



Fig. 9. Block diagram of proposed method.

## 8. Experimental results of web services application

### 8.1. Data sets

We gathered a corpses of web services from SALCentral and webservicelist, two categorized web service indices. These web service indices are multi-level tree-structured hierarchy. The top level of the tree, which is the first level below the root of the tree, contains 11 categories in SALCentral dataset (Table 1) and again 11 categories for web servicelist dataset (Table 2). Each of these categories contains sub-categories that are placed in the second level below the root. We use the top-level categories to label the web services in our experiments.

#### 8.1.1. SALCentral data set

The SALCentral data set is obtained from salcentral.com. This web service index is manually classified by the human experts. The extracted subset consisting of 424 web services, which are distributed among 11 top level categories. The largest category (Country Info) consists of 64 web services; while the smallest category (Mathematics) consists of only 10 web services. The minimum of 3 and 10% of the total pages in a category is taken as training sample, out of 424 web services 43 web services are taken as training samples and rest are considered as test samples. Detailed information about number of pages and number of training and testing are given in Table 1(a). We have also demonstrated the number of individual features in each components of web services. In case of SALCentral data set maximum number of features found in UDDI description (1355) and minimum number of features found in service name (132). Detailed information about the features of SALCentral data set is provided in Table 1(b).

#### 8.1.2. Webservicelist data set

The webservicelist data set is obtained from webservicelist.com. This is another available manually classified web service index. The extracted subset consisting of 444 web services, which are distributed among 11 top level categories. The largest category (Business) consists of 97 web services; while the smallest category (Sales) consists of only 20 web services. The minimum of 3 and 10% of the total pages in a category is taken as training sample, out of 444 web services 48 web services are taken as training samples and rest are considered as test samples. Detailed information about number of pages and number of training and testing are given in Table 2(a). We have also demonstrated the number of individual features in each components of web services. In case of webservicelist data set maximum number of features found in UDDI description (1537) and minimum number of features found in service name (190). Detailed information about the features of webservicelist data set is provided in Table 2(b).

### 8.2. Evaluation measures

We employ the standard measures to evaluate the performance of Web classification, i.e. precision, recall and $F_1$-measure.

Table 1
Class distribution and features of the Salcentral dataset

(a) Categories

| Categories | #services | #train | %train | #test | %test |
|---|---|---|---|---|---|
| Business | 22 | 3 | 13.63 | 19 | 86.36 |
| Communication | 44 | 4 | 9.09 | 40 | 90.90 |
| Converter | 43 | 4 | 9.30 | 39 | 90.69 |
| Country Info | 62 | 6 | 9.67 | 56 | 90.32 |
| Developers | 34 | 3 | 8.82 | 31 | 91.17 |
| Finder | 44 | 4 | 9.09 | 40 | 90.90 |
| Games | 42 | 4 | 9.52 | 38 | 90.47 |
| Mathematics | 10 | 3 | 30 | 7 | 70 |
| Money | 54 | 5 | 9.25 | 49 | 90.74 |
| News | 30 | 3 | 10 | 27 | 90 |
| Web | 39 | 4 | 10.25 | 35 | 89.74 |
| Total | 424 | 43 | 10.14 | 381 | 89.85 |

(b) Features

| Components | Corpus | Train |
|---|---|---|
| UDDI description | 1355 | 453 |
| Service name | 132 | 64 |
| Service description | 1167 | 388 |
| Operations name | 359 | 92 |
| Operations description | 1243 | 403 |
| Parameters | 486 | 161 |
| Total | 4742 | 1561 |
| Union | 3290 | 1131 |
| Additional | 1452 | 430 |

Table 2
Class distribution and features of the Web service list dataset

(a) Categories

| Categories | #services | #train | %train | #test | %test |
|---|---|---|---|---|---|
| Access | 27 | 3 | 11.11 | 24 | 88.88 |
| Locations | 57 | 6 | 10.52 | 51 | 89.47 |
| Business | 97 | 10 | 10.30 | 87 | 89.69 |
| Developers | 54 | 5 | 9.25 | 49 | 90.74 |
| Databases | 24 | 3 | 12.5 | 21 | 87.5 |
| Politics | 56 | 6 | 10.71 | 50 | 89.28 |
| Validations | 26 | 3 | 11.53 | 23 | 88.46 |
| Stock | 31 | 3 | 9.67 | 28 | 90.32 |
| Search | 22 | 3 | 13.63 | 19 | 86.36 |
| Sales | 20 | 3 | 15 | 17 | 85 |
| Retail | 30 | 3 | 10 | 27 | 90 |
| Total | 444 | 48 | 10.81 | 396 | 89.18 |

(b) Features

| Components | Corpus | Train |
|---|---|---|
| UDDI description | 1537 | 548 |
| Service name | 190 | 71 |
| Service description | 1426 | 405 |
| Operations name | 331 | 121 |
| Operations description | 1432 | 476 |
| Parameters | 408 | 179 |
| Total | 5324 | 1800 |
| Union | 3821 | 1246 |
| Additional | 1503 | 554 |

Precision ($P$) is the proportion of actual positive class members returned by the system among all predicted positive class members returned by the system.

Recall ($R$) is the proportion of predicted positive members among all actual positive class members in the data.

$F_1$ measure is the harmonic average of precision and recall. It is computed as:

$$F_1 = \frac{2PR}{P+R}$$

To evaluate the average performance across multiple categories, there are two conventional methods, micro-average and macro-average. Micro-average precision is the global calculation of precision measure regardless of categories. Macro-average precision is the average on precision scores of all categories. Micro-average recall is the global calculation of recall measure regardless of categories. Macro-average recall is the average on recall scores of all categories. Micro-average $F_1$ is the global calculation of $F_1$ measure regardless of categories. Macro-average $F_1$ is the average on $F_1$ scores of all categories. Micro-average gives equal weight to every document, while macro-average gives equal weight to every category, regardless of its frequency. In our experiments, micro precision, micro recall, micro $F_1$ macro precision, macro recall and macro $F_1$ will be used to evaluate the performance of classifications.

Table 3
Classification results using vector similarity and tensor similarity

(a) Salcentral dataset

| Measures | Micro | | Macro | |
|---|---|---|---|---|
| | VSM | TSM | VSM | TSM |
| Precision | 43.17 | 52.67 | 41.62 | 51.82 |
| Recall | 37.50 | 47.94 | 36.23 | 45.30 |
| $F_1$ | 40.13 | 50.20 | 38.73 | 48.34 |

(b) Webservicelist dataset

| Measures | Micro | | Macro | |
|---|---|---|---|---|
| | VSM | TSM | VSM | TSM |
| Precision | 62.56 | 67.64 | 60.67 | 66.80 |
| Recall | 54.57 | 60.68 | 52.19 | 62.93 |
| $F_1$ | 58.29 | 63.97 | 56.11 | 64.80 |

### 8.3. *Comparison of classification results using vector similarity and tensor similarity for k-NN classifier.*

Decisions of many vector space classifiers are based on a notion of distance, e.g., when computing the nearest neighbors in k-NN classification. For evaluation of the tensor space model for web service representation, we have constructed two k-NN classifier. In the first k-NN classifier on vector space representation for web services is considered and vector similarity measure is used to compute nearest neighbor. In the second k-NN classifier on tensor space model for web service representation is considered and tensor similarity measure is used to compute nearest neighbor. distance as the underlying distance. The performance of these two classifier has been observed on above mentioned data set. The classification results of comparison is shown in Tables 3(a) and 3(b). It can be observed from the tables that classification results are better when tensor space model for web service representation is considered compared to classification results when vector space model for representation is considered. The results has been shown in terms of precision, recall and $F_1$ measures.

### 8.4. *Classification results on individual components and combined results.*

In this subsection we have provided the results of experiments regarding classifications of web services. Classifications of web services have been performed on different types of feature sets found in UDDI descriptions and associated WSDL documents, i.e. it has been performed on different components of tensor space model. We have also provided the results of classification performed on, vector space model using union of all features. The combined results of classification is provided using two combination techniques, majority vote and rough set based ensemble classifier (REC). The cases considered are given below.

$A$) Classification based on features of UDDI description.

$B$) Classification based on features of Service name.

$C$) Classification based on features of Service description.

$D$) Classification based on features of Operations name.

$E$) Classification based on features of Operations description.

$F$) Classification based on features of Parameters.

$G$) Classification based on union of features in a VSM.

$H$) Classification based on majority vote on TSM.

$I$) Classification based on Rough set based ensemble classifier.

Note that, all the classifications tasks have been done using a single classification algorithm, in case of combined classifier this algorithm has been treated as base classifier. We have used three such classification algorithms, naive bayes (NB), support vector machine (SVM) and decision tree (DT). The detailed results in terms of micro precision, micro recall, micro-$F_1$, macro precision, macro recall and macro-$F_1$ of A, B, C, D, E, F, G, H and I on salcentral and webservicelist data set, using naive bayes classifier have been reported in Tables 4 and 5. Classification results in terms of $F_1$ measures, using support vector machine and decision tree classifiers have been given in Table 6.

### 8.5. *Comparisons with some recent web services classification techniques*

We have compared our method with other web service classification algorithms. A brief review of three existing web services classification techniques (A, B and C) have been given below. Three classification methods on tensor space model (D, E and F) have been considered for comparison with other methods.

$A_1$) The article, "An Approach to support Web Service Classification and Annotation" [20], proposes an approach to automatically classify services to specific domains and to identify key concepts inside service textual documentation, and build a lattice of relationships between service annotations. Support Vector Ma-

Table 4
Classification results on Salcentral dataset

| Methods | Micro | | | Macro | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ |
| A | 40.51 | 36.70 | 38.51 | 40.27 | 36.11 | 38.07 |
| B | 34.01 | 31.86 | 32.89 | 32.48 | 30.18 | 31.28 |
| C | 39.88 | 35.70 | 37.67 | 40.77 | 37.56 | 39.09 |
| D | 35.31 | 31.27 | 33.16 | 33.72 | 30.88 | 32.23 |
| E | 38.80 | 35.00 | 36.80 | 36.65 | 33.86 | 35.19 |
| F | 36.86 | 34.23 | 35.49 | 34.73 | 31.23 | 32.88 |
| G | 46.86 | 42.42 | 44.52 | 46.34 | 42.02 | 44.07 |
| H | 52.77 | 47.89 | 50.21 | 51.98 | 46.84 | 49.27 |
| I | 64.29 | 59.42 | 61.75 | 63.11 | 60.31 | 61.67 |

Table 5
Classification results on Web service list dataset

| Methods | Micro | | | Macro | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ |
| A | 58.83 | 54.93 | 56.81 | 57.89 | 54.33 | 56.05 |
| B | 48.43 | 44.28 | 46.26 | 48.42 | 43.56 | 45.86 |
| C | 52.90 | 49.17 | 50.96 | 51.43 | 47.87 | 49.58 |
| D | 49.61 | 47.43 | 48.49 | 48.96 | 46.48 | 47.68 |
| E | 56.19 | 52.11 | 54.07 | 55.66 | 51.77 | 53.64 |
| F | 54.88 | 51.84 | 53.31 | 52.60 | 49.72 | 51.11 |
| G | 61.68 | 57.21 | 59.36 | 59.15 | 57.65 | 58.39 |
| H | 68.92 | 62.61 | 65.61 | 67.53 | 63.73 | 65.57 |
| I | 74.35 | 70.38 | 72.31 | 73.16 | 69.34 | 71.19 |

Table 6
Classification results using SVM and Decision Tree, on Salcentral and Web service list dataset

| Methods | Sal-central | | | | Web-service-list | | | |
|---|---|---|---|---|---|---|---|---|
| | SVM | | Decision Tree | | SVM | | Decision Tree | |
| | Mi-$F_1$ | Ma-$F_1$ | Mi-$F_1$ | Ma-$F_1$ | Mi-$F_1$ | Ma-$F_1$ | Mi-$F_1$ | Ma-$F_1$ |
| A | 43.17 | 42.35 | 41.62 | 39.23 | 62.56 | 57.61 | 60.67 | 57.48 |
| B | 36.74 | 35.62 | 35.87 | 34.27 | 52.51 | 44.91 | 49.35 | 46.63 |
| C | 41.95 | 38.80 | 40.81 | 36.84 | 55.41 | 50.86 | 53.02 | 48.25 |
| D | 37.70 | 34.49 | 37.09 | 33.40 | 54.95 | 55.51 | 55.86 | 53.87 |
| E | 40.00 | 36.62 | 39.38 | 38.58 | 59.71 | 55.07 | 58.96 | 54.99 |
| F | 39.88 | 36.24 | 38.56 | 34.04 | 58.35 | 54.72 | 56.70 | 52.79 |
| G | 55.50 | 52.07 | 49.23 | 47.34 | 65.57 | 60.96 | 64.19 | 60.18 |
| H | 58.78 | 56.19 | 56.31 | 53.47 | 73.63 | 67.88 | 70.90 | 67.53 |
| I | 66.27 | 62.93 | 65.35 | 62.96 | 76.35 | 67.58 | 74.97 | 72.78 |

chines and Formal Concept Analysis have been used to perform the two tasks.

$B_1$) The article "Iterative Ensemble Classification for Relational Data: A Case Study of Semantic Web Services" [15], proposes the use two separate classifiers for the intrinsic and the relational (extrinsic) attributes and vote their predictions. It also introduce a new way of exploiting the relational structure.

$C_1$) In the article "A Heuristic Approach to Semantic Web Services Classification" [7], a heuristic based

Table 7
Comparison of classification results on Salcentral dataset

| | Micro | | | Macro | | |
|---|---|---|---|---|---|---|
| Methods | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ |
| $A_1$ | 45.34 | 46.39 | 45.86 | 45.23 | 41.36 | 43.21 |
| $B_1$ | 50.59 | 46.05 | 48.21 | 47.16 | 43.18 | 45.08 |
| $C_1$ | 51.15 | 43.25 | 46.87 | 47.24 | 44.16 | 45.65 |
| $D_1$ | 52.67 | 47.94 | 50.20 | 51.82 | 45.30 | 48.34 |
| $E_1$ | 52.77 | 47.89 | 50.21 | 51.98 | 46.84 | 49.27 |
| $F_1$ | 64.29 | 59.42 | 61.75 | 63.11 | 60.31 | 61.67 |

Table 8
Comparison of classification results on Web service list dataset

| | Micro | | | Macro | | |
|---|---|---|---|---|---|---|
| Methods | Precision | Recall | $F_1$ | Precision | Recall | $F_1$ |
| $A_1$ | 61.37 | 60.83 | 61.10 | 61.35 | 58.24 | 59.76 |
| $B_1$ | 63.79 | 61.39 | 62.57 | 63.43 | 61.72 | 62.57 |
| $C_1$ | 65.06 | 63.94 | 64.49 | 61.65 | 58.64 | 60.11 |
| $D_1$ | 67.64 | 60.68 | 63.97 | 66.80 | 62.93 | 64.80 |
| $E_1$ | 68.92 | 62.61 | 65.61 | 67.53 | 63.73 | 65.57 |
| $F_1$ | 74.35 | 70.38 | 72.31 | 73.16 | 69.34 | 71.19 |

mechanism is proposed, that enables service publishers to classify their services in a service taxonomy, managed by a service repository.

$D_1$) Here we have considered k-NN classification algorithm on tensor space model. The classifier use proposed tensor similarity measure for comparing web services.

$E_1$) Here we have considered naive bayes classification on each different components in the tensor space model. The classification results have been combined using majority voting method.

$F_1$) Here we have considered naive bayes classification on each different components in the tensor space model. The classification results have been combined using rough set based ensemble classifier.

Results in terms of micro precision, micro recall, micro-$F_1$, macro precision, macro recall and macro-$F_1$ of $A_1$, $B_1$, $C_1$, $D_1$, $E_1$ and $F_1$ on salcentral and web service list data set have been reported in Tables 7 and 8 respectively.

## 9. Experimental results of focus crawling application

Our experiments are run over sixteen top level topics that are obtained from the Open Directory Project (ODP). For each topic, we have positive and negative examples. The positive examples are Web pages that have been manually judged to be on the topic, and negative examples are randomly selected Web pages from other topics. We keep the number of negatives to be twice the number of positives. The positive and negative examples are represented in TF-IDF (term frequency-inverse document frequency) vector space. Hence, all of the examples are parsed and tokenized to identify the words within them. The stop-words are removed and the remaining words are stemmed using the Porter stemming algorithm [Porter 1980]. These stemmed words or terms from all of the negative and positive examples form our vocabulary V for the topic. This vocabulary may differ across topics. Both the positive and the negative example pages represented as TF-IDF based feature vectors are used for training a classifier. We call the trained classifier the crawling classifier since we will use it to guide a topical crawler. We have explored combinations of classifiers using rough set that perform well.

### 9.1. Focus crawling data

We obtained topics for our crawls from topic directories of Dmoz and Yahoo. First we have collected the topic listing and the external pages relevant to each topics. These relevant pages have been judged to be relevant to the topic by human experts. In our experi-

Table 9
Categories, related cites, seeds and targets of Dmoz and Yahoo

(a)

| Categories | Cites | Seeds | Targets |
|---|---|---|---|
| Arts | 60 | 20 | 40 |
| Business | 96 | 32 | 64 |
| Computers | 85 | 29 | 56 |
| Games | 79 | 27 | 52 |
| Health | 62 | 21 | 41 |
| Home | 86 | 29 | 57 |
| Sports | 54 | 18 | 36 |
| Kids and Teens | 78 | 26 | 52 |
| News | 63 | 21 | 42 |
| Recreation | 89 | 30 | 59 |
| Reference | 65 | 22 | 43 |
| Regional | 72 | 24 | 48 |
| Science | 74 | 25 | 49 |
| Shopping | 59 | 20 | 39 |
| World | 52 | 18 | 34 |
| Society | 97 | 33 | 64 |

(b)

| Categories | Cites | Seeds | Targets |
|---|---|---|---|
| Arts and Humanities | 86 | 29 | 57 |
| Business and Economy | 79 | 27 | 52 |
| Computers and Internet | 112 | 38 | 74 |
| Education | 84 | 28 | 56 |
| Entertainment | 123 | 41 | 82 |
| Government | 78 | 26 | 52 |
| Health | 106 | 36 | 70 |
| News and Media | 91 | 31 | 60 |
| Recreation and Sports | 88 | 30 | 58 |
| Reference | 108 | 36 | 72 |
| Regional | 102 | 34 | 68 |
| Science | 135 | 45 | 90 |
| Social Science | 85 | 29 | 56 |
| Society and Culture | 126 | 42 | 84 |

ments we have considered top sixteen topics of Dmoz and top fourteen topics of Yahoo. For each topic we have divided the relevant pages into two random disjoint subsets. The first set is the seeds. This set of URLs will be used to initialize the crawl as well as provide the positive examples to train the classifiers. The second set is the targets. These targets will be used only for evaluation of crawl performance. Details about categories, related cites, seeds and targets are given in Tables 9.

### 9.2. Performance metrics

The output of a crawler is a temporal sequence of pages crawled. Any evaluation of crawler performance is hence based on this output. The key question for evaluation is: How do we judge the relevance of crawled pages? If the Web was a small controlled collection, we could use human judges to classify all pages as relevant or irrelevant to a given topic. With this knowledge, we could estimate the precision and recall of a crawler after crawling t pages. The precision would be the fraction of pages crawled that are relevant to the topic and recall would be the fraction of relevant pages crawled. However, the Web is neither controlled nor small. The relevant set for any given topic is unknown and, hence, the true recall is hard to measure [24]. The precision could be measured using manual relevance judgment on the crawled pages. However, the manual relevance judgment for each of the crawled pages is extremely costly in terms of man-hours when we have millions of crawled pages spread over more than 100 topics. Even if we sample for manual evaluation, we will have to do so at various points during a crawl to capture the temporal nature of crawl performance. Hence, even such samples would be costly to evaluate over 100 topics. Hence, the two standard information retrieval (IR) measures, recall and precision, can only be estimated using surrogate metrics. Below, we describe harvest rate which we use as an estimate of precision and target recall which we use as an estimate of recall.

#### 9.2.1. Harvest rate

Harvest rate estimates the fraction of crawled pages that are relevant to a given topic. Since manual relevance judgment of Web pages is costly, we depend on multiple classifiers to make this decision. That is, we use a set of evaluation classifiers that act as an automated judges to decide on the relevance of a crawled page. For each topic, we take one classifier at a time and train it using the pages corresponding to the entire ODP relevant set (instead of just the seeds) as the positive examples. The negative examples are obtained from the ODP relevant sets of the other topics. The negative examples are again twice as many as the positive examples. These trained classifiers are called evaluation classifiers to distinguish them from the crawl-

Table 10
Results of focus crawling with different type of feature sets on Dmoz

| Features | 1000 pages | | 2000 pages | | 3000 pages | | 4000 pages | |
|---|---|---|---|---|---|---|---|---|
| | HR | TR | HR | TR | HR | TR | HR | TR |
| $F_1$ | 0.3574 | 0.1225 | 0.3433 | 0.1546 | 0.3272 | 0.1720 | 0.3131 | 0.1811 |
| $F_2$ | 0.3334 | 0.1071 | 0.3095 | 0.1406 | 0.2897 | 0.1641 | 0.2670 | 0.1852 |
| $F_3$ | 0.3078 | 0.0788 | 0.2827 | 0.1170 | 0.2700 | 0.1266 | 0.2548 | 0.1428 |
| $F_4$ | 0.2842 | 0.0676 | 0.2763 | 0.1032 | 0.2594 | 0.1113 | 0.2499 | 0.1356 |
| $F_5$ | 0.3307 | 0.0861 | 0.3139 | 0.1168 | 0.2942 | 0.1230 | 0.2692 | 0.1652 |
| $F_6$ | 0.1443 | 0.1039 | 0.1357 | 0.1125 | 0.1314 | 0.1341 | 0.1218 | 0.1555 |

Table 11
Results of focus crawling with different type of feature sets on Yahoo

| Features | 1000 pages | | 2000 pages | | 3000 pages | | 4000 pages | |
|---|---|---|---|---|---|---|---|---|
| | HR | TR | HR | TR | HR | TR | HR | TR |
| $F_1$ | 0.3503 | 0.1175 | 0.3396 | 0.1465 | 0.3184 | 0.1660 | 0.3040 | 0.1764 |
| $F_2$ | 0.3243 | 0.0938 | 0.3121 | 0.1344 | 0.3061 | 0.1549 | 0.2990 | 0.1780 |
| $F_3$ | 0.2934 | 0.0699 | 0.2898 | 0.0983 | 0.2665 | 0.1170 | 0.2483 | 0.1326 |
| $F_4$ | 0.3035 | 0.0584 | 0.2878 | 0.0951 | 0.2743 | 0.1109 | 0.2509 | 0.1245 |
| $F_5$ | 0.3206 | 0.0747 | 0.3071 | 0.1079 | 0.2903 | 0.1229 | 0.2784 | 0.1500 |
| $F_6$ | 0.3304 | 0.0584 | 0.3253 | 0.0808 | 0.3113 | 0.0994 | 0.3028 | 0.1273 |

ing classifiers that are used to guide the crawlers. Harvest rate, $H(t)$, after crawling the first $t$ pages is then computed as: $H(t) = 1/t \sum_{i=1}^{t} r_i$ where $r_i$ is the binary (0/1) relevance score for page $i$ based on a majority vote among the evaluation classifiers. In order to obtain the trajectory of crawler performance over time, the harvest rate is computed at different points during the crawl. Here, time is estimated by $t$, the number of pages crawled. We have one such trajectory for each topic and each crawler. However, to provide an idea of the crawler's performance in general, we average the harvest rate at various points over the 100 topics chosen for the experiment and draw this average trajectory over time for each crawler. We also compute the standard error around the average harvest rates. The performance trajectory can help an application designer decide on an appropriate crawler based on the length of crawl required by the application. For example, a crawler may perform well for crawls of a few hundred pages but poorly for longer crawls.

#### 9.2.2. Target recall

Target recall is an estimate of the fraction of relevant pages (on the Web) that are fetched by a crawler. As described earlier, true recall is hard to measure since we cannot identify the true relevant set for any topic over the Web. Hence, we treat the recall of the target set, i.e., target recall, as an estimate of true recall. If targets are a random sample of the relevant pages on the Web, then we can expect target recall to give us a good estimate of the actual recall. The target recall, $R(t)$, after first crawling $t$ pages for a given topic is computed as: $R(t) = \frac{|C(t) \cap T|}{|T|}$ where $C(t)$ is the set of first $t$ pages crawled, $T$ is the set of targets, and $|T|$ is the number of targets. As with harvest rate, we compute the average target recall, the standard error, and represent the crawler's performance as a trajectory over time.

### 9.3. Results of focus crawling with different set of features

Results of crawl based on different type of predictions made by the classifiers trained on different feature sets are shown in Tables 10 and 11. Focused crawler considered hare are $F_1$ (prediction from URL of unvisited page), $F_2$ (prediction from Anchor text), $F_3$ (prediction from URL of parent page), $F_4$ (prediction from title of parent page), $F_5$ (prediction from content of parent page), $F_6$ (prediction from text window (10) around the URL). Crawling process have been performed on sixteen top level ODP categories and average harvest rate and average target recall are

Table 12
Harvest rate of voting and rough set based combined prediction on Dmoz

| Algorithm | 1000 pages | 2000 pages | 3000 pages | 4000 pages | 5000 pages | 6000 pages |
|---|---|---|---|---|---|---|
| Voting | 0.5210 | 0.5078 | 0.4959 | 0.4813 | 0.4685 | 0.4604 |
| RS-Combined | 0.5873 | 0.5822 | 0.5787 | 0.5695 | 0.5639 | 0.5631 |

Table 13
Harvest rate of voting and rough set based combined prediction on Yahoo

| Algorithm | 1000 pages | 2000 pages | 3000 pages | 4000 pages | 5000 pages | 6000 pages |
|---|---|---|---|---|---|---|
| Voting | 0.5210 | 0.5078 | 0.4959 | 0.4813 | 0.4685 | 0.4604 |
| RS-Combined | 0.5873 | 0.5822 | 0.5787 | 0.5695 | 0.5639 | 0.5631 |

Table 14
Target recall of voting and rough set based combined prediction on Dmoz

| Algorithm | 500 pages | 1000 pages | 1500 pages | 2000 pages | 2500 pages | 3000 pages |
|---|---|---|---|---|---|---|
| Voting | 0.0312 | 0.0602 | 0.0755 | 0.0804 | 0.0923 | 0.0995 |
| RS-Combined | 0.0534 | 0.1104 | 0.1535 | 0.1708 | 0.1986 | 0.2094 |

Table 15
Target recall of voting and rough set based combined prediction on Yahoo

| Algorithm | 500 pages | 1000 pages | 1500 pages | 2000 pages | 2500 pages | 3000 pages |
|---|---|---|---|---|---|---|
| Voting | 0.0333 | 0.0575 | 0.0713 | 0.0814 | 0.0918 | 0.1004 |
| RS-Combined | 0.0518 | 0.1123 | 0.1510 | 0.1687 | 0.1961 | 0.2089 |

reported. Note that for each topics same seeds have been considered for all the crawlers. It can be observed from the table that performance corresponding to $F_1$, $F_2$ and $F_5$ are good but performance corresponding to $F_3$, $F_4$ and $F_6$ are relatively poor.

### 9.4. *Comparison of combination techniques on Dmoz and Yahoo topics.*

Average harvest rate of focused crawler using combination of different predictions has been shown in Tables 12 and 13. Performance of two combination techniques, voting and rough set based combination is reported on Dmoz and Yahoo. It cab be observed that average harvest rate of rough set based combination is better than that of voting method with respect to different number pages crawled.

Average target recall of focused crawler using combination of different predictions has been shown in Tables 14 and 15. Performance of two combination techniques, voting and rough set based combination is reported Dmoz and Yahoo. It cab be observed that average target recall of rough set based combination is better than that of voting method with respect to different number pages crawled.

### 9.5. *Comparisons with widely used focus crawling results*

We have compared our method with other focused crawling algorithms. A brief review of these algorithms is given below.

– **Best-first [23]:**
A best-first crawler represents a fetched Web page as a vector of words weighted by occurrence frequency. The crawler then computes the cosine similarity of the page to the query or description provided by the user, and scores the unvisited URLs on the page by this similarity value. The URLs are then added to a frontier that is maintained as a priority queue based on these scores. In the next iteration each crawler thread picks the best URL in the frontier to crawl, and returns with new unvisited URLs that are again inserted in the priority queue after being scored based on the cosine similarity of the parent page.

– **Topic sensitive page rank [13]:**
  In this approach to topic-sensitive PageRank, the importance scores are precomputed with ordinary Page-Rank. A set of scores of the importance of a page with respect to various topics are computed. At query time, these importance scores are combined based on the topics of the query to form a composite PageRank score for those pages matching the query. This score can be used in conjunction with other IR-based scoring schemes to produce a final rank for the result pages with respect to the query. The improvements to PageRank's precision is shown.
– **Info-spider [1]:**
  In InfoSpiders, an adaptive population of agents search for pages relevant to the topic. Each agent is essentially following the crawling loop while using an adaptive query list and a neural net to decide which links to follow. The algorithm provides an exclusive frontier for each agent. In a multi-threaded implementation of InfoSpiders each agent corresponds to a thread of execution. Hence, each thread has a non-contentious access to its own frontier. A back-propagation algorithm is used for learning. Such a learning technique provides InfoSpiders with the unique capability to adapt the link-following behavior in the course of a crawl by associating relevance estimates with particular patterns of keyword frequencies around links.
– **Shark-search [4]:**
  SharkSearch is a version of FishSearch with some improvements. It uses a similarity measure for estimating the relevance of an unvisited URL. However, SharkSearch has a more refined notion of potential scores for the links in the crawl frontier. The anchor text, text surrounding the links or link-context, and inherited score from ancestors influence the potential scores of links. The ancestors of a URL are the pages that appeared on the crawl path to the URL. SharkSearch, like its predecessor FishSearch, maintains a depth bound. That is, if the crawler finds unimportant pages on a crawl path it stops crawling further along that path. To be able to track all the information, each URL in the frontier is associated with a depth and a potential score.

Results on harvest rate and target recall of above mentioned algorithms and RS-combined on Dmoz and Yahoo has been reported in Tables 16 and 17 respectively. Same seeds have been considered for all the algorithms and average results on sixteen top level ODP category is taken for comparison. It can be observed that performance of RS-combined is marginally better than Best-first and Page-rank in terms of average harvest rate and average target recall.

## 10. Conclusion

We discussed the problem of classifying a web service and focused crawling using rough ensemble classifier on tensor space model. The classification of web services is treated as a split merge classification problem. Splits of the features have been performed based on tag set exists in the WSDL documents corresponding to web services. Tensor space model has been used to represent the services according to tag structure. Base level classification has been performed on individual tensor components. Finally combined classification has been obtained by using rough set based ensemble classifier. Two step improvement on the existing classification results of web services has been shown. In the first step we achieve better classification results by using tensor space model. In the second step further improvement of the results has been obtained by using Rough set based ensemble classifier.

On the other hand, several techniques of focused crawling have been proposed recently for topic specific resource discovery. All these crawlers use different types of hypertext features to predict the relevance of unvisited page. A combined method based on rough set theory has been studied in this article. Rough set based attribute reduction has been performed to remove redundant predictions. Rough set based decision rules has been used to decide the category of the unvisited web page. Results reported in the experimental section shows that performance of rough set based combined crawler is better than other focused crawling methods.

## Acknowledgements

The authors would like to thank the Department of Science and Technology, Government of India, for funding the Center for Soft Computing Research: A National Facility. This paper was done when one of the authors, S.K. Pal, was a J.C. Bose Fellow of the Government of India.

Table 16
Harvest rate and Target recall of popular methods and rough set based combined predictions on Dmoz

(a) Harvest rate on Dmoz

| Algorithm | 1000 pages | 2000 pages | 3000 pages | 4000 pages | 5000 pages | 6000 pages |
|---|---|---|---|---|---|---|
| Best-First | 0.5654 | 0.5589 | 0.5520 | 0.5467 | 0.5332 | 0.5229 |
| TSPage-Rank | 0.5381 | 0.5313 | 0.5206 | 0.5058 | 0.4841 | 0.4711 |
| Info-spider | 0.5237 | 0.5169 | 0.5022 | 0.4876 | 0.4716 | 0.4665 |
| Shark-search | 0.5019 | 0.4969 | 0.4824 | 0.4738 | 0.4618 | 0.4532 |
| RS-Combined | 0.5873 | 0.5822 | 0.5787 | 0.5695 | 0.5639 | 0.5631 |

(b) Target recall on Dmoz

| Algorithm | 500 pages | 1000 pages | 1500 pages | 2000 pages | 2500 pages | 3000 pages |
|---|---|---|---|---|---|---|
| Best-First | 0.0614 | 0.0945 | 0.1265 | 0.1538 | 0.1751 | 0.2026 |
| TSPage-Rank | 0.0626 | 0.0843 | 0.1029 | 0.1312 | 0.1502 | 0.1662 |
| Info-spider | 0.0682 | 0.0868 | 0.1167 | 0.1429 | 0.1653 | 0.1871 |
| Shark-search | 0.0539 | 0.0783 | 0.1015 | 0.1381 | 0.1591 | 0.1737 |
| RS-Combined | 0.0628 | 0.1193 | 0.1641 | 0.1822 | 0.2030 | 0.2164 |

Table 17
Harvest rate and Target recall of popular methods and rough set based combined predictions on Yahoo

(a) Harvest rate on Yahoo

| Algorithm | 1000 pages | 2000 pages | 3000 pages | 4000 pages | 5000 pages | 6000 pages |
|---|---|---|---|---|---|---|
| Best-First | 0.4689 | 0.4821 | 0.4576 | 0.4370 | 0.4576 | 0.4567 |
| TSPage-Rank | 0.4649 | 0.4190 | 0.4239 | 0.4077 | 0.4244 | 0.3653 |
| Info-spider | 0.4492 | 0.4594 | 0.4198 | 0.4157 | 0.3871 | 0.3751 |
| Shark-search | 0.4130 | 0.4444 | 0.3918 | 0.4306 | 0.3859 | 0.3435 |
| RS-Combined | 0.4896 | 0.5122 | 0.4764 | 0.5053 | 0.4988 | 0.5160 |

(b) Target recall on Dmoz

| Algorithm | 500 pages | 1000 pages | 1500 pages | 2000 pages | 2500 pages | 3000 pages |
|---|---|---|---|---|---|---|
| Best-First | 0.0534 | 0.0843 | 0.1177 | 0.1384 | 0.1670 | 0.1952 |
| TSPage-Rank | 0.0531 | 0.0736 | 0.0933 | 0.1202 | 0.1459 | 0.1551 |
| Info-spider | 0.0605 | 0.0792 | 0.1101 | 0.1332 | 0.1585 | 0.1764 |
| Shark-search | 0.0464 | 0.0737 | 0.0915 | 0.1310 | 0.1521 | 0.1627 |
| RS-Combined | 0.0533 | 0.1087 | 0.1538 | 0.1769 | 0.2002 | 0.2113 |

## References

[1] C. Aggarwal. Collaborative crawling: Mining user experiences for topical resource discovery, 2002.

[2] Deng Cai, Xiaofei He, and Jiawei Han. Tensor space model for document analysis. *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 625–626, 2006.

[3] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1623–1640, 1999.

[4] Chen, Zhumin Ma, Jun Lei, Jingsheng Yuan, Bo Lian, and Li. An improved shark-search algorithm based on multi-information. In *Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, pages 659–658, Haikou, China, 24-27 Aug. 2007.

[5] Antonella Chirichiello and Gwen Salaün. Encoding process algebraic descriptions of web services into BPEL. *Web Intelligence and Agent Systems, IOS Press*, 5(4):419–434, 2007.

[6] Junghoo Cho, Hector García-Molina, and Lawrence Page. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.

[7] M.A. Corella and P. Castells. A heuristic approach to semantic web services classification. *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 2006.

[8] Alfredo Cuzzocrea. Combining multidimensional user models and knowledge representation and management techniques for

making web services knowledge-aware. *Web Intelligence and Agent Systems, IOS Press*, 4(3):289–312, 2006.

[9] Darren Davis and Eric Jiang. Exploring content and linkage structures for searching relevant web pages. *Advanced Data Mining and Applications, Lecture Notes in Computer Science, Springer Berlin Heidelberg*, 4632:15–22, 2007.

[10] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, pages 527–534, Cairo, Egypt, 10–14 September 2000.

[11] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity search for web services. In *Proc. of VLDB*, 2004.

[12] A. Wong G. Salton and C.S. Yang. A vector-space model for information retrieval. In *Journal of the American Society for Information Science*, 18:13–620, 1975.

[13] Taher Haveliwala. Topic-sensitive pagerank. In *Proceedings of the Eleventh International World Wide Web Conference*, May 2002.

[14] Andreas Heb and Nick Kushmerick. Learning to attach semantic metadata to web services. *The IEEE International Conference*, 2003.

[15] Andreas Heb and Nick Kushmerick. Iterative ensemble classification for relational data: A case study of semantic web services. *ECML/PKDD 2004, Pisa, Italy*, 2004.

[16] Yu Jianjun, Guo Shengmin, Su Hao, Zhang Hui, and Xu Ke. A kernel based structure matching for web services search. *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1249–1250, 2007.

[17] John King, Yuefeng Li, Xiaohui Tao, and Richi Nayak. Mining world knowledge for analysis of search engine content. *Web Intelligence and Agent Systems, IOS Press*, 5(3):233–253, 2007.

[18] Steve Lawrence and C. Lee Giles. Searching the World Wide Web. *Science*, 280(5360):98–100, 1998.

[19] Eric Lo, David Cheung, C.Y. Ng, and Thomas Lee. Wsipl: An xml scripting language for integrating web service data and applications. *Web Intelligence and Agent Systems, IOS Press*, 4(1):24–41, 2006.

[20] M. Bruno, G. Canfora, M. Di Penta, and R. Scognamiglio. An approach to support web service classification and annotation. *The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*, pages 138–143, 2005.

[21] Zakaria Maamar, Quan Z. Sheng, and Boaulem Benatallah. Towards a conversation-driven composition of web services. *Web Intelligence and Agent Systems, IOS Press*, 2(2):145–150, 2004.

[22] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Building domain-specific search engines with machine learning techniques. In *Proc. AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace, 1999*, 1999.

[23] F. Menczer, G. Pant, and P. Srinivasan. Topical web crawlers: Evaluating adaptive algorithms, 2003.

[24] Filippo Menczer, Gautam Pant, Padmini Srinivasan, and Miguel E. Ruiz. Evaluating topic-driven web crawlers. In *Research and Development in Information Retrieval*, pages 241–249, 2001.

[25] Nicole Oldham, Christopher Thomas, Amit P. Sheth, and Kunal Verma. Meteor-s web service annotation framework with machine learning classification. *SWSWPC*, 3387:137–146, 2004.

[26] Z. Pawlak. Rough Sets: Theoretical Aspects of Reasoning About Data. *Kluwer, Boston*, pages 41–48, 1991.

[27] Suman Saha, C.A. Murthy, and Sankar K. Pal. Rough set based ensemble classifier for web page classification. *Fundamentae Informetica*, 76(1-2):171–187, 2007.