Focused Crawling for Automatic Service Discovery, Annotation and Classification in Industrial Digital Ecosystems

Hai Dong, *Student Member, IEEE*, Farookh K. Hussain

***Abstract*—Digital Ecosystems make use of Service Factories for service entities' publishing, classification and management. However, before the emergence of Digital Ecosystems, there existed ubiquitous and heterogeneous service information in the Business Ecosystems environment. Therefore, dealing with the pre-existing service information becomes a crucial issue in Digital Ecosystems. This issue has not been addressed previously in the literature. In order to resolve this issue, in this paper we present a conceptual framework for a semantic focused crawler, with the purpose of automatically discovering, annotating and classifying the service information with the Semantic Web technologies. The technical and evaluation details of the framework are also presented and discussed in this paper.**

***Index Terms*—Digital Ecosystems, semantic focused crawler, service discovery, service annotation, service classification, service ontologies**

I. Introduction

The emergence of Digital (Business) Ecosystems can be attributed to the natural existence of Business Ecosystems [1], along with the evolution of business network and information technology. Services involved in the Digital Ecosystems environment have the features of diversity and geographical dispersal, including individual services such as food and beverage, business/organizational services and web services, etc.. From the perspective of services, species in Digital Ecosystems can play two roles, which are service requester (client) that needs services, and service provider (server) that provides services. In addition, a species can play the two roles simultaneously. A service provider enters a Digital Ecosystem environment by publishing a service entity, which will be stored in distributed service knowledge bases [2]. Here, these service entities are stored in the form of service metadata [3]. Service Factories are a group of functional components within the Digital Ecosystems environment, which allows a service provider to create and test a service entity [4].

Manuscript received July 9, 2009, accepted March 3, 2010.

The authors are with the Digital Ecosystems and Business Intelligence Institute in Curtin University of Technology, Bentley, WA 6102, Australia (phone: 61-08-9266-9270; fax: 61-08-9266-7548; e-mail: {hai.dong, farookh.hussain}@cbs.curtin.edu.au).

When a service provider publishes a service entity by means of Service Factories, the service entity can be annotated by alternative Semantic Web markup languages such as Resource Description Framework (RDF) [5] or Web Ontology Language (OWL) [6], etc., and categorized by domain-specific ontologies provided within Digital Ecosystems, by referencing the Uniform Resource Identifier (URI) of the service metadata to the ontological concepts [3, 7]. However, Service Factories ignore the issue that, before the emergence of Digital Ecosystems, service entities had already been ubiquitous in the Business Ecosystems environment, and were heterogeneous without sufficient ontological support. For instance, in the online Yellowpages®, Yahoo! or Google™ local search or other local business directories, vast service information is available. It is, however, important to note that the existing service information cannot easily be retrieved [8]. One reason for this problem is the lack of semantic annotation (e.g., the retrieved service results may not fulfill users' query requirements, due to the lack of semantic support for denoting users' query intentions and annotating the service information), and service domain knowledge-oriented classification (e.g., service information is usually interspersed with product information under the same category, and it is difficult to distinguish between them). As a result, the lack of methodologies for discovering and semanticizing those service entities is a crucial issue for Digital Ecosystems, in order to ensure the efficient and effective working of Digital Ecosystems as a whole. Moreover, in the existing literature, there is no methodology especially designed for classifying the ubiquitous service entities, which hampers the retrieval performance for these entities. Hence, owing to the huge number of non-semantic service entities, there is an urgent need for an automatic and ontology-based service entity discovering, annotating and classifying methodology. Such a methodology could be used to discover and categorize service entities, thereby resulting in an improvement in service entity collection, management and retrieval in the Digital Ecosystems environment.

Coincidentally, the Semantic Web provides such tools for domain-knowledge-based classification [6]. For example, OWL-DL follows the principle of Description Logic (DL) that represents domain knowledge by defining concepts and specifying relations between concepts, which is employed for the classification of concepts and individuals. The

classification of concepts refers to defining a hierarchical structure of concepts determined by the concept-subconcept relationship. The classification of instances is used to determine whether an individual is an instance of a concept.

A web crawler is a software agent that can automatically browse and download webpages from the web. Web crawlers are usually deployed for retrieving and indexing web documents for search engines, which enables search engines to rank the visiting priority of web documents in terms of topics or user queries [9-16].

As the combined product of the Semantic Web and web crawlers, a semantic focused crawler is a software agent that can search and download web information for certain specific topics by means of the Semantic Web technologies [17]. Here, pertinent to the two issues above, we present a conceptual framework for a semantic focused crawler with the purpose of automatic service discovery, annotation and classification in the Digital Ecosystems environment.

The rest of the paper will be organized as follows: in Section 2, we will briefly review the current researches in the field of semantic focused crawlers and state our research motivations; in Section 3, we will present the overall framework of the proposed semantic focused crawler; in Section 4, we will validate the framework by means of a series of experiments; conclusions are drawn and future works are outlined in the final section.

II. Semantic Focused Crawlers

According to the existing literature, the emerging semantic focused crawlers can be distinguished into two primary categories: ontology-based focused crawlers and metadata abstraction focused crawlers [17]. In the rest of this section, we will discuss the features of the two categories of crawlers and examine their potential issues when crawling in the Digital Ecosystems environment.

A. Ontology-based Focused Crawlers

Ontology-based focused crawlers refer to a group of focused crawlers that link web documents with related ontology concepts, with the purpose of filtering and categorizing web documents [18].

LSCrawler is a typical example of ontology-based focused crawlers, which makes use of ontologies to analyze the semantic similarity between Uniform Resource Locators (URLs) of webpages and topics [19]. Topics are stored in the form of ontologies within an ontology base. Once a query has been provided, the compatible ontology will be retrieved from the ontology base. The query will then be sent to the commercial search engines, and its relevant URLs will be retrieved from the search engines. Next, a multi-thread crawler will be generated in order to fetch webpages based on these URLs. After the webpages have been crawled, all URLs and their surrounding texts will be extracted from the webpages. These texts will then be matched with the concepts of a compatible ontology, in order to determine the relevance of URLs to the query.

Tane et al. [20] proposed an ontology-based focused crawler for an ontology management system – Courseware Watchdog. By means of this crawler, a user can specify individual preference, in terms of assigning weights to some ontology concepts. The weights of the other ontology concepts can then be obtained by considering the interrelations among all concepts in an ontology. Once a webpage has been fetched by the crawler, its text and URL descriptions will be matched with the weighted ontology concepts in order to calculate the similarity between the URLs and ontology concepts. The calculated similarity will be combined with the ontology concept weights as the weight of the webpage. In terms of the above process, webpages can be classified and ranked by the ontology concepts.

Ganesh et al. [21] propose an association metric, with the purpose of optimizing the order of visited URLs for web crawlers. For each URL, an association metric evaluates its semantic content based on a reference domain-specific ontology. In addition, the metric of URL can analyze the link strength between parent and children webpages after the latter has been downloaded, in order to refine itself.

THESUS aims to organize online documents by linking their URLs to hierarchical ontology concepts, which are seen as thematic subsets [22]. A focused crawler is used in the document acquisition component of the system. The working mechanism of this crawler is as follows: first, the crawler extracts the URLs and their descriptive texts from the initial set of documents; then the descriptive text of one URL is matched with one of the ontological concepts, and the URL is linked to the concept. A threshold of maximum times of recursions or maximum number of documents is set as an ending requirement.

The limitation of the ontology-based focused crawlers can be described as follows:

Most of these crawlers fetch the surrounding texts of URLs as the descriptive texts of the URLs, and compute the similarity between the URLs and ontology concepts based on these texts. However, the surrounding texts sometimes cannot be used to correctly or sufficiently describe the URLs, which may increase the fault rate of the similarity computing.

B. Metadata Abstraction Focused Crawlers

Metadata abstraction focused crawlers are the focused crawlers that can abstract meaningful information from relevant webpages and annotate the information with ontology mark-up languages [23].

Francesconi and Peruginelli [24] proposed a metadata abstraction focused crawler for a Vertical Portal system, which is a management system for legal documents. Once a web document has been downloaded, a metadata generator will extract the meaningful information and convert it to metadata. The metadata format is in accordance with the Dublin Core scheme in its XML version. Next, two algorithms – Naive

Bayes and Multiclass Support Vector Machines (MSVM) are adopted respectively for the metadata-based document classification.

Giles et al. [25] proposed a metadata abstraction-focused crawler for a niche e-business information search engine. The focused crawler employs the CiteSeer technique to parse texts from downloaded web documents and generate metadata based on the parsed texts. Eventually, the Support Vector Machine (SVM) algorithm is employed for the metadata-based document classification.

The limitation of the metadata abstraction-focused crawlers is as follows:

These crawlers mostly utilize the supervised classification models such as MSVM and SVM etc., for document categorization. Most of the supervised classification models use predefined classifiers, described in plain texts without enough semantic support, for the metadata-based web document distance computing. This may give rise to two issues: 1) the plain-text classifiers are without semantic support, which may decrease the performance of document classification; 2) the classifiers may not match the actual domain knowledge structure, leading to a situation where the classified web documents cannot fulfill users' requirements for the domain knowledge-based search.

C. *Issues of the Semantic Focused Crawlers for the Digital Ecosystems Environment*

Apart from the respective limitation of each category of semantic focused crawlers, these crawlers share a common shortcoming when meeting the requirement of service entity crawling in the Digital Ecosystems environment, which can be described as follows:

Digital Ecosystems have an urgent requirement for service entity discovery, registry and categorization in its inclusive service domains. However, the existing semantic focused crawlers lack service domain knowledge for the service crawling. Therefore, the task of designing service domain knowledge can be regarded as a research challenge for service crawling in Digital Ecosystems.

In terms of exploring the limitations of the ontology-based focused crawlers and the metadata abstraction crawlers, the issues observed from the two categories could be resolved when we combine the specialties of the two categories of crawlers. To explain in detail, we combine the speciality of ontology-based classification, and the speciality of metadata abstraction and metadata-based web document classification. The former endows classifiers with sufficient semantic support, resulting in the reduction of the matching error rate between classifiers and web documents, and more importantly, precisely matches classifiers with domain knowledge structure. The latter uses metadata as the descriptive texts of URLs instead of simply fetching surrounding texts, which may enrich the semantics of the URL description and enhance the classification effect.

Therefore, in the rest of this paper, based on the research motivation introduced above, we present an innovative semantic focused crawler for service discovery, annotation and classification in the Digital Ecosystems environment. This crawler combines the features of the ontology-based focused crawlers and the metadata abstraction focused crawlers.

III. Overview of the Semantic Focused Crawler

In this section, we provide a detailed overview of the proposed semantic focused crawler, from the perspective of functions, architecture, components, working process, data format and relevant mathematical models.

A. *System Architecture*

Before we introduce the system architecture of the semantic focused crawler, the proposed functions of the crawler need to be stated as follows:

- The crawler is able to retrieve the information regarding service entities from the web, which corresponds to the functionality of service discovery in Digital Ecosystems.
- The crawler is able to annotate the service information with the purpose of semanticizing, and to store the semanticized service information, which corresponds to the functionality of service annotation in Digital Ecosystems.
- The crawler is able to filter and classify the annotated service information by means of specific service domain knowledge, which corresponds to the functionality of service classification in Digital Ecosystems.

In order to realize the proposed functions above, we design a conceptual framework for the semantic focused crawler, which is primarily comprised of two components – a semantic focused crawler and a Service Knowledgebase (Fig. 1).

From Fig. 1, it is observed that the semantic focused crawler consists of five agents: Webpage Fetcher, Policy Centre,

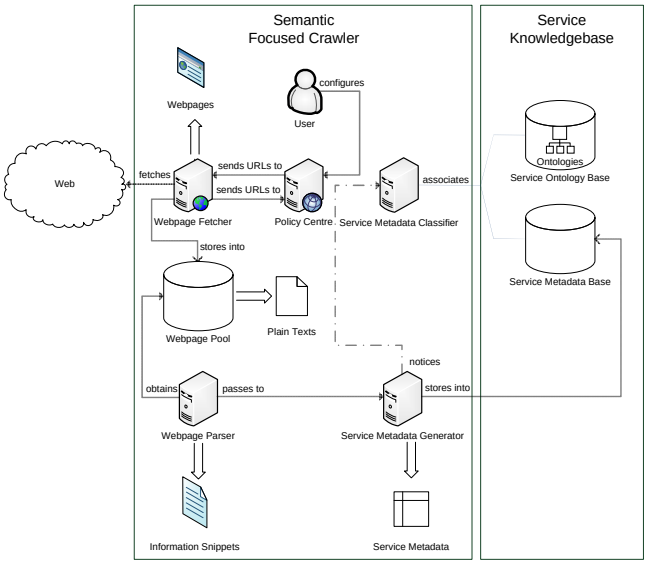

Fig. 1. Conceptual framework of the proposed semantic focused crawler.

Webpage Parser, Service Metadata Generator and Service Metadata Classifier, and a repository – Webpage Pool. Each is described as follows:

Webpage Fetcher. Its function is to selectively download webpages by using the multi-threading technology which allows several webpage fetching processes to work concurrently. Given a list of Uniform Resource Locaters (URLs), the Webpage Fetcher can download the webpages linked by the URLs. In addition, the Webpage Fetcher can extract the URLs from the downloaded webpages, and send them to the Policy Centre for further analysis.

Policy Centre. The function of the Policy Centre is to control the behavior of the Webpage Fetcher by configuring several policies introduced as follows:

- Selection Policy is defined in order to regulate the initial URLs that a Webpage Fetcher needs to visit, and to define the fetching boundary in order to ensure that a Webpage Fetcher does not escape from a configured website. Additionally, the policy can also be used to rank the priorities of visiting URLs, by means of setting up a set of heuristic rules for analyzing URL annotations (e.g., URLs annotated with "*next page*" or page numbers etc. may have higher visiting priorities).
- Maximum Visiting Policy is designed to regulate the maximum depth to which a website can be explored, in order to avoid overloading a Webpage Fetcher.
- Parallelization Policy is used to coordinate the multi-threading fetching processes. The main task is to distribute the initial visiting URLs to each process and, by means of tracking the visited URLs, to avoid repetitively visiting the same URLs.

Webpage Pool. Webpage Pool is designed to store the web documents downloaded by the Webpage Fetcher. Here, all the embedded webpage mark-up language tags, such as Hypertext Mark-up Language (HTML) tags and Extensible Mark-up Language (XML) tags, and scripting language tags, such as JavaScript tags, are removed from the webpages. As a result, webpages are stored in the form of plain texts.

Webpage Parser. The task of the Webpage Parser is to extract meaningful information snippets from the web documents stored in the Webpage Pool. This is realized by following a set of heuristic rules for the text processing. In order to deal with the information heterogeneity in the large number of webpages, we define heuristic rules by referring to actual webpage layouts in websites and a general service metadata format, since service information in a website usually maintains a consistent style.

Service Metadata Generator. Service Metadata Generator is employed to produce metadata by annotating the information snippets obtained by the Webpage Parser with the ontology mark-up languages. The annotation process follows a general service metadata format, which will be discussed in the next section.

Service Metadata Classifier. Service Metadata Classifier has the mission of employing structured domain knowledge to classify the generated service metadata, by means of associating the metadata with predefined service domain ontology concepts. The association is based on the similarities between the metadata and each service concept. There is a specially designed mathematical model for computing the similarities, which will be discussed in Section 3C.

The Service Knowledgebase is primarily comprised of two components – a **Service Metadata Base** and a **Service Ontology Base**. The former is designed to store the generated service metadata, and the latter is employed to store the predefined service domain ontologies. In Digital Ecosystems, a service domain ontology is a hierarchy of service concepts, in which each concept is an abstraction of the service entities that share some common features. The concepts are related by the concept-subconcept relationship, in which a concept is the abstraction of its subconcepts. For the purposes of metadata classification, only the bottom-level service concepts have the privilege of associating with service metadata, as a higher-level concept comprises all the metadata associated by its subconcepts.

Therefore, the whole working process of the system is completed by the collaboration between the semantic focused crawler and the Service Knowledgebase, which can be described as follows:

Step 1. Before the crawler starts to work, users need to configure the initial URLs of visiting websites (usually the websites' domain names), and the depth of exploring the websites in the Policy Centre. Once the configuration has been completed, the Policy Centre will send the commands to the Webpage Fetcher for the webpage crawling.

Step 2. The Webpage Fetcher will start to obtain webpages after it receives the URL list. Once the Webpage Fetcher has downloaded a webpage, it will extract the URLs in the webpage and send them to the Policy Centre for further analysis. The webpage will be sent to the Webpage Pool for storage purposes.

Step 3. When the Policy Centre receives the URLs from the Webpage Fetcher, it will determine whether or not they are within the crawling boundary, by analyzing their domain names. After that, the Policy Centre will rank the URLs by their visiting priorities and send them back to the Webpage Fetcher. Steps 2 and 3 are a recursive process until the user-defined website exploration depth has been reached.

Step 4. Once a webpage has been passed to the Webpage Pool, all its embedded tags will be removed and the webpage will be stored in the form of plain texts.

Step 5. The Webpage Parser will obtain the processed webpage information from the Webpage Pool, extract the meaningful information snippets from each webpage, and pass them to the Service Metadata Generator.

Step 6. The Service Metadata Generator will annotate the delivered information snippets with the ontology mark-up languages, in order to create service metadata. The service metadata will then be stored into the Service Metadata Base. In addition, once a metadata has been generated, the Service Metadata Generator will send a message to the Service Metadata Classifier.

Step 7. On receiving the message from the Service Metadata Generator, the Service Metadata Classifier will compute the similarities between the generated metadata and each bottom-level ontology concept of a compatible ontology. If a similarity is above a threshold value, the corresponding concept can be regarded as being relevant to the metadata. Then the Service Metadata Generator will associate the metadata with the concept. The technical details regarding the similarity computation and the association will be discussed in the next two sections.

B. General Service Metadata Format and General Service Concept Format for Metadata-Concept Association

As introduced previously, the Service Metadata Generator needs to follow a general service metadata format in order to annotate information snippets into metadata. An example of the service metadata format coded in OWL is represented in Fig. 2

Fig. 2 shows that each service metadata consists of at least two primary properties, which are defined as follows:

metadataDescription is a datatype property of a service metadata, which refers to the description of a service entity. Because a service entity may have more than one description in the real environment, this property can be extended in accordance with the number of information snippets that are relevant to the service descriptions. Moreover, this property can be employed for the similarity computation between the service metadata and service concept, which will be discussed in the next section.

linkedConcepts is an object property of a service metadata, which is used to store the URIs of the associated concepts. The Service Metadata Classifier utilizes this property for the metadata-concept association, which will be discussed in the next section.

In correspondence with the general service metadata format, when defining a service concept within the Service Ontology Base, a simple and extensible general service concept format needs to be followed, which consists of the two primary properties introduced as follows:

conceptDescription is a data type property of a service concept, which refers to the predefined contexts that define and describe the concept. This property is the counterpart of the metadataDescription property of the service metadata. Similarly, this property can also be extended to more than one according to the actual definitions of service concepts. This property will be incorporated with the metadataDescription property for the forthcoming metadata-concept similarity computation.

linkedMetadata is an object property of concept, which is used to store the URIs of metadata semantically similar to the concept. This property is the counterpart of the linkedConcepts property of a service concept, which will be used for the metadata-concept association.

An example of the general service concept format is represented in Fig .3.

It is noted that the general format of the service metadata and service concepts are flexible and therefore can be modified according to different service domain knowledge.

C. Metadata-Concept Similarity Computation and Association

As introduced in Section 3A, when a service metadata is generated, the Service Metadata Classifier will compute the similarity between the metadata and each service concept of a compatible service ontology. Here, we design an Extended Case-based Reasoning (ECBR) model to achieve this goal, which can be represented by Equations (1) and (2) as follows:

$$sim(S,C) = \max_{sd_i \in S}\left(\max_{cd_j \in C}\left(\sum_{t_{jh} \in cd_j} \frac{f(sd_i, t_{jh})}{l_{cd_j}}\right)\right) \quad (1)$$

```
<owl:Class rdf:ID="Service_Metadata"/>
  <owl:DatatypeProperty
rdf:ID="serviceDescription_1">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#strin
g"/>
    <rdfs:domain rdf:resource="#Service_Metadata"/>
  </owl:DatatypeProperty>
……
  <owl:DatatypeProperty
rdf:ID="serviceDescription_N">
……
  <owl:ObjectProperty rdf:about="#linkedConcepts">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="linkedMetadata"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Service_Metadata"/>
    <rdfs:range rdf:resource="#Service_Concept"/>
  </owl:ObjectProperty>
</owl:Class>
```

Fig. 2. An example of the general service metadata format coded in OWL.

```
<owl:Class rdf:ID="Service_Concept"/>
  <owl:DatatypeProperty rdf:ID="conceptDescription_1">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"
/>
    <rdfs:domain rdf:resource="#Service_Concept"/>
  </owl:DatatypeProperty>
……
  <owl:DatatypeProperty rdf:ID="conceptDescription_M">
……
  <owl:ObjectProperty rdf:ID="linkedMetadata">
    <owl:inverseOf rdf:resource="#linkedConcepts"/>
    <rdfs:domain rdf:resource="#Service_Concept"/>
    <rdfs:range rdf:resource="#Service_Metadata"/>
  </owl:ObjectProperty>
</owl:Class>
```

Fig. 3. An example of the general service metadata format coded in OWL.

$$f(sd_i, t_{jh}) = \begin{cases} 1 \ if \ \ t_{jh} \in sd_i \\ 0 \ otherwise \end{cases} \quad (2)$$

where S is a service metadata, C is a service concept, sd_i is the value of a serviceDescription property of the metadata S, cd_i is the value of a conceptDescription property of the concept C, t_{jh} is a term that appears in the conceptDescription cd_j and l_{cdj} is the total number of terms that appears in the conceptDescription cd_j.

The principle of the ECBR model is to match the features of objects in order to find the maximum similarity between two objects. In this research, the ECBR model aims at comparing the terms in the metadataDescription properties of a service metadata and the terms in the conceptDescription properties of a service concept, in order to find the maximum similarity between the two groups of properties, which is represented by Fig. 4.

The Extended CBR model is simple to implement, and it does not need to generate index terms before matching, which saves pre-processing time. It can also adapt to the frequent update of the ontologies, which often need the regenerating of index terms in most of the index term-based algorithms. Since the model is independent of index terms, it does not have the issue of index term dependency.

After the metadata-concept similarity has been obtained, the followed association process can be represented by Fig. 5.

IV. System Evaluation

In this section, firstly, we will introduce seven performance indicators from the traditional information retrieval field: secondly, we will implement a series of experiments to evaluate the conceptual framework of our semantic focused crawler.

A. Performance Indicators

In order to thoroughly evaluate the performance of our proposed semantic focused crawlers, we employ seven indicators from the field of information retrieval: Harvest Rate, Precision, Mean Average Precision, Recall, Harmonic Mean,

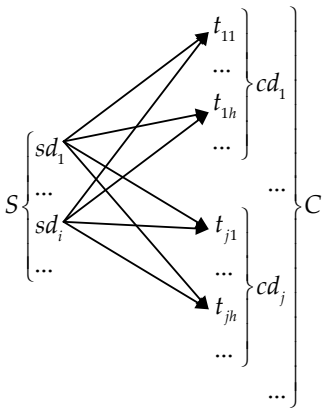

Fig. 4. The matching process between a service metadata and a service concept.

Input: S is a service metadata, $C = (C_1, C_2 \ldots C_m)$ is the whole collection of the bottom-level service concepts in a service ontology.

Procedure:

Fetch the serviceDescription values of S and store them in a bi-dimensional array sd.
For j = 1 **to** m
  Fetch the conceptDescription values of C_j and store them in a bi-dimensional array cd_j.
  Compute the similarity between sd and cd_j by the ECBR and store the value in s_{ij}.
  If s_{ij} > threshold **then**
    Assign URI of C_j into the linkedConcepts property of S.
    Assign URI of S into the linkedMetadata property of C_j.
  End if
End for

Fig. 5. The metadata-concept association process.

F-measure, and Fallout Rate. Here we provide their definitions for the forthcoming experiments.

Harvest Rate in the information retrieval is used to measure the crawling ability of a crawler. In this experiment, Harvest Rate is the proportion of associated metadata in the whole collection of generated metadata, which can be mathematically represented as:

$$\text{Harvest Rate} = \frac{\text{Number of associated metadata}}{\text{Number of generated metadata}} \quad (3)$$

Precision in the information retrieval is used to measure the preciseness of a retrieval system [26]. In this experiment, Precision for a single concept – Precision(S), is the proportion of the relevant metadata associated by this concept in all the metadata associated by this concept, which can be mathematically represented as:

$$\text{Precision(S)} = \frac{\text{Number of associated and relevant metadata}}{\text{Number of associated metadata}} \quad (4)$$

With regard to the whole collection of concepts in an ontology, the Precision(W) is the sum of the Precision for each concept normalized by the number of concepts in the collection, which can be represented as:

$$\text{Precision(W)} = \frac{\sum_{i=1}^{n} \text{Precision}(S_i)}{n} \quad (5)$$

where n is the number of concepts in the collection.

Before we introduce the definition of Mean Average Precision, the concept of Average Precision should be defined. Average Precision for a single concept – Average Precision(S), is the average of precision values after truncating a ranked

metadata list associated by this concept after each of the relevant metadata for this concept [26]. This indicator emphasizes the earlier return of more relevant metadata, which can be represented as:

$$\text{Average Precision(S)} = \frac{\text{Sum(Precision @ Each relevant metadata in the list)}}{\text{Number of associated and relevant metadata in the list}} \quad (6)$$

Mean Average Precision measures how quickly and precisely a crawler works, being the average of the average precision values for the collection of concepts in an ontology, which can be represented as:

$$\text{Mean Average Precision} = \frac{\sum_{i=1}^{n} \text{Average Precision}(S_i)}{n} \quad (7)$$

where *n* is the number of concepts in the collection.

Recall in the information retrieval refers to the measure of effectiveness of a query system [26]. In this experiment, Recall for a single concept – Recall(S) is the proportion of the relevant metadata associated by this concept in all the relevant metadata of this concept in the collection of generated metadata, which can be represented as:

$$\text{Recall(S)} = \frac{\text{Number of associated and relevant metadata}}{\text{Number of relevant metadata}} \quad (8)$$

With regard to the whole collection of concepts in an ontology, the Recall(W) is the sum of the Recall for each concept normalized by the number of concepts in the collection, which can be represented as:

$$\text{Recall(W)} = \frac{\sum_{i=1}^{n} \text{Recall}(S_i)}{n} \quad (9)$$

where *n* is the number of concepts in the collection.

It is important to note that the number of relevant metadata can be determined only by a peer-reviewed method, as the estimation of relevance between metadata and concept requires detailed knowledge of all concepts and metadata in a Service Knowledgebase, which can only be manually implemented in the current situation.

Harmonic Mean in the information retrieval is used as an aggregated performance scale for a search system [26]. In this experiment, Harmonic Mean is the mean of Precision and Recall, which can be represented as:

$$\text{Harmonic Mean} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \quad (10)$$

When the Harmonic Mean reaches the highest, it means the integrated value between Precision and Recall reaches to the highest at the same time.

F-measure is another measure that combines precision and recall, and the difference is that users can specify the preference on Recall or Precision by configuring different weights [26]. In this experiment, we employ F-measure (β=2) that weights Recall twice as much as Precision, which is close to the fact that most search engines concern recall more than precision, as a result of most users' purposes in obtaining information [27]. The F-measure (β=2) can be represented below:

$$\text{F-measure } (\beta=2) = \frac{(1+\beta^2) \cdot \text{Precision} \times \text{Recall}}{\beta^2 \cdot \text{Precision+Recall}} = \frac{5 \times \text{Precision} \times \text{Recall}}{4 \times \text{Precision+Recall}} \quad (11)$$

All of the above indicators have the same limitation – they do not consider the number of non-relevant metadata in an associated metadata collection. Furthermore, if there is no relevant metadata in the associated collection, Recall cannot be defined. To solve this issue, we need another performance indicator – **Fallout Rate** [26]. In this experiment, Fallout Rate for a single concept – Fallout Rate(S) is the proportion of non-relevant concept associated by this concept in the whole collection of non-relevant metadata for this concept in the generated metadata, which can be represented as:

$$\text{Fallout Rate(S)} = \frac{\text{Number of associated and non-relevant metadata}}{\text{Number of non-relevant metadata}} \quad (12)$$

With regard to the whole collection of concepts, the Fallout Rate(W) is the sum of the Fallout Rate for each concept normalized by the number of concepts in an ontology, which can be represented as:

$$\text{Fallout Rate (W)} = \frac{\sum_{i=1}^{n} \text{Fallout Rate}(S_i)}{n} \quad (13)$$

where *n* is the number of concepts in the collection.

In contrast to other performance indicators, the lower the Fallout Rate value, the better is the crawler's performance.

Therefore, the following experiments will be implemented in relation to the seven performance indicators.

B. Experiments

As introduced previously, in the metadata-concept association process (Fig. 5), after the similarity between a metadata and a concept has been obtained, a threshold needs to be established in order to determine whether or not they are relevant. Hence, there are two primary objectives involved in the forthcoming experiments – apart from evaluating the overall performance of the proposed semantic focused crawler, an optimal threshold for the association process needs to be established.

Experimental environment setup

It is known that at least one service domain ontology needs to be installed into the Service Ontology Base before the semantic focused crawler can start working. Nevertheless, currently no such ontology is available. Therefore, we develop a transport service ontology in Protégé-OWL. The background knowledge for the ontology is obtained from the Wikipedia (http://en.wikipedia.org/) and Open Directory Project (http://www.dmoz.org/). The transport service ontology is a four-tier hierarchical structure which consists of 304 transport service concepts. Each concept is defined by a certain number of concept descriptions and the concepts are related by the concept-subconcept relationship. As discussed in Section 3A, only the bottom-level concepts have the property of linkedMetadata, which are allowed to associate with service metadata. The abbreviated view of the Transport Service Ontology can be viewed in Fig .6.

After we build a prototype of the semantic focused crawler in JAVA, we choose the Australian Yellowpages® website (http://www.yellowpages.com.au/) as the data source for crawling service metadata.

Experimental results

We run the semantic focused crawler prototype to download 2000 business webpages under the category of transport in the Australian Yellowpages® website, and the crawler generates 4243 service metadata in total. The crawler totally uses 216.92s to complete the whole process, which is 0.108s/webpage and 0.051s/metadata. These statistical data preliminarily prove its capability to deal with large-scale ontologies. The following investigations are conducted based on this crawling activity.

In order to obtain the optimal threshold for the crawler, we set the initial threshold at 0.5, and increase it by 0.05 each time. A series of experiments based on the seven performance indicators is implemented at each time of increment. The test results can be found in Fig. 7 to Fig. 13 (See Appendix).

Discussion

The discussion of the experimental results is comprised of two parts: 1) evaluating the performance of the semantic focused crawler prototype from the perspective of the seven performance indicators; 2) choosing an optimal threshold for non-relevant metadata filtering.

Harvest Rate. It can be observed that the crawler delivers an outstanding performance for associating metadata with concepts, as the overall trend of the curve is relatively stable and the lowest value is above 97%, which indicates that the effect of the changes to the threshold value is not too serious on the Harvest Rate. This experiment preliminarily proves the crawling ability of the semantic focused crawler.

Precision. The Precision increases from 13.11% to 86.63%, which indicates that the variation of the threshold value evidently impacts upon the Precision, especially in the initial interval (0.5 – 0.8). However, when the threshold value exceeds 0.8, the Precision remains relatively stable on the top. This is because the crawler filters few non-relevant metadata when the threshold value is higher than 0.8. This experiment proves that the crawler can work precisely when its threshold value is no lower than 0.8.

Mean Average Precision. Analogous to the curve of the Precision, the curve of the Mean Average Precision shows a similar trend, which continues to increase until the threshold value reaches 0.8. The only difference is that the top Mean Average Precision value is 0.9% lower than the top Precision value. This experiment proves that the crawler can work quickly and precisely when the threshold is no lower than 0.8.

Recall. It is observed that the trend of the Recall is similar to the trend of the Harvest Rate where the score continues to decrease as the threshold values increases. Nevertheless, the variation interval of the Recall is relatively narrow (1.25%). Therefore, it can be deduced that the Recall is not heavily affected by the threshold value. This is because the relatively higher threshold values do not affect the exclusion of the non-relevant metadata. Here the average score of the Recall of the crawler is higher than 98%, which preliminarily proves the effectiveness of the proposed crawler in this experiment.

Harmonic Mean. As introduced previously, the Harmonic Mean is an aggregated metric between the Precision and Recall. In the experiments above, we found that the Precision experiences a remarkable variation, in contrast to the relative stability of the Recall. Hence, the Precision significantly impacts upon the Harmonic Mean, and the curve of the

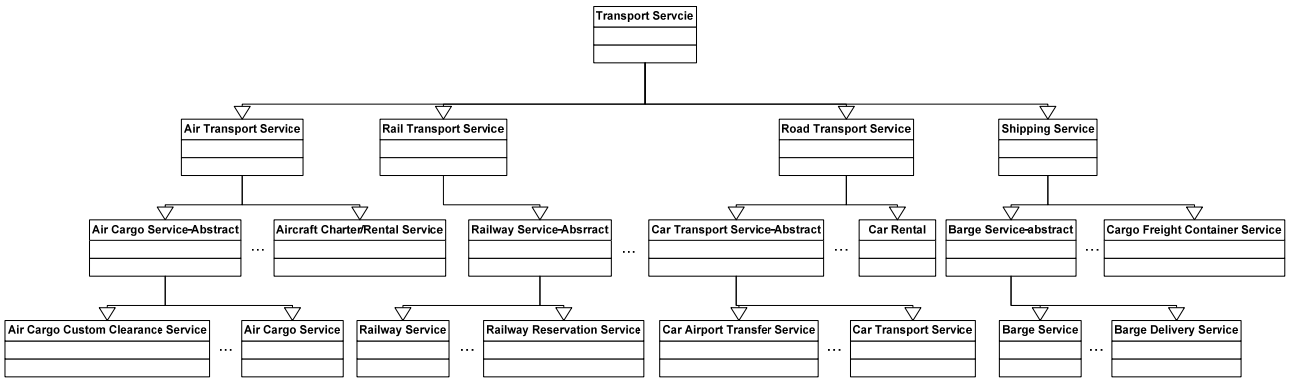

Fig. 6. An abbreviated view of the Transport Service Ontology.

Harmonic Mean is almost parallel to the curve of the Precision, which reaches to the top and remains steady when the threshold value is higher than 0.8. Due to the high performance of the Recall and the Precision, the top value of the Harmonic Mean is above 90%, which proves the comprehensive performance of our proposed methodology at this stage.

F-measure (β=2). Similar to the trend of the Harmonic Mean, the trend of the F-measure (β=2) also experiences a climbing process. As a result of the preference on the Recall that has a more outstanding performance than the Precision, the score of the F-measure (β=2) is higher than the simultaneous Harmonic Mean. Furthermore, this result proves that our crawler can fulfill the requirement of the recall-preferred search engines.

Fallout Rate. Since the higher threshold values serve as a barrier to more non-relevant metadata, the Fallout Rate is positively affected by the increase of the threshold value. Moreover, the change of the Fallout Rate is obvious, which varies from 5.31% to 0.07%. Moreover, when the threshold value reaches 0.8, the Fallout Rate is close to 0, which indicates that the higher threshold values can maintain the crawler's low fault rate.

As a conclusion, in terms of the analysis above, we find that, there are two primary types of trends in these figures. One of the trends is that the performance of the crawler declines when the threshold value increases. This group of indicators includes the Harvest Rate and Recall. However, these declines are in relatively small intervals (2.08% for the Harvest Rate and 1.25% for the Recall), which cannot heavily influence the performance of the crawler on these parameters. Another group of indicators includes the Precision, Mean Average Precision, Harmonic Mean and F-measure (β=2) and Fallout Rate. More interestingly, if we rotate the curve of the Fallout Rate 180 degrees, we find that its curve is nearly parallel to the curves of the other indicators from this group, namely, the increase of the threshold value can enhance the performance of the crawler on this group of metrics, and this sort of impact is extremely distinct. By analyzing Equations (4) and (8), we conclude that the impact of the increase of the threshold value on the number of associated metadata is heavier than it is on the number of associated and relevant metadata. In other words, the increase of threshold value heavily impacts upon the reduction of the number of associated and non-relevant metadata, which can also be observed from the curve of Fallout Rate (Fig. 13). Furthermore, in terms of the two aggregated metrics – the Harmonic Mean and F-measure (β=2), we can draw the second conclusion – the relatively higher threshold values are beneficial to the overall performance of the crawler. For instance, at the relatively higher threshold values (e.g., 0.8), the performance of the crawler is more convincing on all of the seven indicators (Harvest Rate: 97.07%, Precision: 86.63%, Mean Average Precision: 85.61%, Recall: 98.01%, Harmonic Mean: 91.91, F-measure (β=2): 95.50%, Fallout Rate: 0.07%).

For the second task, it is observed that 0.8 is a milestone for all the indicators above. At this point, the scores of the second group of indicators are all in their relatively higher values; on the other hand, the first group of indicators are not heavily affected by the variation and still keep their outstanding performance. After this point, the impact of the increase of the threshold value on all the indicators is nearly invisible and the indicators all remain in a relatively high-level status, especially for the two aggregated indicators – the Harmonic Mean and F-measure (β=2), which have more weights on deciding the optimal threshold value. Based on this consequence, we determine that 0.8 is an optimal threshold value for the semantic focused crawler in this experiment.

V. Conclusion and Future Work

In this paper, we have designed a conceptual framework for a semantic focused crawler, which combines the speciality of ontology-based metadata classification from the ontology-based focused crawlers and the speciality of metadata abstraction from the metadata abstraction crawlers, in order to achieve the goal of automatic service discovery, annotation and classification in the Digital Ecosystems environment. The main functions of the crawler include: discovering service information from the web; parsing, annotating, and storing service information; classifying the annotated service information based on specific service domain knowledge. We have defined a general format for service metadata and service concept, which enables the function of similarity computation and the association between metadata and concepts. In addition, an ECBR model is designed in order to compute the similarity between metadata and concepts. Subsequently, we built a prototype in JAVA and developed a transport service ontology for the metadata classification. In order to evaluate the conceptual framework, we conducted a series of experiments based on seven performance indicators from the traditional information retrieval field. From the experiments, we have drawn two conclusions: 1) The increase of the threshold value can reduce the amount of associated and non-relevant metadata, and 2) the relatively higher threshold values can benefit the overall performance of the crawler.

Our further research will concentrate on the following four aspects:

- studying the effects of ontology evolution on the ontology-based metadata classification methodology;
- designing ontologies for other service domains in order to obtain wider searching scope of the semantic focused crawler;
- testing the crawler by using other data sources; and
- revising the ECBR algorithm to achieve better performance in metadata classification.

References

[1] J. F. Moore, "Predators and prey: a new ecology of competition," *Harvard Business Review* vol. 71, pp. 75-86, 1993.

[2] E. Chang and M. West, "Digital Ecosystem - A next generation of the collaborative environment," in *iiWAS2006*, Yogyakarta, 2006.

[3] H. Boley and E. Chang, "Digital Ecosystems: Principles and semantics," in *IEEE DEST 2007*, Cairns, 2007, pp. 398-403.
[4] T. Heistracher, T. Kurz, C. Masuch, P. Ferronato, M. Vidal, A. Corallo, G. Briscoe, and P. Dini, "Pervasive service architecture for a digital business ecosystem," in *18th European Conference on Object-Oriented Programming (ECOOP 2004)*, Oslo, 2004, pp. 71-80.
[5] Z. Stejic, Y. Takama, and K. Hirota, "Relevance feedback-based image retrieval interface incorporating region and feature saliency patterns as visualizable image similarity criteria," *IEEE Transactions on Industrial Electronics,* vol. 50, pp. 839-852, 2003.
[6] J. L. M. Lastra and M. Delamer, "Semantic web services in factory automation: fundamental insights and research roadmap," *IEEE Transactions on Industrial Informatics,* vol. 2, pp. 1-11, 2006.
[7] P. Malone, "DE services in Ecosystem Oriented Architectures," in *Digital Business Ecosystems*, F. Nachira, P. Dini, A.Nicolai, M. L. Louarn, and L. R. Lèon, Eds.: European Commission, 2007.
[8] H. Dong, F. K. Hussain, and E. Chang, "A service search engine for the industrial digital ecosystems," *IEEE Transactions on Industrial Electronics,* to be published. DOI: 10.1109/TIE.2009.2031186
[9] W. Hu, G.-P. Liu, D. Rees, and Y. Qiao, "Design and implementation of Web-based control laboratory for test rigs in geographically diverse locations," *IEEE Transactions on Industrial Electronics,* vol. 55, pp. 2343-2354, 2008.
[10] A. Leva and F. Donida, "Multifunctional remote laboratory for education in automatic control: the Crautolab experience," *IEEE Transactions on Industrial Electronics,* vol. 55, pp. 2376-2385, 2008.
[11] J. Prieto-Blazquez, J. Arnedo-Moreno, and J. Herrera-Joancomarti, "An integrated structure for a virtual networking laboratory," *IEEE Transactions on Industrial Electronics,* vol. 55, pp. 2334-2342, 2008.
[12] T. T. Quan, S. C. Hui, and A. C. M. Fong, "Automatic fuzzy ontology generation for semantic help-desk support," *IEEE Transactions on Industrial Informatics,* vol. 2, pp. 155-164, 2006.
[13] Y. Takama and S. Hattori, "Mining association rules for adaptive search engine based on RDF technology," *IEEE Transactions on Industrial Electronics,* vol. 54, pp. 790-796, 2007.
[14] S.-C. Wang and Y.-H. Liu, "Software-reconfigurable e-learning platform for power electronics courses," *IEEE Transactions on Industrial Electronics,* vol. 55, pp. 2416-2424, 2008.
[15] A. C. Weaver and M. W. Condry, "Distributing Internet services to the network's edge," *IEEE Transactions on Industrial Electronics,* vol. 50, pp. 404-411, 2003.
[16] M. Wu, J.-H. She, G.-X. Zeng, and Y. Ohyama, "Internet-based teaching and experiment system for control engineering course," *IEEE Transactions on Industrial Electronics,* vol. 55, pp. 2386-2396, 2008.
[17] H. Dong, F. K. Hussain, and E. Chang, "State of the art in semantic focused crawlers," in *ICCSA 2009*, Yongin, 2009, pp. 890–904.
[18] H. Dong, F. K. Hussain, and E. Chang, "A survey in semantic web technologies-inspired focused crawlers," in *Third International Conference on Digital Information Management 2008 (ICDIM 2008)*, East London, 2008, pp. 934-936.
[19] M. Yuvarani, N. C. S. N. Iyengar, and A. Kannan, "LSCrawler: a framework for an enhanced focused web crawler based on link semantics," in *the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI'06)*, 2006, pp. 794-800.
[20] J. Tane, C. Schmitz, and G. Stumme, "Semantic resource management for the web: an elearning application," in *WWW2004*, New York, 2004.
[21] S. Ganesh, M. Jayaraj, V. Kalyan, and G. Aghila, "Ontology-based web crawler," in *the International Conference on Information Technology: Coding and Computing (ITCC'04)*, Las Vegas, 2004, pp. 337-341.
[22] M. Halkidi, B. Nguyen, I. Varlamis, and M. Vazirgiannis, "THESUS: organizing web document collections based on link semantics," *The VLDB Journal,* vol. 12, pp. 320–332, 2003.
[23] H. Dong, F. K. Hussain, and E. Chang, "State of the art in metadata abstraction crawlers," in *2008 IEEE International Conference on Industrial Technology (IEEE ICIT 2008)*, Chengdu, 2008, pp. 1-6.
[24] E. Francesconi and G. Peruginelli, "Searching and retrieving legal literature through automated semantic indexing," in *ICAIL '07*, Standford, 2007, pp. 131-138.
[25] C. L. Giles, Y. Petinot, P. B. Teregowda, H. Han, S. Lawrence, A. Rangaswamy, and N. Pal, "eBizSearch: A niche search engine for e-business," in *SIGIR'03*, Toronto, 2003, pp. 213-214.
[26] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. New York: Addison-Wesley, 1999.
[27] L. T. Su, "The relevance of recall and precision in user evaluation," *Journal of the American Society for Information Science and Technology,* vol. 45, pp. 207-217, 1999.

Hai Dong (S'08) received a Bachelor of Management Degree in Information Management in 2003 from Northeastern University, Shenyang, China. In 2006, he received his Master of Commerce Degree in Information Technology from Curtin University of Technology.

He is currently a doctoral candidate in the Digital Ecosystems and Business Intelligence Institute, Curtin University of Technology. His research interest includes: ontology engineering, web semantics, information retrieval, digital ecosystems, semantic search, focused crawling, and service computing.

Farookh Khadeer Hussain received a Bachelor of Technology Degree in Computer Science and Computer Engineering. Later, he received his Master's Degree in Information Technology from La Trobe University in Melbourne, Australia. In 2006, he received a PhD in Information Systems from Curtin University of Technology, Perth, Australia.

He is a research fellow in the Digital Ecosystems and Business Intelligence Institute. His areas of active research are trust, reputation, trust ontologies, data modeling of public and private trust data. He works actively in the domain of making informed business decisions (business intelligence) through the use of trust and reputation technology. He is interested in the application of trust and reputation as a technology, as a business analysis and intelligence tool, and the applications of trust and reputation to various domains. His other areas of research interests are in Web 2.0, social networks and mashup.

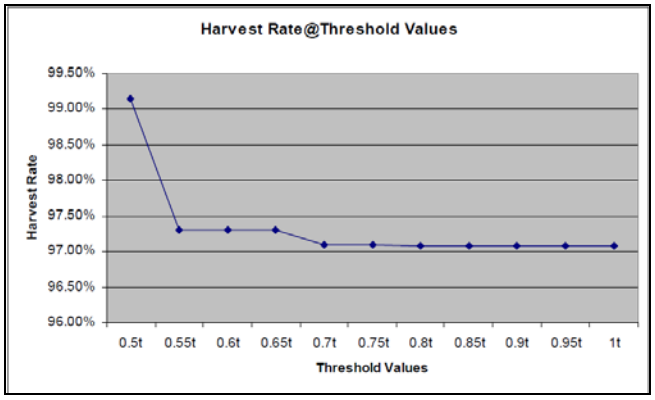

Fig. 7. Harvest Rate @ threshold values.

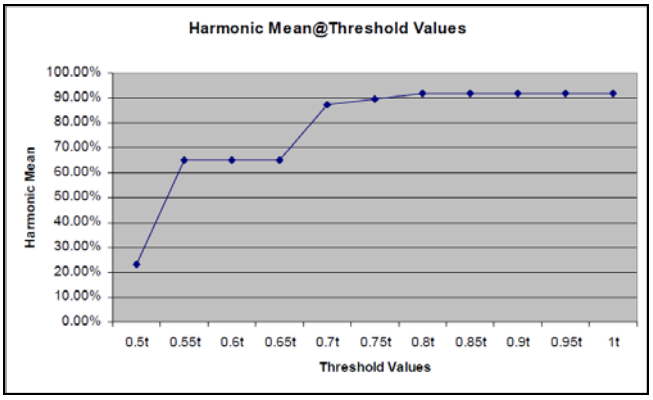

Fig. 11. Harmonic Mean @ threshold values.

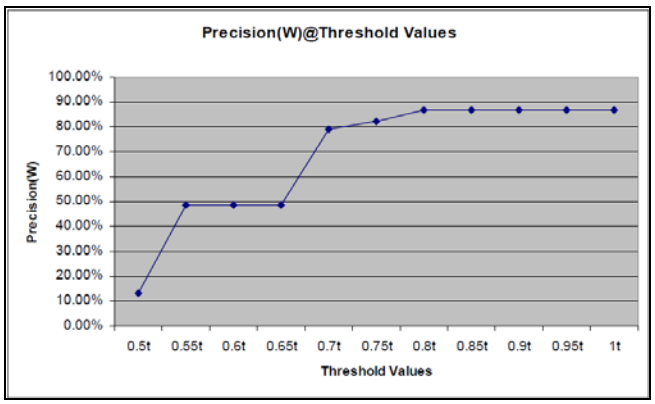

Fig. 8. Precision(W) @ threshold values.

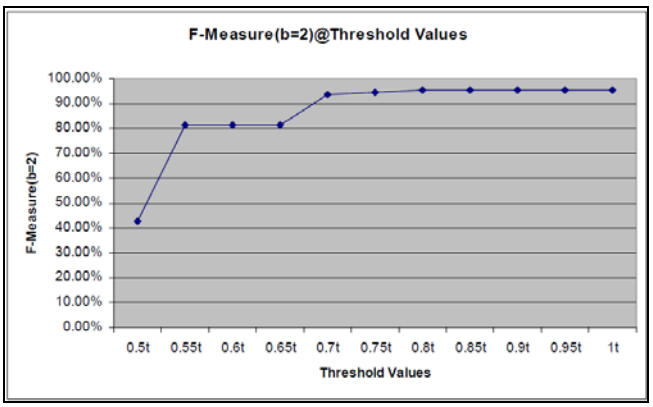

Fig. 12. F-measure(β=2) @ threshold values.

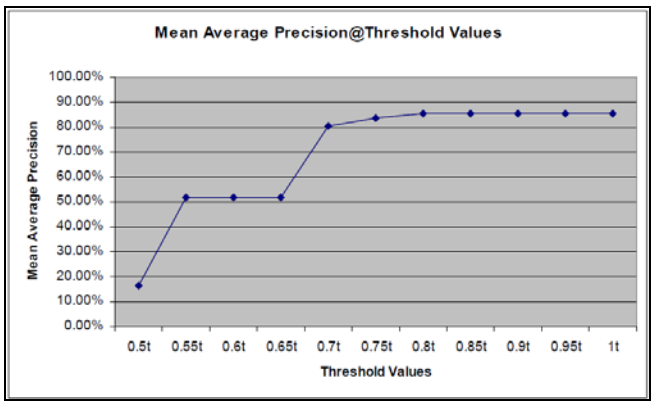

Fig. 9. Mean Average Precision @ threshold values.

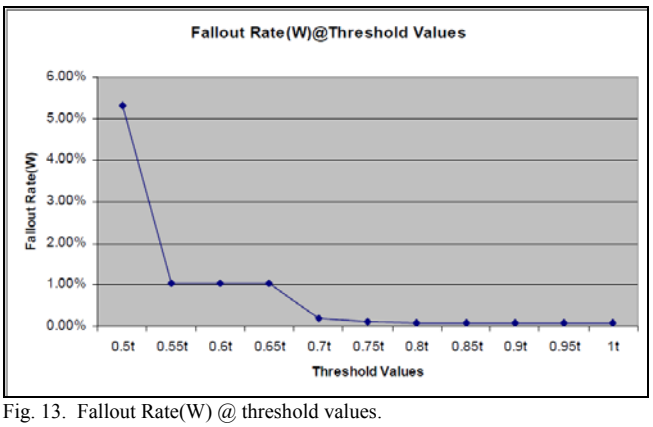

Fig. 13. Fallout Rate(W) @ threshold values.

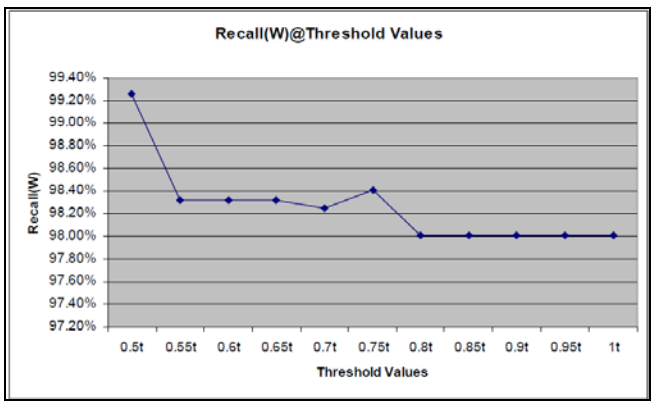

Fig. 10. Recall(W) @ threshold values.