A Semantic Crawler Based on an Extended CBR Algorithm

Hai Dong, Farookh Khadeer Hussain, Elizabeth Chang

Digital Ecosystems and Business Intelligence Institute, Curtin University of Technology,
GPO Box U1987 Perth, Western Australia 6845, Australia
{hai.dong, farookh.hussain, elizabeth.chang}@cbs.curtin.edu.au

Abstract. A semantic (web) crawler refers to a series of web crawlers designed for harvesting semantic web content. This paper presents the framework of a semantic crawler that can abstract metadata from online webpages and cluster the metadata by associating them with ontological concepts. The clustering is based on a CBR algorithm which is adopted in the field of problem solving. We reveal the technical details with regard to ontological concept and metadata format, and the extended CBR algorithm. In addition, the system implementation and evaluation details are provided in detail, finalized by our conclusion and further works.

Keywords: semantic crawler, metadata abstraction, ontological concepts, extended CBR algorithm

1 Introduction

A semantic (web) crawler refers to a series of web crawlers designed for harvesting semantic web content [4]. The semantic web content is normally marked by the ontology mark-up languages (e.g. RDF, XML, OWL, and so forth). In this paper, we will present the conceptual model of a semantic crawler. This semantic crawler uses an extended CBR algorithm to associate the harvested metadata with ontological concepts, in order to realize the purpose of metadata clustering.

The rest of the paper is organized as follows: first of all, the system architecture of the whole crawling system is presented; then we provide the ontological concept and metadata format in the OWL format, followed by the introduction of the core part of the semantic crawler – a CBR algorithm and its extended version in the field of semantic information retrieval; following that, we reveal the implementation details; next, to evaluate the performance of the semantic crawler, we choose a benchmark from the traditional information retrieval evaluation methods, and present its variations and purposes in our testing environment; the testing results are then presented and analysed; the conclusion and further works are summarized in the final section.

2 System Architecture

The design of crawling system has two primary objectives as below:

- Generating metadata by extracting structured and meaningful information from downloaded webpages.
- Clustering metadata by adding references to ontological concepts.

The whole system consists of two main parts – a semantic crawler and a knowledge base, which will be introduced in detail in the following paragraphs (Fig. 1).

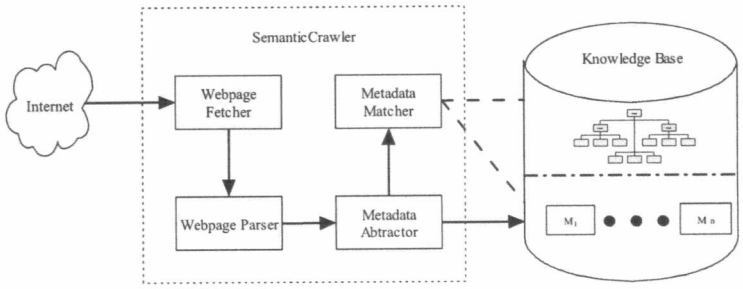

Fig. 1. System architecture

The semantic crawler architecture is composed of four agents, which are a Webpage Fetcher, a Webpage Parser, a Metadata Abstractor, and a Metadata Matcher. The function of each agent is described as follows:

- The Webpage Fetcher's mission is to selectively download webpages in a given website, by configuring the visiting URLs. Once the URL of a given website is passed to the Webpage Fetcher, it will visit and download the webpage located by the URL, and then analyze all hyperlinks, in order to choose further visiting webpages. This can be realized by setting up a set of webpage fetching rules for analyzing HTML hyperlink tags and their annotations (e.g., "next page", numbers, and so forth).
- The Webpage Parser's function is to parse the web documents in the downloaded webpages, and to filter meaningless information (e.g., hyperlinks, footnotes, unimportant HTML tags, and so on) in the parsed documents. Similar to the Webpage Fetcher, a set of webpage parsing rules are set up. According to the rules, the agent can parse the web documents based on the regulated HTML tags.
- The Metadata Abstractor's goal is to abstract the structured and meaningful information from the parsed web documents, and to form metadata based on these information. Similarly, the rules of information abstraction need to be configured, with the purpose of analyzing information from the downloaded webpages. These rules configure the HTML tags and annotations whose contexts may contain meaningful information. Based on the rules, the agent can extract the meaningful information, and treat them as the properties of metadata. By adding OWL tags, the metadata can be formed.

- The Metadata Matcher is used to cluster the metadata with predefined ontological concepts, by means of associating the semantically similar metadata with concepts. The semantic similarity between metadata and concepts are determined by an Extended CBR algorithm. The associating process is realized by inserting the URI of each side into a property of the other.

The knowledge base is used to store the predefined ontological concepts and metadata. The format of ontological concepts and metadata will be described in the next section.

Thus, the workflow of the whole system can be concluded as follows:

- First of all, the Webpage Fetcher will download a webpage according to a given URL. By means of the predefined webpage fetching rules, it will then extract the useful URLs from the webpage for the further visit. The above is a recursive process.
- For the downloaded webpages, the Webpage Parser will obtain all web documents from them, and filter the meaningless information according to the webpage parsing rules. The parsed documents will be passed to the Metadata Abstractor.
- After receiving the parsed documents, the Metadata Abstractor will extract the meaningful information upon the metadata abstracting rules, then form metadata by using these information. The metadata will then be stored into the knowledge base.
- Once a metadata is stored into the knowledge base, the Metadata Matcher will use the Extended CBR algorithm to determine the semantic similarity between the metadata and all ontological concepts in the knowledge base. If the metadata and a concept is similar, the URI of each side is then stored into the property of the other side. Therefore, the metadata are associated and clustered with ontological concepts.

3 Ontological Concept and Metadata Format

In this section, the format of ontological concepts and metadata will be represented in the form of OWL.

3.1 Ontological Concept Format

Each ontological concept has two basic properties (the number of properties can be extended according to the real environment), which are conceptDescription and linkedMetadata.

conceptDescription is a data type property of concept, which refers to the predefined contexts that define and describe an ontological concept. It normally consists of several descriptive phases, which can be used for computing semantic similarity values with metadata (will be talked in the next section).

associatedMetadata is an object property of concept, which is used to store the URIs of semantic similar metadata to the concept.

The OWL code of ontological concept format is as below:

```
<owl:Class rdf:ID="Concept"/>
  <owl:DatatypeProperty rdf:ID="conceptDescription">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/
>
    <rdfs:domain rdf:resource="#Concept"/>
  </owl:DatatypeProperty>
  <owl:ObjectProperty rdf:ID="associatedMetadata">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="associatedConcepts"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Concept"/>
    <rdfs:range rdf:resource="#Metadata"/>
  </owl:ObjectProperty>
```

3.2 Metadata Format

Each metadata has two primary properties (also can be extended), which are metadataDescription and associatedConcepts.

metadataDescription is a data type property of metadata, which refers to the description of a metadata. The content of this property is formed by the Metadata Abstractor, by extracting meaningful information from webpages. Similar to the counterpart in concepts, this property is also used to compute similarity values between metadata and concepts.

associatedConcepts is an object property of metadata, which is used to store the URIs of associated concepts. This property is the inverse of the associatedMetadata property in concepts. In other words, if a metadata stores a concept's URI in the associatedConcepts property, the concept must automatically have the metadata's URI in its associatedMetadata property.

The OWL code of metadata format is as below:

```
<owl:Class rdf:ID="Metadata"/>
  <owl:DatatypeProperty rdf:ID="metadataDescription">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/
>
    <rdfs:domain rdf:resource="#Metadata"/>
  </owl:DatatypeProperty>
  <owl:ObjectProperty rdf:about="#associatedConcepts">
    <owl:inverseOf rdf:resource="#associatedMetadata"/>
    <rdfs:domain rdf:resource="#Metadata"/>
    <rdfs:range rdf:resource="#Concept"/>
  </owl:ObjectProperty>
```

4 Extended Case-Based Reasoning Algorithm

In this section, we will introduce the case-based reasoning algorithm and its extended version adopted in the semantic crawler.

4.1 Case-Based Reasoning (CBR) Algorithm

CBR model is used to retrieve and reuse the existing problem solutions for emerging problems, which has four sub-processes as below [1]:

Retrieve: a new problem is matched with cases in database.

Reuse: if there are matched cases, the solutions to the retrieved cases are reused as the solutions of the emerging problem.

Revise: if the retrieved cases cannot completely match the problems, the solutions to the problem need to be revised.

Retain: the new case, incorporating with both problems and solutions, is stored in database.

Every feature extracted from incident reports is awarded an equal weight. Every feature in a new incident is compared with the corresponding feature in each of the other incidents. If the features match, a score of 1 is awarded. If the features do not match, a score of 0 is awarded. A similarity score is calculated by:

1. Finding the sum of the matching features;
2. Dividing this sum by the number of features contained in the incident, as in the equation (1) below:

$$sim(T,S) = \frac{\sum_{i=1}^{n} f_i(T_i, S_i)}{n} \tag{1}$$

Then a threshold is set up to determine whether the two incidents are matched or not.

4.2 Extended CBR Algorithm

Based on the principle of CBR algorithm, we design an Extended CBR algorithm, in order to apply it in the field of information retrieval. The Extended CBR algorithm is used to compute the similarity values between metadata and concepts, by mutually matching the contents of their metadataDescription and conceptDescription properties. If the similarity values are above a predefined threshold, the metadata and concept are determined to be associated.

The pseudocode of Extended CBR algorithm is shown as below:

```
Input: M = (k1, k2...km), where M is the description of a
metadata consisting of a group of keywords k; C = (c1,
```

```
c2...cn), where C is an array of concepts in KB. c = (d1,
d2...dk), where d is a concept description of concept c.

Output: C`, where C` is an array of selected concepts
based on their semantic similarity values with the
metadata M.

Algorithm:

BEGIN
  SET C` TO NULL;
  FOR i = 1 TO n
    SET sim TO 0
    FOR j = 1 TO k
      SET aj TO 0
      FOR h = 1 TO m
        IF kh in dj THEN
          ADD 1 TO aj
        END IF
      END FOR
      SET aj TO aj / length of dj
    ENE FOR
    CHOOSE the maximum aj
    SET sim TO aj
    IF sim >= threshold THEN
      PUT ci INTO C`
    END IF
  END FOR
END
```

The Extended CBR model is very simple to implement, and it does not need to generate index terms before matching, which saves the preprocessing time. It also can adapt to the flexibility of concepts that often needs regenerating index terms in most of index term-based algorithms. Since the Algorithm is independent of index terms, it does not have the issue of index term independency.

5. System Implementation and Evaluation

In this section, we will implement the semantic crawling system, and make evaluations for its performance.

5.1 System Implementation

According to the system architecture, the implementation can be divided into two parts – knowledge base and semantic crawler. The knowledge base is built in Protégé-OWL; the semantic crawler is realized in JAVA.

5.2 Crawler Benchmarks

To evaluate the performance of our semantic crawler, we choose a benchmark – precision, and then perform a series of experiments based on it.

Precision for a single concept is the proportion of associated, at the same time, and semantically similar metadata in all associated metadata to the concept, which can be represented as below:

$$\text{Precision(S)} = \frac{\text{number of associated and semantically similar metadata}}{\text{number of associated metadata}}$$

With regard to the whole collection of concepts, the whole precision is the sum of precision for each concept normalized by the number of concepts in the collection, which can be represented in equation (2) below:

$$\text{Precision(W)} = \frac{\sum_{i=1}^{n} \text{Precision}(S_i)}{n} \quad (2)$$

The purpose of precision is to test the effectiveness of the semantic crawler.

5.3 Experiment

As mentioned before, after the similarity values between a concept and a metadata is obtained, a predefined threshold is used to determine whether the concept and metadata should be associated or not. To obtain the most proper threshold value, our evaluation concentrates on testing the benchmark along with different threshold values.

To test the crawler, we create a transport service ontology with 262 ontological concepts. Then we use our semantic crawler to crawl 400 webpages under the category of transport in the Australian Yellowpages® website. From the 400 webpages, our crawler abstracts 736 metadata in total. The benchmark results are shown in Fig. 2.

From the precision @ threshold figure (Fig. 2), it is observed that the rise of precision is analogous to the rise of threshold – the precision quickly jumps from 15.86% to 93.53% when the threshold rises from 0.5 to 1 (totally 11 values). The precision reaches the top point and keeps steady after the threshold value reaches to 0.8. It is recommended that the threshold must be kept at 0.7 at least, since the precision keeps in a good performance (>= 80%) in this condition.

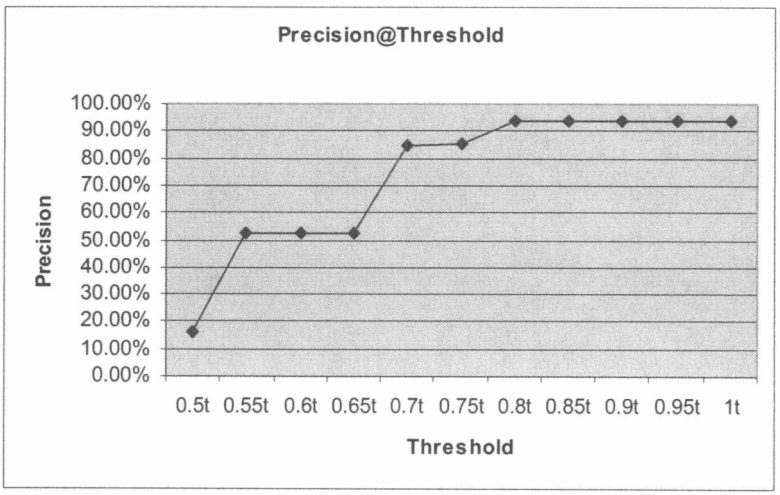

Fig. 2. Precision @ threshold

By means of observing the benchmark's result, it is concluded that 0.8 could be the most proper threshold value for the semantic crawler, since the precision level jumps 77.67%, compared with its value when the threshold is 0.5. The precision is above 90% in this point, which is a convincing performance in the present stage.

6 Related Works

Generally the semantic crawlers can be divided into two categories. The first category of crawlers is the ones that directly harvest semantic web documents from the internet. Another category is the ones that abstract semantic web documents from the normal web documents – "metadata abstraction crawler" [5].

SWoogle Crawler belongs to the first category, which can harvest, parse and analyze semantic web documents or semantic web information pieces embedded in web documents [3]. A SWoogle Analyzer is used to analyze the content of harvested semantic web documents for indexing and ranking purpose.

Slug is another example of the first category, which is a project for harvesting RDF documents from the internet [4]. It is realized in JENA API, and no performance evaluation details are provided by this project team.

For the metadata abstraction crawlers, the most typical example is Ontobroker. Ontobroker is a crawling system designed with the purpose of extracting, reasoning and generating RDF-annotated metadata [2]. Here an Ontocrawler is used to extract the formal knowledge from HTML web pages. Two different approaches are implemented here. For similarly structured HTML files, a wrapper is used to generate their formal descriptions, by means of referring to an ontology in a knowledge base; for the specially structured HTML files, an annotation language is used.

Handschuh and Staab design a framework of metadata creator – CREAM. A RDF crawler is utilized to find references for created metadata, with the purpose of avoiding duplication [6]. In the CREAM, when the metadata creator wants to find whether an instance already exists or not, the RDF crawler retrieves the instance from the local knowledge base, which stores the RDF files harvested from the semantic web [7]. If a URI with regards to the instance is returned by the RDF crawler, the creator will then be aware that the relational metadata is created [8].

Overall speaking, the semantic crawler's research is still in the beginning stages, and currently not too many semantic crawlers are found. Most semantic crawlers do not provide their evaluation details [5]. However, along with the increasing popularity and maturity of semantic web technologies, it is reasonably believed that the semantic crawler research will soon step into a new era.

7 Conclusions and Further Works

In this paper, we present a semantic crawler based on the extension of the CBR algorithm into the field of semantic information retrieval. The whole crawling system mainly contains two main parts – a semantic crawler and a knowledge base. The semantic crawler consists of four agents – a Webpage Fetcher, a Webpage Parser, a Metadata Abstractor and a Metadata Matcher. The Webpage Fetcher is responsible for downloading webpages from the internet; the Webpage Parser is adopted for parsing web documents and filtering meaningless information from the downloaded webpages; the Metadata Abstractor is to abstract meaningful information and to use them to build metadata; the Metadata Matcher uses the Extended CBR algorithm to associate the metadata with ontological concepts. The knowledge base is utilized to store the collection of predefined ontological concepts and abstracted metadata. Based on the semantic crawler's mechanism, we design the OWL format for the ontological concept and metadata, with the purpose of realizing metadata abstraction, and metadata-concept association. The CBR algorithm originates from the datamining field, which is used to retrieve and reuse the existing problem solutions for emerging problems. Here we regard concepts as existing problem solutions, and metadata as emerging problems, to apply the algorithm. By comparing the property of conceptDescription from concepts and the property of metadataDescription from metadata, the semantic similarity values between the metadata and concepts can be computed. If the semantic similarity value is greater than a predefined threshold, the metadata and concept can be determined as associated, and the URI of each side is stored into each other's opposite property. Then we realize the semantic crawler by means of Protégé-OWL and JAVA. To test the performance of the semantic crawler, we create a transport service ontology, and choose the Australian Yellowpages® website as testing data source. We use the semantic crawler to crawl 736 metadata from the 400 business webpages under the category of transport in this website. Based on a benchmark – precision from the traditional information retrieval evaluation methods, we find the most proper threshold for the semantic crawler. The figures show that, in the point of this threshold, the precision is above 90%, which is a superior performance in the present stage.

One issue of our research is that the evaluation's size is limited. Therefore our further works will focus on exploring more websites and crawling more webpages for evaluating purpose. We also need to create ontologies in other domains. In addition, more benchmarks will be selected from the information retrieval field, to prove the semantic crawler in multiple perspectives.

References

1. Carthy, D.C.J., Drummond, A., Dunnion, J., Sheppard, J.: The use of data mining in the design and implementation of an incident report retrieval system. Systems and Information Engineering Design Symposium. IEEE, Charlottesville (2003) 13-18
2. Decker, S., Erdmann, M., Fensel, D., Studer, R.: Ontobroker: Ontology based access to distributed and semi-structured Information. In: Meersman, R. (ed.): Database Semantics: Semantic Issues in Multimedia Systems. Kluwer Academic Publisher (1999) 351–369
3. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V.C., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. the Thirteenth ACM Conference on Information and Knowledge Management ACM Press, Washington D.C. (2004)
4. Dodds, L.: Slug: a semantic web crawler. (2006)
5. Dong, H., Hussain, F.K., Chang, E.: State of the art in metadata abstraction crawlers. 2008 IEEE International Conference on Industrial Technology (IEEE ICIT 2008). IEEE, Chengdu (2008)
6. Handschuh, S., Staab, S.: Authoring and annotation of web pages in CREAM. WWW2002. ACM Press, Honolulu (2002) 462-473
7. Handschuh, S., Staab, S.: CREAM: CREAting Metadata for the Semantic Web. Computer Networks 42 (2003) 579-598
8. Handschuh, S., Staab, S., Maedche, A.: CREAM — Creating relational metadata with a component-based, ontology-driven annotation framework. K-CAP'01. ACM Press, Victoria (2001) 76-83