Semantic Retrieval System Based on Ontology

JIANBO XU[1], ZHONGLIN XU[2], JIAXUN CHEN[3]
1,3: Information College
Donghua University
Yan'an Road 1882, Shanghai, 200051
PEOPLE'S REPUBLIC OF CHINA

2: Aviation University of Air Force
PEOPLE'S REPUBLIC OF CHINA

Abstract : The recall factor is low when keywords are used to retrieve information, and many related documents are omitted. Semantic annotation is used to comment documents to improve the recall factor. While extremely large instances querying requirements may crash ABox reasoner. In this research, a method is proposed to improve the efficiency of semantic retrieving via combining ABox reasoning and database retrieving. A definition of semantic relatedness between two concepts is proposed by way of computing the semantic distance. In order to avoid repeatedly computing the relatedness in information retrieving, semantic relatedness is computed outline and saved. We note documents with Annotea editing system and extract these annotations to store semantic relatedness information in database. And an algorithm is proposed to retrieve semantic information in database.

keywords : semantic annotation, relatedness , ABox reasoning, ontology

1 Introduction

Only documents included these keywords will be retrieved in conventional retrieving with keywords. While many documents don't include these keywords but include the semantic information accordant to the user's requirement, and they will not be retrieved. In this condition, the recall factor is debased. The core problem is that the retrieval system can't recognize the concerns of users correctly only depending on the keywords, so it is necessary to recognize the semantic information which is expressed in those keywords by some efficient processing methods.

The Semantic Web is the technology to make Web resources more accessible to automated processes by adding "semantic annotations"—metadata (data about data) that describes their content[1]. The semantics in semantic annotations will be given by ontologies, which will provide a source of precisely defined terms (vocabularies) that are amenable to automated reasoning.

A standard for expressing ontologies in the Semantic Web has already emerged: the ontology language *OWL* [2], which became a W3C recommendation. One of the main features of OWL is that there is a direct correspondence between (two of the three "species" of) OWL and *Description Logics* (*DL*s) [3]. This means that DL reasoners can be used to reason about OWL ontologies and about annotations that are instances of concept descriptions formed using terms from an ontology.

In order to improve the recall factor, we use semantic annotation based on Ontology to describe the information source[4]. Using semantic reasoning based on Ontology, we can get more information related to user's intention during the process of retrieval.

Unfortunately, while existing techniques for *TBox* reasoning (i.e., reasoning about the concepts in an ontology) seem able to cope with real world ontologies [5], it is not clear if existing techniques for *ABox* reasoning (i.e., reasoning about the individuals in an ontology) will be able to cope with realistic sets of instance data. This difficulty arises not so much from the computational complexity of ABox reasoning, but from the fact that the number of individuals (e.g., annotations) might be extremely large.

In this paper we describe *the semantic retrieval system based on the relatedness of ontology concepts*, which provides an approach to transfer ABox reasoning to database retrieving combined with a DL reasoner . The result is a system that can deal with very large ABoxes, and is able to provide sound and complete answers to instance retrieval queries over such ABoxes.

Through considering the relatedness between ontology concepts, we can get more related concepts in the ontology hierarchy with respect to the user's intention. So, many documents which have semantic relatedness with the user-specified keywords will be retrieved.

It is the target of the paper to provide a complete system to retrieve the related information in a big corpus. Section 2 describes how to deal with the relatedness of these ontology concepts, section 3 describes the processing of document annotation, and how to retrieve related information through semantic annotation and concepts' relatedness, section 4 describes some related works, and section 5 provides the research conclusion.

2 Semantic Relatedness Computing

Ontology is defined as a formal, explicit specification of a shared conceptualization [6]. The ontology structure O, proposed by Maedche [7], can be described by a 5-tuple O: = {C, R, H, rel, Å }. C is for concepts; R is for relations; H is for concept hierarchy; rel is a function relating the concepts non taxonomically; and Å is a set of ontology axioms expressed in appropriate logical language.

As an explicitly defined and machine-processable abstract model, ontologies were developed for the purpose of knowledge sharing to provide shared common understanding about domain knowledge. Recently, Web Ontology Language (OWL) is already being used as an open standard for deploying large scale ontologies on the Web [8].

It is viable to annotate domain knowledge with Ontology. Through these ontology annotation, the machine can understand the semantic content of the marked information. Based on the understanding to these semantic information, the machine can analysis the relatedness of these concepts or resources according to the domain knowledge.

For example, if the content of a html page is about the construction of recycle bin collecting waste paper and plastic. When a user specifies the keywords of “city environment pollution and protection” to retrieve some information, the retrieval engine using keyword-matching technology will omit the above html page. But, if the computer can understand the relationship between recycle bin collecting waste paper and the environmental protection, when the retrieval engine deals with some queries about environment pollution, it should extend the querying scope to all documents relating to environmental protection, including the html page about waste paper and plastic reclaiming and the recycle bin construction. This example indicates that AI retrieval process needs semantic relatedness reasoning according to the user's requirement to search these document which have semantic relatedness with the requirement in the corpus.

In the process of building ontology hierarchy system, there are all kinds of relationship between ontology concepts, such as is–a relation, has–part, is–a–part–of, and antonyms, and many administrator-specified relations, for example, the relation of “deal with” between waste plastic and environmental treatment. These relations influence the relatedness features between concepts. There is close relationship between semantic relatedness and the semantic distance. In this paper, we compute the relatedness by computing the semantic distance between ontology concepts. The shorter the distance the closer the concepts are. While, for various relatedness between concepts, considering that different relationship between concepts have different influences to relatedness, we specify different weight to different relations. For example, weight of is–a relation is 2, weight of is-a-part-of relation is 1.5, and default relation weight is 1.

Semantic distance : For ontology hierarchy system with various relations between ontology concepts, when we compute the semantic distance between any nodes c1 and c2, this hierarchy system can be considered as a connected graph, there maybe are many connected path between nodes c1 and c2. Considering edge weight, if the sum of all weights along the connected path between node c1and c2 is smallest, this connected path is the *shortest path*, and the *semantic distance* between node c1 and c2 is the smallest sum corresponding to the *shortest path*[9]. In this condition, we can compute the semantic distance between c1 and c2 by using Dijkstra algorithm[10].

The following are the computing equations for *semantic distance*:

$$oneDis_{(C1,C2)} = \sum_{edge(i,next(i)) \in Path(C1,C2)} W_{edge(i,next(i))} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

$$Dis_{(C1,C2)} = \min oneDis_{(C1,C2)}, \quad where\ Path(C1,C2) \in PathSet(C1,C2) \quad \ldots\ldots\ldots\ldots\ldots\ldots(2)$$

In the equations, edge(i, next(i)) is the edge which connects from node i to the adjacent node next(i), $W_{edge(i, next(i))}$ is the weight of this edge, Path(C1,C2) is a path from node c1 to node c2, which is the set of all edges along the path, PathSet(C1,C2) is the set of all path from node c1 to node c2.

Semantic relatedness : subtracting the semantic distance between node c1 and node c2 from the biggest semantic distance in the ontology hierarchy system, then the result is the semantic relatedness between node c1 and node c2.

In order to compare to human being intuition easily, the semantic relatedness is mapped to the scope of [0,1]. The equation follows:

$$\mathrm{Rel}_{(C1,C2)} = (\mathrm{D}_{\max} - Dis_{(C1,C2)}) / \mathrm{D}_{\max} \quad \ldots\ldots\ldots\ldots\ldots(3)$$

In the equation, D_{max} is the biggest semantic distance in the ontology hierarchy system.

In the applications of Ontology, such as electronic government, environmental protection, biological genes information, and so on, the ontology system of a domain is relatively stable. After being built by the domain knowledge specialist, the ontology system for a domain keeps stable and will not be changed for a long time. So, the semantic relatedness should be computed outline, and saved in database. When the relatedness between two nodes is required, it should be retrieved from database, in case the computing process is operated every time when the relatedness is considered. Then the computing complex about the semantic relatedness decreases.

The table structure for saving semantic relatedness in database follows:

```
Table T_SemanticDis(
    Id bigint not null,
    Concept1 varchar(20),
    Concept2 varchar(20),
    semanticRel float,
    primary key(Id)
);
```

Fig 1 Database table structure for semantic relatedness

In experiment of computing relatedness, it is showed that some relatedness is very small. This kind of small relatedness has very little influence to semantic retrieval, and could be omitted. So in experiment a threshold of semantic relatedness is needed. In this retrieval system, the threshold is set as 0.6.

3 document annotation and retrieval

Documents' semantic annotation is the basis of semantic retrieval. Semantic annotation is to comment on the concepts or instances in web pages with the ontology built by domain knowledge specialist[4]. The annotation description of notes will be used in semantic retrieval. These annotations usually are kept in an alone document, or in the original document as comments. In this retrieval system the annotations are saved in database. In this way, annotations could be retrieved by powerful database engine.

Annotea is a Web-based shared annotation system based on a general-purpose open RDF infrastructure, where annotations are modeled as a class of metadata[11]. Annotations are viewed as statements made by an author about a Web document. Annotations are external to the documents and can be stored in one or more annotation servers. It is reached mostly by combining RDF with XPointer, XLink, and HTTP. An instance of the system is implemented using the Amaya editor/browser and a generic RDF database, access database, accessible through an Apache HTTP server.

In Annotea protocol the annotation body is a text paragraph or an alone XHTML document. In this research, all annotation body are organized as a text paragraph, not an alone XHTML document to simplify the process of annotation. At the same time, the normative format of RDF/XML is recommended as the description format of annotation body for that well-format document is more acceptable for document parsing or semantic analysis.

Through parsing annotation document "Annota_1" with RDF parser, the annotation body is extracted, marked as Annota_body_1.

The annotation body "Annota_body_1" includes ontology concepts and ontology instances. These ontology concepts and instances could be extracted out with ABox reasoning, and saved in different database tables. The structures of these tables are designed as following tables shown in fig. 2.

```
Table T_Annotation(
    Annotation_Id bigint PRIMARY KEY,
    targetDoc_URI    varchar(254) UNIQUE
);
Table T_instances(
    Instance_Id bigint PRIMARY KEY,
    instance varchar(50),
    annotation_Id bigint ,
    INDEX index_instance(instance)
);
Table T_Concepts_Annota(
    Concept_Annota_Id bigint PRIMARY KEY,
    concept varchar(50),
    annotation_Id bigint,
    INDEX index_concept( concept )
);
```

Fig. 2 the structures of database tables for concepts and instances of annotation body

Ontology concepts extracted from annotation body are saved in table "T_Concepts_Annota", while ontology instances are saved in table "T_instances", and the correlation between annotation and target document is saved in table "T_Annotation".

When a concept "concept_A" is retrieved from table "T_Concepts_Annota" or a instance "instance_a" is retrieved from table "T_instances", it is shown that there is semantic relatedness between corresponding target document and user's intention.

Through extracting ontology concepts and instances in the period of semantic annotation and saving in database, it is decreased that requirement to ABox reasoning in the period of information retrieval, and that requirement to ABox reasoning is transferred to the requirement to general database retrieving. The number of instances might be too extremely large to crash the reasoner if ABox reasoning is used to query instances in the period of information retrieving, on the other side, the efficient of ABox reasoning is far low than that of database retrieving, so transferring the requirement to ABox reasoning from the period of information retrieving to the period of document annotation could decrease the time consumption of information retrieval. Compared to repeated calling to ABox reasoner in the period of information retrieval, there is only one time calling to ABox reasoner in the period of annotation.

3.1 the algorithm of saving annotation body

Step 0 START

Step 1 the annotation "Annota_1" of target document "URI_1" is submitted by admin;

Step 2 the system receives the annotation "Annota_1" and extracts annotation body "Annota_body_1" from "Annota_1";

Step 3 the system retrieves URI_1 in database table "T_Annotation" and gets URI_Set;

Step 4 IF URI_Set = ∅, THEN //It indicates that the "Annota_1" is a new annotation of "URI_1";

Step 5 to insert the value of "URI_1" into table

"T_Annotation" and annotation_id "annotation_id_1" is created;

Step 6 to extract ontology concepts set "Annota_concept_set" and instances set "Annota_instance_set" from annotation body "Annota_body_1" with ABox reasoner;

Step 7 to save the concept in the concepts set "Annota_concept_set" into table "T_Concepts_Annota", including the corresponding annotation_id "annotation_id_1";

Step 8 to save the instance in the concepts set "Annota_instance_set" into table "T_instances", including the corresponding annotation_id "annotation_id_1";

Step 9 ELSE // "Annota_1" is a revision to old annotation of "URI_1";

Step 10 to retrieve the value of annotation_id corresponding to "URI_1" from table "T_Annotation" and get the result "annotation_id_2";

Step 11 to delete rows from table "T_Concepts_Annota"、table "T_instances" and table "T_Ancestors" if the value of annotation_id is "annotation_id_2";

Step 12 to extract ontology concepts set "Annota_concept_set" and instances set "Annota_instance_set" from annotation body "Annota_body_1" with ABox reasoner;

Step 13 to save the concept in the concepts set "Annota_concept_set" into table "T_Concepts_Annota", including the corresponding annotation_id "annotation_id_2";

Step 14 to save the instance in the concepts set "Annota_instance_set" into table "T_instances", including the corresponding annotation_id "annotation_id_2";

Step 15 END

3.2 The algorithm to query semantic annotation in database

For a user-specified concept "user_concept_a", it is efficient to retrieve semantic annotations saved in database to get the target documents related to the user-specified concept.

Querying arithmetic:

```
SELECT T_Annotation.TargetDoc_URI
FROM T_Annotation, T_concepts_Annota
WHERE (T_annotation.Annotation_Id = T_concepts_Annota.Annotation_Id and T_Concepts_Annota.concept = user_concepts_a)
```

4 Related works

As already mentioned, the idea of supporting DL style reasoning using databases is not new. One example is [12], where an architecture and algorithms are presented which can handle DL inference problems by converting them into a collection of SQL queries. This approach is not limited to role-free ABoxes, but the DL language supported is much less expressive, and the database schema must be customised according to the structure of the given TBox.

Another example is the Parka system [13]. Parka is not limited to role-free ABoxes and can deal with very large ABoxes. However, Parka also supports a much less expressive description language, and is not based on standard DL semantics.

5 Conclusion

When keywords are used to retrieve information, the recall factor is low, and many related documents are omitted. For this situation, semantic annotation is used to describe the information to improve the recall factor. Facing extremely large instances querying requirements, ABox reasoner can't provide efficient service alone. In this paper, a method combining ABox reasoning and database retrieving is proposed to improve the efficient of semantic retrieving. Considering the semantic relatedness of ontology concepts, many semantic related documents can be retrieved. Experimental results show that the service is ameliorated.

Reference

[1] Tim Berners-Lee. *Weaving the Web*. Harpur, San Francisco, 1999.
[2] Mike Dean, Dan Connolly, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language 1.0 reference, July 2002. Available at http://www.w3.org/TR/owl-ref/.
[3] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Proc. of the 2nd International Semantic Web Conference (ISWC)*, 2003.
[4] The KIM Platform: Semantic Annotation, http://www.ontotext.com/kim/semanticannotation.html
[5] Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647,
1998.
[6] Gruber, T.R., 1993. A translation approach to portable ontology specification. Knowledge Acquisition, 5:199-220.
[7] Maedche, A., 2002. Ontology Learning for the Semantic Web. Kluwer Academic Publishers.
[8] Berners-Lee, T., 2003. News release: World wide web consortium issues web ontology language candidate recommendations, http://lists.w3.
org/Archives/Public/www-webont-wg/2003Aug/
0091.html, Aug., 2003.
[9] Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. In: IEEE Transactions on Systems, Man and Cybernetics. (1989) 17—30.
[10] Yan Weiming, Wu Weiming, 《Data Structure》: language C, Beijin : Tsinghua Publish House ,1996, P168,189.
[11] Kahan, J., Koivunen, M., Prud'Hommeaux, E., and Swick, R. Annotea: An Open RDF Infrastructure for Shared Web Annotations, in Proc. of the WWW10 International Conference, Hong Kong, May 2001 http://www10.org/cdrom/papers/488/index.html
[12]. Alexander Borgida and Ronald J. Brachman. Loading data into description reasoners. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 217–226, 1993.
[13]. W. A. Andersen, K. Stoffel, and J. A. Hendler. Parka: Support for extremely large knowledge bases. In G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proceedings of the First International KRUSE Symposium*, pages 122–133, 1995.