A Methodology for Domain Ontology Construction Based on Chinese Technology Documents

Xing Zhang and Guoping Xia

School of Economics and Management, Beihang University, Beijing 100083, P.R. China
zin@sem.buaa.edu.cn gxia@buaa.edu.cn

Abstract. Ontology is considered as one of the most important roles in knowledge sharing and reusing. However, how to effectively construct the Chinese domain ontology is a difficult problem. This paper proposed a pattern-learning Chinese domain ontology construction approach based on the fixed and simple characteristic of Chinese syntactic patterns in technology documents. The first step of this method is to construct a prototype of the domain ontology having several representative conceptions and relations. And then it makes a mapping between the prototype and the documents to find syntactic patterns. Finally, it expands the prototype of the domain ontology by using the patterns on the documents. In the end of the paper, it gives an experimental result to show that the proposed method can effectively construct Chinese domain ontology from the technology documents.

Keywords: *Domain ontology construction, Knowledge management, Chinese syntactic pattern learning, Thesaurus, Technology document*

l. INTRODUCTION

An ontology is an explicit specification of the conceptualization [1]. The domain ontology is considered as a collection of key conceptions and their inter-relationships. Most knowledge-based applications, including Knowledge Management Systems, agent systems and e-commerce platforms, need specific domain ontology to describe concepts and relations in that domain. So the domain ontology is a crucial factor for the success of those applications.

Various domain ontology construction methods have been presented recently. For instance, T. Gruber and G. Olsen [2] developed an Ontolingua ontology for mathematical modeling in engineering. L. Zhou [3] proposed a customizable collaborative system to construct the domain ontology. D. Elliman [4] proposed a method for constructing the ontology to represent a set of web pages on a specified site, using the Self-Organization Map (*SOM*) to construct the hierarchy. These methods depend too much on domain experts. Although there are some ontology tools, like Protege-2000 and Ontolingua which can ease the process of ontology construction, domain ontology construction is a boring and time-consuming task and needs human efforts. Especially for the construction of Chinese domain ontology, no effective and efficient automatic approaches have yet been found. Additionally, a

Chinese term may comprise many words and different combinations of words may have different meanings, which makes the construction more difficult.

What the domain ontology means to represent is the terms and their inter-relationships in that domain. In the real world, the best entities of terms and their relationships exist in the technology documents and we can take technology documents as a resource for domain ontology construction. In technology documents, most words of sentences are formal and fixed terms which can be easily identified by the thesaurus, and we can classified the set of words in sentences, V_D, into two subsets: $V_D = V_t \bigcup V_c$, where $V_t \bigcap V_c = \varnothing$; V_t denotes the set of terms; V_c denotes the set of those non-term words. We noted that, in technology documents, the terms from V_t contribute little to their semantic relationships and their relationships is mainly decided by meaning of words from V_c and syntax of this sentence. It also means that, when the two factors, non-term words and syntax of a sentence, are fixed, the relationships among terms in this sentence are fixed.

We also noted that there are some special characteristics of technology documents:

- The number of words in V_c is limited and the *POS* (part-of-speech), meanings and using ways of words in V_c are fixed and simple;
- Most sentences in technology documents are simple sentences, and the rest complex sentences, can be easily converted into simple sentences. Additionally, the syntax of those sentences is very simple and not variable.

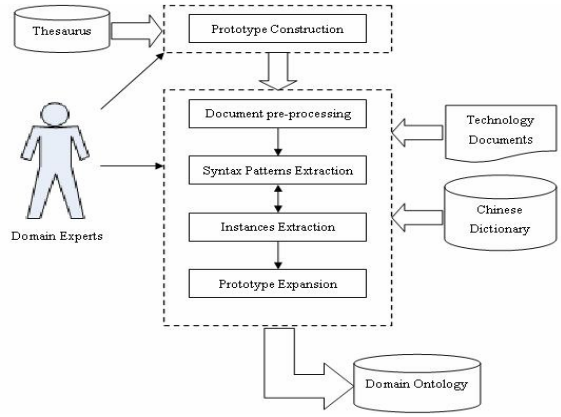

Figure 1. Flowchart of Pattern-learning Domain Ontology Construction Based on Chinese Technology Documents

It inspires us to construct such a pattern that can reflect the two fixed factors mentioned above, and then use this kind of pattern to extract the terms and instances of their relationship from the documents. Based on the analysis of the technology documents, we propose a semi-automatic methodology for domain ontology construction. The first step of this method is to construct a prototype of the domain ontology having representative conceptions and relations. And then it makes a mapping between the prototype and the documents to learn syntactic patterns. Finally,

it expands the prototype of the domain ontology by using the patterns on the documents. Figure.1 shows the flowchart of the method.

The rest of the paper is organized as follows: Section 2 briefly presents how to develop a prototype of the domain ontology by using the thesaurus. Section 3 shows the details of how to expanding the prototype with the technology documents, including document pre-processing, syntax patterns extraction, instances extraction and ontology expansion. The experimental results are discussed in the section 4. Finally, the conclusions are drawn in section 5.

2. PROTOTYPE CONSTRUCTION

In this section, we will briefly introduce the construction process for prototype of domain ontology based on the thesaurus. We first give the definition of domain ontology as follows:

Definition1 (Domain Ontology) Domain ontologies are reusable in a given domain. They provide vocabularies about the concepts within a domain and their relationships, about the activities that take place in that domain, and about the theories and elementary principles governing that domain.[5] So, here, we simply define it as a couple $On = (C, R)$, where C denotes the set of representational terms called conceptions, and R denotes the set of relationships among those conceptions.

According to ISO 2788, a thesaurus is defined as "the vocabulary of a controlled indexing language, formally organized so that the a priori relationships between concepts (for example 'broader' and 'narrower') are made explicit." [6] The thesaurus consists of terms and relationships among them, using a set of indicators to display and distinguish these relationships. And three kinds of semantic relationships, equivalence, hierarchical, and associative relationships exists in the thesaurus.

Compared with the definition of the domain ontology mentioned above, we can conclude some similarities between the thesaurus and the domain ontology: first of all, they both describe common and formal knowledge which is located on a specific domain; secondly, the terms in the thesaurus are collected and defined by the domain experts, which are common recognition, and they are very like the ones in domain ontology; finally, they both have the hierarchical relationship on the concepts. So, it is possible to develop a domain ontology by using the thesaurus and various construction methods have been present in recently years. In this phase, we simply modified the approach developed by Chang Chun [7] and adopt it to construct a prototype. The main process of the approach can be briefly described as follows: Firstly, the core terms are picked out by the domain experts according to the main aim of ontology construction; secondly, the core terms are converted to the core concepts of the prototype and their six semantic references items are also converted to different kinds of relationships between these concepts, including the hierarchical relationship; finally, domain experts supplement some typical relationships on the prototype and some concepts related would be added to the prototype. Figure.2 displays the flowchart of this approach.

The success of the prototype construction depends on the participation of domain

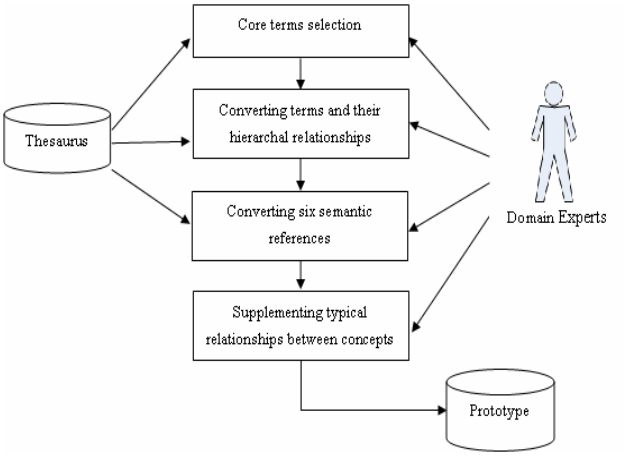

Figure 2. Flowchart of Thesaurus-based Prototype Construction of Domain Ontology

experts, so it is very important to inform the domain experts the followings: the prototype for the domain ontology need not to be perfect, and it is just a frame with representative concepts and relationships, will be expanded in the next phase.

3. EXPANSION WITH TECHNOLOGY DOCUMENTS

This section shows the details of how to expanding the prototype with the technology documents, including the *Document Pre-processing, Syntax Patterns Extraction, Instances Extraction and Prototype Expansion*.

According to the analysis of the technology documents, we intend to expand the prototype with the following steps:

1. *Document Pre-processing*: the main task is to segment the text into sentences and words, and implement part-of-speech tagging;
2. *Patterns Extraction*: make a mapping between instances and sentences in documents and then extract such patterns that can reflect the two fixed factors mentioned above;
3. *Instances Extraction*: use the patterns to extract new instances from documents;
4. *Prototype Expansion*: expand the prototype with concepts and their relations in new instances.

Because new instances would be generated in the third step and we can re-extract new patterns with new instances from the documents, so the last three steps can be cyclic implemented to expand the prototype until there is no new pattern or instance.

3.1 Document Pre-processing

The main task of this process is to separate the technology documents into sentences, and implement the words segmentation and *POS* tagging.

In this paper, we definite the word v as 2-tuple $v = (vv, pos)$, where vv denotes the Chinese characters of word v, pos denotes the part-of-speech of word v. The symbol S denotes the set of all sentences in documents. A sentence s in the document is represented by an ordered set of words $s =< st_1, st_2, ..., st_n > \quad st_i \in V_D$, where st_i denotes 2-tuple of a word in this sentence. And a document is represented by an ordered set of sentences $D =< s_1, s_2, ..., s_n > \quad s \in S$.

The process has three steps and can be described as follows:

1. *Sentences segmentation*. Documents are segmented into simple sentences according to the punctuation and some sentences need supplementing the subjects and replacing pronouns with other words;
2. *Words segmentation and POS tagging*. The principle of words segmentation is Maximum Chinese characters matching. As mentioned above, most words in technology documents are terms, which can be easily identified by thesaurus of this domain. Hence, terms can be tagged with the thesaurus first, just like inserting breakpoints in sentences, which can alleviate the difficulty of words segmentation.
3. *Words filtration*. In this step, the *Words Filter* is used to filter those words with certain *POS* tags and convert certain *POS* tags to other tags, which means to reduce significantly the number of Chinese words and alleviate the computation of the following processes. The *POS* tags of preserved words we suggest are, c(conjunction),m (numerals), n(noun), s(location), t(time), u(auxiliary),v(verb), and in certain conditions, some words with certain tags are combined. For example, if words tagged q (quantifier) and words tagged m (numerals) are adjoining, they should be combined together and re-tagged as m (numerals). However, whether a word is preserved or filtered depends on the domain and applications.

After the three steps, documents are processed into a set of sentences in which words have been filtered, segmented and *POS* tagged. An example of a sentence processed is shown below.

“1#，3#坝段布置了一孔封堵门槽，该封口门槽的孔口净宽24m。”

"A closure gate slot is deployed between 1#monlith and 3# monlith, and the clear width of its opening is 24 meter. "

After the *Document Pre-Processin*g, it is separated into two short simple sentences, shown as follows:

$s_1 =< (1\#3\#, m), (坝段, n), (布置, v), (一孔, m), (封堵门槽, n) >$

$s_2 =< (封堵门槽, n), (的, u), (孔口净宽, n), (24\text{m}, m) >$

The *POS* symbols used in this paper are followed by the Specification for Corpus Processing at Peking University [8].

3.2 Syntax Patterns Extraction

As we mentioned in section1, when the two factors, non-term words and syntax of a sentence, are fixed, the relationships among terms in this sentence are fixed. The definition of pattern that we construct to reflect the two factors is as follows:

Definition2 (Syntax Pattern)we defined the syntax pattern as an ordered set of 2-tuple spt : $sp =< spt_1, spt_2, ..., spt_n >$, $spt_i = (spv, pos)$.

Tuples spt in sp can be divided into two categories:

- kst : It denotes keyword tuple and spv in kst is the Chinese characters of keyword in this pattern
- cst : It denotes candidate concept tuple and every spv in cst is marked as cc_i . The symbol of cc_i represents a word as a candidate concept and its subscript i is same as the subscript of the candidate concepts in the instance.

Tuple kst reflects the factor of those non-term words, and we simplified the factor of the syntax pattern of the sentence to be an ordered sequence of kst and cst .

Definition3 (Instance of the Relation) Domain ontology may have many different kinds of relationships, such as attribute-of and part-of, and every kind of relationship has its own set of instances. An instance ins of the relation r is defined as 2-tuple

$$ins = (insNa, insC)$$

$insNa$ denotes the type name of r and $insC =< ic_1, ic_2, ..., ic_n >$ denotes an ordered set of concepts that ins included.

Definition4 (Matching between Sentence and Instance) Given an instance ins of the relation r and a sentence s :

$ins = (r, < ic_1, ic_2, ..., ic_n >)$ $s =< st_1, st_2, ..., st_n >$

If ins and s meet the following description:

for each ic_j in ins, it exists the corresponding st_k in s qualified the two conditions:

(1) $st_k.vv = ic_j$; (2) $not\ every\ st_k.pos = OC$.

Then we can say that the sentence s match with the instance ins .

In this section, it operate a mapping between the instances of relationships and documents that have been preprocessed, and find out those sentences that match with instances and then use them to extract syntax patterns of relationships by the following steps:

1. If one word st_j in the matching sentence s is judged as a key words by domain experts, we use st_j to generate a keyword tuple kst for the syntax pattern sp ;If one word st_j in the matching sentence s exists in the instance ins , we use st_j to generate a concept tuple cst for sp ;
2. According to the order of tuples in s , sort the order of tuples in sp ;
3. Domain experts correct and validate the syntax patterns.

The pseudocode for Syntax Pattern Extraction is shown by Figure.3.

For example, there is the "*attribute-of*" relationship between the "封堵门槽 (sealing gate slot)" and "孔口净宽(clear width of the opening)", and it can be expressed as $ins = (attri - of, <$ 封堵门槽,孔口净宽 $>)$.

For the sentence $s =<$ (封堵门槽,n),(的,u),(孔口净宽,n),(24m,m) $>$, it is obvious that s match with ins , and word "的" is judged as a keyword by the domain experts. So we can extract a pattern $sp =< (cc_1, n), (的, u), (cc_2, n) >$.

```
Process SP_Extraction(S, Ins, SP)
/*use the sentence set S and an instance set Ins to extract a syntax pattern set SP*/
Begin
 for each ins in Ins do:
    for each s_i in S do:
        if s_i match with ins
            then {generate sp;
                  for each st_j in s_i do:
                      if st_j.vv is judged as a keyword by domain experts
                          then {generate new tuple kst;
                                set kst.vv = st_j.vv and kst.pos = st_j.pos;
                                add kst to sp;}
                       else
                         {for each it_k in ins do:
                             if st_j.vv = it_k.vv
                               then { generate new tuple cst;
                                      set cst.vv = cc_k and cst.pos = st_j.pos;
                                      set st_j.pos = OC;/* avoid extract same pattern again */
                                      add cst to sp;
                                      break;}
                           End}
                  End
                  if sp not in SP
                     then add sp to SP;
                 }
    End
 End
 call domain experts to correct and validate SP
 return SP
End
```

Figure 3. The Pseudocode for Syntax Pattern Extraction

3.3 Instances Extraction

The initial step of *Instances Extraction* is to find sentences which match with syntax patterns extracted in section3.2, and we give the definition of matching between syntax pattern and sentence as follows:

Definition5 (Matching between Syntax Pattern and Sentence) Given a syntax pattern sp and a sentence s : $sp =< spt_1, spt_2, ..., spt_n >$, $s =< st_1, st_2, ..., st_n >$. We use the symbol to $\leq pos_{sp}$ denote the sequence of *POS* in sp and $\leq pos_s$ denote the sequence of *POS* in s .If every kst in sp exists in s , and $\leq pos_{sp}$ is contained by $\leq pos_s$, we can say that the sentence s match with the syntax pattern sp .

After we get the matched sentences, we can use them to extract instances of relationships according to the position and *POS* of the candidate concepts in syntax patterns and it can be described as follows:

1. generate an instance of relationship ins ;
2. scan every word tuple st_i in sentence s , if *POS* and the position in s of st_i is same as that of one tuple cst_j in sp , we then generate an concept tupleof ins and set the $st_i.vv = it_k$, and set the subscript of ic_k equals to that of cc_m in cst_j ;

3. According to the subscript of ic, sort tuples ic in ins and return ins.

Figure.4 displays the pseudocode of this process.

```
Process Ins _ Extraction(S, SP, r, Ins)
/*use a set of syntax pattern SP of r and the sentence set S to extract new instance set Ins*/
Begin
 for each sp in sp do:
    for each s_i in S do:
       if s_i match with sp
        then {generate an new ins;
              set ins.insNa = r;
              for each cst_j in sp do:
                 for each st_k in s_i do:
                    if st_k.pos = cst_j.pos and st_k.pis = cst_j.pis
                     then {generate new tuple ic_m of ins
                           set ic_m.vv = t_j.vv
                           set the subscript m of ic_m equals to the subscript of cc_m in cst_j;
                           add ic_m to ins and sort tuples ic in ins with the subscript of them;
                           break;}
                 End
              End}
    if ins not exsits in Ins
      then add ins to Ins
 End
 return Ins;
End
```

Figure. 4. The Pseudocode for Instance Extraction

For example, given a syntax pattern of relationship *attribute-of* and a sentence as follows:

$sp =< (cc_1, n), (的, u), (cc_2, n) >$, $s =< (江垭大坝, n), (的, u), (坝高, n), (136.5\text{m}, m) >$

The kst in sp is $(的, u)$ and it exists in s.

For the sequences of POS, $\leq pos_{sp} =< n, u, n >$, $\leq pos_s =< n, u, n, m >$, and there is no doubt that $\leq pos_{sp} \subseteq \leq pos_s$, so we can conclude that s match with sp. And then according to the position and POS, we scan word tuples in s with every cst in sp, finally, an instance ins of attribute-of is drawn as follows:

$ins = (attri - of, <$江垭大坝，坝高$>)$

3.4 Prototype Expansion

In this section, we use the instances obtained in section3.3 to expand the concepts and their relationships of the prototype, which can be described as follows:

1. When all concepts in an instance ins of the relationship r have already existed in the prototype and no relationship r existed among those concepts, we create the relationship r for those concepts;

2. When not all concepts in an instance *ins* of the relationship *r* have existed in the prototype, we first add those new concepts to the prototype and then create the relationship *r* among those concepts;
3. Call the domain experts to correct and validate the new concepts and relationships and check if it can create a new type of relationships with the new concepts and the old concepts.

This process is manually accomplished all by domain experts, especially for the creation of new type of relationship which mainly depend on the experiences and domain knowledge of the experts and is a time-consuming and boring task. In the future study, we will consider combining of the method of Sequential Pattern Discovery with the method of lexical co-occurrence to find new type of relationships automatically.

4. EXPERIMENTAL RESULT

The thesaurus that we adopted to construct the prototype is the Water Resources and Hydroelectric Power Thesaurus published by Yellow River Water Conservancy Press in 2000. 170 terms from the four word families: dam, material, concrete, concrete placing, are picked out as core concepts, and 34 terms is supplemented by domain experts, in sum ,there is 204 concepts in the prototype, and the experts creates 8 types of relationships between them, which is shown in Table.1

Table 1. Eight Types of Relationships in the Prototype

Type of Relationships	Example
attribute-of	Dam and height
cause	Earthquake damage of dam and Dam vibration
instance-of	Rolled dam and Jiangya Rolled Dam
kind-of	Concrete and Asphalt Concrete
material	Fibre concrete and Steel fibre
part-of	Dam and Monolith
synonymy	Slurry fill dam and Hydraulic fill dam
user	Asphalt concrete test and Asphalt Concrete

The input data is 3 technology documents related dam construction. After the document pre-processing, we got 6316 sentences. We implemented our approach with these sentences, the experimental result is as follows: 1617 instances are obtained, including 2713 concepts. After the examination, 1297 instances are confirmed by the domain experts and the precision is 80.2% and compared with precision, the recall is relatively low, only 1297 valid instances are extracted from 6316 sentences; 2614 of 2713 concepts are valid, and the precision is very high, up to 96.3%. The analysis of the experimental result is as follows:

– We ascribed the high precision of concepts to appropriately using the thesaurus in *Document Pre-processing*, which alleviate the difficulty of words segmentation. And the principle of words segmentation, Maximum Chinese characters matching, defends the integrity of concepts;

– As for the performance of *Instance Extraction*, we conclude it to the simple structure of syntax pattern. The syntax pattern of the sentence is expressed by an ordered sequence of concept tuples and keyword tuples, and in some cases, it is too simple to perform well and will leads to the failure of Instance Extraction. For example, the relationship *attribute-of* and *part-of* both have a pattern $sp = <(cc_1, n), (的, u), (cc_2, n)>$, and for the sentence "江垭大坝的坝高为136.5m (The height of Jiangya Dam is 136.5m)", we obtained two relationships between Jiangya Dam(江垭大坝) and height(坝高) and it is obviously that the relationship between two concepts should be *attribute-of*.

Although the result is not perfect, some intermediate results generated by this approach can help domain experts validate the domain ontology and discover further domain knowledge.

5. CONCLUSIONS

In this paper, we proposed a pattern-learning Chinese domain ontology construction approach based on the fixed and simple characteristic of Chinese syntactic patterns in technology documents, and gave an experimental result to show that the proposed method can effectively construct Chinese domain ontology from the technology documents. However, for some special cases, such as some relations with same syntax patterns, our method still needs some improvement.

REFERENCES

1. C. Chang, *Construction and Conversion of Ontology in Agricultural Information Management*, Scientech Documentation and Information Center, Chinese Academy of Agricultural Sciences (2004).
2. D. Elliman, J. Rafael, and G. Pulido, Automatic derivation of on-line document ontology, MERIT 2001, in *Proc. of the 15th European Conference on Object Oriented Programming* (Budapest, Hungary, 2001).
3. ISO 2788, *Documentation: Guidelines for the Development and Establishment of Monolingual Thesauri* (1986).
4. L. Zhou, Q.E. Booker, and D. Zhang, ROD-toward rapid ontology development for underdeveloped domains, in *Proc. of the 35th Annual Hawaii International Conference on System Sciences* (2002).
5. R. Mizoguchi, J. Vanwelkenhuysen, and M.I. Task, *Ontology for Reuse of Problem Solving Knowledge* in: *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing*, eds. N.J.I. Mars (IOS Press: Amsterdam, NL, 1995), pp.46-57.
6. S. Yu, Specification for Corpus Processing at Peking University: Word Segmentation, POS Tagging and Phonetic Notation, *Journal of Chinese Language and Computing*. Volume 13, Number 2, pp.121-158, (2003).
7. T. Gruber and G. Olsen, *An ontology for engineering mathematics*, Technical report, Knowledge Systems Laboratory, Stanford University, CA (1994).
8. T. Gruber, A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*. Volume 5, pp.199-220, (1993).