A Pattern-based Annotation Transformation Schema for Knowledge Exchange

Tianyong Hao, Yuanyuan Mu, and Chunshen Zhu

***Abstract*—In teamwork of semantic annotation, the task is usually assigned to different annotators, and the process may last for a long period of time due to the complexity of its nature and coverage. Moreover, the annotation criteria may vary from time to time and from person to person. The annotators have to make continuous efforts to keep the annotation data consistent. In this paper, we propose a new annotation transformation method, called PATS, based on the patterns automatically detected from the source and target annotation data. The transformation patterns can process annotation items by means of structure mapping as well as flow control such as iteration. The PATS can also be applied to knowledge exchange, in which the concepts, attributes and relations of annotation resources are mapped into a certain specific format such as OWL. As a preliminary experiment, a Chinese poetry annotation project is used to test the effectiveness of PATS. The transformation result is represented in standard OWL format automatically and can be operated by Protégé directly. The experiment proves the effectiveness and efficiency of the pattern-based annotation transformation schema.**

***Index Terms*—Semantic annotation, PATS, pattern pair, knowledge transformation.**

I. Introduction

In recent years, more and more systems tend to use accumulated and represented knowledge in order to provide more precise information. For example, major web search services like Google and Yahoo! are using ontology-based approaches to find and organize contents on the Web. Therefore, knowledge acquisition, formalization, presentation and sharing have become increasingly a real concern [1], [2]. Since a large amount of knowledge exists in text form, knowledge acquisition from text becomes one of the very important research fields in artificial intelligence [2]. Because of the high complexity of natural language, i.e. with a large number of concepts and relations in free texts being too complicated to be formalized by automatic methods, acquisition of high quality knowledge for the purpose of fine-grained data processing still mainly relies on costly manual labour at present.

Annotation can potentially become a bottleneck if there are many demands on the time of knowledge workers [3]. Few organizations have the capability of employing professional annotators to complete the annotation tasks in a fixed period of time for the sake of consistency, and therefore the annotation results may vary from person to person and from time to time. Besides, some organizations develop and use theirs own annotation languages instead of standardized annotation or ontology language, e.g. W3C's RDF annotation schema [4], Web Ontology Language OWL [5]. Due to the long-drawn process of corpus annotation, the annotation criteria or standards may also change or be modified from time to time. Therefore, the annotated resources with such variations are in need of unification and standardization for the sake of information sharing and knowledge exchange. Also, there is an urgent requirement to transform and reuse the previously annotated resources so as to reduce manual labour as much as possible.

Using standard formats is preferred since the investment in marking up resources and future proofing is considerable [3]. However, due to such factors as security or convenience, many knowledge acquisition groups are still using personalized representation languages. Human-based conversion or transformation to keep resources consistent may be very tedious and time-consuming. Variations in the formats of the annotation texts especially in terms of context-related annotations make the transformation procedures even more complicated. The related annotations with varied formats would thus make information sharing and knowledge exchange difficult. Therefore, it is important and a great challenge to develop a schema which can transform the annotated resources into a consistent or specific standard format directly and promptly.

This paper proposes a new method called pattern-based annotation transformation schema (PATS) for annotators to transform previously annotated resources into any specific format to keep the data consistent. In addition to the transformation of annotation data, this schema can also be used to transform annotated knowledge to any other representation format, such as the standard web ontology language OWL. To achieve this objective, this transformation schema PATS is equipped with functions such as concept transforming, attribute transforming and relation transforming, apart from an automatic pattern learning algorithm based on validated annotation cases. This learning procedure can identify not only pattern tags but also flow control and relations between different cases to enhance the effectiveness of transformation.

The system is tested for its efficiency in a preliminary experiment to process a corpus of Chinese poetry annotated with the aid of different annotation resources in the last three years by different annotators using different annotation languages. In the process, PATS aims to automatically generate mapping patterns in the aspects of concept, attribute and relation for each knowledge item. The generated pattern pairs are then used to transform the knowledge items automatically into the target format (OWL). The resulting OWL file is directly imported into Protégé to test the transformation accuracy. The high accuracy result proves the effectiveness of the proposed

Manuscript received March 10, 2012; revised April 22, 2012.

The authors are with the Department of Chinese, Translation and Linguistics, City University of Hong Kong, Hong Kong, China. (email: haotianyong@gmail.com.)

PATS.

II. Pattern-based Annotation Transformation Schema

Human-operated semantic annotation is important for constructing a fine-grained high quality knowledge base. But human errors, such as those caused by lack of consistency among individual annotators in using terms and formats can give rise to various kinds of complications, and coordination and integration of annotations will become very difficult. To solve the problem, we design a pattern-based annotation transformation schema (PATS) improve information transformation by which annotations in this language can be transformed into any commonly used knowledge representation languages, such as RDF or OWL.

A. Schema Description

In PATS, a pattern pair is a mapping strategy which contains a source pattern and a target pattern. An annotated source item to be transformed is defined as *s*, and its pattern as p_s, while the target representation of the item is defined as *t*, and its pattern as p_t. As to the semantic content of the item, the part to be mapped onto the target text, ether in word or phrase form, is defined as "[SLOT *id*]". The slot can be used in both the source pattern p_s and the target pattern p_t in a knowledge representation language such as OWL for ease of transformation. There is also a "[CHANGEABLE *id*]" standing for the changeable part within either a source or a target item, which means: changeable part is not used for mapping purposes but for matching changeable content within the item. A fixed symbol in the source pattern is represented as "<S=>" and the symbol is definitely not shown in target pattern. The symbol part is not used for transformation between source and target pattern but for extraction of sub patterns in the source pattern itself.

In addition to the slot-to-slot transformation, in more complex cases where, for instance, the source and the expected target text do not match in slot number, a flow control mechanism is designed to further extend the expression capability to solve the circulation annotation transformation, for example, when two items containing different numbers of slots are to be conflated. In such cases, the slot numbers cannot be fixed since PATS cannot operate the transform by using a source-target pattern set. The problem is solved with a type of flow control, e.g. "<WHILE>", to enhance the structure flexibility of the pattern. The flow control is similar to the normal development language such as C# but can be applied and changed quickly without code changing. In this way, the logic judgment with flow control mechanism extends the expression capability to deal with such complicated transformation situations.

A transformation process can become even more complicated when a key slot is found empty, that is, when there is no semantic content (i.e. a part missing) to fill in the slot, because of, say, the incompleteness of the annotation item. In such cases, the system needs either to find the related parts (attributes or relations) for the item elsewhere in the corpus or to treat it as an incomplete data item in preprocessing, with the slot remaining empty. For example, the annotators may leave out a part since the same semantic element already exists in previous or subsequent annotations. To enable the system to detect and confirm the existence of a related semantic element in the annotations, two labels are introduced, namely "[PRE SLOT *id*]" and "[NEXT SLOT *id*]", to identify element concerned in previous or subsequent annotations.

In the source pattern, SLOT with its id can be mapped onto the same SLOT in the target pattern, so that the corresponding content can be replaced in the target representation. In this way, the annotation item can be transformed into any other knowledge representation languages such as RDF or OWL automatically.

B. Case-Based Automatic Pattern Learning

To learn transformation pattern pairs, the validated transformation cases are required in advance. Technically, pattern pairs can be extracted from a single case. However, at least three cases are needed to avoid coincidence or confusion caused by overlapping terms in different annotation languages, i.e. a word is used both in the annotation paralanguage and the user's annotations proper.

To train those patterns, we propose a case-based automatic pattern learning algorithm. This algorithm firstly generates pattern pairs in each case and then computes the frequency of those generated pairs by using Apriori algorithm [6]. The pair with the highest frequency is regarded as the final pair for the transformation type of the case. The learning procedure is described in detail as follows:

Step 1: The algorithm identifies and pairs up repeated contents in an annotated source and target case (as sentences). For each word identified in both source and target case, it is tagged and replaced with tag "[SLOT *id*]" which contains a unique id.

Step 2: The fixed symbols are further identified in comparison with a symbol list and are replaced with tag "<S=>".

Step 3: Within each case, changeable contents (words or phrases), repeatedly shown as same structure but different content, are identified and each replaced with tag "[CHANGEABLE *id*]" containing a unique id.

Step 4: From each extracted pattern, the algorithm mines for all related sub patterns. The sub patterns are then converted to flow control by feature comparison.

Step 5: In different cases, the algorithm compares all parts to be mapped and identifies missing parts, which are then converted into relation tags "[PRE SLOT *id*]" or "[NEXT SLOT *id*]" containing a unique id.

For example, there are two annotation cases "Object:moon (Poetry:001) {nature;heaven;light}" and "Object:wind (Poetry:008) {nature;air}", in which "moon" and "wind" are the annotated words, "nature", "heaven", "light", and "air" belong to the paralanguage used for annotating the words in the above form of labels, and "001" and "008" are poem IDs. The two cases are transformed into the target format (OWL) as "<Word=moon>….<Annotation>light</Annotation…>".
By using the case-based automatic pattern learning algorithm, a source and target pattern pair is extracted from the two cases as follows:

TABLE I: THE PATTERN PAIRS EXTRACTED BY THE CASE-BASED AUTOMATIC PATTERN LEARNING ALGORITHM.

Source pattern	Target pattern
Object<**S**=:>[**SLOT** *id*=1] <**S**=(>Poetry<**S**=:>[**CHANGEABLE** *id*=1]<**S**=)> <**S**={><**LOOP**>[**SLOT** *id*=*id*+1] <**S**=;></**LOOP**><**S**=}>	<Word=[**SLOT** *id*=1]> <**LOOP**> <Annotation>[**SLOT** *id*=*id*+1]</Annotation > </**LOOP**> </Word>

C. Pattern-Based Knowledge Transforming

In addition to pattern-based annotation transformation, PATS can also transform knowledge or even ontology into a specific representation to achieve the goal of knowledge sharing or exchange.

Take a normal ontology as an example, which is defined as follows: An ontology is a 6-tuple $O = (C, R, A, D, Th, f)$, where $C = \{c_1, c_2\ldots, c_k\}$ is a nonempty set of concepts; $R = \{r_1, r_2, \ldots, r_l\}$ $r_i \subseteq C \times C$ is a taxonomy relation; $A = \{a_1, a_2\ldots, a_m\}$ is a set of concept attributes and $attr(c) \subseteq A$ defines the function which contains all attributes of a concept c; $D = \{d_1, d_2\ldots, d_n\}$ is a domain set; Th is a set of first-order logic propositions.

Based on the ontology definition, PATS mainly focuses on annotation transformation in the following three aspects: concept, attribute, and relation. These aspects also depend on actual situations and may vary in accordance with specific requirements. Given a knowledge item and its transformed and validated target as an actual case, PATS checks the three parts in the order of "concept, attribute, relation" to extract all pattern pairs, the order in which is shown in Table III below. The patterns in each aspect can be processed in transformation by pattern mapping.

TABLE II: AN EXAMPLE OF PATTERN-BASED KNOWLEDGE TRANSFORMATION.

	Pattern pair	Knowledge items
Source	Def [**SLOT** *id*=1] <**S**={> **<LOOP>** <**SLOT** *id*=*id*+1><**S**=:>[**SLOT** *id*=*id*+1]<**S**=//>[**CHANGABLE** *id*=*id*+1] **</LOOP>** <**S**=}>	Def morning { Temperature:cold//temperature attribute Pinyin:chén// }
Target	<owl:Class rdf:ID="[**SLOT** *id*=1]"> **<LOOP>** <owl:Restriction> <owl:onProperty rdf:resource="**#[SLOT** *id*=*id*+1]" /> <owl:hasValue rdf:resource="#[**SLOT** *id*=*id*+1]" /> </owl:Restriction> **</LOOP>** </owl:Class>	<owl:Class rdf:ID="morning "> <owl:Restriction> <owl:onProperty rdf:resource="#Temperature " /> <owl:hasValue rdf:resource="#cold " /> </owl:Restriction> <owl:Restriction> <owl:onProperty rdf:resource="#Pinyin" /> <owl:hasValue rdf:resource="#chén" /> </owl:Restriction> </owl:Class>

For example, there is a learned pattern pair from the previous knowledge transformation in the aspect of attribute only, and a knowledge item of "morning" with two annotated attributes "Temperature:cold" and "Pinyin(Chinese phonetic spelling):chén" needs to be transformed into an OWL format. With pattern mapping, the knowledge item is processed as shown in Table II below:

III. EXPERIMENT

As a preliminary experiment, PATS is applied in a Chinese poetry annotation project with annotation resources which has been on going over the last three years with annotations done by different annotators using different annotation languages to test its effectiveness in automatically generating pattern pairs in the three aspects of concept, attribute and relation for each annotation item. Take an ontology structure of "plant" as an example, the extracted patterns for concept, attribute, and relation transformation into an OWL format as the target representation are shown in Table III.

TABLE III: EXAMPLES OF TRANSFORMATION PATTERNS FOR THE CHINESE POETRY ANNOTATION PROJECT.

	Source pattern	Target pattern
Concept transformation	[**CHANGEABLE** *id*=1] [**SLOT** *id*=1] <**S**=[>[**CHANGEABLE** *id*=1] <**S**=]>	<Declaration> <Class IRI="#[**SLOT** *id*=1]"/> </Declaration>
Attribute transformation	[**SLOT** *id*=1] <**S**=[><**LOOP** *id*=1>[**SLOT** *id*=*id*+1]<**S**=,></**LOOP**><**S**=]>	<**LOOP** *id*=1> <Declaration> <DataProperty IRI="#[**SLOT** *id*=*id*+1] "/> </Declaration> <SubDataPropertyOf> <DataProperty IRI="#[**SLOT** *id*=*id*+1]"/> <DataProperty abbreviatedIRI="owl:topDataProperty"/> </SubDataPropertyOf> <DataPropertyDomain> <DataProperty IRI="#[**SLOT** *id*=*id*+1]"/> <Class IRI="#[**SLOT** *id*=*id*] "/> </DataPropertyDomain> **</LOOP>**
Relation transformation	[**CHANGEABLE** *id*=1] [**SLOT** *id*=1]<S=[>[**CHANGEABLE** *id*=1]<S=]>	<SubClassOf> <Class IRI="#[**PRE SLOT** *id*=1]"/> <Class IRI="#[**SLOT** *id*=1]"/> </SubClassOf>

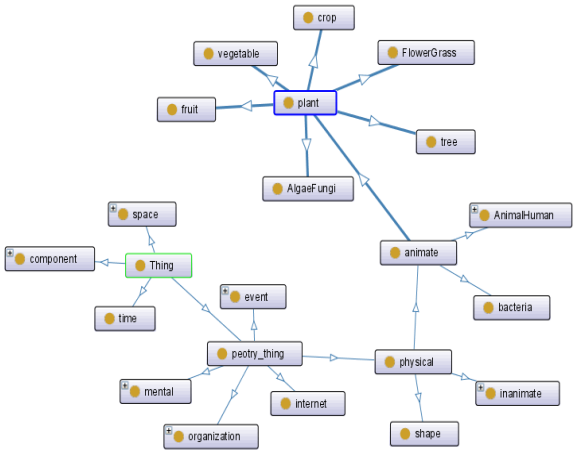

Fig. 1. Visualization of transformed ontology in Protégé using PATS.

All knowledge items in the project corpus that need to be

converted to OWL are processed by using PATS. After that, the result in the OWL format is imported into Protégé after adding file headings, which is to test the effectiveness of the pattern-based knowledge transformation since Protégé can identify and mark all errors in an OWL file. Human editors can then check the errors and calculate the transformation accuracy, which is defined as the number of correct transformations over all transformations. From the result, all PATS transformations are correct and the resulting ontology is visualized and shown as Fig.1 below.

IV. Summary and Future Work

Since semantic annotation is a complicated and time-consuming operation, the task is usually assigned to different developers and may last for a long period of time, and therefore annotations made during different periods of a particular knowledge item may vary with the improvement of annotation methods, or due to variations in annotation standards on an individual or team basis. This paper proposes a new pattern-based annotation transformation schema (PATS) to transform such kind of annotation data into any specific but standard representation format for the purpose of unifying annotations produced by discrete annotators or over various stages of an annotation project. By using PATS, pattern pairs can be automatically extracted from validated transformation cases. The technique can also be used in knowledge or ontology transformation from one language to any other commonly used presentation language. By such pattern mapping, the concepts, attributes and relations in source knowledge items are correctly mapped and transformed, as proved by our preliminary experiment with a Chinese poetry annotation project. Looking ahead, we plan to enhance the flow control to cover more complicated situations and test more data to further validate the effectiveness of the method.

References

[1] P. Hendriks, "Why Share Knowledge? The Influence of ICT on the Motivation for Knowledge Sharing," *Journal of Knowledge and Process Management*, vol. 6, no. 2, pp. 91-100, 1999.

[2] P. R. Bowden, P. Halstead, and T. G. Rose, "Extracting Conceptual Knowledge from Text Using Explicit Relation Markers," Advances in Knowledge Acquisition, LNAI, vol. 1076. Springer-Verlag, Berlin, pp.147-162, 1996.

[3] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, "Semantic annotation for knowledge management: Requirements and a survey of the state of the art," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, no. 1, pp.14-28, 2006.

[4] R. Swick, E. Prud'hommeaux, M. R. Koivunen, and J. Kahan, "Annotea Protocols," http://www.w3.org/2001/Annotea/User/Protocol. 2012.

[5] D. L. McGuinness, and F. van Harmelen, "OWL Web Ontology Language Overview," http://www.w3.org/TR/owl-features/. 2012.

[6] Wikepedia: Apriori algorithm. http://en.wikipedia.org/wiki/Apriori_algorithm, 2012.