Study of Design Issues on an Automated Semantic Annotation System

Yihong Ding
Departments of Computer Science
Brigham Young University
ding@cs.byu.edu

Reference to this article should be made as follows:
Ding Y., (2005), Study of Design Issues on an Automated Semantic Annotation System,
AIS SIGSEMIS Bulletin, Vol. 2, Issue (3&4), 2005, pp: 45-51

Abstract

The semantic annotation process turns ordinary HTML web pages into machine-understandable semantic web pages. We have proposed a semantic annotation system to automate annotation of web pages by ontologies. In this paper, we present a study of five design issues on this system, which include: (1) the covering of web pages; (2) the general paradigm of annotation process; (3) the measurements of annotation performance; (4) the compatibility to semantic web standards; and (5) the resolution for lack of ontologies. To address these issues, our proposed system intends to automate annotation for data-rich web pages; simply the annotation process and improve the degree of automation by adapting the ontology-based data recognizer; measure to obtain accurate annotations, to run fast, and to be resilient to page layouts through a two-layer annotation process; be compatible to the semantic web standards by using OWL ontologies; and benefit an interactive ontology creation process by automatically choosing relevant ontology components according to an annotation task from an ontology knowledge base.

1. Introduction

The semantic web provides a machine-understandable environment [2]. Machines, however, do not really understand web page content unless its meaning is *explicitly* specified in a *formal*, *unambiguous* way. To establish machine understandability, people use *ontologies*, which are explicit, formal specification of conceptualizations [8]. A *semantic annotation* process is thus to label web page content explicitly, formally, and unambiguously using ontologies.

After the emergence of the semantic web, more and more people have accepted it to be the next-generation of World Wide Web. With machine-understandable semantic web pages, we can develop more practical and interoperable web applications. For example, when the semantic web comes to reality, users can directly search web content using database-like queries. To establish the semantic web, however, is difficult. There are billions of pages in the ordinary web and only very few of them have been converted to become machine-understandable. It is impractical to ask web developers to rewrite all their web pages with respect to new semantic-web standards, especially if it involves tedious manual labeling of documents. Hence automatic semantic annotation becomes the important bridge that links the ordinary web to the fascinating semantic web.

By the study of existing semantic annotation approaches (e.g., [1, 3, 9, 10, 12, 13]), we have summarized a typical process for current automated semantic annotation systems. It usually takes an annotation task and a domain ontology as inputs, where an annotation task is a set of web pages waiting to be annotated. The system sends these inputs to an automated data recognizer, which is an adapted information extraction (IE) tool, to extract data instances from web pages. After extracting,

the system usually performs "a set of heuristics for post-processing and map-ping of the IE results to an ontology." [10] An annotation generator takes these mappings to eventually create explicit annotations that ontology-aware machine agents can process. These annotations may be stored either within the original web pages or in separate files.

Although this typical annotation process works, there are some problems that limit its practi-cability. For example, Kiryakov et. al. have pointed out that the requirement of the "post-processing and mapping" between the extracted results and ontologies is, as their words, "the main drawback" for these existing automated annotation systems [10]. To solve this problem and along with several other improvements, we have proposed a new automated semantic annotation system using extraction ontologies [4]. As Figure 1 shows, this proposed system takes web pages and domain ontologies as inputs. When there are no input ontologies, the system embeds an ontology assembler that provides an interactive ontology creation process by automatically choosing relevant ontology components according to the annotation task from an ontology knowledge base. Depending on the number of web pages and the degree of domain diversity, the system either hands the task to the conceptual annotator, which annotates documents by ontology-based domain specifications, or hands it to the structural annotator, which annotates documents by page-layout specifications. The two annotators share a common annotation gene-rator. The ontology converter in the figure assures that our system is compatible to a semantic web standard, which is OWL (Web Ontology Language) [14].

The focus of this paper is to present five design issues we have studied when proposing this automated annotation system, which include: (1) the covering of web pages our system works for; (2) the general paradigm of semantic annotation process we have simplified to improve the degree of automation of our system; (3) three performance measurements (accuracy, speed, and resiliency) our system aims to achieve; (4) the compatibility to the semantic web standards our system holds; and (5) the resolution our system provides to handle annotating when there are no input ontologies. Through the discussion of these design issues, we not only describe the details of our proposed system, but also state the reasons that our improvements are effective to achieve practical semantic annotations.

In the rest of this paper, we start with the discussion of the web page coverage issue in Section 2. In Section 3, we present the reason that the adaptation of the ontology-based data recognizer increases the degree of automation of our annotation system by simplifying the paradigm of the typical annotation process. Section 4 presents the three performance measurements and the reason our two-layer annotation model improves their evaluations. Section 5 addresses the issue that our system is compatible to the semantic web standards. In Section 6, we present a mecha-nism in our system to semi-automatically assemble a domain ontology when no ontology inputs. In the end, we conclude in Section 7.

2. Web Pages Coverage

The first design issue we have addressed is the covering of web pages our automated annotation system works for, which is determined the data recognizers we are going to adapt. There are no automated IE tools that can effectively perform on all web pages [11]. Each individual IE approach has its favorite covering of web pages, from which it is effective to extract data. For example, it is favorable to use NLP (natural language processing) based IE tools to extract unstructured free-text documents, while HTML-aware IE tools are effective to extract data from fully structured HTML web pages [11].

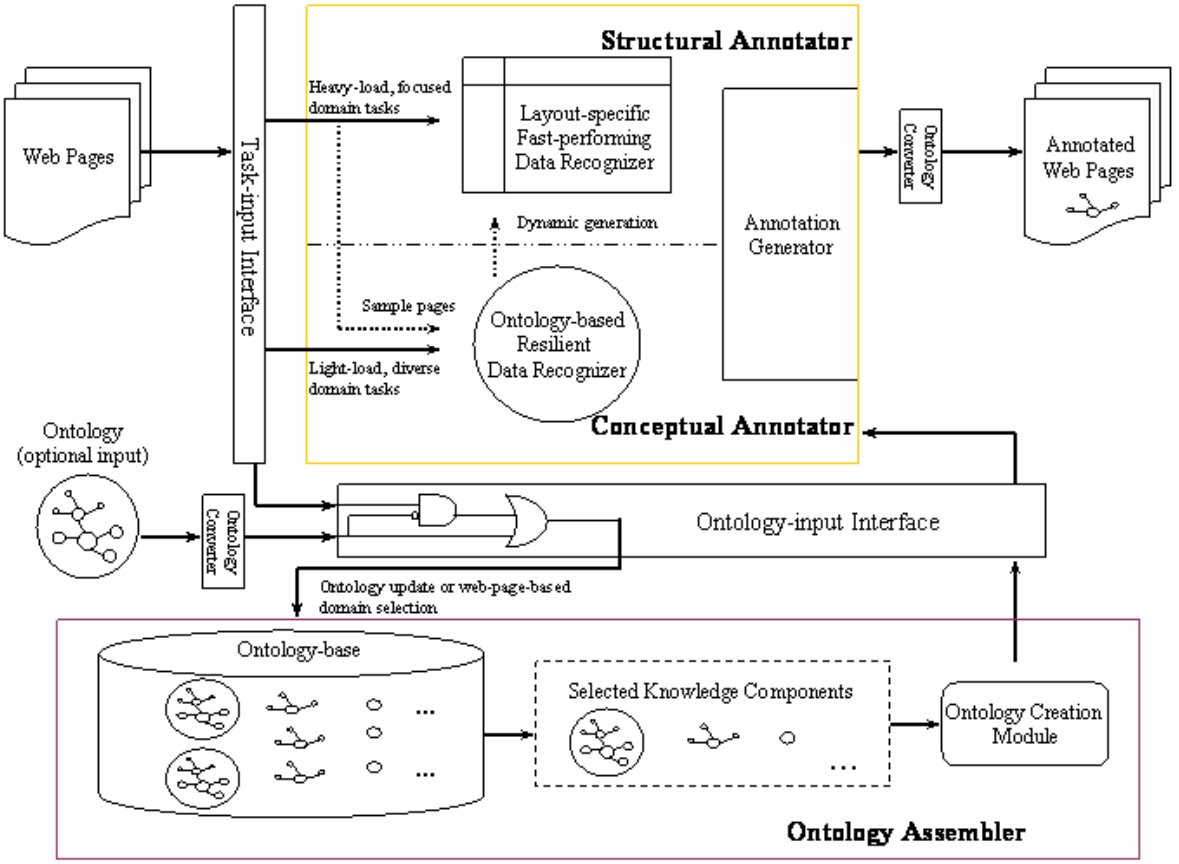

Figure 1: Framework of Automated Semantic Annotation System Using Extraction Ontologies

Moreover, the importance of this coverage issue is not only for the anxiety of determining applicable web pages, but also for the curiosity of knowing appropriate domains the annotation system can effectively manage. A fact in the web is that each domain usually has its typical web representing format. For example, news is usually written in free-text web pages; and shopping categories are often presented within complex HTML tables without many complete natural language sentences. Therefore, when an annotation system adapts an NLP-based IE tool, it can perform well on the news domain but with less accuracy on the shopping domain. Similarly, an annotation system with an HTML-aware data recognizer can effectively annotate the shopping domain but not so effective on annotating the news domain.

We plan to design a system that can effectively annotate semi-structured and fully structured data-rich web pages that each have a relatively narrow domain. The ontology-based IE tool matches this purpose [6, 7]. We must mention that this type of web page coverage is not unique for our annotation approach (see, for example, [12]). Moreover, this type of web page is common on the web (shopping, product portals, for example).

3. Annotation Process Paradigm

The second system design problem we encountered is to establish the entire annotation process. We have mentioned earlier that a significant problem in the typical annotation process is the requirement of the "post-processing and mapping of the IE results to an ontology." The reason causing this problem is the independency between ontologies and the non-ontology-based IE wrappers (to become the data recognizers). According to the survey written by Laender et. al., all the automated IE approaches except the ontology-based ones do not extract data with respect to ontologies [11]. Since

ontology is a mandatory factor in the semantic annotation scenario,[3] this "post-processing and mapping" problem is unavoidable for the annotation systems with adapted non-ontology-based data recognizers.

To solve this problem, we have integrated domain ontologies with extraction engines so that the extraction process is executed directly with respect to the ontological declarations [5]. We declare extraction patterns to be the extensional semantics within domain ontologies, through which the ontology-based data recognizer extracts data instances. Hence these data instances are directly categorized into their corresponding ontology definitions without the needs of further "post-processing and mapping." Through this new paradigm, we do improve the system's degree of automation because the original "post-processing and mapping" often requires much human involvement. Our resolution actually fulfills what Kiryakov et. al. have suggested in [10].

4. Performance Measurements

After establishing the general semantic annotation process, we did a measurement study to ensure the overall performance of our system. We believe that three performance measurements, which are accuracy, speed, and resiliency, are equally important to a semantic annotation system. There is no problem that accuracy and speed measurements are crucial. For automated semantic annotation systems, high resiliency to web page layouts is also important because otherwise a system may have to regenerate data recognition patterns each time for a new page layout. This type of regeneration usually decreases the degree of automation for the system.

To achieve good performance on all the three measurements, we have proposed a two-layer annotation model that contains two annotators—a lower-layer conceptual annotator and an upper-layer structural annotator. The conceptual annotator employs an ontology-based data recognizer to perform resilient annotating on web pages with varied layouts. The structural anno-tator employs one or more layout-specific data recognizers, where each recognizer can annotate web pages with a common layout fast and accurate. Figure 1 shows that the two annotators are separated by the dot-dash line.

The resiliency property for the ontology-based data recognizer is inherited from the ontology-based IE approach [6]. Our ontology-based IE tool can continuously work on different web page layouts so long as the pages are for the same domain because its extraction process is based on the declarative domain-oriented extensional semantics without encoding of layout information. As a trade-off to be resilient, our ontology-based data recognizer requires a large number of computational cycles to enumerate all possible candidate instances and resolve ambiguities. Hence, its execution speed is relatively slow. Also, although the ontology-based data recognizer is designed to achieve good accuracy in general cases, it does not take into account local structural patterns, which can lead to higher extraction accuracy.

On the contrary, a layout-specific data recognizer performs very fast because it usually requires only a single pass through an entire document to do extraction. It also assures very high accuracy because it only processes web pages that match a known layout structure. Therefore, layout-specific data recognizers are not resilient. They usually fail to perform correctly when layouts are unknown or change. When there is a new layout pattern, the system needs a regeneration process to build a new layout-specific data recognizer.

[3] Ontologies are optional within the traditional IE paradigm. Due to the difficulty of ontology generation, many traditional IE researchers do not input ontologies to their automated IE systems.

Figure 1 illustrates how we integrate these two annotators. When the system needs to annotate large numbers of web pages, and especially if these web pages are for a focused domain and hold a common layout,[4] the system takes a small set of web pages out of the input task to be samples, which are sent to the conceptual annotator (as the long dotted arrow-head line in Figure 1). After the conceptual annotator annotates them, the system processes a structural analysis on the annotated sample pages, through which the system dynamically creates a layout-specific data recognizer according to the presented page layout (as the short dotted arrow-head line in Figure 1). This created layout-specific data recognizer and the annotation generator together compose the structural annotator. The system then sends the vast majority of the input web pages to the created structural annotator to take the benefit of fast speed and high accuracy annotation. On the contrary, when the number of input web pages is small and the layouts are varied, it is too expensive to process dynamic generation of multiple data recognizers, while each of them annotates only a very small number of web pages. The system therefore simply let the conceptual annotator annotates the whole task to take the benefit of resiliency.

We must further point out two characteristics about this dynamic generation of the structural annotator. First, it holds the property of resiliency. When the set of sample pages contains multiple layouts, the conceptual annotator can annotate them continuously due to its resiliency. Using the annotated web pages, the system can simultaneously create a layout-specific data recognizer for every input layout. Second, this dynamic generation process does not conflict to the elimination of the "post-processing and mapping" we have just discussed. Layout-specific data recognizers are generated from annotated web pages that already contain correct mappings between data instances and ontology concepts. Therefore, the mappings between extraction categories in the generated layout-specific data recognizers and ontology concepts are ensured.

5. Compatibility to Semantic Web Standards

When we design our semantic annotation system, we want it to be compatible to the semantic web standards so that it can be directly used by the rest of the semantic web society. However, the adapted ontology-based data recognizer requires OSMX (Object-oriented Systems Model in XML) ontologies [5, 6], which is not a semantic web standard. We thus need to do a conversion between OSMX and a semantic web standard. Since OWL is widely accepted to be a standard semantic web ontology language, we choose it to be the semantic web standard in our system.

Fortunately, the OSMX and OWL representations are quite similar and compatible to each other. Many conversions are straightforward. For example, an *object set* in OSMX is a *class* in OWL; a *relationship set* in OSMX is an *ObjectProperty* in OWL; a *participation constraint* in OSMX is a *Cardinality* restriction in OWL; an *isa* hierarchical relationship in OSMX is a *subClassOf* specialization in OWL. There are, however, some unique specifications in each language. For example, the data frames, which describe extensional semantics of ontology concepts in OSMX, are not well defined in OWL, while OSMX does not explicitly support the subproperty feature in OWL. Our ontology converter needs to address and solve these specialties.

As Figure 1 shows, we put the converter on both the input and output sides of our system. On the input side, when users input an OWL ontology, the converter transforms it to its OSMX representations. Otherwise, the system simply ignores the converter and takes the input OSMX

[4] This is quite common in the ordinary web. For example, the auto-generated web pages within many large commercial web sites, such as amazon.com or ebay.com, hold common layouts and domain.

ontology to the data recognizers. On the output side, when the original input ontology is in OWL, the system generates annotations directly with respect to the original OWL ontology, and thus, no conversion is needed. Otherwise, when there are no input OWL ontologies, the system converts the OSMX representations used by the data recognizers to their OWL representations, and outputs annotations with respect to the converted OWL ontology.

6. Resolution for Lack of Input Ontologies

Until now, we assume that there are input ontologies, which is also the assumption most of the current automated semantic annotation systems make [1, 3, 9, 10, 12, 13]. However, ontology creation is difficult and most of current ontologies are constructed manually. Many times users simply cannot find an existing ontology that is appropriate for their annotation task. Our system, therefore, propose the ontology assembler to help users build an ontology semi-automatically on the absence of input ontologies. The theme underlying our ontology assembler is to maximize the reuse of existing ontologies and minimize the work of constructing new ontologies.

The bottom part of Figure 1 shows our ontology-input interface and the ontology assembler. The logic circuit inside the ontology-input interface shows that the interface alternately inputs a set of descriptive web pages, which illustrate the domain of interest, to the ontology assembler when there are no input ontologies. The ontology assembler consists of two parts—an ontology-base and an ontology creation module. The ontology-base consists of pre-used and pre-constructed ontologies, snippets of ontology, and single concept recognizers. The ontology creation module in our system is an ontology editor that users can view and manually create or modify ontologies.

When there is an input ontology, the assembler simply updates the ontology-base with the input ontology and displays it by the ontology editor. With users' approval, the system sends the ontology to the annotators to accomplish the annotation task. Otherwise, there are no input ontologies but a set of descriptive web pages. The assembler performs a knowledge-selection process to look for relative ontology components within the ontology-base with respect to the descriptive web pages. These ontology components could be pre-existing ontologies, snippets of ontology, or single concept recognizers, as the dashed box inside the ontology assembler shows. The assembler thus sends these selected components to the ontology creation module, through which users can view these components, integrate the appropriate ones, and build missing parts, if necessary. Finally, users assemble the appropriate components to be a unified ontology, which is the name ontology assembler coming from.

7. Concluding Remarks

This paper presents a brief introduction of our automated semantic annotation system. Through the discussion, we present five design issues that we have addressed when proposing this system.

- The web page coverage issue affects both the web page types and potential applicable domains an annotation system can well-perform. It decides the theme of an annotation system. Our system focuses on annotating semi-structured and structured data-rich web pages, which are common on the ordinary web.
- With the focused theme, we need to figure out an effective process to accomplish annotation tasks. The ontology-based data recognizer helps to improve the degree of automation of our system by eliminating the requirement of the "post-processing and mapping of the IE results to an ontology."

- Having an effective annotation process, the success of our system is closely related to the performance measurements. The two-layer annotation model assures our system to obtain accurate annotations, to run fast, and to be resilient to page layouts.
- Nevertheless, the acceptance of our annotation system depends on whether our system is compatible to the semantic web standards. The ontology converter assures our system to accept OWL ontologies and to produce annotations with respect to OWL representations.
- The existence of ontologies is not promising, while the requirement of ontologies is demanding. When there are no input ontologies, our ontology assembler helps to build a task-oriented ontology through an interactive process by automatically choosing relevant ontology components according to an annotation task from an ontology knowledge base.

Upon to the time we submit this paper, this is an on-going project. Our papers [4] and [5] have described more details of our proposed system and what we have done. There is also an online demo of our annotator.[5] Through this study, we expect to deliver the vision that it is convincible to develop practical semantic annotation systems that can automatically accommodate the huge quantity of existing data-rich web pages on the ordinary web.

References

[1] L. Arlotta, V. Crescenzi, G. Mecca, and P. Merialdo, "Automatic annotation of data extracted from large web sites," In *Proceedings of Sixth International Workshop on the Web and Database (WebDB 2003)*, pp. 7-12, San Diego, California, June 2003.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, 36(25):34-43, May 2001.

[3] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K.S. McCurley, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien, "A Case for Automated Large Scale Semantic Annotations," *Journal of Web Semantics*, 1(1):115-132, December 2003.

[4] Y. Ding, "Annotating Documents for The Semantic Web Using Data-Extraction Ontologies," *PhD dissertation proposal*, Brigham Young University, September 2005.

[5] Y. Ding, D.W. Embley, S.W. Liddle, "Semantic Annotation Based On Extraction Ontologies," 2005. (submitted to review)

[6] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith, "Conceptual-model-based data extraction from multiple-record web pages," *Data & Knowledge Engineering*, 31(3):227-251, November 1999.

[7] D.W. Embley, C. Tao, and S.W. Liddle, "Automating the extraction of data from HTML tables with unknown structure," *Data & Knowledge Engineering*, 54(1):3-28, July 2005.

[8] T.R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, 5(2):199-220, 1993.

[9] S. Handschuh, S. Staab, and F. Ciravegna, "S-CREAM Semi-automatic CREAtion of Meta-data," In *Proceedings of European Conference on Knowledge Acquisition and Management (EKAW-2002)*, pp. 358-372, Madrid, Spain, October, 2002.

[10] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff, "Semantic Annotation, Indexing, and Retrieval," *Journal of Web Semantics*, 2(1):49-79, December 2004.

[11] A.H.F. Laender, B.A. Ribeiro-Neto, A.S. da Silva, and J.S. Teixeira, "A brief survey of web data extraction tools," *SIGMOD Record*, 31(2):84-93, June 2002.

[12] S. Mukherjee, G. Yang, and I.V. Ramakrishnan, "Automatic Annotation of Content-Rich HTML Documents: Structural and Semantic Analysis," In *Proceedings of Second International Semantic Web Conference (ISWC 2003)*, pp. 533-549, Sanibel Island, Florida, October, 2003.

[13] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna, "MnM: Ontology Driven Tool for Semantic Markup," In *Proceedings of Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002)*, pp. 43-47, Lyon, France, July, 2002.

[14] W3C (World Wide Web Consortium) *OWL Web Ontology Language Reference*. URL: http://www.w3.org/TR/owl-ref/.

[5] http://www.deg.byu.edu/