Data & Knowledge Engineering 69 (2010) 1043–1061

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Evaluating ontology extraction tools using a comprehensive evaluation framework

Jinsoo Park [a,*], Wonchin Cho [b], Sangkyu Rho [a]

[a] Graduate School of Business, Seoul National University, 599 Gwanangno, Gwanak-Gu, Seoul, 151-916, Republic of Korea
[b] College of Business Administration, Seoul National University, 599 Gwanangno, Gwanak-Gu, Seoul, 151-916, Republic of Korea

ARTICLE INFO

Article history:
Received 14 February 2009
Received in revised form 3 July 2010
Accepted 6 July 2010
Available online 14 July 2010

Keywords:
Ontology
Ontology extraction tool
Tool evaluation

ABSTRACT

Ontologies are a key component of the Semantic Web; thus, they are widely used in various applications. However, most ontologies are still built manually, a time-consuming activity which requires many resources. Several tools such as ontology editing tools, ontology merging tools, and ontology extraction tools have therefore been proposed to speed up ontology development. To minimize building time, one promising solution is the automation of the ontology development process. Consequently, the need for an automatic ontology extraction tool has increased in the last two decades and many tools have been developed for this purpose. However, there is still no comprehensive framework for evaluating such tools. In this paper, we proposed a set of criteria for evaluating ontology extraction tools and carried out an evaluation experiment on four ontology extraction tools (i.e., OntoLT, Text2Onto, OntoBuilder, and DODDLE-OWL) using our proposed evaluation framework. Based on the results of our experiment, we concluded that ontology extraction tools still lack the ability to automate the extraction process fully and thus require functional performance improvement.

1. Introduction

The Web is evolving toward the Semantic Web, in which the semantics of Web content is defined, making the Web meaningful, understandable, and machine-processable [2]. Ontologies are considered a crucial component of the Semantic Web because they are the backbone of knowledge representation [57]. An ontology is an explicit formal conceptualization of a specific domain of interest [29]. Ontologies have been studied by many researchers in various fields. In particular, they are widely used in various applications such as data annotation, data integration, and intelligent system development [47].

Although many efforts have been made to develop ontologies, several issues still need to be resolved in order to construct an ontology effectively and efficiently. First, since most ontology development processes are carried out manually, building an ontology is a time-consuming activity. This causes a bottleneck problem which results from the lack of a fast and efficient way to build an ontology. Recently, several studies attempted to resolve this problem [22,36,55], and as a result, several tools have been developed for the automation of the ontology development process. These tools can be broadly classified into ontology editing tools, ontology merging tools, and ontology extraction tools. Ontology editing tools help the ontology engineer acquire, organize, and visualize domain knowledge before and during ontology construction. Ontology merging tools are used to create a single coherent ontology by unifying two or more existing ontologies. Ontology extraction tools support the automatic extraction of concepts and/or their relations by applying some techniques such as natural language processing or machine learning.

The second issue is related to tool evaluation. Several evaluation frameworks have been proposed to measure the performance of ontology editing tools [39] and ontology merging tools [34]. Although these tools might reduce the ontology-building time, the

* Corresponding author. Tel.: +82 2 880 9385; fax: +82 2 872 6366.
E-mail addresses: jinsoo@snu.ac.kr (J. Park), cool97@snu.ac.kr (W. Cho), srho@snu.ac.kr (S. Rho).

0169-023X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2010.07.002

most promising solution is the automation of the ontology development process using ontology extraction tools. Thus, when building an ontology from scratch, ontology extraction tools are considered a fundamental and critical solution for resolving the bottleneck problem. However, to the best of our knowledge, no evaluation framework exists on ontology extraction tools, which may be partly due to the inherent difficulty of the evaluation process.

The last issue deals with the quality of a constructed ontology. Evaluating ontology quality refers to assessing how well an ontology reflects the real world using concepts, relations, and axioms, among others. Since ontology quality may have a direct impact on the quality of the outcome of the processes in which it is used, the ontology user needs to evaluate its quality before applying it to a specific application. Accordingly, some studies suggested various criteria to measure the quality of an ontology regardless of whether it is built manually or automatically [8,27].

In this paper, we propose a novel evaluation framework that will guide us in measuring the various aspects of ontology extraction tools, considering both tool performance and the quality of a constructed ontology. In particular, ontology quality is measured based on three dimensions as adopted from the work of Burton-Jones et al. [8]: syntactic, semantic, and pragmatic qualities. Using our proposed evaluation framework, we carried out an experiment and reviewed four popular extraction tools: OntoLT, Text2Onto, OntoBuilder, and DODDLE-OWL.

2. Related work

2.1. Automatic ontology extraction tools

Although some methodologies for building ontologies have been proposed to improve the ontology development process [11,18,25,55], building ontologies manually is a time-consuming and laborious activity that requires the work of highly trained ontology engineers. Moreover, an ontology that is built manually tends to be biased toward its developer's view. Therefore, researchers have been interested in the automation of the ontology extraction process and have developed several ontology extraction tools.[1]

Ontology extraction techniques rely on methods borrowed from various fields such as machine learning, knowledge acquisition, natural language processing, statistics, and information retrieval. Before the introduction of statistical and machine learning techniques into the natural language processing domain, it was difficult to analyze corpora from a specific domain owing to the lack of resources and adequate approaches for large-scale computation. Nowadays, many languages can be processed by tools that allow the recognition and analysis of sentences' major constituents and their relations. Furthermore, techniques that are combined with a natural language's standard pattern matching and machine learning techniques also allow ontology engineers to extract concepts and their relations [13].

Typically, ontologies can be generated from various data types such as textual data, dictionaries, knowledge bases, semi-structured schemata, and relational schemata. Most works related to automatic ontology construction have been directed toward extracting an ontology from texts [56]. A typical approach in ontology extraction from text first involves term extraction from a domain-specific corpus at hand. Then, the extracted terms are clustered into groups with the purpose of identifying the taxonomies of potential classes. Relations can also be extracted by computing a statistical measure of connectedness between identified clusters [7,36]. Several systems have been developed in this area such as OntoLearn [40,41], Text2Onto [35], ASIUM [23], OntoBuilder [26], SOAT [60], SVETLAN [9], and Mo'K Workbench [3].

2.2. Evaluation framework for ontology tools

The evaluation of the functionality of ontology tools is very important because it will help the ontology engineer choose an appropriate tool for his/her specific need. Measuring various aspects of tools requires evaluation criteria. While methodologies such as OntoClean [30] have been proposed for validating the ontological adequacy of taxonomic relationships based on philosophy, to the best of our knowledge, no comprehensive evaluation framework exists for ontology extraction tools. However, we believe that frameworks for ontology editing and merging tools would still be useful references when we develop an evaluation framework for ontology extraction tools. Thus, in this section, we review those designated for ontology editing and ontology merging tools. We also discuss the work of Navigli et al. [41]. Although Navigli et al. did not provide any evaluation criteria, they did try to evaluate ontology extraction tools.

Lambrix and Edberg [34] evaluated two of the most popular ontology merging tools, Protégé-2000 with PROMPT [44] and Chimaera [45]. The objective of their study was to measure how well these tools were suited for merging currently existing bio-ontologies. For this, they defined evaluation criteria such as availability, stability, representation language, functionality, and user interface. They tested these tools using literature and empirical studies. To define the availability, stability, representation language, and functionality of Protégé with PROMPT and Chimaera, they studied some relevant literature. They then tested these tools based on the criteria mentioned above. On the other hand, the user interfaces of Protégé with PROMPT and Chimaera were evaluated through eight test participants. After evaluating the tools with their proposed criteria, they concluded that both Protégé with PROMPT and Chimaera were well-suited to model most of the current bio-ontologies.

Murshed and Singh [39] proposed an evaluation framework to help ontology engineers choose a proper editing tool for their purpose. Their evaluation criteria included functionality, reusability, data storage, complexity, association, scalability, resilience,

[1] Some researchers use the term “ontology learning” instead of “ontology extraction” primarily because their tools gave rise to concepts, and their relations were obtained through a learning process. However, since many ontology merging tools are also operated through learning and, furthermore, some ontology extraction tools have worked based on linguistics instead of learning, we used the term “ontology extraction” in our paper.

reliability, robustness, learnability, availability, efficiency, and visibility. Based on these criteria, they compared three popular ontology editing tools, Protégé, Metis, and OntoEdit. They concluded that Protégé can be used to construct small-sized ontologies; OntoEdit is suited for developing medium to large-sized ontology; and Metis is good for enterprise architecture modeling.

Navigli et al. [41] proposed a twofold evaluation strategy for OntoLearn. First, they conducted a detailed quantitative analysis of OntoLearn's main algorithms. Their goal was to compute the accuracy of OntoLearn under different learning circumstances. The three basic algorithms (terminology extraction, ontology learning, and semantic annotation) have all been individually evaluated in the economy and tourism domains. Most of the algorithms had good performances, while some algorithms, such as semantic annotation algorithm needed to be improved by updating the set of relations and enriching the training corpus with manually tagged examples from the new domain. Second, they conducted a per-concept qualitative analysis by domain specialists. They automatically generated natural language descriptions of formal concept specifications since the domain specialists would find it hard to evaluate the formal aspects of a computational ontology. Two specialists participated in the per-concept qualitative analysis. The evaluation result showed that some of the descriptions would not be appropriate to take them over in an ontology just as they are. However, most of them turned out to be quite helpful, serving as a basis for building the ontology.

In summary, the previous research on the evaluation of ontology editing and merging tools proposed a set of criteria and empirically tested tools based on their proposed evaluation criteria. However, many issues still remain problematic. First, most of the criteria are centered on the usability issue. Thus, they do not seriously consider the outputs produced by ontology editing and merging tools. It is important to judge whether the ontology created by the tools satisfies quality requirements before it is effectively used for its own distinct purpose. Second, most of the works fail to show the evaluation result objectively using right comparison methods. Thus, they mainly show results of a straightforward analysis on the tools' capabilities, and discuss the pros and cons of these tools without any numerical and experimental comparison. In order to help users identify which tool is most suitable, however, the evaluation of ontology tools should provide a hands-on comparison of their functional performance using proper comparison methods.

Similarly, the foregoing issues should also be considered carefully when performing evaluations on the ontology extraction tools. As a first attempt, we addressed these issues by, first, proposing an evaluation framework consisting of various criteria that consider both usability and quality of a constructed ontology. In particular, to assess the ontology quality constructed by an extraction tool, we partly adopted Burton-Jones et al.'s well-defined criteria. Second, to show an objective comparison result, we performed an empirical test using the Analytic Hierarchy Process (AHP) method, a widely used and well-tested method among available techniques in solving multi-attribute comparison problems [49].

2.3. Ontology quality

Some efforts have been made to evaluate the quality of constructed ontologies [8,27,33,46]. Brank et al. [4,5] grouped various evaluation approaches into four categories. The first approach, called the gold-standard approach, attempts to evaluate the quality of constructed ontologies using a “gold standard” ontology. In this approach, the gold standard ontology is regarded as a well-constructed one. It could be another existing ontology, or it could be taken statistically from a corpus of documents or prepared by a domain expert. The concepts of a constructed ontology are evaluated by comparing them with those of a gold standard ontology, which are considered good representations of the concepts for the problem domain under consideration [17]. Typically, the gold standard approach is used to evaluate an ontology generated by a learning process. The second one is an application-based approach in which the quality of the ontology is evaluated based on its actual use in a real-world application [62]. The output of the application or its performance on the given task might be better or worse depending on the ontology used in it. Ontologies may therefore be evaluated simply by plugging them into an application and evaluating the results of such application. However, if an ontology is only a small component of the application, it is difficult to judge its quality using the application-based approach because its effect on the outcome may be relatively small and indirect [5]. The third approach is data-driven because it evaluates the quality of an ontology by measuring the fit between the ontology and the corpus of a problem domain to which it refers. Thus, this approach evaluates an ontology by measuring the amount of overlap between the domain-specific terms in the corpus and terms appearing in the ontology [6]. Since an ontology is a fairly complex structure, it should be evaluated on the lexical, semantic, syntactic, and context levels. In the data-driven approach, however, an ontology is evaluated only on the lexical level [5]. The final approach relies on human judgment. In this approach, the evaluation is done by domain experts who try to assess how well the ontology meets a set of predefined criteria, standards, and requirements. Although this evaluation requires a longer time, this approach can evaluate an ontology in various perspectives including lexical, semantic, syntactic, and context levels. Our approach belongs to the last category.

Burton-Jones et al. [8] proposed a suite of metrics to assess ontology quality. Their research is based on the semiotic theory and focuses on four aspects of ontological quality: syntactic, semantic, pragmatic, and social. In this research, we adopted some of their criteria in our framework. These four aspects are discussed in detail in Section 4.3. Burton-Jones and his colleagues operationalized evaluation metrics from these four aspects and implemented them in a prototype tool called the “Ontology Auditor.” The validation results indicate that the metrics are feasible. In addition, the results highlight a wide variation in quality among ontologies. In a similar manner, Köhler et al. [33] proposed the metrics of circularity and intelligibility to evaluate the quality of the Gene Ontology (GO), which would allow ontology engineers to pinpoint flawed terms or definitions in ontologies in a systemic way. They demonstrated the potential of these methods by applying them to isolate a subset of problematic GO terms. By automatically aligning GO with other ontologies and taxonomies, they could propose alternative synonyms and definitions for some of these problematic terms. They conclude that their methods provide triable indications of the quality of terms and

definitions in ontologies. Gangemi et al. [27] defined three dimensions for ontology quality evaluation, namely, structural, functional, and usability-related. The structural dimension focuses on syntax and formal semantics. The functional dimension focuses on measuring how well an ontology is serving its purposes. The usability-related dimension checks whether the ontology profile has metadata about the ontology and its elements. For example, it measures how easy it is for the user to recognize its properties and how easy it is to determine which one is more computationally suitable for a given task. Orme et al. [46] examined the quality, completeness, and stability of ontology data as ontologies evolve. They compared automatic measurements and human evaluations of tourist ontologies and found that the software complexity metrics could be successful in automatically evaluating the complexity and cohesiveness of an ontology. They proposed a metrics suite based on standard software quality concepts to measure the complexity and cohesion of ontology data. They concluded that several of their metrics successfully determined ontology complexity or cohesion.

The features of ontology extraction tools are different from those of ontology merging tools and ontology editing tools. The main feature of the ontology merging tool is to create a single coherent ontology by identifying and matching the structures and semantics of two different ontologies. Meanwhile, the ontology editing tool helps ontology engineers organize domain knowledge and make easy for them to edit existing ontologies. In contrast, the ontology extraction tool's main functions are to identify terms from the corpora, and then, to automatically produce concepts and their relations from the identified terms using appropriate extraction algorithms. Therefore, it is critical to develop an evaluation framework for ontology extraction tools. Furthermore, a new evaluation framework should consider both the functionality of a tool and the quality of a constructed ontology. As such, we attempt to propose a comprehensive framework addressing both in this paper.

3. Automatic ontology extraction tools

To evaluate ontology extraction tools using our proposed framework, we selected OntoLT, Text2Onto, OntoBuilder, and DODDLE-OWL for two reasons. First, these tools are well known as successful ontology extraction tools in the literature [7,14,19,28,52]. Second, most of other extraction tools are not available because these tools were developed as laboratory prototypes and are therefore not available to general users.

3.1. OntoLT

OntoLT [69] was developed at the DFKI GmbH, a German research center for artificial intelligence [7]. It is available as a Protégé plug-in. Protégé [70] is a widely used ontology development tool. OntoLT enables the definition of mapping rules with which concepts (i.e., Protégé classes) and attributes (i.e., Protégé slots) can be extracted from linguistically annotated text collections (see Fig. 1). However, since linguistic annotation functions are not included in OntoLT, the corpus should be annotated in advance in order to extract an ontology using OntoLT, as shown in Fig. 2. This linguistic annotation is currently provided by SCHUG [16], a rule-based system for German and English analysis. SCHUG provides annotation of part-of-speech and morphological inflection and decomposition. SCHUG is not an integrated part of OntoLT, but it can be accessed through a Web service.

The ontology extraction process with OntoLT is as follows. OntoLT provides a precondition language for defining mapping rules, which allows for the selection of particular linguistic entities in the annotated documents. A number of mapping rules are included in this tool. For example, in OntoLT, two rules are defined for mapping information from the linguistic annotation to potential Protégé classes and slots, HeadNounToClass_ModToSubclass (mapping a head-noun to a class and in combination with its modifier(s) to one or more sub-class(es)) and SubjToclass_PredToSlot_DObjToRange (mapping a linguistic subject to a class, its predicate to a corresponding slot for this class and the direct object to the range of the slot). In addition, users can simply generate mapping rules for all possible XML-elements in the linguistic annotation either manually or by integrating a machine learning process. The preconditions can also be used to check certain conditions of linguistic entities such as whether the subject corresponds to a certain semantic class. These preconditions are implemented as XPATH expressions over the linguistic annotation. If the precondition is satisfied, the mapping rule activates one or more operators which create classes, slots, and instances from linguistic entities. Depending on which preconditions are satisfied, the corresponding operator will be activated to create classes and slots. Through this interactive process, classes and slots are automatically generated into a new ontology or integrated into an existing one [7].

3.2. Text2Onto

Text2Onto [71], a successor of Text-To-Onto, was developed at the AIFB Institute of the University of Karlsruhe in Germany [35]. As shown in Fig. 3, it interoperates with graphical environments for ontology-based applications such as KAON [64] and the NeOn toolkit [66], which can be used by clients such as Text2Onto and OI-Modeler [67]. OI-Modeler is a graphical ontology editor for creating and maintaining ontologies, which was also developed at the AIFB Institute.

Text2Onto combines machine learning approaches with basic linguistic processing techniques such as tokenization, lemmatizing, and shallow parsing. Text2Onto is based on the GATE[2] framework [12] for linguistic processing. Linguistic processing in Text2Onto begins with tokenization and sentence splitting. The resulting annotation set serves as an input for a POS tagger that assigns appropriate syntactic categories to all tokens. Finally, lemmatizing or stemming is conducted through a

[2] As a framework for language engineering, General Architecture for Text Engineering (GATE) has provided a reusable design for a language engineering software system and a set of prefabricated software building blocks that language engineers can use, extend, and customize for their specific needs [12].

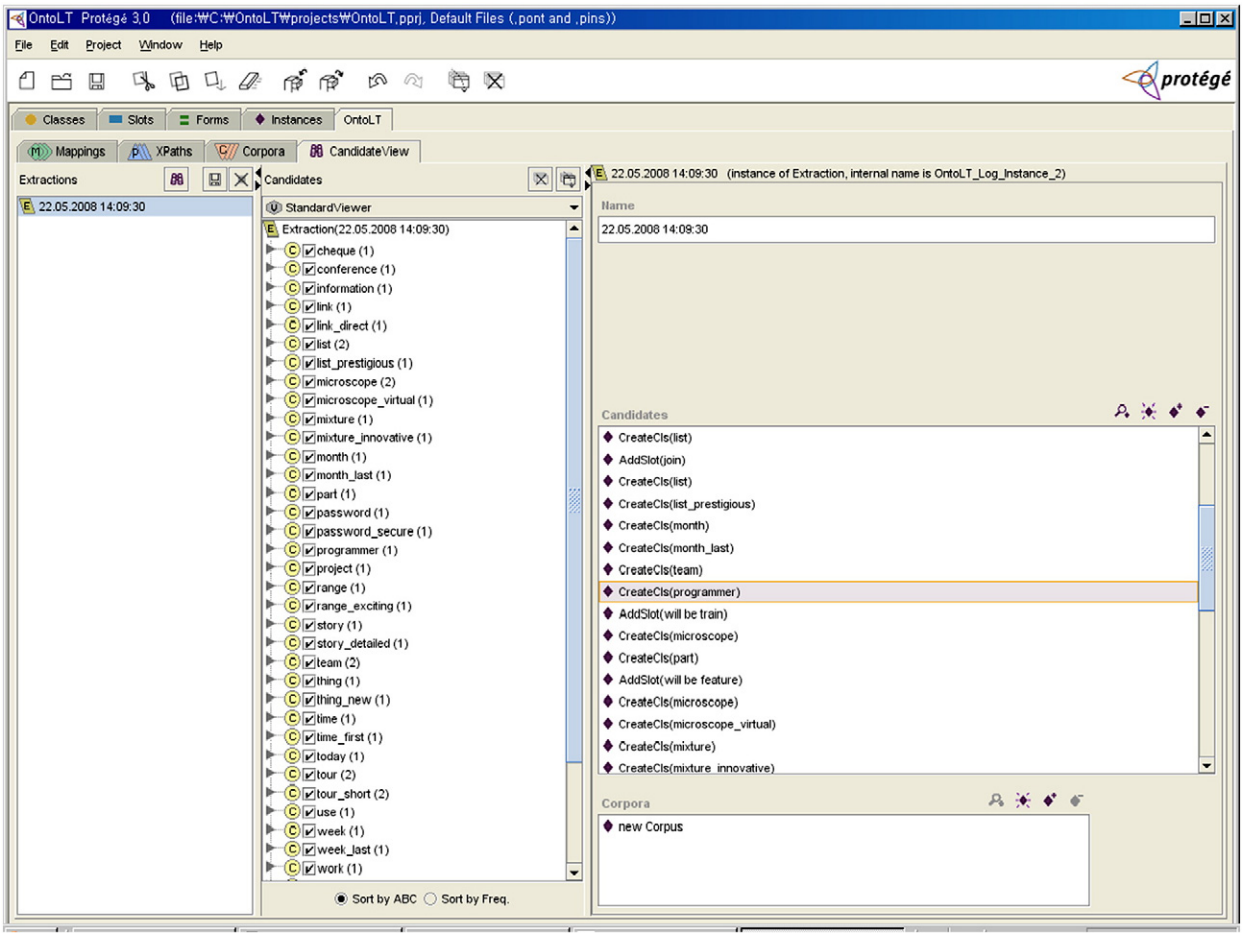

Fig. 1. The extraction result of the OntoLT plug-in of Protégé.

morphological analyzer and stemmer, respectively. The learning process then begins based on machine learning and linguistic heuristics in order to identify concepts and relations. Text2Onto includes several measures to assess the relevance of a certain term with respect to the corpus in question. In particular, Text2Onto contains several algorithms calculating the following measures:

```
    <phrase from="34" id="14" phraseType="PP" to="36"/>
    <phrase from="41" id="15" phraseType="SUBORD_CLAUSE" to="69"/>
</phrases>
<text>
    <token id="16" pos="NN" str="Adelaide">
        <lemma id="17">Adelaide</lemma>
    </token>
    <token id="18" pos="MD" str="can">
        <lemma id="19">can</lemma>
    </token>
    <token id="20" pos="MD" str="be">
        <lemma id="21">be</lemma>
    </token>
    <token id="22" pos="RB" str="pretty">
        <lemma id="23">pretty</lemma>
    </token>
    <token id="24" pos="JJ" str="damn">
        <lemma id="25">damn</lemma>
```

Fig. 2. Annotated source file for OntoLT.

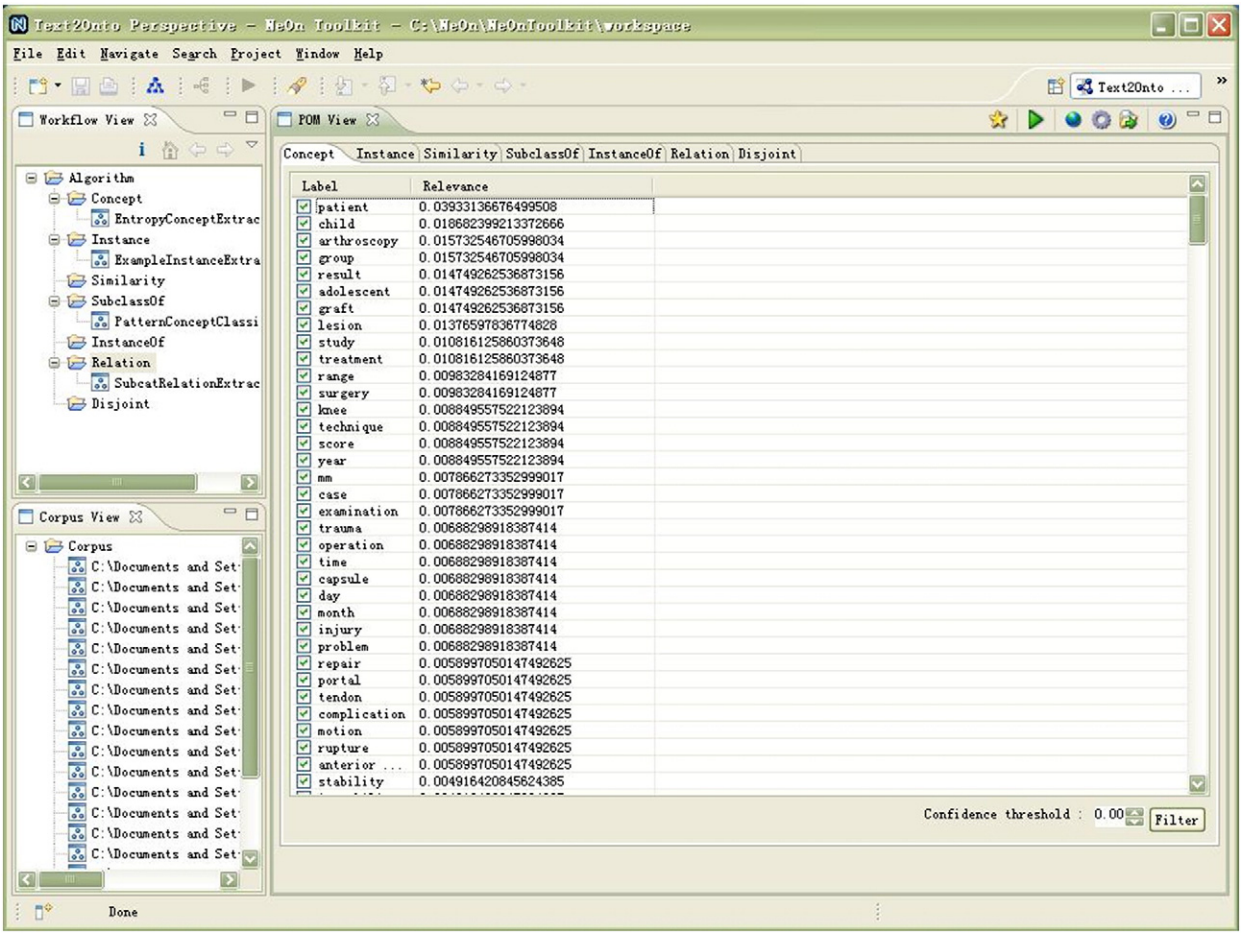

Fig. 3. The term extraction process of Text2Onto based on the NeOn toolkit.

Relative Term Frequency (RTF), Term Frequency Inverse Document Frequency (TFIDF), and entropy. Furthermore, there are various algorithms exploiting the hypernym structure of WordNet [24], matching Hearst patterns [31], and applying linguistic heuristics [35] in order to learn relations.

Text2Onto uses free texts, semi-structured texts, dictionaries, legacy ontologies, and databases as its input sources. The output of the extraction process (also called a "learning process") is a domain ontology that contains domain-specific and domain-independent concepts. Domain-independent concepts are removed to better adjust the vocabulary of the domain ontology. Therefore, the result of the process is a domain ontology that contains only the domain concepts learned from the input sources. The whole process, supervised by ontology engineers, is a cycle which is iterated to refine and complete an ontology.

3.3. OntoBuilder

OntoBuilder [68] was developed by the Israel Institute of Technology. It was originally developed for evaluating and improving automatic schema matching algorithms. OntoBuilder supports the extraction of ontologies from Web interfaces, ranging from simple search engine forms to multiple pages of reservation systems. It extracts concepts and their relations[3] based on heuristic methods learned from a training set of HTML documents. This system can create ontologies automatically and combine them into a global ontology [26].

OntoBuilder was designed to operate like a Web browser (see Fig. 4). To navigate through a page, the user simply enters the URL. Once the Web page from which the user wants to extract concepts is loaded into the OntoBuilder, it begins to build an ontology. Once a Web site is accessed by the OntoBuilder browser, each page is parsed into a data structure called a document object model. After identifying the elements of a page, OntoBuilder generates a dictionary of terms by extracting labels and field names from the Web page. It also recognizes unique relations among terms and these relations are used in the matching algorithms.

[3] The authors explain that OntoBuilder extracts an ontological structure [26]. In this paper, we used the terms "concept" and "relation" instead of "ontological structure."

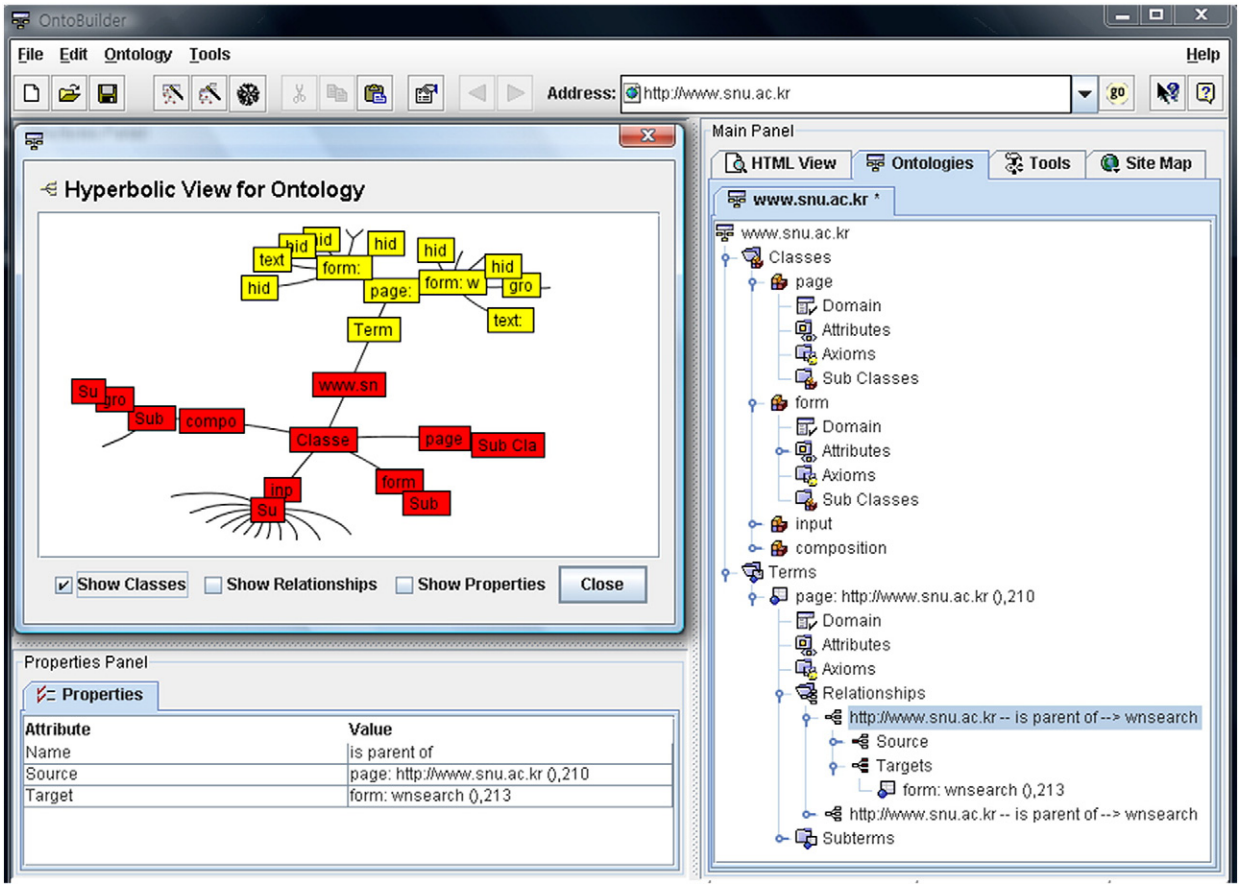

Fig. 4. The ontological structure of "www.snu.ac.kr" extracted by OntoBuilder.

3.4. DODDLE-OWL

Domain Ontology rapiD DeveLopment Environment-OWL extension (DODDLE-OWL) [63] was developed by Morita et al. [38]. It is extended from DODDLE and DODDLE II in order for it to be used for the Semantic Web. DODDLE-OWL reuses existing ontologies and supports the semi-automatic construction of taxonomic and other relations using domain-specific documents and ontologies.

It consists of four modules: input, construction, refinement, and translation. First, in the input module, a user selects several terms that are extracted by DODDLE-OWL and identifies the meaning of each term to map it to the corresponding concept in WordNet[4] (see Fig. 5). Then, it generates the basis of an ontology, that is, an initial concept hierarchy and a set of concept pairs, by referring to the reference ontologies and documents. This is done through the construction module. In the refinement module, the initial ontology generated by the construction module is refined by the user through interactive support from DODDLE-OWL. Lastly, in the translation module, the ontology constructed by DODDLE-OWL is exported into OWL language. Furthermore, DODDLE-OWL can connect with MR^3 [65] which provides a graphical editor [38].

4. The evaluation framework for ontology extraction tools

According to Hevner et al. [32], newly developed systems need to be justified in terms of their usefulness and performance using well-defined evaluation metrics. The evaluation of a built IT system requires the appropriate criteria and, possibly, the gathering and analysis of appropriate data. Hevner et al. defined two main activities, system building and justification/evaluation, in their proposed framework for information system research. Moreover, they emphasized that each developed system must yield utility for the specified problem. Hence, thorough evaluation of the system is a crucial activity.

We identified important tool evaluation criteria and developed an evaluation framework that will guide us in assessing the quality of ontology extraction tools. Our proposed framework includes criteria to evaluate both the functional performance of the extraction tools and the quality of the constructed ontology. In this section, we discuss our proposed evaluation framework.

[4] DODDLE-OWL uses two reference ontologies, WordNet [24] and EDR Electronic Dictionary [61]. It requires WordNet when constructing English ontologies, while the EDR Electronic Dictionary is used for Japanese ontologies.

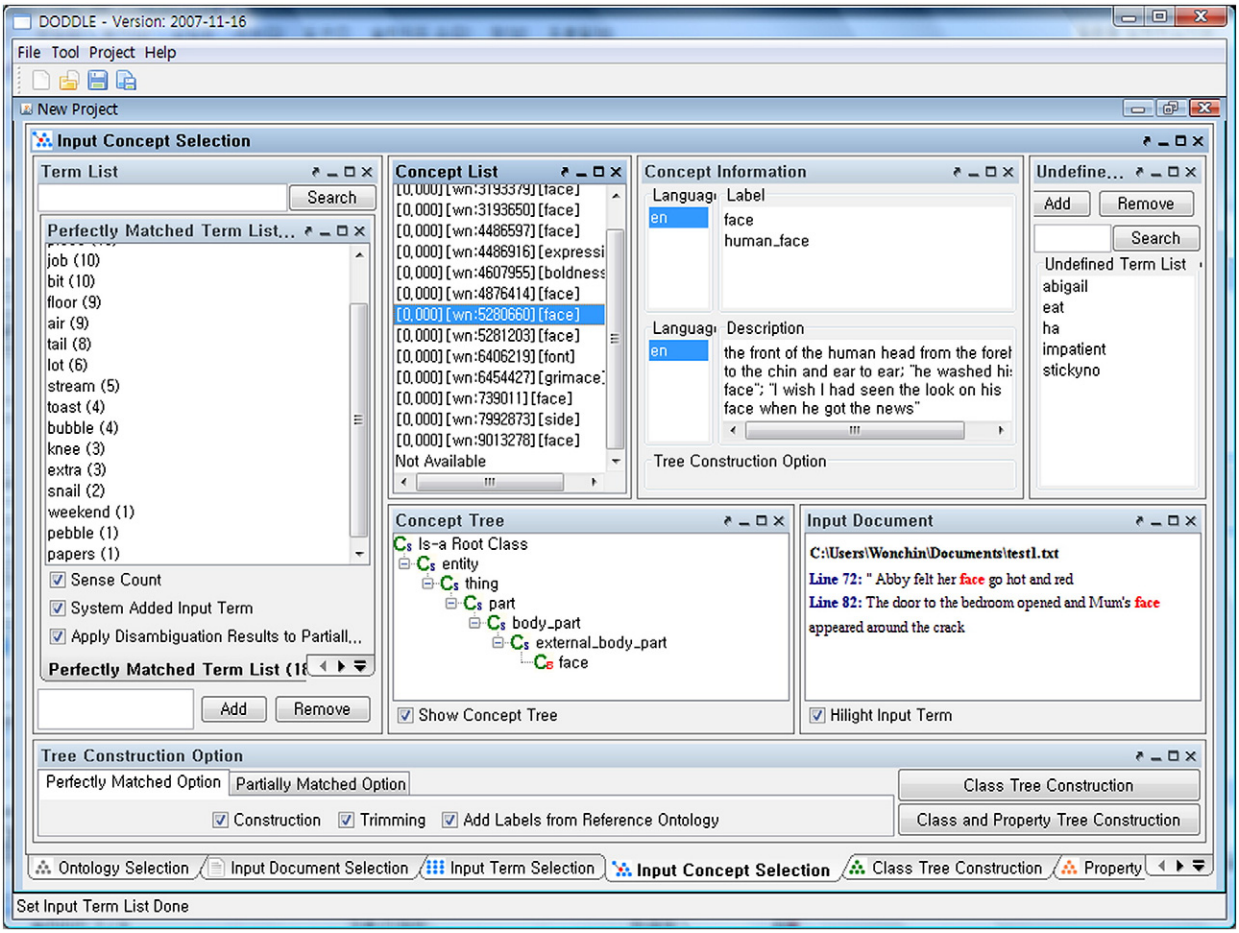

Fig. 5. Identifying the meaning of each term in WordNet with DODDLE-OWL.

As shown in Fig. 6, the criteria are grouped into three features. The first one deals with general features that are related to a tool's interface and convenience. The general features evaluate the built-in functions of each tool and have three criteria: user interface, availability, and time-to-first-use. These features may be measured during the entire process it is used. However, users may also assess these general features without actually building ontologies. In other words, users tend to handle and investigate the software programs' general and exterior features such as user interface before they get down to using the main functions of each program (see Section 4.1). The second set of features are used to evaluate the main functions used for ontology extraction such as whether

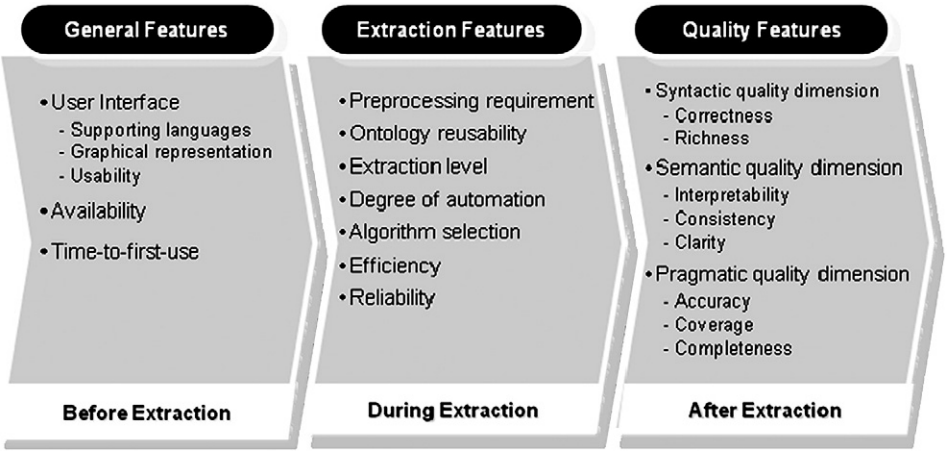

Fig. 6. The proposed evaluation framework for ontology extraction tools.

the users can extract concepts and their relations using an ontology extraction tool, and whether it is possible to select an extraction algorithm among various ones. Hence, this part consists of seven criteria, namely, preprocessing required, ontology reusability, extraction level, degree of automation, algorithm selection, efficiency, and reliability. These extraction features are evaluated during an ontology extraction process. The extraction features are introduced in Section 4.2. The last set of features check ontology quality. In this stage, the task of extracting an ontology has been completed, and as the next step, the quality of the constructed ontology is measured. Ontology quality is measured based on three dimensions as adopted from the work of Burton-Jones et al. [8]: syntactic, semantic, and pragmatic qualities. These consist of two or three subcriteria and details are introduced in Section 4.3.

4.1. General features

This part deals with a tool's exterior features and the convenience of its use. The first criterion is *user interface*. In many studies, user interface is considered an important feature when evaluating programs or tools [3,21,34,39]. In fact, it has been argued that user interface is the most crucial element of a computer system as its design determines eventual user acceptance and utilization [15,43]. We define three properties of the user interface: supporting languages, graphical representation, and usability. The first property, *supporting languages*, evaluates whether a tool supports an interface for multiple languages. If a tool has an option for the user to choose a familiar interface language among various languages, we consider it more flexible than those that provide only one language. *Graphical representation* checks whether a tool can display an extracted ontology visually. If a tool represents an ontology visually, we can consider that users could understand its structure easily. Furthermore, users might edit the elements of an ontology more easily with a visual aid. Lastly, we measure the *usability* of the user interface. If the user interface of a tool is complex, it may require more time to master the tool's functions [42,53].

The second criterion of this category is related with a tool's *availability*. If we can acquire a tool without too much effort, the tool is presumed to be available. For instance, if we can download a tool free of charge, we can consider it highly available. Although there exist several ontology extraction tools, most of these tools are not easily obtainable. Hence, we should first consider the availability of each tool before testing which extraction tool is better.

The last criterion is the *time-to-first-use*. This concerns the following questions: “Is the tool easy to install on a PC?,” “Does it specify additional platform requirements for installation?,” and “Does it provide installation and user manuals, and/or a help function?” If a tool is difficult to set up on a PC because it needs additional platform requirements, a longer time will be needed for its first-time use.

4.2. Extraction features

Since our framework is developed to evaluate ontology extraction tools, examining the various aspects of an extraction process performed by a tool is considered the most important part of this study. When evaluating the extraction features, we first suggest two criteria pertinent to input data. First, we need to consider whether a tool requires specific data types for source documents. For example, if a tool uses only HTML files as input data, additional efforts should be made to convert the source data type into HTML format if the source data contains plain texts. Moreover, some ontology extraction tools require the source data to be annotated linguistically. If a tool requires extra effort, it has more restraint on the tool than the ones that do not require preprocessing. Therefore, we should consider whether a tool requires additional preprocessing effort (*preprocessing requirement*). In many situations, it is more convenient to use concepts from existing ontologies when constructing a new ontology. Hence, we also need to investigate whether a tool allows the user to refer to existing ontologies, and whether it allows him/her to reuse them when constructing a new one (*ontology reusability*).

In addition, we should also consider the *extraction level* of a tool. When building an ontology, ontology engineers have to define and organize concepts and their relations. While some tools automatically or semi-automatically generate concepts only, other tools can extract various relations between concepts as well. Therefore, the extraction level examines whether a tool can automatically or semi-automatically extract concepts or both concepts and their relations. We can conclude that it is better if an ontology extraction tool can automatically extract both concepts and their relations.

In addition to the extraction level, we also consider the *degree of automation* during the concept and relation extraction. Generally, automatic extraction tools are considered to be one which requires no human intervention during the actual extraction process, but may require the user to first define some extraction rules, or to set initial parameters. In our framework, an *automatic* tool is defined as the one that performs the extraction process without human intervention, but may require the user to select the proper extraction algorithms before the extraction process starts. On the other hand, if the extraction rules must be defined by the user before extracting concepts/relations or any human intervention is required during the extraction process, these tools are classified as semi-automatic tools. When using an automatic extraction tool, whether the user defines additional extraction rules for his/her specific purpose is optional. When using a semi-automatic tool, however, the user is forced to define the extraction rules before extracting concepts and/or relations, or he/she should intervene to set parameters. Note that in many cases, a tool named “automatic ontology extraction tool” is not fully automatic. We therefore need to investigate whether those tools that claim to be “automatic” are actually automatic. In this sense, the degree of automation literally measures whether a tool can extract concepts (and relations) automatically or semi-automatically.

Next, we need to consider whether a tool provides the user with an option for selecting a desirable extraction algorithm for his/her specific need. We assume that it is better if the user has an option to select an algorithm from various extraction algorithms. Each algorithm might be developed to deal with a distinct type of source data. If a tool provides various algorithms, the user is able

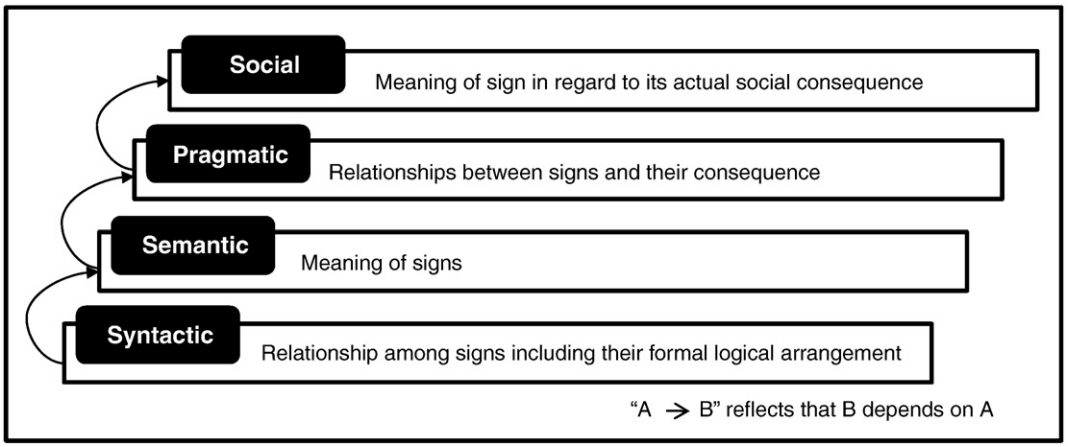

Fig. 7. Semiotic framework proposed by [8].

to extract a more suitable ontology for what he/she wants to construct. For example, some algorithms are good for distinguishing various forms of a verb, whereas the extraction result may include redundant and unnecessary forms. Other algorithms are more suitable for extracting fundamental terms, but the result of extraction may not be fully exploited. Hence, we need to look into whether a tool provides various algorithms from which the user can choose.[5] This criterion is called *algorithm selection*. Note that adding additional algorithms to a tool might be realized using plug-ins. In other words, a tool could be extended with new modules (i.e., new algorithms) by adding plug-ins.

Finally, like other software programs, the *efficiency* of a tool should be considered. Efficiency measures how fast a tool can extract concepts and/or relations from source data under stated conditions. This measure computes the total time taken by the tool to read the source data and to present the target output to the user. Some tools may take a longer time to extract concepts due to their inefficient extraction algorithm. If tools present similar extraction results, it is better to choose more efficient ones. Another measure is *reliability*. Reliability reviews whether the output remains consistent over repeated tests with the same input data under identical conditions. If a tool shows different extraction results from the same input data, we can conclude that the tool and its results are not reliable. Note that this criterion examines whether a tool produces consistent results, not the quality of the output.

4.3. Quality features

After evaluating the ontology extraction process, the next step is to assess the quality of an ontology constructed using a tool. A constructed ontology employed by a specific application is typically used for its own distinct purpose; thus, it is important to judge whether the ontology satisfies quality requirements to be effectively deployed in a practical application. Hence, the quality features measure the utility of an ontology. To evaluate an ontology, some studies employed a formal ontology as a benchmark [10,58]. However, there is no way to validate whether such a formal ontology is ideal for benchmarking [59]. Moreover, formal ontologies are either too high level or too philosophical to reflect pragmatic issues. On the other hand, Burton-Jones et al. [8] developed a set of measures to evaluate ontology quality. Their research was based on a semiotic framework (the more general theoretical framework derived from linguistics) which explicitly deals with pragmatic issues. Therefore, in this research, we adopt their well-defined criteria to measure the quality aspects of an ontology. The semiotic framework proposed by Burton-Jones et al. consists of criteria for syntactic, semantic, pragmatic, and social qualities (see Fig. 7). The latter quality is a higher level, and each higher-level quality depends on its preceding one; for example, semantic quality is a higher level than syntactic quality and it depends on syntactic quality. Among these qualities, social quality deals with ontology maintenance and evolution issues. It measures the extent to which other ontologies rely on the ontology that the user wants to evaluate as well as the number of times it has been used. The social quality feature is excluded in our study because it requires continuous monitoring and examination not feasible within our limited experimental timetable. Thus, in our framework, we only adopt their syntactic, semantic, and pragmatic qualities.

First, the *syntactic quality dimension* measures the syntax[6] of the ontology generated by a tool. It has two attributes: correctness and richness. *Correctness* checks the degree to which the syntax of an ontology conforms to the specific syntactic rule of the target language. In other words, it measures the correctness of the syntax. Not all ontology editors have syntactic error-checking capabilities. An ontology cannot be read and used properly without correct syntax. *Richness* refers to the proportion of all syntactic constructs used in a constructed ontology, which are provided by the target ontology language. In other words, it measures how many syntactic vocabularies are used to describe an ontology. For example, let us assume that an ontology uses only two types of syntactic vocabularies: *subclass* and *type*. Another uses various types of syntactic vocabularies such as *class* and *subclass*, *property*

[5] Note that it is only useful to have this option if the user is given some guidance on the kind of algorithm used in a specific situation, or if the user is familiar with the algorithms in the ontology extraction tool.

[6] Note that it means ontology syntax not linguistic syntax.

and *subproperty*, *inverse*, *intersection*, *domain*, *range*, and *cardinality*. In such a case, we can conclude that the latter ontology is richer than the former in terms of syntactic quality. The richness measures the breadth of syntax being used.

Second, the *semantic quality dimension* examines the meaning of terms expressed in a constructed ontology. It has three attributes: interpretability, consistency, and clarity. *Interpretability* investigates whether each concept in the ontology has a proper meaning as used in the real world. It is measured by examining the existence of the individual word by checking the lexicon used in the real world. *Consistency* measures whether the meaning of individual terms is consistently used throughout the ontology. For example, if an ontology claims that *X* is a subclass of *Y*, and *Y* is a property of *X*, *X* and *Y* have semantically inconsistent meanings and thus no semantic value. Lastly, *clarity* evaluates whether the context of each term in an ontology is not ambiguous. In other words, clarity checks how well the name of a class or a property is expressed by explicit words, not polysemous words. A polysemous word is one having more than one meaning and can be used in different contexts to express two or more different meanings. For example, the word "bear" is polysemous. If an ontology claims that the class "bear" has a property "stock," a software agent should know that "bear" describes a selling party in a stock market, not an animal. There are various ways to measure clarity. For example, Burton-Jones et al. measure clarity by counting the number of word senses in WordNet for the class or property name as a whole.

Finally, the *pragmatic quality dimension* evaluates an ontology's usefulness for users or their software agents regardless of its syntax and semantics. It has three attributes: accuracy, coverage, and completeness. *Accuracy* measures whether the claims the ontology makes are true. For example, let us assume that there is a UNIVERSITY ontology. If this ontology models *Student* as a subclass of *Department*, *Student* should inherit all the properties of *Department*. However, this is not an accurate representation of the real world. In general, we can think that a student belongs to a department; thus, a student is a subset of a department. However, because a student is not a subclass of a department, "student" should not inherit the properties of "department." *Coverage*[7] measures the overall size of a constructed ontology. If the coverage of an ontology is larger, it is more likely to provide more knowledge to the user. Lastly, *completeness*[8] measures whether an ontology provides the syntactic vocabularies needed by a specific application. For example, if an ontology provides only class/subclass and property information, it is not a complete ontology for an application that needs cardinality information.

5. An experiment of automatic ontology extraction tools

In this section, we present the evaluations of the ontology extraction tools. We tested four ontology extraction tools introduced in Section 3. The evaluations of the tools were carried out using two methods. First, we examined each tool's functionality and then compared the performance of the tools based on our evaluation framework in Section 5.1. To compare the tools objectively, we used the same corpus consisting of 15 files related to Asia tourism. The original dataset was HTML documents, which were used as an input source for OntoBuilder. Then, for Text2Onto and DODDLE-OWL, we extracted plain text files from HTML documents. Finally, the text files for OntoLT were annotated because the corpus for OntoLT has to be linguistically annotated in advance.

Second, we used an empirical approach to assess and rank the ontology extraction tools. Four participants evaluated them using the Analytic Hierarchy Process (AHP) method. Since the evaluation process of ontology extraction tools may be considered the selection process of choosing the right tool, we applied the AHP method, which is one of the more popular multi-criteria decision-making methods. The AHP method [49] is especially suitable for a comparison of decision elements that are difficult to quantify. This is a widely used and well tested method among available techniques to solve the multi-attribute problems. Two experts and two PhD candidates participated in our experiment. The two experts have extensive experience in ontology construction. One expert has been involved in several Semantic Web development projects as well as in teaching ontology and the Semantic Web to graduate students. The other is an ontology engineer who has several years of working experience in developing Semantic Web platforms. The two PhD candidates are currently conducting ontology-related research, and have developed ontologies for their theses. We asked the participants to use four ontology extraction tools. After using the tools, they evaluated them using the AHP method. The empirical test results using the AHP method are presented in Section 5.2.

5.1. Evaluation of ontology extraction tools

The primary goal of our experiment is to evaluate the functional performance of current ontology extraction tools. In this section, we discuss the characteristics of ontology extraction tools according to our proposed evaluation criteria.

5.1.1. General features

The general features include three criteria: user interface, availability, and time-to-first-use. Meanwhile, the user interface has three attributes: supporting languages, graphical representation, and usability. All the tools that we tested only support English as the interface language. An evaluation of the second attribute, graphical representation, revealed that Text2Onto and OntoBuilder provide the ability to display a constructed ontology visually, which makes it easier for the user to understand the ontology structure. In contrast, DODDLE-OWL and OntoLT do not include a visualization module. DODDLE-OWL can represent the

[7] Originally, the name of this criterion was "comprehensiveness" in the work of Burton-Jones et al. [8]. However, we think that "coverage" is more appropriate than "comprehensiveness" as a criterion for measuring the overall size of a constructed ontology. In this paper, we use "coverage" instead of "comprehensiveness."

[8] Burton-Jones et al. [8] used the term "relevance" as a criterion for evaluating whether or not an ontology provides the information types needed by a specific application. However, completeness is closer than relevance to describe the degree of providing various types of information. Therefore, we use completeness instead of relevance in our framework.

Table 1
The extraction levels of four ontology extraction tools.

	OntoLT	Text2Onto	DODDLE-OWL	OntoBuilder
Concept	✓	✓	✓	✓
Relation	✓	✓		✓
Taxonomy		✓	✓	

constructed ontology visually by connecting with MR[3]. OntoLT can use the visualization module in Protégé. Lastly, the user interface of OntoLT is somewhat less user-friendly than other tools. In OntoLT, uploading corpus and extracting candidate concepts or relations are executed in separate tabs. However, we think that it does not need to keep a tab for uploading corpus apart from a tab for extracting candidate concepts. It is more convenient and efficient to upload its input corpus once in an extraction tab. On the other hand, OntoLT shows both concepts and relations in a candidate view. However, in this case, it is better to show the extracted concepts apart from the extracted relations because it looks complicated and distracting if it displays too many candidate results.

The next criterion is availability of tools. All the tools we tested are available on the Web. In particular, Text2Onto and DODDLE-OWL can be obtained from SourceForge.net. SourceForge.net is the world's largest open source software development Web site and provides free access to all the registered tools. On the other hand, OntoLT and OntoBuilder are available from project Web sites.

The last criterion of the general features is the time-to-first-use. All the tools require a Java virtual machine because they are all Java-based. It is not difficult to install the Java virtual machine on a PC and it is available on the Web free of charge. OntoBuilder is not difficult to set up on a PC because it does not require another platform in addition to the Java virtual machine and provides installation and user manuals in English. However, OntoLT is a little cumbersome to use. As mentioned earlier, OntoLT is a plug-in of Protégé; thus, Protégé must be pre-installed on a PC and the user must know how to install plug-ins. DODDLE-OWL uses text files as source data by default. However, if the user wants to extract words from documents not in text format, he/she can use xdoc2txt [72]. xdoc2txt can convert Microsoft Word, Microsoft Excel, Microsoft Power Point, and Adobe PDF documents into text files automatically. Note that converting formats such as Microsoft Word, or Adobe PDF document into text files is often inaccurate, especially where complex tables and figures are included.

5.1.2. Extraction features

As discussed in Section 4.2, the extraction features include seven criteria: preprocessing requirement, ontology reusability, extraction level, degree of automation, algorithm selection, efficiency, and reliability. There were several tools we tested which did not require the input data to be preprocessed. However, OntoLT requires preprocessing. In OntoLT, the corpus must be annotated using its proprietary XML format which includes part-of-speech tags and morphological analysis. As the most common form of corpus annotation, part-of-speech tags mark up the individual words in a text which correspond to a particular part of speech based on both their definitions and contexts. Morphological analysis is a basic procedure in natural language processing, which examines the internal structure of words.

Some of the tools we tested reuse existing ontologies. Text2Onto can employ existing ontologies as its input source. DODDLE-OWL uses WordNet and EDR, which are linguistic ontologies in English and Japanese, respectively. Furthermore, DODDLE-OWL supports the semi-automatic construction of taxonomic and other relations referring to other ontologies.

As shown in Table 1, the evaluation results of extraction-level show that Text2Onto, OntoLT, and OntoBuilder are capable of extracting both concepts and relations from a corpus. DODDLE-OWL extracts both concepts and their taxonomies.[9]

The tools we examined have different degrees of automation. With OntoLT, relations are extracted only by a semi-automatic method. When the user constructs an ontology using OntoLT, he/she must check to see whether the extracted relations are meaningful. In contrast, Text2Onto is capable of extracting both concepts and their relations semi-automatically and automatically. In other words, with Text2Onto, the user has the option of choosing automatic extraction or semi-automatic extraction when constructing an ontology. If the user selects the semi-automatic method, he/she can examine and verify the result before sending it to the ontology editor. DODDLE-OWL extracts concepts automatically but taxonomies semi-automatically, while OntoBuilder can extract only concepts automatically.

Table 2 shows the numerical comparison of the extracted outputs using the same corpus related to tourism. Each text file contains around 800–1500 words and one of them is shown in Fig. 8. Note that the outputs in Table 2 are the result of the fully automated extraction without any user intervention. All experiments were carried out on a Windows 2002 with Intel Pentium processors (3.40 GHz) and 1 Gb of memory.

As shown in Table 2, DODDLE-OWL takes a longer time to extract concepts. If extraction time takes less than 30 s, the efficiency is regarded as “high,” 30–60 s is “normal,” and more than 60 s is “low.” The results reveal that OntoLT, Text2Onto, and OntoBuilder are highly efficient while the efficiency of DODDLE-OWL is lower than other tools.

Our evaluation of the algorithm selection shows that OntoLT and Text2Onto are more flexible than the other tools. Both OntoLT and Text2Onto include, by default, various extraction algorithms and allow the user to define additional algorithms. On the other hand, when using OntoBuilder and DODDLE-OWL, users have no option to select a desirable extraction algorithm because they do not incorporate various ones.

[9] DODDLE-OWL generates a concept hierarchy by referring to its reference ontology (i.e., WordNet). A concept hierarchy is constructed as taxonomic relations.

Table 2
The numerical comparison of the automatically extracted outputs.

	OntoLT	Text2Onto	DODDLE-OWL	OntoBuilder
Concepts	152	370	711	15
Relations	24	10		1
Elapsed time	2″	2″	109″	15″

5.1.3. Quality features

As stated earlier, we adopted three criteria from Burton-Jones et al.'s framework [8] to evaluate ontology quality: syntactic, semantic, and pragmatic quality. The syntactic quality dimension has two attributes: correctness and richness. The correctness examines whether the syntax of the ontology constructed using each tool tested in this study conforms to each language's syntactic rules. All the tools we tested produced ontologies that were syntactically correct. In terms of richness, all the tools except OntoBuilder used various types of syntactic vocabularies to describe ontologies. Ontologies constructed using OntoBuilder used limited syntactic vocabularies such as class, domain, attribute, and axiom. On the other hand, ontologies built using other tools used various types of syntactic vocabularies including class and subclass, property and sub-property, inverse, cardinality, and so on.

The next criterion is semantic quality. It consists of three attributes: interpretability, consistency, and clarity. As the first evaluation attribute of semantic quality, interpretability measures whether the concepts in an ontology represent a correct meaning as used in the real world. The concepts in ontologies constructed by OntoLT, Text2Onto, and OntoBuilder have proper meanings that can be interpreted in the real world. However, we did not evaluate the interpretability of the ontologies created by DODDLE-OWL. Since DODDLE-OWL does not automatically assign the meaning of each term, unlike other tools doing so automatically, it is not feasible to evaluate the interpretability of pure concepts produced by DODDLE-OWL. In other words, when using DODDLE-OWL, the ontology engineer should manually identify and check the meaning of each term in order to map it to the corresponding concept in WordNet. In terms of consistency, the terms generated by OntoLT, Text2Onto, and OntoBuilder have consistent meanings in ontologies. The last attribute of semantic quality is clarity. This attribute evaluates how well the name of a class or property is expressed by explicit words, not polysemous words. If a term is polysemous, its context may be ambiguous. To measure the clarity, we used the Burton-Jones et al.'s method that measures clarity by counting the number of word senses in WordNet for concept or relation names. We examined terms of each ontology's top 20 concepts that were most frequently occurred in a corpus. Most of top 20 concepts in each ontology were general and universal terms related to travel and geographic location, and they did not have multiple meanings. This result may be due to the simple tourism data set we used as an input source. Depending on the input data, the evaluation result of clarity could be different from this. If an input dataset contains many technical or specialized terms with multiple meanings, the clarity of ontology constructed using the input data might naturally be unclear.

Lastly, pragmatic quality is measured by three attributes: accuracy, coverage, and completeness. Accuracy is determined by checking the information given by the ontology against existing knowledge known to be true. Thus, this attribute must be subjectively evaluated by a human expert. All the tools we tested were accurate. In terms of coverage, most of the tools were capable of building a large ontology. On the other hand, the coverage of the ontology generated by OntoBuilder is limited. OntoBuilder was originally developed in an attempt to map between the structure of a specific Web site's HTML page and that of another site's HTML page. Thus, OntoBuilder's objective is not to investigate and extract all the relations among concepts but to understand the Web site's overall structure by extracting concepts and their relations existing in the HTML page. Although we tested OntoBuilder with various sizes of Web sites, the extracted ontology was not large, and therefore its coverage was limited. Our evaluation of the completeness showed that the ontology created by OntoBuilder is less complete than the rest of tools. As mentioned in the richness criteria of the syntactic quality dimension, the ontologies generated by OntoBuilder use a limited set of syntactic vocabularies that include only class, domain, attribute, and axiom. Therefore, if a specific application needs cardinality information, the ontology constructed by OntoBuilder is not complete. Thus, since the ontologies constructed using OntoBuilder use a few types of syntactic vocabularies, it is less complete than the other tools.

```
Seoul is beautiful at all times of the year--your visit depends
on your tastes. Autumn (September-November) is the most popular
time, with fine weather and amazingly colorful forests. Winter-
-if you can stand the cold--is also magnificent. The ice and
snow show off Seoul in a flattering way. There is a short,
magnificent cherry blossom season in early spring (February).
Summer is probably the least attractive time to visit--it's
warm and very wet.

The worst times to be in Seoul as a traveler are during holiday
periods. Accommodation is expensive, transport is crowded and
people are everywhere. Especially avoid Lunar New Year (First
day of the first moon).

                                .
                                .
                                .
```

Fig. 8. A partial example of an input text file.

5.2. Application of the AHP for selecting the ontology extraction tool

The AHP developed by Saaty [49] is a practical approach in solving relatively complex multi-criteria decision-making problems. Based on mathematics and psychology, it provides a comprehensive and rational framework for structuring a decision problem, representing and quantifying its elements, relating those elements to overall goals, and evaluating alternative solutions. It is used in a wide variety of decision situations in fields such as government, business, industry, and education [20,48].

The AHP is a selection process that consists of the following four steps:

1. Decide upon the criteria for selection.
2. Rate the relative importance of these criteria using pair-wise comparisons.
3. Rate each potential choice relative to each other choice on the basis of each selection criterion (this is achieved by performing pair-wise comparisons of the choices).
4. Combine the ratings derived in steps 2 and 3 to obtain an overall relative rating for each potential choice.

To apply the AHP method, first, we created a logical hierarchy and MS EXCEL-based questionnaire as shown in Figs. 9 and 10. The main criteria and sub-criteria for the evaluation and selection of ontology extraction tool consist of the evaluation criteria we proposed in Section 4. As recommended by Saaty [50], the scale we used is 1 through 9, with 1 meaning no difference in importance of one criterion in relation to the other and 9 meaning one criterion is extremely more important than the other with increasing

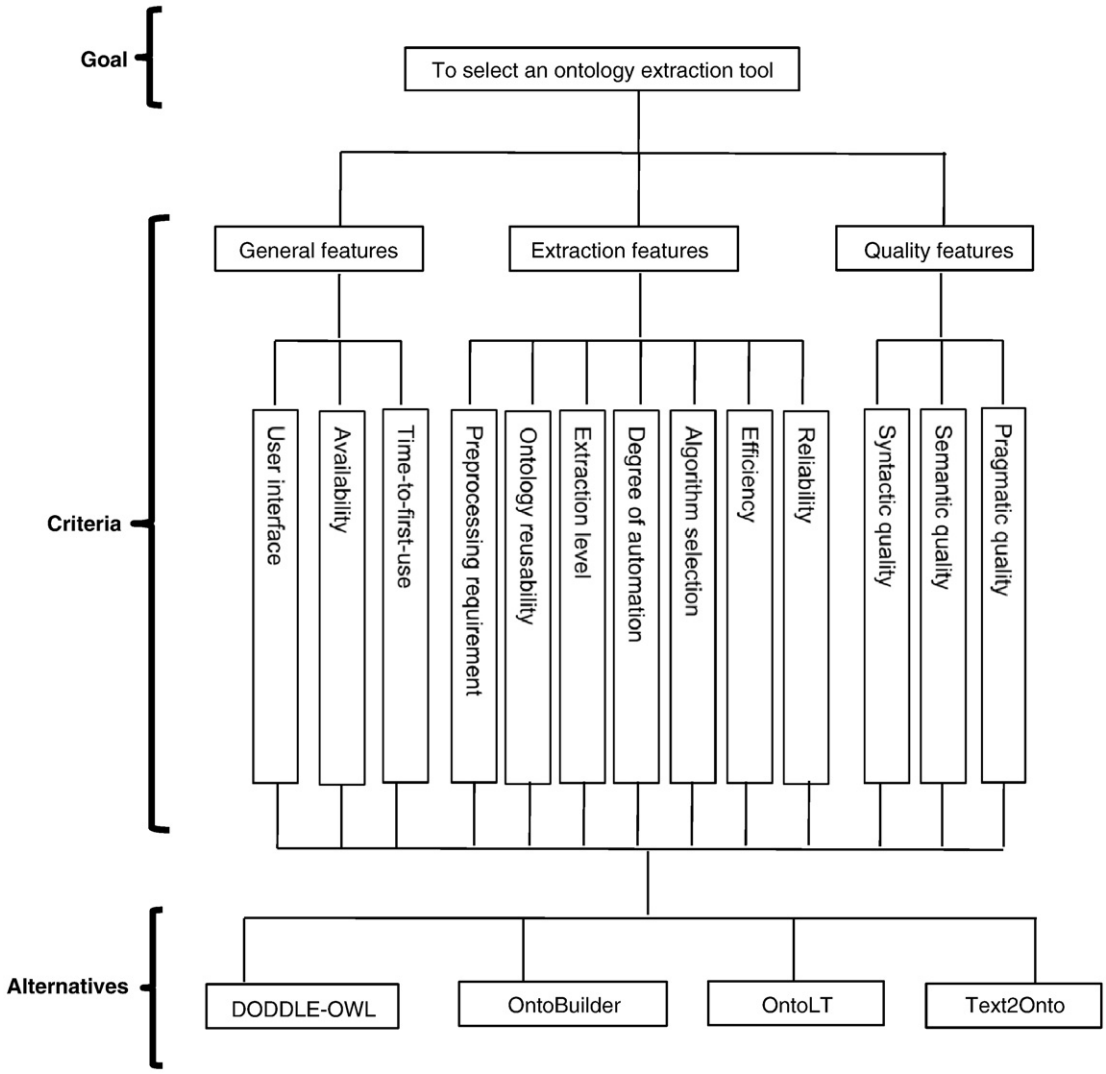

Fig. 9. Schematic diagram of the AHP model for ontology extraction tool selection.

Pair-wise Comparison

1. Compare the relative importance with respect to **"Ontology Extraction Tool Selection"**

	Extreme	Very Strong	Strong	Moderate	Equal	Moderate	Strong	Very Strong	Extreme	
General Features								o		Extraction Features
General Features							o			Quality Features
Extraction Features				o						Quality Features
	9	7	5	3	1	1/3	1/5	1/7	1/9	

2. Compare the relative importance with respect to **"General Features"**

	Extreme	Very Strong	Strong	Moderate	Equal	Moderate	Strong	Very Strong	Extreme	
User Interface							o			Availability
User Interface						o				Time-to-first-use
Availability			o							Time-to-first-use
	9	7	5	3	1	1/3	1/5	1/7	1/9	

3. Compare the relative importance with respect to **"Extraction Features"**

	Extreme	Very Strong	Strong	Moderate	Equal	Moderate	Strong	Very Strong	Extreme	
Preprocessing Requirement			o							Ontology Reusability
Preprocessing Requirement						o				Extraction Level

Fig. 10. Pair-wise comparison questionnaire (part).

degrees of importance in between. In addition, individual scores that are rated by four participants are aggregated in order to have a single weight for each criterion and single priority for each ontology extraction tool by calculating the arithmetic mean.[10]

The four participants then systematically evaluated various criteria and ontology extraction tools through pairwise comparison. Each participant's evaluation result is summarized in the pairwise comparison matrix. The pairwise comparison matrix is constructed as follows:

Let *C*1, *C*2 … *Cn* be the set of elements, while *aij* represents a quantified judgment on a pair of elements *Ci* and *Cj*. The pairwise comparisons are made in terms of which element dominates another (i.e., based on the relative importance of elements). If element *Ci* dominates over element *Cj*, then the whole number integer (*aij*) is entered in row *Ci*, column *Cj* and reciprocal (1/*aij*) is entered in row *Cj*, column *Ci* [49].

Having done all pairwise comparisons, weight for each criterion is determined. The column of numbers is normalized by dividing each entry by the sum of all entries. The average of normalized values of each row is taken as the weight of the respective criterion of the row. Table 3 indicates that the selection decision is dominated by three criteria that have more weight compared to others. Those criteria are extraction level, reliability, and semantic quality.

Finally, numerical priorities were calculated for each of the ontology extraction tools. The priority score of each ontology extraction tool is calculated as follows:

$$a_j = \sum_i w_i k_{ij}$$

where

$a_j =$ overall relative rating for ontology extraction tool j
$w_i =$ average normalized weight for criterion i
$k_{ij} =$ average normalized rating for ontology extraction tool j with respect to criterion i.

The priority score represents the ontology extraction tools' relative ability to achieve the decision goal. As shown in Table 4, the best suited ontology extraction tool is concluded to be Text2Onto with a global priority of 0.3623.

Using Fleiss' Kappa index, we checked the consistency between heterogeneous inclinations toward expertise. The Kappa index may be used as a proxy instrument to conduct a power test, which is widely adopted in order to acknowledge a proper sample size with statistical assumptions. It ranges from 0 (chance agreement) to 1 (perfect agreement), and generally a Kappa>0.7 is

[10] Generally, there are two methods for aggregating the individual values when multiple raters participate in the AHP test: geometric mean and arithmetic mean.

Table 3
Weights for criteria.

Main criteria	Sub-criteria	Importance	Weight [a]
General features (0.0738 [b])	User interface	0.1022	0.0075
	Availability	0.6864	0.0506
	Time-to-first-use	0.2114	0.0156
Extraction features (0.6434 [b])	Preprocessing requirement	0.0883	0.0568
	ontology reusability	0.0337	0.0217
	Extraction level	0.2453	**0.1579**
	Degree of automation	0.1490	0.0959
	Algorithm selection	0.0575	0.0370
	Efficiency	0.0206	0.0132
	Reliability	0.4055	**0.2609**
Quality features (0.2828 [b])	Syntactic quality	0.2605	0.0737
	Semantic quality	0.6333	**0.1791**
	Pragmatic quality	0.1062	0.0300

[a] Weight = Main criteria importance × Sub-criteria importance.
[b] Main criteria importance.

considered satisfactory. In our experiment, the reliability for the participants was found to be Kappa = 0.72, implying that the participants' four rating scores are significantly consistent.

To sum up, in our experiments, Text2Onto ranked the best among four ontology extraction tools. This may be due to the following reasons: First, Text2Onto not only extracts both concepts and relations automatically, but also provides various input data types, including text files, XML, HTML, and Adobe PDF. Second, Text2Onto is very flexible because it provides a variety of algorithms from which the users can choose from. In addition, Text2Onto has good scores in terms of quality. Ontologies constructed by Text2Onto use various types of syntactic vocabularies and are semantically clear since Text2Onto identifies the meanings of polysemous words using WordNet-based extraction algorithms. Although Text2Onto received the best score among four tools in our experiments, our evaluation result is not carved in stone. In other words, it is difficult to assert that Text2Onto is the best tool because it largely depends on what the user wants to do with it. For example, if the user cares about generating as many concepts as possible, or if the user wants to extract concepts from Japanese texts, DODDLE-OWL is the best among the four tools. If the user cares about generating more relations, OntoLT may be the top option.

5.3. Current issues of ontology extraction tools

This section discusses current issues of ontology extraction tools based on the results of our experiments. Current ontology extraction tools have good reliability and graphical representation, and do not require much preprocessing effort. However, the following should be improved.

First, although ontology extraction has been studied extensively, further efforts must be made to develop and improve algorithms used for ontology extraction. Such algorithms typically employ either a statistical or a symbolic approach. Statistical approaches are easier to compute and implement, while symbolic ones are more precise and provide more reasonable results because generally, they perform semantic analysis and reasoning on the basis of background knowledge [52]. Although statistical approaches are usually general and can be used for different domains or languages, their main disadvantage is that it is difficult to extract general concepts contained in technical texts or specific technical concepts contained in general texts. On the other hand, symbolic approaches, such as linguistic-based or pattern-driven methods, are mostly language-dependent and need domain

Table 4
Priority weights of alternatives for criteria and overall priority scores.

Criteria	DODDLE-OWL	OntoBuilder	OntoLT	Text2Onto
User interface (0.0075 [a])	0.3025	0.0490	0.2369	**0.4117**
Availability (0.0506 [a])	**0.2500**	**0.2500**	**0.2500**	**0.2500**
Time-to-first-use (0.0156 [a])	**0.6497**	0.2133	0.0738	0.0632
Preprocessing requirement (0.0568 [a])	0.1855	0.2149	0.1471	**0.4525**
Ontology reusability (0.0217 [a])	**0.5695**	0.1347	0.0979	0.1978
Extraction level (0.1576 [a])	0.0934	0.0463	0.3989	**0.4614**
Degree of automation (0.0959 [a])	0.0583	0.1021	0.3854	**0.4542**
Algorithm selection (0.0370 [a])	0.0934	0.0463	**0.4614**	0.3989
Efficiency (0.0132 [a])	0.0722	0.0722	**0.4635**	0.3921
Reliability (0.2609 [a])	**0.3000**	0.1000	**0.3000**	**0.3000**
Syntactic quality (0.0737 [a])	0.3099	0.0587	**0.3773**	0.2541
Semantic quality (0.1791 [a])	0.1896	0.1362	0.2903	**0.3838**
Pragmatic quality (0.0300 [a])	0.1279	0.0595	0.3639	**0.4487**
Priority score	0.2116	0.1079	0.3182	**0.3623**

[a] Weight for criteria.

experts or linguists. In addition, the cost of adapting extracted patterns or semantic knowledge to a new domain may be high. Thus, although symbolic approaches have good performance in particular domains for extracting specific concepts or relations, they are limited and inflexible. Consequently, some tools based on the symbolic approach are less efficient because this approach may require considerable time in extracting concepts or relations. On the other hand, tools based on statistical approaches are less precise. There might be different ways to enhance the performance of ontology extraction tools, such as a new hybrid approach combining both statistical and symbolic approaches.

Second, various aspects of ontology extraction tools must be improved to make them user-friendly. Ontology extraction tools were first developed around a decade ago. Only a few were developed for commercial purpose, while most were developed for research purposes. Since research prototype tools are typically developed with an emphasis on the extraction features such as the extraction algorithms, their interfaces are less user-friendly, and hence, do not meet users' expectations. If the interface of an ontology extraction tool is complex to use, it may require a longer period to master the tool's functions. Therefore, a tool's interface needs to be improved to enhance usability. In addition, for the user to maneuver the tool easily, the ontology extraction tool must provide an installation and user guide. Currently, only a few ontology extraction tools provide such a manual.

Third, ontology extraction tools should provide various extraction levels. Despite most tools being capable of extracting concepts using existing ontologies or dictionaries, only a few can extract relations automatically or semi-automatically [1,51]. Therefore, ontology extraction tools need to be improved further, and should be able to extract all the ontology elements such as concepts, taxonomies, and relations. Furthermore, they should support a fully automated extraction process. Although a few fully automatic tools are already available, they are very restricted, can only function under limited circumstances, and have lower performance as compared to semi-automatic or cooperative extraction tools [52]. In addition, in many cases, a tool labeled as an “automatic ontology extraction tool” is not fully automatic. The ultimate aim of automatic ontology extraction tool is to produce satisfactory and complete ontologies without human intervention. Although several ontology extraction tools have been developed, most of the tools are still not fully automatic. For example, when building an ontology using OntoLT, the ontology engineer has to preprocess input data manually, or with the help of other linguistic tools. Of course, ontology extraction tools may show better results with human expert intervention. However, it is inefficient for an expert to intervene in the extraction processes that can be easily automated such as preprocessing of input data, visualization of extracted outputs, and statistical analysis of extracted concepts or relations. Automatic ontology extraction tools should be improved to show more acceptable results, thus minimizing human intervention.

Fourth, it is obvious that the reuse of an existing ontology is more efficient than building a new one [54]. However, even when building a new ontology is inevitable, it would be very effective to use concepts and relations currently used in existing ontologies. Although a few tools such as SPRAT [37] can reuse and modify an existing ontology to create a new one, many ontology extraction tools still fail to support ontology reuse. Therefore, to generate ontologies more efficiently, ontology extraction tools should be designed to aid ontology reuse.

Finally, ontology extraction tools should be syntactically rich in displaying concepts or relations. Some ontology extraction tools offer simple syntactic vocabularies, such as property, class, sub-class, range, and domain. However, if ontology extraction tools used various syntactic vocabularies including cardinality, axiom, and subproperty, in addition to those simple syntactic terms, the constructed ontologies would be syntactically richer. Furthermore, it is desirable for ontology extraction tools to support a function that allows the user to export an ontology in various ontology languages. Currently, most ontology extraction tools can export extracted outcomes to OWL or RDF. However, if more diverse languages such as XTM, XOL, and F-Logic are supported by tools, the exported ontologies can be more usable for more various applications.

6. Conclusion

Along with ontology editing and ontology merging tools, ontology extraction tools play a key role in the development of the Semantic Web. However, there is no widely accepted evaluation framework that allows users to objectively compare the functional performances of various ontology extraction tools. In this paper, we proposed a comprehensive evaluation framework to evaluate ontology extraction tools considering both the tool's performance during the construction time and the quality of the created ontology.

Using the proposed evaluation framework, we carried out an experiment and evaluated four popular ontology extraction tools. Based on the results of our experiment, we learned that most of the existing ontology extraction tools still lack the ability to fully automate the extraction process. If an ontology extraction tool is fully automatic, an ontology engineer can construct an ontology in a faster and more efficient way. In addition, an ontology created by an ontology extraction tool tends to be less biased toward its developer because the ontology constructed by the extraction tool may be less affected by its developer's individual thoughts and experiences during the construction process. Several tools still need functional performance improvement. For example, OntoLT requires the source data to be annotated linguistically before extracting concepts and their relations. However, since linguistic annotation functions are not included in OntoLT, the corpus should be annotated in advance with the help of other annotation tools. Moreover, most of the tools, except for Text2Onto, do not provide various extraction algorithms. Each algorithm might be developed to deal with a distinct type of source data; therefore, if the user does not have the option of selecting the desirable extraction algorithm for a specific source data, the extracted ontology may be neither satisfactory nor acceptable.

Ontology technologies have recently become popular and have attracted more attention than ever before because they are the cornerstones for the realization of the Semantic Web. Establishing a way to develop ontologies effectively and efficiently is what is most needed to improve these ontology technologies. It is because of this that some researchers have paid great attention to standardizing ontology development and evaluation methodologies. In addition, others have tried to minimize the ontology-building time. To accomplish this, various ontology tools have been developed. In particular, ontology extraction tools are

considered a major solution since they can speed up the ontology construction time through the automation of the ontology development process. When developing a new ontology extraction tool, if a standard that the tool has to meet is well set up, developers can clearly comprehend the requirement and hence easily undertake tool development. Furthermore, developers may also receive essential feedback about the ontology extraction tool under development. Our framework can be applied as a useful benchmark or guidance when developing an ontology extraction tool.

References

[1] M. Abulaish, L. Dey, Biological relation extraction and query answering from MEDLINE abstracts using ontology-based text mining, Data & Knowledge Engineering 61 (2007) 228–262.
[2] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Scientific American 284 (2001) 34–43.
[3] G. Bisson, C. Nedellec, D. Canamero, Designing clustering methods for ontology building, The Mo'K Workbench, In proceedings of the Workshop on Ontology Learning, 14th European Conference on Artificial Intelligence (ECAI), Berlin, Germany, 2000, pp. 13–19.
[4] J. Brank, M. Grobelnik, D. Mladenic, A survey of ontology evaluation techniques, Proceedings of the Conference on Data Mining and Data Warehouses, 2005.
[5] J. Brank, D. Mladenic, M. Grobelnik, Gold standard based ontology evaluation using instance assignment, Proceedings of the 4th International Workshop on Evaluation of Ontologies for the Web (EON) at the 15th International World Wide Web Conference, Edinburgh, UK, 2006.
[6] C. Brewster, H. Alani, S. Dasmahapatra, Y. Wilka, Data driven ontology evaluation, Proceedings of the Language Resources and Evaluation, 2004.
[7] P. Buitelaar, D. Olejnik, M. Sintek, A protégé plug-in for ontology extraction from text based on linguistic analysis, Proceedings of the 1st European Semantic Web Symposium (ESWS), Heraklion, Greece, 2004.
[8] A. Burton-Jones, V. Storey, V. Sugumaran, P. Ahluwalia, A semiotic metrics suite for assessing the quality of ontologies, Data & Knowledge Engineering 55 (2005) 84–102.
[9] G. Chaelandar, B. Grau, SVETLAN'—a system to classify words in context, Proceedings of the Workshop on Ontology Learning, 14th European Conference on Artificial Intelligence (ECAI), Berlin, Germany, 2000, pp. 19–24.
[10] S. Chidamber, C. Kemerer, A metrics suite for object-oriented design, IEEE transactions on software engineering 20 (6) (1994) 476–493.
[11] O. Corcho, M. Fernandez, A. Gomez-Perez, Methodologies, tools and languages for building ontologies. Where is the meeting point? Data & Knowledge Engineering 46 (2003) 41–64.
[12] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, GATE: a framework and graphical development environment for robust NLP tools and applications, Proceedings of the 40th Annual Meeting of the ACL, 2002.
[13] W. Daelemans, M. Reinberger, Shallow text understanding for ontology content evaluation, IEEE Intelligent Systems (July/August 2004) 76–77.
[14] M. Dahab, H. Hassan, A. Refea, TextOntoEx: automatic construction form natural English text, Expert Systems with Applications 34 (2) (2008) 1474–1480.
[15] S. Davis, R. Bostrom, Training end users: an experimental investigation of the roles of the computer interface and training methods, MIS Quarterly 17 (1) (1993) 61–85.
[16] T. Declerck, A set of tools for intergrating linguistic and non-linguistic information, Proceedings of the SAAKM workshop at ECAL, 2002.
[17] K. Dellschaft, S. Staab, On how to perform a gold standard based evaluation of ontology learning, Proceedings of the ISWC, 2006.
[18] A. De Nicola, M. Missikoff, R. Navigli, A software engineering approach to ontology building, Information Systems 34 (2) (2009) 258–275.
[19] Y. Ding, S. Foo, Ontology research and development: part 1 — a review of ontology generation, Journal of Information Science 28 (5) (2002) 375–388.
[20] P. Drake, Using the analytical hierarchy process in engineering education, International Journal of Engineering Education 14 (3) (1998) 191–196.
[21] A. Duineveld, R. Stoter, M. Weiden, B. Kenepa, V. Benjamins, Wonder tools? A comprehensive study of ontological engineering tool, Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management, Nanff, Canada, 1999.
[22] J. Euzenat, P. Shvaiko, Ontology Matching, Springer, 2007.
[23] D. Faure, C. Nedellec, Knowledge acquisition of predicate argument structures from technical texts using machine learning: the system ASIUM, Proceedings of the 11th European Workshop (EKAK), LNAI 1621, 1999, pp. 329–334.
[24] C. Fellbaum, WordNet: an Electronic Lexical Database, MIT Press, 1998.
[25] M. Fernandez, A. Gomez-Perez, N. Juristo, Methontology: from ontological art to ontological engineering, Workshop on Ontological Engineering. Spring Symposium Series (AAAI '97) Stanford, USA, 1997.
[26] A. Gal, G. Modica, OntoBuilder: fully automatic extraction and consolidation of ontologies from Web sources, Proceedings of the 20th International Conference on Data Engineering (ICDE), 2004.
[27] A. Gangemi, C. Catenacci, M. Ciaramita, J. Lehmann, A theoretical framework for ontology evaluation and validation, Proceedings of the Semantic Web Applications and Perspectives (SWAP), 2nd Italian Semantic Web Workshop, Trento, Italy, 2005.
[28] A. Gomez-Perez, D. Manzano-Macho, An overview of methods and tools for ontology learning form texts, The Knowledge Engineering Review 19 (3) (2005) 187–212.
[29] T. Gruber, Towards principles for the design of ontologies used for knowledge sharing, International Journal of Human–Computer Studies 43 (4–5) (1995) 907–928.
[30] N. Guarino, C. Welty, An overview of OntoClean, in: S. Staab, R. Studer (Eds.), The Handbook on Ontologies, 2004, pp. 151–172.
[31] M. Hearst, Automatic acquisition of hyponyms from large text corpora, Proceedings of the 14th International Conference on Computational linguistics, 1992.
[32] A. Hevner, S. March, J. Park, S. Ram, Design science in information systems research, MIS Quarterly 28 (1) (2004) 75–106.
[33] J. Köhler, K. Munn, A. Rüegg, A. Skusa, B. Smith, Quality control for terms and definitions in ontologies and taxonomies, BMC Bioinformatics 7 (212) (2006).
[34] P. Lambrix, A. Edberg, Evaluation of ontology merging tools in bioinformatics, Proceedings of the Pacific Symposium on Biocomputing, 8, 2003, pp. 589–600.
[35] A. Maedche, R. Volz, The Text2Onto ontology extraction and maintenance environment, Proceedings of the ICDM Workshop on Integrating Data Mining and Knowledge Management, San Jose, California, USA, 2001.
[36] A. Maedche, S. Staab, Ontology learning, In Handbook on Ontologies, In S. Staab and R. Studer, eds., The Handbook on Ontologies. Springer, 2004, 173–190.
[37] D. Maynard, A. Funk, W. Peters, SPRAT: a tool for automatic semantic pattern-based ontology population, Proceedings of the International Conference for Digital Libraries and the Semantic Web, Trento, Italy, 2009.
[38] T. Morita, N. Fukuta, N. Izumi, T. Yamaguchi, DODDLE-OWL: a domain ontology construction tool with OWL, Lecture Notes in Computer Science 4185 (2006) 537–551.
[39] A. Murshed, R. Singh, Evaluation and ranking of ontology construction tools," Technical Report DIT-05-013, University of Trento, 2005.
[40] R. Navigli, P. Velardi, Learning domain ontologies from document warehouses and dedicated web sites, Computational Linguistics 30 (2) (2004) 151–179.
[41] R. Navigli, P. Velardi, A. Cucchiarelli, F. Neri, Quantitative and qualitative evaluation of the OntoLearn ontology learning system, Proceedings of the 20th International Conference on Computational Linguistics (COLING), 2004, pp. 1043–1050.
[42] J. Nielsen, Usability Engineering, Harcourt Brace Company, 1993.
[43] D. Norman, S. Draper, User centered system design — new perspectives on human–computer interaction, Lawrence Erlbaum Associates, 1986.
[44] N. Noy, M. Musen, The PROMPT suite: interactive tools for ontology merging and mapping, International Journal of Human–Computer Studies 59 (6) (2003) 983–1024.
[45] D. McGuinness, R. Fikes, J. Rice, S. Wilder, An environment for merging and testing large ontologies, Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning, Breckenridge, CO, USA, 2000, pp. 483–493.
[46] A. Orme, H. Yao, L. Etzkorn, Indicating ontology data quality, stability, and completeness throughout ontology evolution, Journal of Software Maintenance and Evolution: Research and Practice 19 (2007) 49–75.

[47] J. Park, Ontology, in: Gordon B. Davis (Ed.), Management Information Systems, Second Ed., The Blackwell Encyclopedia of Management, Vol. VII, Blackwell Publishing, Cambridge, Massachusetts, 2005, pp. 233–236.
[48] H. Perera, W. Costa, Analytic hierarchy process for selection of ERP software for manufacturing companies, Vision 12 (4) (2008) 1–11.
[49] T. Saaty, The Analytic Hierarchy Process, McGraw-Hill, NewYork, 1980.
[50] T. Saaty, How to make a decision: the analytical hierarchy process, Inferences 24 (6) (1994) 19–43.
[51] D. Sanchez, A. Moreno, Learning non-taxonomic relationships from web documents for domain ontology construction, Data & Knowledge Engineering 64 (2008) 600–623.
[52] M. Shamsfard, A. Barforoush, The state of the art in ontology learning: a framework for comparison, The Knowledge Engineering Review 18 (4) (2003) 293–316.
[53] B. Shneiderman, Designing the User Interface: Strategies for Effective Human Computer Interaction, Addison-Wesley, 1992.
[54] E. Simperl, Reusing ontologies on the Semantic Web: a feasibility study, Data & Knowledge Engineering 68 (2009) 905–925.
[55] Y. Sure, S. Staab, R. Studer, On-To-Knowledge methodology (OTKM), in: Steffen Staab, Rudi Studer (Eds.), The Handbook on Ontologies, 2004.
[56] C. Tao, D. Embley, Automatic hidden-web table interpretation, conceptualization, and semantic annotation, Data & Knowledge Engineering 68 (2009) 683–703.
[57] P. Vossen, Ontologies, in: R. Mitkov (Ed.), the Oxford Handbook of Computational Linguistics, 2003, pp. 464–482.
[58] Y. Wand, R. Weber, On the deep structure of information systems, Journal of Information Systems 5 (1995) 203–223.
[59] R. Weber, Ontological issues in accounting information systems, Researching Accounting as an Information System Discipline (2002) 13–33.
[60] S. Wu, W. Hsu, SOAT: a semi-automatic domain ontology acquisition tool from Chinese corpus, Proceedings of the 19th International Conference on Computational Linguistics, Taipei, Taiwan, 2002, pp. 1–5.
[61] T. Yokoi, The EDR electronic dictionary, Communications of the ACM 38 (11) (1995) 42–44.
[62] J. Yu, J. Thom, A. Tam, Requirements-oriented methodology for evaluating ontologies, Information Systems 34 (2009) 766–791.
[63] DODDLE-OWL <http://doddle-owl.sourceforge.net/>.
[64] KAON <http://kaon.semanticweb.org/>.
[65] MR^3 <http://mr3.sourceforge.net/>.
[66] NeOn <http://www.neon-project.org/>.
[67] OI-Modeler <http://kaon.semanticweb.org/documentation/>.
[68] OntoBuilder <http://iew3.technion.ac.il/OntoBuilder/>.
[69] OntoLT <http://olp.dfki.de/OntoLT/OntoLT.htm/>.
[70] Protégé <http://protege.stanford.edu/>.
[71] Text2Onto <http://ontoware.org/projects/text2onto/>.
[72] xdoc2txt <http://www31.ocn.ne.jp/~h_ishida/xdoc2txt.html/>.

Jinsoo Park is Associate Professor of Information Systems in the Graduate School of Business at Seoul National University. He was formerly on the faculties of University of Minnesota and Korea University. He holds a Ph.D. in MIS from the University of Arizona. His research interests include ontology, semantic interoperability, metadata management, data modeling, and process modeling. His research has been published in MIS Quarterly, IEEE Transactions on Knowledge and Data Engineering, IEEE Computer, ACM Transactions on Information Systems, and others. He currently serves on the editorial boards of Journal of Database Management and International Journal of Principles and Applications in Information Science and Technology.

Wonchin Cho received the B.S. and M.S. degree in Business from Ajou University in 2001 and 2003. She received her Ph.D. from the Seoul National University in 2010. Her research interests include data mining, Semantic Web, ontology, and automatic keyword generation.

Sangkyu Rho is Professor of Information Systems in the Graduate School of Business at Seoul National University. He received his Ph.D. from the University of Minnesota. His research interests include Internet business, ontology development, data mining, ranking, and social networking. He has published papers in such journals as IEEE Transactions on Knowledge and Data Engineering, Information Systems, Annals of Operations Research, and Strategic Management Journal.