Towards Semi-automatic Ontology Building Supported by Large-scale Knowledge Acquisition

Yimin Wang and **Johanna Völker** and **Peter Haase**
Institute AIFB, University of Karlsruhe (TH)
76128, Karlsruhe, Germany
{ywa, jvo, pha}@aifb.uni-karlsruhe.de

Abstract

Knowledge acquisition is usually the first step in building ontologies. On the one hand, knowledge is typically implicitly contained in large collections of unstructured documents. Therefore it is extremely troublesome to manually identify relevant concepts. On the other hand, users are often not fully satisfied with the results of automated state-of-the-art ontology learning techniques. In this paper we present a technique for large-scale Knowledge Acquisition supported Semi-automated Ontology building (KASO) and a corresponding software system. By applying KASO and using this software, users are able to bootstrap the process of building high quality ontologies by automatically acquiring concepts from large-scale document collections and to make use of traditional knowledge acquisition approaches to refine and organize the machine-generated concepts. Evaluation studies and user experiences indicate the applicability of KASO in bootstrapping ontology construction.

Introduction

With the growing population of digital knowledge over the web, finding information with effectiveness, efficiency and accuracy becomes an increasingly challenging task, especially since most knowledge is contained in large collections of unstructured textual documents. Ordinary approaches are to acquire knowledge from the documents manually and then structure the knowledge at the conceptual level. However, conventional knowledge acquisition approaches are usually driven by humans, which means that they are labor-intensive, time-consuming and troublesome. Thus, automated or semi-automated knowledge acquisition techniques are urgently required.

Recently, machine learning and text mining methods have been introduced to assist users in this process. **Text2Onto**[1] (Cimiano & Völker 2005), a framework for ontology learning from textual resources, is able to automatically create ontologies from a corpus of documents within a certain domain. This approach is a machine-centered task, so major human tasks are to define the input knowledge resource, tune the parameters and select the output destination.

[1]Text2Onto is distributed under the LGPL and can be obtained from `http://ontoware.org/projects/text2onto/`.

Consider the following scenario of large-scale knowledge acquisition: A knowledge manager, or a domain expert, is going to build a knowledge base from a large set of domain-specific documents. The traditional way is to seek through the documents for the key terms and map them onto concepts using knowledge development tools, such as Protégé (Noy *et al.* 2001), which is typical manual tooling. Alternatively, we have automated knowledge acquisition approaches such as Text2Onto, which gathers concepts from large set of documents and directly outputs as intermediate ontologies. The user has to interact with this intermediate representation in order to transform it into a "real" ontology.

Both of the two aforementioned approaches have their particular advantages and disadvantages. People using manual acquisition usually build ontologies for specific purposes so that they can directly use the ontologies in their applications – they know well about the ontology structure, interconnections, rules, semantics and so on. But it's really difficult if there are thousands of concepts to manage for the manual acquisition. Automatic ontology generation systems are essentially quite fast and effective in use, but the automatically manipulated ontologies are not meant to be used directly by the end ontology users. Thus we are going to find a balance point that combines the success of conventional knowledge acquisition approaches and automatic ontology learning framework to facilitate the ontology engineering life-cycle support.

The main contribution of this paper is a novel semi-automatic ontology building technique based on large-scale knowledge acquisition – **KASO**, presented in Section "The KASO Technique", and a corresponding software system implemented as a Protégé plug-in which is described in Section "Implementation". This plug-in allows for automatically eliciting knowledge from large documents in combination with traditional knowledge acquisition approaches to refine and organize the elicited concepts. Through investigating several existing knowledge acquisition technologies (Section "Related Works and Challenges"), card sorting and laddering methods are selected for performing tasks after the automated acquiring phase, for which we decided to use Text2Onto. For the user interface, we applied *User-centered Design* techniques (Section "User-centered Design for KASO") to help users in creating ontologies in a convenient and straightforward manner. The implementation of of KASO was thoroughly evaluated against the requirements (Section "Evaluation and Comparison"). The evaluation and comparison results show that this software successfully implements this novel technique and indeed saves users a considerable amount of time.

Related Work and Challenges

There are two major research areas involved in KASO, namely, the domains of conventional *Knowledge Acquisition* and *Ontology Learning*.

Conventional Knowledge Acquisition

In the past several decades, people tried to develop knowledge acquisition techniques to acquire knowledge with effectiveness, efficiency and correctness. A number of these methods were borrowed from *Cognitive Science* and other disciplines such as *Anthropology*, *Ethnography*, and *Business Administration* (Boose 1988; Hoffman 1987). Knowledge acquisition techniques were effectively used in the early 1990s, when graphical interfaces for personal computers became more and more popular (Shadbolt, Hara, & Crow 1999)

Card sorting is a comprehensive technique which is concurrently being used in several disciplines such as *Knowledge Engineering*, *Psychology*, and *Marketing* (Upchurch, Rugg, & Kitchenham 2001). In the field of *Knowledge Acquisition*, card sorting is considered to be one of the most effective ways for eliciting the domain expert's idea about the knowledge structure. Many evidences show that card sorting has a lot of positive aspects in making a useful and reasonable elicitation experiment, including: (i) helping the respondents to recall the domain concepts; (ii) providing a structured concepts pile for future processing – such as laddering; (iii) fast acting and (iv) easy handling (Shadbolt, Hara, & Crow 1999). Currently the card sorting method is mainly performed manually and is not widely applied as a computer tool. Figure 5 shows a snapshot of typical card sorting task along with the prototype in section "Implementation".

Laddering has been widely used in knowledge acquisition activities in recent years. The basic purpose of the laddering method is to elicit people's goals and values (Shadbolt, Hara, & Crow 1999). People from the knowledge acquisition community have developed a range of well-established formal semantics, procedures and notations for building ladders. Based on the Rugg and McGeorge's (Rugg *et al.* 2002) categorization, the laddering can be used for three major purposes – the elicitation of sub-classes, explanations and goals/values. An independent tool for laddering is provided by the PCPACK[2] toolkit which supports the CommonKADS methodology (Schreiber *et al.* 1994). But the implementation in PCPACK lacks the full support of the standard OWL language[3], and the well integration with state-of-the-art ontology engineering tools, like Protégé. In this paper, we are going to use the laddering method to build up the data and object properties for OWL classes. A typical conceptual ladder example for the Pizza related concepts is depicted by Figure 5.

There are also some other popular human-driven acquisition techniques, such as **repertory grid**, a matrix-based approach, which was developed mainly for eliciting and analyzing knowledge and for self-help and counseling purposes (Boose 1988). Another informal approach, **user observation** (Ericsson & Simon 1984), was developed decades ago and is still widely used for interviews, questionnaires and other common social science methods. Both of the approaches are difficult to be directly applied to ontology engineering. Therefore, card sorting and laddering will be deployed to combine with the ontology learning techniques, which is one of our major concerns in this paper.

Ontology Learning

Several ontology learning frameworks have been designed and implemented in the recent years. For instance, the Mo'K workbench (Bisson, Nedellec, & Canamero 2000) basically relies on unsupervised machine learning methods to induce concept hierarchies from text collections. An ontology learning plug-in for Protégé called OntoLT (Buitelaar, Olejnik, & Sintek 2003) is targeted more at end users and heavily relies on linguistic analysis. The framework by Velardi et al., OntoLearn (Velardi *et al.* 2005), mainly focuses on the problem of word sense disambiguation. Finally, TextToOnto (Maedche & Staab 2004), the predecessor of Text2Onto, is a framework implementing a variety of algorithms for diverse ontology learning subtasks. In particular, it implements diverse relevance measures for term extraction, different algorithms for taxonomy construction as well as techniques for learning relations between concepts (Maedche & Staab 2000).

Common to all the above mentioned frameworks is some sort of natural language processing to derive features on the basis of which to learn ontological structures.

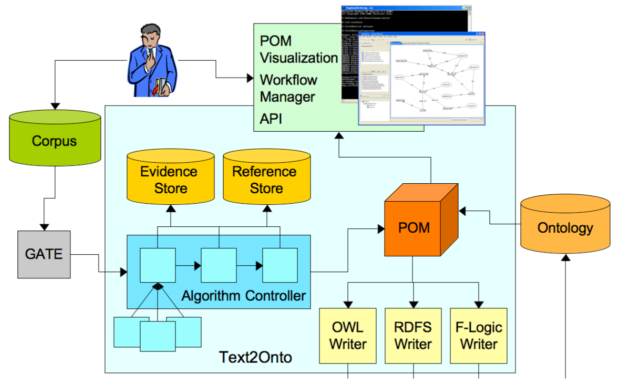


Figure 1: The scenario of using Text2Onto.

As shown in Figure 1, three main features distinguish Text2Onto from its predecessor, TextToOnto, as well as other state-of-the-art ontology learning frameworks. The first is to represent the learned knowledge at a meta-level

[2]http://www.epistemics.co.uk/Notes/55-0-0.htm

[3]http://www.w3.org/2004/OWL/

in the form of instantiated modeling primitives within a so called *Model of Possible Ontologies* (POM). Second, by incorporating strategies for data-driven change discovery, we avoid processing the whole corpus from scratch each time it changes, only selectively updating the POM according to the corpus changes instead. Third, user interaction is a central aspect of Text2Onto's architecture, although it is still not sufficient enough to build ontologies for end users. Actually, the fact that every instantiated modeling primitive gets assigned a probability or confidence factor by the system permits us to design sophisticated visualizations on top of the Text2Onto API in which the user can prune or filter the POM on the basis of the system's confidence.

Challenges

Truly, both the human-driven knowledge acquisition techniques, like card sorting and laddering, and the existing ontology learning frameworks, like Text2Onto, are able to construct initial ontologies. However, neither of them could fully satisfy the users. Whereas human-driven acquisition approaches are short of the effectiveness and efficiency, ontologies generated by ontology learning algorithms is highly required to be refined.

The challenge hereby is to create a novel technique which combines the advantages of these two approaches and presents users a new technique of building ontologies. Another major challenge consists in applying this novel technique to a real application, which on the one hand, gives people exact view of the new ontology building approach, and on the other hand, proves the soundness, correctness and applicability of this technique.

The KASO Technique

The technique introduced in this section aims to (i) reduce the work-load for knowledge engineers and domain experts, (ii) increase the reusability of laddering and card sorting processes, (iii) effectively manage the knowledge acquisition tasks, and (iv) seamlessly integrate with existing software systems.

Our work is to find a midpoint so that we can balance the trade-off between the automatic and purely manual ontology building approaches. The technique introduced here goes well with this demanding goal. We use a large collection of documents as a knowledge source, apply the Text2Onto technology to elicit the concepts, and then manually organize the raw outcome in an intuitive way by adopting traditional knowledge acquisition methods, i.e., card sorting and laddering. Thereafter, we have the basic ontology structure which can be output in a standard OWL ontology format for future editing. As shown in Figure 2, users perform tasks from top to bottom, starting with the KASO technique, which is integrated from conventional knowledge acquisition and ontology learning techniques, to get and refine the ontology structure and generate the OWL output.

To be more specific, in Figure 3, the basic taxonomy of the ontology is built by using the card sorting procedure, which is equal to sorting cards with concept labels. The laddering method is used afterwards to construct the relationships

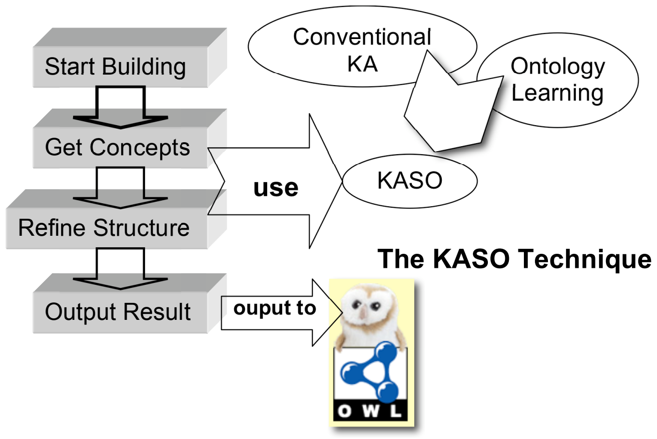


Figure 2: Task roadmap of KASO.

between the concepts that are typically object properties in OWL. The ontology skeleton is completed mainly by deploying KASO, whereas in normal cases, ontologies consist of many other elements like restrictions, annotations, rules, etc. So other knowledge acquisition procedures can be included which are not explicitly defined. For example, ontology engineers could add restrictions, annotations and rules with help of ontology engineering tools. These editing activities are summarized as "Future Editing" and their corresponding acquisition methods are stated as "Others" in the Figure 3.

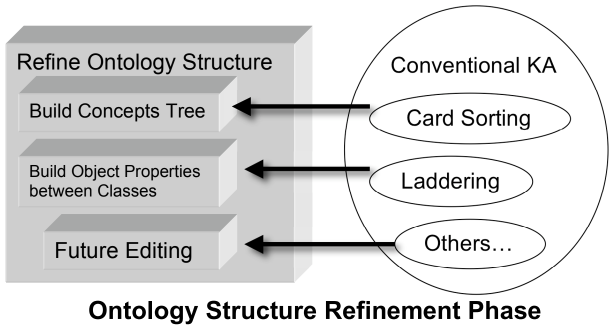

Refine Ontology Structure
Build Concepts Tree
Build Object Properties between Classes
Future Editing
Conventional KA
Card Sorting
Laddering
Others…
Ontology Structure Refinement Phase

Figure 3: The human-driven knowledge acquisition as a refining procedure in KASO involves different detailed knowledge acquisition techniques and their corresponding usages in constructing ontologies.

KASO applies the successful elements of two major ontology building approaches, whose background technologies have been developed in many actual systems, as indicated in section "Related Work and Challenges". Thus, we believe that the technique indicated in this section is practical for implementation and real use.

User-centered Design for KASO

In this section we discuss design considerations we took into account for defining the user interface of KASO. In the

Human-Computer Interaction (HCI) research domain, *User-centered Design* is one of the most essential methodologies and now widely used in various disciplines, including *Software Engineering, Knowledge Management and Information Systems* (Shneiderman 1997).

One aspect of user-centered design techniques is to make users involved in the software design process, by interviewing various groups of users based on certain requirements, such as age, occupation, gender, culture and so on. Therefore, we interviewed potential users of KASO and collected interview results in order to discover the goals and needs of the target user group. The techniques of participatory design are necessary in implementing KASO to a real software, because the target user group mainly consists of scientific researchers with different disciplines, requirements, personalities, ways of working and thinking.

Participatory design includes observing and recording the manual activities, such as using the rectangles on paper as software window frames; cutting the paper into rectangles with different size as dialogues and menus; choosing difference colors to indicate the selection status; drawing, dragging while necessary to modify the interface; taking the picture while performing activities and many other actions. All these activities are performed by the **real** target group of users. The picture below was taken from the interview activities within the software design process.

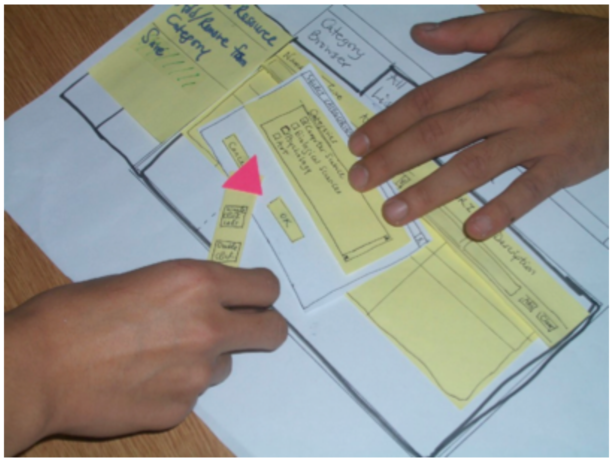


Figure 4: A user participant design case with hand-draw window frames

By adopting these design techniques, we successfully collected enough data for the implementation. Eleven people with varying backgrounds participated in the interview. The same people later participated in the user evaluation in order to discover whether the outcome meets their requirements. The interview results show many variances among people due to different academic backgrounds and level of expertise and also indicate that the cultural background plays an essential role. For example, a first year Asian undergraduate student may have difficulties in modeling an ontology about the domain of pizzas, because he is normally not familiar with different kinds of pizzas and does not have much experience on knowledge systems.

After analyzing the interview results, we discovered that certain number of people asked for a quick execution for concepts input and some comprehensive approaches to organize them, as well as standard output and good user interface. Thus, we derived the following functionalities as requirements:

1. a text processor for a series of documents belonging to a certain domain as knowledge source for building ontologies;
2. a flexible and straightforward user interface;
3. human-driven knowledge acquisition techniques integrated with graphical manipulation;
4. a reusable and well-formatted output;
5. integration with other ontology editors.

When implementing KASO, we took into account the aforementioned requirements derived from the user interviews.

Implementation

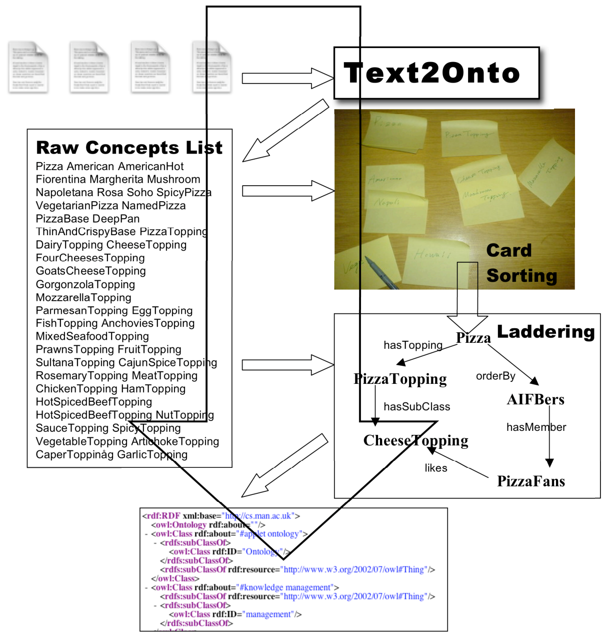
We implemented the KASO technique by firstly creating a prototype with the main functionality and afterwards integrating the prototype into an ontology engineering environment.

Prototype for KASO Implementation

There are two ways to generate the knowledge source for eliciting concepts. One is to gather a large collection of documents within a domain separately, for example, from the sports news we can easily find enough football reports and save them to a corpus folder. The other way is to find a large document, perhaps a book, divide the book into many sections and put them into a same corpus folder. This task is so straightforward that a user with basic PC and internet knowledge can successfully handle this.

As soon as the corpus has been prepared for the ontology learning process, the users can begin to build their ontologies. The first step is generally automatic – the users just need to select the corpus and run the program. After a while, a list of concepts will be extracted from the documents ready for future organizing. Conventional knowledge acquisition activities are applied to help users in this process. In particular, the card sorting technique is used to select concepts and insert them into a taxonomy, while laddering aims to initialize the relations between the concepts by building a conceptual ladder.

After addressing the aforementioned issues, we get a prototype for this plug-in as depicted by Figure 5. Card sorting is mandatory because the basic taxonomy of an ontology has to be settled by arranging cards with concepts, whereas laddering sometimes is optional due to actual use cases and different goals for building ontologies. What is worth mentioning is that both the card sorting and the laddering approaches are implemented as working panels that allow a style of interaction, which is largely similar to that of domain experts working on the desk.


Text2Onto
Raw Concepts List
Pizza American AmericanHot
Fiorentina Margherita Mushroom
Napoletana Rosa Soho SpicyPizza
VegetarianPizza NamedPizza
PizzaBase DeepPan
ThinAndCrispyBase PizzaTopping
DairyTopping CheeseTopping
FourCheesesTopping
GoatsCheeseTopping
GorgonzolaTopping
MozzarellaTopping
ParmesanTopping EggTopping
FishTopping AnchoviesTopping
MixedSeafoodTopping
PrawnsTopping FruitTopping
SultanaTopping CajunSpiceTopping
RosemaryTopping MeatTopping
ChickenTopping HamTopping
HotSpicedBeefTopping
HotSpicedBeefTopping NutTopping
SauceTopping SpicyTopping
VegetableTopping ArtichokeTopping
CaperToppinág GarlicTopping
Card Sorting
Laddering
Pizza
hasTopping
PizzaTopping
orderBy
AIFBers
hasSubClass
hasMember
CheeseTopping
likes
PizzaFans

```
<rdf:RDF xml:base="http://cs.man.ac.uk">
  <owl:Ontology rdf:about=""/>
 - <owl:Class rdf:about="#applet ontology">
   - <rdfs:subClassOf>
       <owl:Class rdf:ID="Ontology"/>
     </rdfs:subClassOf>
     <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
 - <owl:Class rdf:about="#knowledge management">
     <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
   - <rdfs:subClassOf>
       <owl:Class rdf:ID="management"/>
     </rdfs:subClassOf>
```

Figure 5: The software prototype for KASO, the big up-down arrow points out the overall process, while the small arrows lead to different sub-steps.

The KASO Protégé Plugin

Considering the requirements raised by the users during the interview session, we find a trade-off between working with existing mature ontology development toolkits and running as a standalone application. While it is comparatively easier to implement this system as a self-stand application, the users may find it difficult to access the software and get used to it. Otherwise, the output of the software will probably be edited in other ontology editors, so the users may find it troublesome to switch between different applications. Finally, we decided to implement the KASO prototype as a plugin for Protégé – one of the most popular ontology engineering environments.

The prototype is implemented as a Protégé plug-in and depends on the Protégé OWL plug-in API (Knublauch *et al.* 2004), executing smoothly as depicted by Figure 6. The button of "Auto Elicit" enables users to select the corpus folder in the local storage system, which is marked as "1". The window with number "3" shows the internal text files of the selected folder, while window with the number "2" is a sample of a text file in window "3". Now the users just need to click the "OK" button in window "1". The system will call the Text2Onto engine to process the selected corpus and return the result to the user interface.

The user interface consists of card sorting and laddering working panels, as well as other controlling functionalities. Figure 7 indicates the conventional knowledge acquisition tasks performed after automatically eliciting the concepts. We can see that as opposed to Figure 6, the card sorting window with mark number "2" is now running as a separate window to facilitate users to perform card sorting and laddering simultaneously. The outputs of the Text2Onto engine now directly appear as cards with concept labels on the card sorting panel. The window with number "4" displays the Text2Onto runtime status, while Text2Onto runs in the background.

Focus on Figure 7, users usually start in the card sorting panel, inserting the concepts into the ontology tree following the bigger arrow from right to the left. The selection and the status of the cards are rendered by means of different colors. At the same time, users can ladder up the concepts with different relationships to the laddering window with number "1", to construct object properties in the OWL language. After constructing the basic contents of the ontology, the users can save the result to an OWL file as opened in a browser's window marked with number "3" – this step is represented by the arrow pointing from right to left.

Following the steps described above, in brief, the sample text file in Figure 6 contributes several concepts to the card sorting panel of Figure 7, from which the users refine the ontology skeleton and output to the OWL formatted document. Hence, on the one hand, domain ontology engineers can successfully create a high quality ontology from a large set of domain-specified documents without searching the key concepts from scratch, which is extremely troublesome. On the other hand, Text2Onto users can easily filter out the concepts which were not appropriately selected by the system, to make their auto-generated ontology more applicable.

Evaluation and Comparison

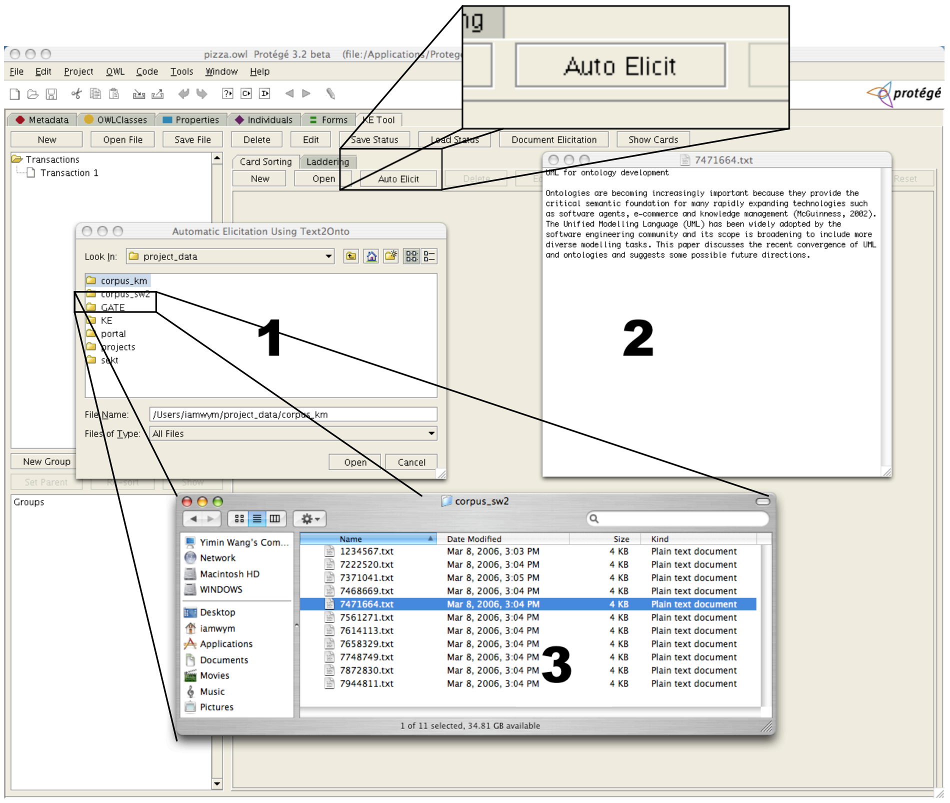
In this section, we explicitly evaluate the software which applied the KASO by performing user evaluation based on the usability requirements, and moreover, we also compare different ontology building approaches, to prove the correctness of KASO and the quality of this implementation. The feedbacks from user evaluation were treated as an essential guideline for the application testing and debugging.

User Evaluation Results

The user evaluation has two different parts. One is the **interface evaluation** which mainly concerns the software's graphical user interface (GUI), including look and feel, interface layout, ease of use, and flexibility. And the other is **functional evaluation** which focuses on the application for KASO. This evaluation procedure aims to detect the user's comments on three basic aspects in the domain of *User-centred Design* – the software should be powerful, flexible and robust.

Eleven people were involved in the user evaluation activities. They were diverse in academic and cultural background. In order to quantify the results, a grading system similar to evaluating university examination was borrowed, that is, 5 is a pass, 6 is a good pass, above 7 is a distinction and 10 is the full score. In the arrays of the scores introduced below, the first five scores in each array come from the experts or skilled users of knowledge systems. The participants are marked with "E" for expert and "N" for "Non-expert" plus the reference number.

In terms of the user interface design, the grading results will be given to four different aspects as interface evaluation.



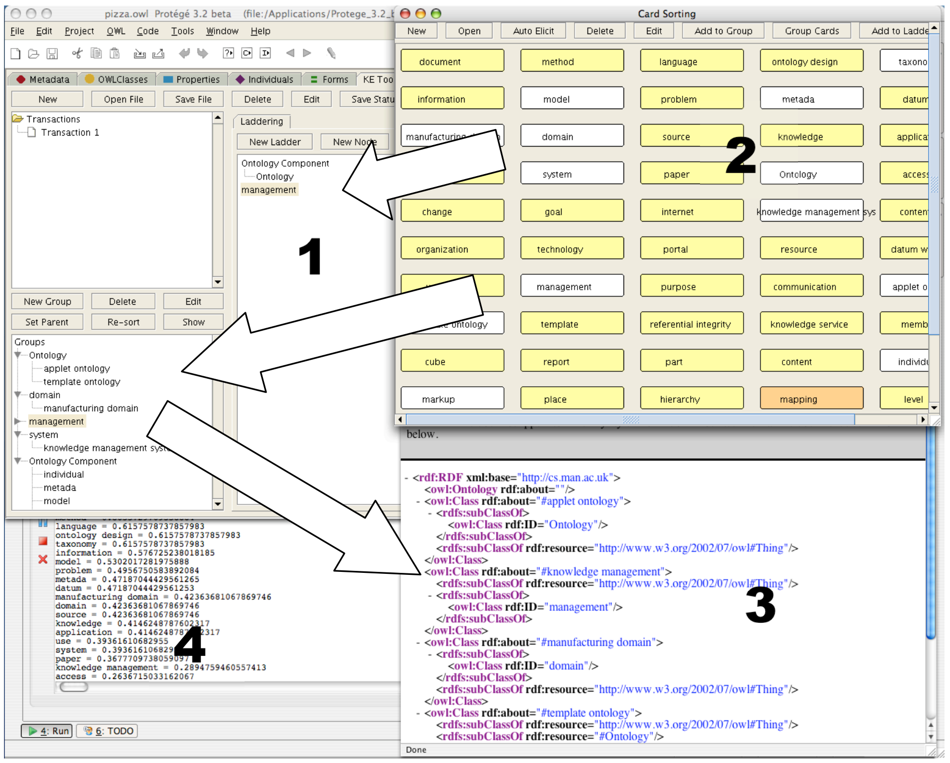
pizza.owl Protégé 3.2 beta (file:/Applications/Protege
File Edit Project OWL Code Tools Window Help
Metadata OWLClasses Properties Individuals Forms KE Tool
New Open File Save File Delete Edit Save Status Load Status Document Elicitation Show Cards
Auto Elicit
Transactions
Transaction 1
Card Sorting Laddering
New Open Auto Elicit
Automatic Elicitation Using Text2Onto
Look In: project_data
corpus_km
corpus_sw2
GATE
KE
portal
projects
sekt
File Name: /Users/iamwym/project_data/corpus_km
Files of Type: All Files
Open Cancel
New Group
Groups
1
2
3
7471664.txt
corpus_sw2
Yimin Wang's Com...
Network
Macintosh HD
WINDOWS
Desktop
iamwym
Applications
Documents
Movies
Music
Pictures
1 of 11 selected, 34.81 GB available

```
UML for ontology development

Ontologies are becoming increasingly important because they provide the
critical semantic foundation for many rapidly expanding technologies such
as software agents, e-commerce and knowledge management (McGuinness, 2002).
The Unified Modelling Language (UML) has been widely adopted by the
software engineering community and its scope is broadening to include more
diverse modelling tasks. This paper discusses the recent convergence of UML
and ontologies and suggests some possible future directions.
```

Name	Date Modified	Size	Kind
1234567.txt	Mar 8, 2006, 3:03 PM	4 KB	Plain text document
7222520.txt	Mar 8, 2006, 3:04 PM	4 KB	Plain text document
7371041.txt	Mar 8, 2006, 3:05 PM	4 KB	Plain text document
7468669.txt	Mar 8, 2006, 3:04 PM	4 KB	Plain text document
7471664.txt	Mar 8, 2006, 3:04 PM	4 KB	Plain text document
7561271.txt	Mar 8, 2006, 3:04 PM	4 KB	Plain text document
7614113.txt	Mar 8, 2006, 3:04 PM	4 KB	Plain text document
7658329.txt	Mar 8, 2006, 3:04 PM	4 KB	Plain text document
7748749.txt	Mar 8, 2006, 3:04 PM	4 KB	Plain text document
7872830.txt	Mar 8, 2006, 3:04 PM	4 KB	Plain text document
7944811.txt	Mar 8, 2006, 3:04 PM	4 KB	Plain text document

Figure 6: Screenshot of initializing the concepts eliciting. The different numbers are mapped to the components in the figure.



```
language = 0.6157578737857983
ontology design = 0.6157578737857983
taxonomy = 0.6157578737857983
information = 0.576725238018185
model = 0.5302017281975888
problem = 0.4956750583892084
metada = 0.4718704442956l265
datum = 0.4718704442956l253
manufacturing domain = 0.42363681067869746
domain = 0.42363681067869746
source = 0.42363681067869746
knowledge = 0.4146248787602317
application = 0.414624878[illegible]2317
use = 0.39361610682955
system = 0.39361610682[illegible]
paper = 0.367770973805909[illegible]
knowledge management = 0.2894759460557413
access = 0.2636715033162067
```

```
- <rdf:RDF xml:base="http://cs.man.ac.uk">
    <owl:Ontology rdf:about=""/>
  - <owl:Class rdf:about="#applet ontology">
    - <rdfs:subClassOf>
        <owl:Class rdf:ID="Ontology"/>
      </rdfs:subClassOf>
      <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    </owl:Class>
    <owl:Class rdf:about="#knowledge management">
      <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    - <rdfs:subClassOf>
        <owl:Class rdf:ID="management"/>
      </rdfs:subClassOf>
    </owl:Class>
  - <owl:Class rdf:about="#manufacturing domain">
    - <rdfs:subClassOf>
        <owl:Class rdf:ID="domain"/>
      </rdfs:subClassOf>
      <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    </owl:Class>
  - <owl:Class rdf:about="#template ontology">
      <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
      <rdfs:subClassOf rdf:resource="#Ontology"/>
```

Figure 7: Working with human-driven knowledge acquisition to manipulate OWL ontology. The numbers and arrows show the steps after eliciting concepts from documents with the help of Text2Onto.

Participant	E1	E2	E3	E4	E5	E6	N1	N2	N3	N4	N5	Average
Look and feel	9	7	7	8	9	7	7	6	7	8	6	7.3
Interface layout	7	7	9	6	8	6	9	5	6	5	6	7.3
Ease of use	7	7	6	7	8	7	6	6	6	6	5	6.4
Flexibility	7	8	8	6	5	6	6	6	9	6	8	6.8

Table 1: Interface evaluation results

Participant	E1	E2	E3	E4	E5	E6	N1	N2	N3	N4	N5	Average
Card sorting	9	8	7	9	9	7	8	8	7	7	8	7.9
Laddering	7	6	7	7	7	7	6	7	7	9	8	7.0
Relationship setting	6	7	6	6	6	8	7	9	7	8	8	7.0

Table 2: Functional evaluation results

The users were asked for the grades of the four points, and their grading results are listed in table 1. To be statistically accurate, the average score was calculated by eliminating the highest and the lowest scores in each array.

The functional evaluation involves the grading of each basic component, including card sorting, laddering and relationship setting. They are three major components provided by this software and users are easily getting familiar with them. So the grading of these components is direct.

The overall score is calculated by calculating the average value. We get **7.0** points for interface evaluation and **7.3** points for functional evaluation. For the component of concept eliciting, we refer to the existing evaluation from Text2Onto (Völker, Vrandečić, & Sure 2006). The Text2Onto engine has been widely evaluated and is undoubtedly considered to be a stable and reliable automatic ontology constructing system using text-based documents as knowledge source.

The Comparison

Notwithstanding the evaluation indicates the success of KASO in practical usage, compared to the ordinary human-based ontology editing, we can also see the advantages owned by KASO.

This comparison is performed by recording the **approximate** time required to build an ontology with fixed number of concepts by using pure conventional knowledge acquisition techniques and KASO, respectively. Because the times required to organize the ontology structure vary based on different cases, it is not necessary and practical to have more accurate timing scale. The timing by minutes is suitable for this comparison.

In the table 3, the conventional knowledge acquisition technique stands for selecting the concepts from documents using the mouse instead of using Text2Onto. The sample is a university website that mainly consists of a bunch of HTML documents. The requirement of using KASO techniques is to build an KASO ontology that is similar to the manually constructed ontology, which contains rich semantics and is not practical to be generated just by using ontology learning tools.

In addition, many redundant concepts are suggested by Text2Onto, which means that many of the auto-elicited concepts are not selected for inclusion into the ontology by the user. While performing the KASO technique, the experimenter's experience showed that the redundancy rate is around 30% – 40%. However, this means that the users definitely need human-driven acquisition approaches as a complementary part for fully automated ontology learning approaches when they build ontologies.

Methodology	Concepts	Time
Conventional KA	25	14 minutes
KASO	25	12 minutes
Conventional KA	100	1 hours
KASO	100	20 minutes
Conventional KA	200	2 hours+
KASO	200	30 minutes

Table 3: The comparison of ontology building approaches, "Concepts" column shows the fixed number of concepts for an ontology and "Time" indicates the approximate time required for building ontologies.

Analysis and Discussion

The evaluation scores illustrate that users are mostly satisfied with the functionalities of this software and the implementation of the technique. With respect to the interface of this software, although the score is comparatively moderate, the users generally have positive comments. To discover more based on the evaluation results, we find that the look and feel of the interface, concept eliciting and card sorting have the highest ratings and are thought to be best implemented. Meanwhile, the elements related to the ease of use principle and interface layout arrangement require some future improvements.

It is worthwhile to mention the evaluation of this software system from the independent *UK Freshwater Life Biological Association*[4]. Their comment on this software was: *"It was good to see what he has been doing and looks like a potentially very useful tool. We really liked to get our hands on a copy to play around with. Even in its current state it could save us considerable time."*

[4]`http://www.freshwaterlife.org/`

In a nutshell, the KASO technique is proved to be sound and useful in assisting ontology engineer's daily work. The corresponding software is commonly considered to be a well-implemented and powerful tool in **real** use. The target user group of this software – knowledge system experts acknowledge that the software interface is straightforward and intuitive. All the evidences obtained from this user evaluation procedure indicate that people are very willing to see the future development of this software.

Conclusion and Outlook

In this paper, we presented KASO, a technique for ontology building by automatically acquiring raw knowledge from large-scale document collections, and refining the taxonomy by using integrated traditional knowledge acquisition techniques, i.e. card sorting and laddering. We also developed a usable software as a use case for our technique, and then integrated it as a Protégé plug-in. The implementation outcome and evaluation results demonstrate that our technique is well applicable and this software plug-in is welcomed by most real users.

Future work includes the extension of this software with more capabilities provided by the Text2Onto toolkit and to apply further ontology learning and ontology evaluation techniques to heuristically assist users in building the fine-grained ontologies. The NeOn project(Dzbor *et al.* 2005) requirements also provide us with an opportunity to align our technique to the forthcoming networked ontology model, specifically, posing as the collaborative ontology engineering environment with support from large-scale knowledge acquisition techniques.

Acknowledgements

The authors' current research is supported by the EU-IST-027595 NeOn project and EU-IST-506826 SEKT project. The research reported in this paper was partly supported in part by the CO-ODE project funded by the UK Joint Information Services Committee (JISC). We would like to thank our colleagues for fruitful discussions.

References

Bisson, G.; Nedellec, C.; and Canamero, L. 2000. Designing clustering methods for ontology building - The Mo'K workbench. In *Proc. of the ECAI Ontology Learning WS.*

Boose, J. H. 1988. Knowledge acquisition techniques and tools: Current research strategies and approaches. In *Proceedings of Fifth Generation Computer Systems*, 1221–1235.

Buitelaar, P.; Olejnik, D.; and Sintek, M. 2003. OntoLT: A protégé plug-in for ontology extraction from text. In *Proceedings of the International Semantic Web Conference (ISWC).*

Cimiano, P., and Völker, J. 2005. A framework for ontology learning and data-driven change discovery. In *Proc. of the NLDB'2005.*

Dzbor, M.; Motta, E.; Studer, R.; Sure, Y.; Haase, P.; Gmez-Prez, A.; Benjamins, R.; and Waterfeld, W. 2005. Neon - lifecycle support for networked ontologies. In *Proceedings of 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies (EWIMT-2005)*, 451–452. London, UK: IEE.

Ericsson, K., and Simon, H. 1984. *Protocol Analysis: Verbal Reports as Data.* Cambridge, MA USA: MIT Press.

Hoffman, R. R. 1987. The problem of extracting the knowledge of experts from the perspective of experimental psychology. *AI Magazine* 8(2):53–67.

Knublauch, H.; Fergerson, R. W.; Noy, N. F.; and Musen, M. A. 2004. The protégé owl plugin: An open development environment for semantic web applications. In *International Semantic Web Conference*, 229–243.

Maedche, A., and Staab, S. 2000. Discovering conceptual relations from text. In Horn, W., ed., *Proceedings of the 14th ECAI'2000.*

Maedche, A., and Staab, S. 2004. Ontology learning. In Staab, S., and Studer, R., eds., *Handbook on Ontologies.* Springer. 173–189.

Noy, N. F.; Sintek, M.; Decker, S.; Crubézy, M.; Fergerson, R. W.; and Musen, M. A. 2001. Creating semantic web contents with protégé-2000. *IEEE Intelligent Systems* 16(2):60–71.

Rugg, G.; Eva, M.; Mahmood, A.; Rehman, N.; Andrews, S.; and Davies, S. 2002. Eliciting information about organizational culture via laddering. *Journal of Information System* 12(3):215–230.

Schreiber, G.; Wielinga, B. J.; Akkermans, H.; de Velde, W. V.; and Anjewierden, A. 1994. CML: The CommonKADS conceptual modelling language. In *Proceedings of 8th European Knowledge Acquisition Workshop (EKAW)*, 1–25.

Shadbolt, N.; Hara, K. O.; and Crow, L. 1999. The experimental evaluation of knowledge acquisition techniques and methods: history, problems and new directions. *International Journal of Human-Computer Studies* 51:729–755.

Shneiderman, B. 1997. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Upchurch, L.; Rugg, G.; and Kitchenham, B. 2001. Using card sorts to elicit web page quality attributes. *IEEE Software* 18(4):84–89.

Velardi, P.; Navigli, R.; Cuchiarelli, A.; and Neri, F. 2005. Evaluation of ontolearn, a methodology for automatic population of domain ontologies. In *Ontology Learning from Text: Methods, Applications and Evaluation.* IOS Press.

Völker, J.; Vrandečić, D.; and Sure, Y. 2006. Data-driven change discovery - evaluation. Technical report, Universität Karlsruhe. SEKT Deliverable 3.3.2.