Visualisation of Document and Concept Spaces

Sam Holden, Judy Kay, Andrew Lum
School of Information Technologies
University of Sydney, NSW 2006
Australia
E-mail {sholden, judy, alum}@it.usyd.edu.au

Abstract

Collections of documents with conceptual relationships exist in many domains. Teaching systems often contain numerous learning resource documents. University policies are often large collections of related documents. The visualisation of the structure of these collections can be useful as it allows the exploration of the collection.

This paper describes a graphical interface for visualising document spaces. The interface makes it simple for the user to explore the documents and the relationships between them.

Keywords metadata, ITS, ontology extraction, user modelling, visualisation

1 Introduction

There are large numbers of existing document collections which are valuable resources, but are often difficult to explore. From university course materials on the Internet, to policy documents on a corporate intranet.

In the C++ STL domain, for example, there are numerous online resources, from textbooks [3], to short tutorials [5], to programmer reference manuals [8, 7]. The value of these resources to a learner is limited by the difficulty in finding the ones that meet the current learning need, and do not rely on material the learner does not know.

In order to make these documents more useful metadata is required. General metadata such as LOM [2] and Dublin Core [1], provides information about the author, type, subject area, etc.

When documents are used in specialised contexts, such as learning resources in a course, extra metadata is useful. This metadata indicates prerequisite and outcome information for the documents. With the availability of such metadata, a learner can find out which documents teach the concepts they wish to learn. And also determine which documents are available due to the learner understanding all the prerequisites.

Proceedings of the 7th Australasian Document Computing Symposium,
Sydney, Australia, December 16, 2002.

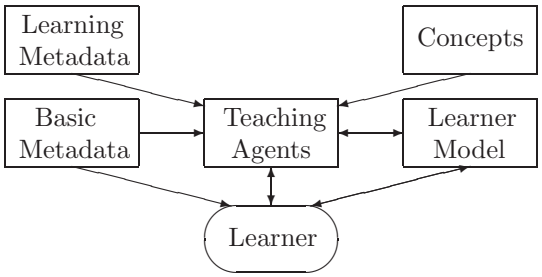

Figure 1: SITS Component Interaction

If the learner does not understand the prerequisites for a document they wish to use, they must learn those prerequisites. This involves viewing documents which teach those prerequisite concepts. If there are yet more prerequisites which are needed then the process will continue in a recursive manner.

Doing this manually would be complicated and tedious. Instead, a teaching system should be able to assist the learner. This paper describes a graphical interface for helping a learner explore the document space.

The teaching system used to host the interface will be briefly described. Then the visualisation interface will be presented, with a report on a user evaluation. Finally the benefits and some possible improvements to the interface will be discussed. We summarise the current state of this work.

2 SITS

Scrutable Intelligent Tutoring System (SITS) is a teaching system designed to reuse existing learning documents.

Figure 1 shows the main components of SITS.

The Concepts are the pieces of knowledge that SITS models in the learner model. The concepts are simply a vocabulary; no relationship structure between concepts is used in SITS. For example, in a C++ STL course some of the concepts might be: sort, copy, vector, and list.

The Basic Metadata is a database of references to documents. A reference to a LOM format metadata source is associated with the document if it is available. These documents are independent of SITS, and are simply reused by SITS.

The Learning Metadata is metadata relating documents in the Basic Metadata to the Concepts. The Learning Metadata is simply a list of Concepts associated with each Document. The concepts whose understanding is required in order to understand the document are listed as *prereq* concepts. The concepts that are taught by the document are listed as *shows* concepts. Finally, the concepts that are referenced by the document, but whose understanding is not necessary in order to make use of the document are listed as *uses* concepts.

The Learner Model uses an evidence-based approach based on um [4] in which evidence that the user knows or does not know concepts is stored. The learner model keeps track of the learner's knowledge of each of the concepts in the Concept vocabulary and also preference information for the learner.

The Teaching Agents are the central components of the system. Multiple Teaching Agents exist, though the learner selects a particular one to use at any given time. The Teaching Agent has the task of determining which concept the learner should learn next. Then it needs to choose which document the learner should view in order to progress toward the goal concepts. SITS does not place limits on how teaching agents go about their tasks. They are free to extend the interface beyond the SITS basics.

3 SV - a visualisation-based teaching agent

SV is the tool we have developed to visualise the documents and concepts in SITS. It is a heavily modified version of VlUM [9], a web-based tool for visualising large user models. The VlUM interface is designed as a Java applet which is loaded in a web browser so that the user can view their complete user model. The applet sits in a frame on the left side of the screen, and page contents are shown on the right.

SV accepts a graph where each node represents either a document or a concept. Each document connects only to concept nodes representing the concepts that the user must know, i.e. prerequisites, in order to be able to understand that particular document. Conversely, concepts have connections to documents that teach the concept. We use this to enable the user to see the relationships between the documents and concepts associated with them. SV displays the graph in two separate columns, with documents on the left hand side and concepts on the right.

Selecting a document will expand the prerequisite concepts. Selecting a concept will expand documents that teach that concept, allowing users to navigate back and forth between the document

Figure 2: SV with document *SGI: priority_queue* currently selected. The most visible concepts on the right column - *Assignable*, *LessThanComparable*, *DefaultConstructable*, *RandomAccessContainer*, *Sequence* and *Container* - are the prerequisite concepts the user should know before reading *SGI: priority_queue*. The user can see that they still have to learn the concepts displayed in red - *DefaultConstructable*, *RandomAccessContainer* and *Sequence*.

and concept space. For the document column, the colour represents whether the user is ready to read it or not - red means they have not fulfilled the prerequisite requirements, green means they are ready to read that document. Once a document has been read, its colour is changed to yellow and it is indented slightly to the right. The concept column uses green to mean that the user has learnt enough about this concept, and red to indicate that they have not and need to read the relevant documents.

As an example, a student Jane wants to read a particular document *SGI: priority_queue*, but the title appears red meaning she has not fulfilled all the prerequisites. She selects it (Figure 3), expanding the prerequisite concepts on the right hand side. The concepts *Assignable*, *LessThanComparable* and *Container* are green, indicating Jane has learnt these concepts. The remaining concepts *DefaultContstructable*, *RandomAccessContainer* and *Sequence* are shown in red, indicating Jane has not yet learn them. She then clicks on the unknown

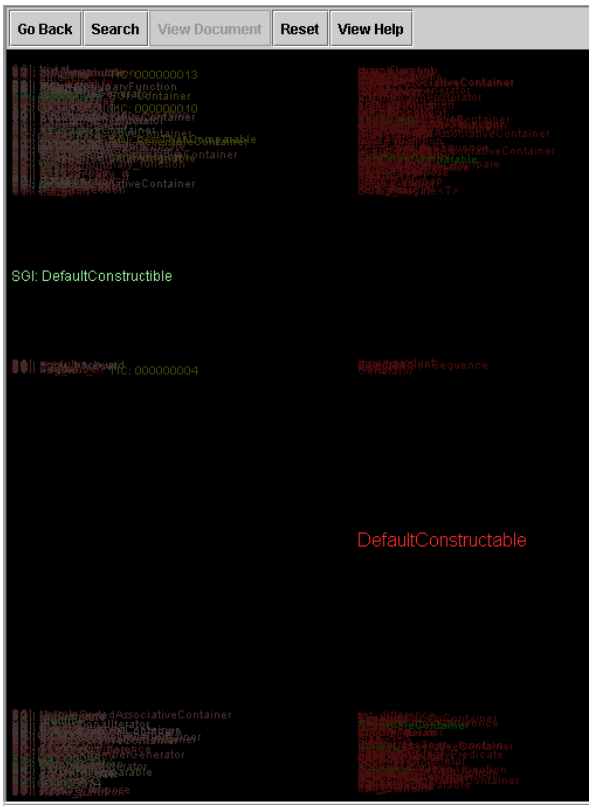

Figure 3: SV with concept *DefaultConstructable* currently selected on the right column. We can see that the document *SGI: DefaultConstructable* teaches this concept. The user is ready to read this document (as it is in green), indicating they already know the prerequisite concepts.

concept *DefaultContructable* to see what documents show this concept, causing the *SGI: DefaultConstructable* document to be displayed (Figure 3 on the preceding page). Jane decides she will read the *SGI: DefaultConstructable* document as it is green. This example shows how users can navigate through their user model and explore the relationship between the documents and the concepts they teach.

A SITS teaching agent has been implemented which used VlUM by generating a VlUM consumable RDF file of the metadata. Additional elements were added to the RDF for typing of the components and the relationships. Each component is designated as either a document or a concept by its own *componentType* element.

4 Evaluation

A think aloud experiment was performed with the recommended five users [6] for a basic qualitative evaluation of usability. All the users were undergraduate students studying computer science.

The users were asked to pretend they were a particular user who wanted to know about the *priority_queue* concept. For the purposes of the experiment they were told to assume they learnt all concepts *shown* by a document when viewing it. All the users managed to complete the task successfully. The colour and positional meanings of the concepts and documents in SV were explained to the users at the beginning.

Table 1: Think Aloud Results Summary

User	# Docs	Notes
User1	5	Found the colours too similar
User2	3	Used back button to observe changes to learner model
User3	5	Found interface easy to use
User4	7	Got confused between documents and concepts
User5	5	Liked the way the green increased as you progressed

A user model was constructed and each user was asked to use the system to 'learn' the *priority_queue* concept. The document, metadata, and user model were designed so that there were two documents that could be viewed to learn the *priority_queue* concept. Both those documents had *prereq* concepts which the user model indicated were not known. The first document had six *prereq* concepts, half of which were not known. Learning all of those required documents for which yet more *prereq* concepts were required because they were not initially known. The second document that taught the *priority_queue* concept had only one *prereq* concept. That concept was not known according to the user model.

It was possible to 'learn' the *priority_queue* by viewing two documents. None of the users tried to find a quick path through the documents. Instead they all headed depth first down the first path.

Table 1 shows the number of documents each user viewed in order to 'learn' the *priority_queue* concepts. As well as a representative comment on the users experience with the interface.

The main difficulty that all the users had was getting lost and losing track of the sub-task they were performing. This occurred after reaching a depth of about two *prereq* concepts (which involved clicking on a document, then a *prereq*, then another document, and then clicking another *prereq*). The users could not remember exactly why they had selected the concept. This was resolved in all cases by clicking the back button a few times and regaining their bearings. The fact that the users not actually using the system to learn, probably meant they weren't paying much attention to the actual concepts which would account for some of this problem. A real learner would probably not forget which concept they were trying to learn.

The users gave many suggestions as to why they got lost. The lack of a visible history indication to show the path they had followed was one complaint.

Another was that when a document is selected, the *prereq* concepts are indicated, but no indication of the *shows* concepts is provided. The lack of that indication caused some users to click back to remind themselves why they have clicked on the document.

One user confused the document and concept columns. They suggested column headers as a simple solution to the problem. Column headers would also be useful in providing the users with an indication of whether the concepts being indicated, are *prereq* concepts of the selected documents, or *show* concepts of the indicated documents.

All the users completed the task in a few minutes, and most of that time was spent waiting for the web server to respond to page requests. The VlUM applet itself was fast enough, with SITS being the speed bottleneck. This is due to the fact that SITS has not been optimised for speed. Simple caching techniques will speed SITS up, in a later test a speed improvement of about 20x was realised by caching just one database query between sessions. The entire graph they were exploring contained 171 documents, 155 concepts, and 620 edges, all of which were accessible from within SV.

The think aloud experiment indicated that the interface is usable, since all the users succeeded at the task without any assistance. SV provides an interface to a complicated graph which is fast to learn and use.

5 Discussion and conclusion

The system enables the user to explore a range of documents that explain the concepts the user is interested in. The user simply selects a concept they are interested in and suitable documents are indicated along with a colour indication of whether the user model indicates the user understands the required prerequisites. The users understood this, and successfully performed these tasks. One user was confused for a short time by the fact that green concepts are known, whereas green documents are ready to be used, and not already used.

The system also enables the user to explore the concepts which are required to understand a document. The user simply selects the document and those concepts are indicated, and coloured to indicate how well the user understands them. All the users has no problems understanding and using this prerequisite indication.

Navigation through the documents and concepts makes it easy to find the documents that teach concepts that are required by the document the user wants to view.

The main limitation of the interface is that it is not suited to long sequences, such as when the user needs to study a chain of five or six documents deep in order to learn a concept required for the wanted document. The interface does not provide a way to find out that a long sequence exists, other than clicking on the concepts and documents in the chain.

The user test showed that the interface is understandable and usable. It also identified some areas for improvement for the future.

The strength of the interface is that it provides a useful view of a set of documents and concepts. It allows the user to explore the document structure without having to understand the entire, possibly complex, graph.

A number of ideas for improvements came out of the user testing. Such as using a different colours for the documents and concepts, to keep the distinction between them more obvious.

The horizontal indentation of the documents could be used to indicate how suitable the documents are based on external metadata preference information, a certain author might be liked by a particular learner, for example. Indentation could replace colour entirely, by having items which are currently green near the center, and items which are currently red near the edges.

References

[1] Dublin core metadata element set, IETF RFC 2413.

[2] IEEE 1484.12.1-2002, learning objects metadata.

[3] Bruce Eckel. Thinking in c++ 2nd edition (volume 2). http://www.mindview.net/Books/TICPP/.

[4] Judy Kay. The um toolkit for cooperative user modelling. *User Modelling and User-Adapted Interaction*, Volume 4, Number 3, pages 149–196, 1995.

[5] Jak Kirman. A modest stl tutorial. http://www.cs.brown.edu/people/jak/proglang/cpp/stltut/tut.html.

[6] A J Nielsen. Estimating the number of subjects needed for a thinking aloud test. *International Journal of Human-Computer Studies*, Volume 41, Number 1–6, pages 385–397, 1994.

[7] P. J. Plauger. Dinkum c++ library reference. http://www.dinkum.com/refcpp.html.

[8] Silicon Graphics Computer Systems. Standard template library programmer's guide. http://www.sgi.com/tech/stl/.

[9] James Uther. *On the Visualisation of Large User Model in Web Based Systems.* Ph.D. thesis, University of Sydney, 2001.