

國立交通大學

資訊科學系

碩士論文

符合 SCORM 標準之學習資源庫
的管理機制之研究

A Content Management Scheme in
SCORM Compliant Learning Object Repository

研究生：宋昱璋

指導教授：曾憲雄 教授

中華民國九十四年六月

符合 SCORM 標準之學習資源庫的管理機制之研究

A Content Management Scheme in
SCORM Compliant Learning Object Repository

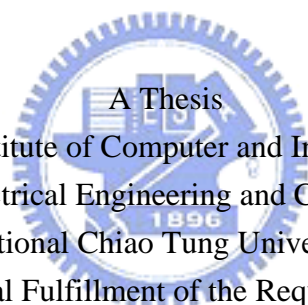
研究生：宋昱璋

Student：Yu-Chang Sung

指導教授：曾憲雄

Advisor：Shian-Shyong Tseng

國立交通大學
資訊科學系
碩士論文



A Thesis

Submitted to Institute of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

符合 SCORM 標準之學習資源庫 的管理機制之研究

研究生：宋昱璋

指導教授：曾憲雄教授

國立交通大學資訊科學研究所

摘要

隨著網際網路的發展，網路學習(e-Learning)也越來越普及。為了促進學習資源在不同網路學習系統間的分享與再利用，近年來有許多國際性組織提出了各種格式的標準，其中最被廣泛應用的是 SCORM。另外，在 e-learning 系統中的學習資源通常都存放在資源庫(Learning Object Repository (LOR))中，而當資源庫中存放著大量物件時，隨即會面臨到大量物件的管理問題。因此在本篇論文中，我們提出了一個階層式的管理機制(Level-wise Content Management System (LCMS))來有效地管理符合 SCORM 標準的學習資源庫。LCMS 的流程可分為“建構”與“搜尋”兩大部份。在建構階段(Constructing Phase)，我們先運用 SCORM 標準中所提供的資訊，將學習資源轉換成一個樹狀架構。接著考慮到 SCORM 中的詮釋性資料(Metadata)對一般人的複雜度，另外提出了一個方式來輔助使用者來加強學習資源中各學習物件的詮釋性資訊。而後藉由分群的技術，我們針對資源庫中的學習物件建立了一個多層有向非環圖，稱為 Level-wise Content Clustering Graph (LCCG)來儲存物件的資訊以及學習物件間的關聯。在搜尋階段(Searching Phase)，提出了一個搜尋機制以利用已建立的 LCCG 找出使用者想要的學習物件。除此之外，考量到使用者在下搜尋關鍵字時的難處，在此亦基於 LCCG 提出了一個方式來輔助使用者改善搜尋用詞以在學習資源庫中找出相關的物件。最後，我們實作了一個雛形系統並進行了一些實驗。由實驗結果可知，LCMS 的確能有效地管理符合 SCORM 標準的學習資源庫。

關鍵字：學習資源庫， e-Learning， SCORM， 內容管理

A Content Management Scheme in SCORM Compliant Learning Object Repository

Student: Yu-Chang Sung

Advisor: Dr. Shian-Shyong Tseng

**Department of Computer and Information Science
National Chiao Tung University**

Abstract

With rapid development of the Internet, e-learning system has become more and more popular. Currently, to solve the issue of sharing and reusing of learning contents in different e-learning systems, several standards formats have been proposed by international organizations in recent years, and Sharable Content Object Reference Model (SCORM) is the most popular one among existing international standards. In e-learning system, learning contents are usually stored in database, called *Learning Object Repository* (LOR). In LOR, a huge amount of SCORM learning contents including associated learning objects will result in the issues of management over wired/wireless environment. Therefore, in this thesis, we propose a management approach, called *Level-wise Content Management Scheme* (LCMS), to efficiently maintain, search, and retrieve the learning contents in SCORM compliant LOR. The LCMS includes two phases: *Constructing Phase* and *Searching Phase*. In *Constructing Phase*, we first transform the content tree (CT) from the SCORM content package to represent each learning materials. Then, considering about the difficulty of giving learning objects useful metadata, an information enhancing module is proposed to assist users in enhancing the meta-information of content trees. Afterward, a multistage graph as Directed Acyclic Graph (DAG) with relationships

among learning objects, called *Level-wise Content Clustering Graph* (LCCG), will be created by applying incremental clustering techniques. In *Searching phase*, based on the LCCG, we propose a searching strategy to traverse the LCCG for retrieving the desired learning objects. Besides, the short query problem is also one of our concerns. In general, while users want to search desired learning contents, they usually make rough queries. But this kind of queries often results in a lot of irrelevant searching results. So a query expansion method is also proposed to assist users in refining their queries and searching more specific learning objects from a LOR. Finally, for evaluating the performance, a web-based system has been implemented and some experiments also have been done. The experimental results show that our LCMS is efficient and workable to manage the SCORM compliant learning objects.

Keywords: Learning Object Repository (LOR), E-learning, SCORM,
Content Management



誌謝

這篇論文的完成，必須感謝許多人的協助與支持。首先必須感謝我的指導教授，曾憲雄老師，由於他耐心的指導和勉勵，讓我得以順利完成此篇論文。此外，在老師的帶領下，這兩年來，除了學習應有的專業知識外，對於待人處世的方面也啟發不少，而研究上許多觀念的釐清更是讓我受益匪淺，真的十分感激。同時，必須感謝我的口試委員，黃國禎教授、楊鎮華教授與袁賢銘教授，他們對這篇論文提供了不少寶貴的建議。

此外要感謝兩位博士班的學長，蘇俊銘學長和翁瑞鋒學長。除了在數位學習領域上讓我了解不少的知識外，在研究上或是系統的發展上都提供了不少的建議及協助，且這篇論文能夠順利完成也得力於學長們的幫忙。

另外也要感謝實驗室的學長、同學以及學弟們，王慶堯學長、楊哲青學長、陳君翰、林易虹。不管是論文上或是系統的建置上都給我許多的協助與建議。同時也感謝其他的同學，黃柏智、陳瑞言、邱成樑、吳振霖、李育松，陪我度過這忙碌以及充實的碩士生涯。

要感謝的人很多，無法一一詳述，在此僅向所有幫助過我的人，致上我最深的謝意。

Table of Contents

摘要.....	i
Abstract.....	ii
誌謝.....	iv
Table of Content.....	v
List of Figure.....	vi
List of Example.....	vii
List of Definition.....	viii
List of Algorithm.....	ix
Chapter 1 Introduction.....	1
Chapter 2 Background and Related Work.....	4
2.1 SCORM (Sharable Content Object Reference Model).....	4
2.2 Document Clustering/Management.....	6
2.3 Keyword/phrase Extraction.....	8
Chapter 3 Level-wise Content Management Scheme (LCMS).....	9
3.1 The Processes of LCMS.....	9
Chapter 4 Constructing Phase of LCMS.....	12
4.1 Content Tree Transforming Module.....	12
4.2 Information Enhancing Module.....	15
4.2.1 Keyword/phrase Extraction Process.....	15
4.2.2 Feature Aggregation Process.....	19
4.3 Level-wise Content Clustering Module.....	22
4.3.1 Level-wise Content Clustering Graph (LCCG).....	22
4.3.2 Incremental Level-wise Content Clustering Algorithm.....	24
Chapter 5 Searching Phase of LCMS.....	30
5.1 Preprocessing Module.....	30
5.2 Content-based Query Expansion Module.....	31
5.3 LCCG Content Searching Module.....	34
Chapter 6 Implementation and Experiments.....	37
6.1 System Implementation.....	37
6.2 Experimental Results.....	40
Chapter 7 Conclusion and Future Work.....	46

List of Figures

Figure 2.1: SCORM Content Packaging Scope and Corresponding Structure of Learning Materials	5
Figure 3.1: Level-wise Content Management Scheme (LCMS)	11
Figure 4.1: The Representation of Content Tree.....	13
Figure 4.2: An Example of Content Tree Transforming	13
Figure 4.3: An Example of Keyword/phrase Extraction.....	17
Figure 4.4: An Example of Keyword Vector Generation.....	20
Figure 4.5: An Example of Feature Aggregation	21
Figure 4.6: The Representation of Level-wise Content Clustering Graph	22
Figure 4.7: The Process of ILCC-Algorithm	24
Figure 4.8: An Example of Incremental Single Level Clustering.....	26
Figure 4.9: An Example of Incremental Level-wise Content Clustering.....	28
Figure 5.1: Preprocessing: Query Vector Generator.....	30
Figure 5.2: The Process of Content-based Query Expansion	32
Figure 5.3: The Process of LCCG Content Searching.....	32
Figure 5.4: The Diagram of Near Similarity According to the Query Threshold Q and Clustering Threshold T.....	35
Figure 6.1: System Screenshot: LOMS configuration.....	38
Figure 6.2: System Screenshot: Searching.....	39
Figure 6.4: System Screenshot: Searching Results.....	39
Figure 6.5: System Screenshot: Viewing Learning Objects	40
Figure 6.6: The F-measure of Each Query.....	42
Figure 6.7: The Searching Time of Each Query	42
Figure 6.8: The Comparison of ISLC-Alg and ILCC-Alg with Cluster Refining	42
Figure 6.9: The precision with/without CQE-Alg.....	44
Figure 6.10: The recall with/without CQE-Alg.....	44
Figure 6.11: The F-measure with/withour CQE-Alg.....	44
Figure 6.12: The Results of Accuracy and Relevance in Questionnaire.....	45

List of Examples

Example 4.1: Content Tree (CT) Transformation	13
Example 4.2: Keyword/phrase Extraction	17
Example 4.3: Keyword Vector (KV) Generation.....	19
Example 4.4: Feature Aggregation	20
Example 4.5: Cluster Feature (CF) and Content Node List (CNL).....	24
Example 5.1: Preprocessing: Query Vector Generator	30



List of Definitions

Definition 4.1: Content Tree (CT)	12
Definition 4.2: Level-wise Content Clustering Graph (LCCG).....	22
Definition 4.3: Cluster Feature	23
Definition 5.1: Near Similarity Criterion.....	34



List of Algorithms

Algorithm 4.1: Content Package to Content Tree Algorithm (CP2CT-Alg).....	14
Algorithm 4.2: Keyword/phrase Extraction Algorithm (KE-Alg).....	18
Algorithm 4.3: Feature Aggregation Algorithm (FA-Alg).....	21
Algorithm 4.4: Incremental Single Level Clustering Algorithm (ISLC-Alg).....	26
Algorithm 4.5: Incremental Level-wise Content Clustering Algorithm (ILCC-Alg) ..	29
Algorithm 5.1: Content-based Query Expansion Algorithm (CQE-Alg)	33
Algorithm 5.2: LCCG Content Searching Algorithm (LCCG-CSAlg)	36



Chapter 1 Introduction

With rapid development of the internet, e-Learning system has become more and more popular. E-learning system can make learners study at any time and any location conveniently. However, because the learning materials in different e-learning systems are usually defined in specific data format, the sharing and reusing of learning materials among these systems becomes very difficult. To solve the issue of uniform learning materials format, several standards formats including SCORM [SCORM], IMS [IMS], LOM [LTSC], AICC [AICC], etc. have been proposed by international organizations in recent years. By these standard formats, the learning materials in different learning management system can be shared, reused, extended, and recombined.



Recently, in SCORM 2004 (aka SCORM1.3), ADL outlined the plans of the Content Object Repository Discovery and Resolution Architecture (CORDRA) as a reference model which is motivated by an identified need for contextualized learning object discovery. Based upon CORDRA, learners would be able to discover and identify relevant material from within the context of a particular learning activity [SCORM][CETIS][LSAL]. Therefore, this shows how to efficiently retrieve desired learning contents for learners has become an important issue. Moreover, in mobile learning environment, retransmitting the whole document under the connection-oriented transport protocol, such as TCP, will result in lower throughput due to the head-of-line blocking and Go-Back-N error recovery mechanism in an error-sensitive environment. Accordingly, a suitable management scheme for managing learning resources and providing teachers/learners an efficient search service to retrieve the desired learning resources is necessary over the wired/wireless

environment.

In SCORM, a content packaging scheme is proposed to package the learning content resources into learning objects (LOs), and several related learning objects can be packaged into a learning material. Besides, SCORM provides user with plentiful metadata to describe each learning object. Moreover, the structure information of learning materials can be stored and represented as a tree-like structure described by XML language [W3C][XML]. Therefore, in this thesis, we propose a *Level-wise Content Management Scheme (LCMS)* to efficiently maintain, search, and retrieve learning contents in SCORM compliant learning object repository (*LOR*). This management scheme consists of two phases: Constructing Phase and Searching Phase. In Constructing Phase, we first transform the content structure of SCORM learning materials (Content Package) into a tree-like structure, called *Content Tree (CT)*, to represent each learning materials. Then, considering about the difficulty of giving learning objects useful metadata, we propose an automatic information enhancing module, which includes a *Keyword/phrase Extraction Algorithm (KE-Alg)* and a *Feature Aggregation Algorithm (FA-Alg)*, to assist users in enhancing the meta-information of content trees. Afterward, an *Incremental Level-wise Content Clustering Algorithm (ILCC-Alg)* is proposed to cluster content trees and create a multistage graph, called *Level-wise Content Clustering Graph (LCCG)*, which contains both vertical hierarchy relationships and horizontal similarity relationships among learning objects.

In Searching phase, based on the LCCG, we propose a searching strategy called *LCCG Content Search Algorithm (LCCG-CSAlg)* to traverse the LCCG for retrieving the desired learning content. Besides, the short query problem is also one of

our concerns. In general, while users want to search desired learning contents, they usually make rough queries. But this kind of queries often results in a lot of irrelevant searching results. So a *Content-base Query Expansion Algorithm (CQE-Alg)* is also proposed to assist users in searching more specific learning contents by a rough query. By integrating the original query with the concepts stored in LCCG, the *CQE-Alg* can refine the query and retrieve more specific learning contents from a learning object repository.

To evaluate the performance, a web-based *Learning Object Management System (LOMS)* has been implemented and several experiments have also been done. The experimental results show that our approach is efficient to manage the SCORM compliant learning objects.



This thesis is organized as follows: Chapter 2 introduces the related works. Overall system architecture will be described in Chapter 3. And Chapters 4 and 5 present the details of the proposed system. Chapter 6 follows with the implementation issues and experiments of the system. Chapter 7 concludes with a summary.

Chapter 2 Background and Related Work

In this chapter, we review SCORM standard and some related works as follows.

2.1 SCORM (Sharable Content Object Reference Model)

Among those existing standards for learning contents, SCORM, which is proposed by the U.S. Department of Defense's Advanced Distributed Learning (ADL) organization in 1997, is currently the most popular one. The SCORM specifications are a composite of several specifications developed by international standards organizations, including the IEEE [LTSC], IMS [IMS], AICC [AICC] and ARIADNE [ARIADNE]. In a nutshell, SCORM is a set of specifications for developing, packaging and delivering high-quality education and training materials whenever and wherever they are needed. SCORM-compliant courses leverage course development investments by ensuring that compliant courses are "RAID:" Reusable: easily modified and used by different development tools, Accessible: can be searched and made available as needed by both learners and content developers, Interoperable: operates across a wide variety of hardware, operating systems and web browsers, and Durable: does not require significant modifications with new versions of system software [Jonse04].

In SCORM, content packaging scheme is proposed to package the learning objects into standard learning materials, as shown in Figure 2.1. The content packaging scheme defines a learning materials package consisting of four parts, that is, **1) Metadata:** describes the characteristic or attribute of this learning content, **2) Organizations:** describes the structure of this learning material, **3) Resources:** denotes the physical file linked by each learning object within the learning material,

and **4) (Sub) Manifest:** describes this learning material is consisted of itself and another learning material. In Figure 2.1, the organizations define the structure of whole learning material, which consists of many organizations containing arbitrary number of tags, called *item*, to denote the corresponding chapter, section, or subsection within physical learning material. Each item as a learning activity can be also tagged with activity metadata which can be used to easily reuse and discover within a content repository or similar system and to provide descriptive information about the activity. Hence, based upon the concept of learning object and SCORM content packaging scheme, the learning materials can be constructed dynamically by organizing the learning objects according to the learning strategies, students' learning aptitudes, and the evaluation results. Thus, the individualized learning materials can be offered to each student for learning, and then the learning material can be reused, shared, recombined.

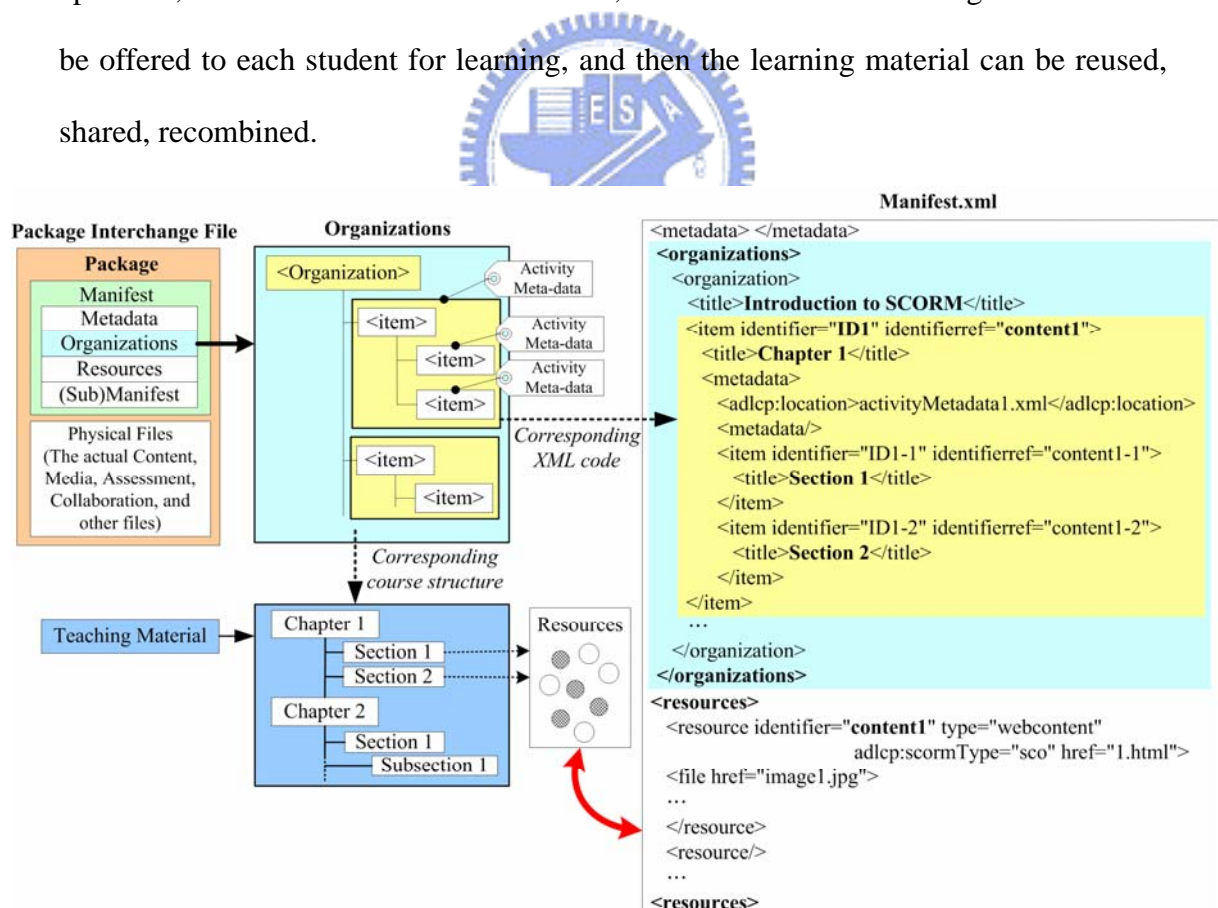


Figure 2.1: SCORM Content Packaging Scope and Corresponding Structure of Learning Materials

2.2 Document Clustering/Management

For fast retrieving the information from structured documents, Ko et al. [KC02] proposed a new index structure which integrates the element-based and attribute-based structure information for representing the document. Based upon this index structure, three retrieval methods including 1) top-down, 2) bottom-up, and 3) hybrid are proposed to fast retrieve the information from the structured documents. However, although the index structure takes the elements and attributes information into account, it is too complex to be managed for the huge amount of documents.

How to efficiently manage and transfer document over wireless environment has become an important issue in recent years. The articles [LM+00][YL+99] have addressed that retransmitting the whole document is a expensive cost in faulty transmission. Therefore, for efficiently streaming generalized XML documents over the wireless environment, Wong et al. [WC+04] proposed a fragmenting strategy, called Xstream, for flexibly managing the XML document over the wireless environment. In the Xstream approach, the structural characteristics of XML documents has been taken into account to fragment XML contents into an autonomous units, called Xstream Data Unit (XDU). Therefore, the XML document can be transferred incrementally over a wireless environment based upon the XDU. However, how to create the relationships between different documents and provide the desired content of document have not been discussed. Moreover, the above articles didn't take the SCORM standard into account yet.

In order to create and utilize the relationships between different documents and provide useful searching functions, document clustering methods have been extensively investigated in a number of different areas of text mining and information retrieval. Initially, document clustering was investigated for improving the precision or recall in information retrieval systems [KK02] and as an efficient way of finding the nearest neighbors of the document [BL85]. Recently, it is proposed for the use of searching and browsing a collection of documents efficiently [VV+04][KK04].

In order to discover the relationships between documents, each document should be represented by its features, but what the features are in each document depends on different views. Common approaches from information retrieval focus on keywords. The assumption is that similarity in words usage indicates similarity in content. Then, the selected words seen as descriptive features are represented by a vector, and one distinct dimension assigns one feature respectively. The way to represent each document by the vector is called Vector Space Model method [CK+92]. In this thesis, we also employ the VSM model to encode the keywords/phrases of learning objects into vectors to represent the features of learning objects.

2.3 Keyword/phrase Extraction

As those mentioned above, the common approach to represent documents is giving them a set of keywords/phrases, but where those keywords/phrases comes from? The most popular approach is using the *TF-IDF* weighting scheme to mining keywords from the context of documents. *TF-IDF* weighting scheme is based on the term frequency (*TF*) or the term frequency combined with the inverse document frequency (*TF-IDF*). The formula of *IDF* is $\log(n/df)$ where n is total number of documents and df is the number of documents that contains the term. By applying statistical analysis, *TF-IDF* can extract representative words from documents, but the long enough context and a number of documents are both its prerequisites.

In addition, a rule-based approach combining fuzzy inductive learning was proposed by Shigeaki and Akihiro [SA04]. The method decomposes textual data into word sets by using lexical analysis, and then discovers key phrases using key phrase relation rules training from amount of data. Besides, Khor and Khan [KK01] proposed a key phrase identification scheme, which employs the tagging technique to indicate the positions of potential noun phrase and uses statistical results to confirm them. By this kind of identification scheme, the number of documents is not a matter. However, a long enough context is still needed to extracted key-phrases from documents.

Chapter 3 Level-wise Content Management Scheme (LCMS)

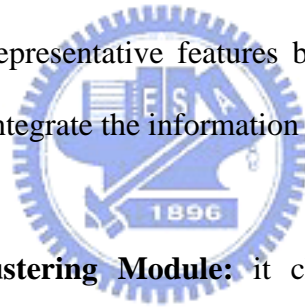
In an e-learning system, learning contents are usually stored in database, called Learning Object Repository (LOR). Because the SCORM standard has been accepted and applied popularly, its compliant learning contents are also created and developed. Therefore, in LOR, a huge amount of SCORM learning contents including associated learning objects (LO) will result in the issues of management. Recently, SCORM international organization has focused on how to efficiently maintain, search, and retrieve desired learning objects in LOR for users. In this thesis, we propose a new approach, called, *Level-wise Content Management Scheme (LCMS)*, to efficiently maintain, search, and retrieve the learning contents in SCORM compliant LOR.

3.1 The Processes of LCMS

As shown in Figure 3.1, the scheme of LCMS is divided into *Constructing Phase* and *Searching Phase*. The former first creates the content tree (CT) from the SCORM content package by *Content Tree Transforming Module*, enriches the meta-information of each content node (CN) and aggregates the representative feature of the content tree by *Information Enhancing Module*, and then creates and maintains a multistage graph as Directed Acyclic Graph (DAG) with relationships among learning objects, called *Level-wise Content Clustering Graph (LCCG)*, by applying clustering techniques. The latter assists user to expand their queries by *Content-based Query Expansion Module*, and then traverses the LCCG by *LCCG Content Searching Module* to retrieve desired learning contents with general and specific learning objects according to the query of users over wire/wireless environment.

Constructing Phase includes the following three modules:

- **Content Tree Transforming Module:** it transforms the content structure of SCORM learning material (Content Package) into a tree-like structure with the representative feature vector and the variant depth, called Content Tree (CT), for representing each learning material.
- **Information Enhancing Module:** it assists user to enhance the meta-information of a content tree. This module consists of two processes: 1) *Keyword/phrase Extraction Process*, which employs a pattern-based approach to extract additional useful keywords/phrases from other metadata for each content node (CN) to enrich the representative feature of CNs, and 2) *Feature Aggregation Process*, which aggregates those representative features by the hierarchical relationships among CNs in the CT to integrate the information of the CT.
- **Level-wise Content Clustering Module:** it clusters learning objects (LOs) according to content trees to establish the *level-wise content clustering graph* (LCCG) for creating the relationships among learning objects. This module consists of three processes: 1) *Single Level Clustering Process*, which clusters the content nodes of the content tree in each tree level, 2) *Content Cluster Refining Process*, which refines the clustering result of the Single Level Clustering Process if necessary, and 3) *Concept Relation Connection Process*, which utilizes the hierarchical relationships stored in content trees to create the links between the clustering results of every two adjacent levels.



Searching Phase includes the following three modules:

- **Preprocessing Module:** it encodes the original user query into a single vector, called query vector, to represent the keywords/phrases in the user's query.
- **Content-based Query Expansion Module:** it utilizes the concept feature stored in the LCCG to make a rough query contain more concepts and find more precise learning objects.
- **LCCG Content Searching Module:** it traverses the LCCG from these entry nodes to retrieve the desired learning objects in the LOR and to deliver them for learners.

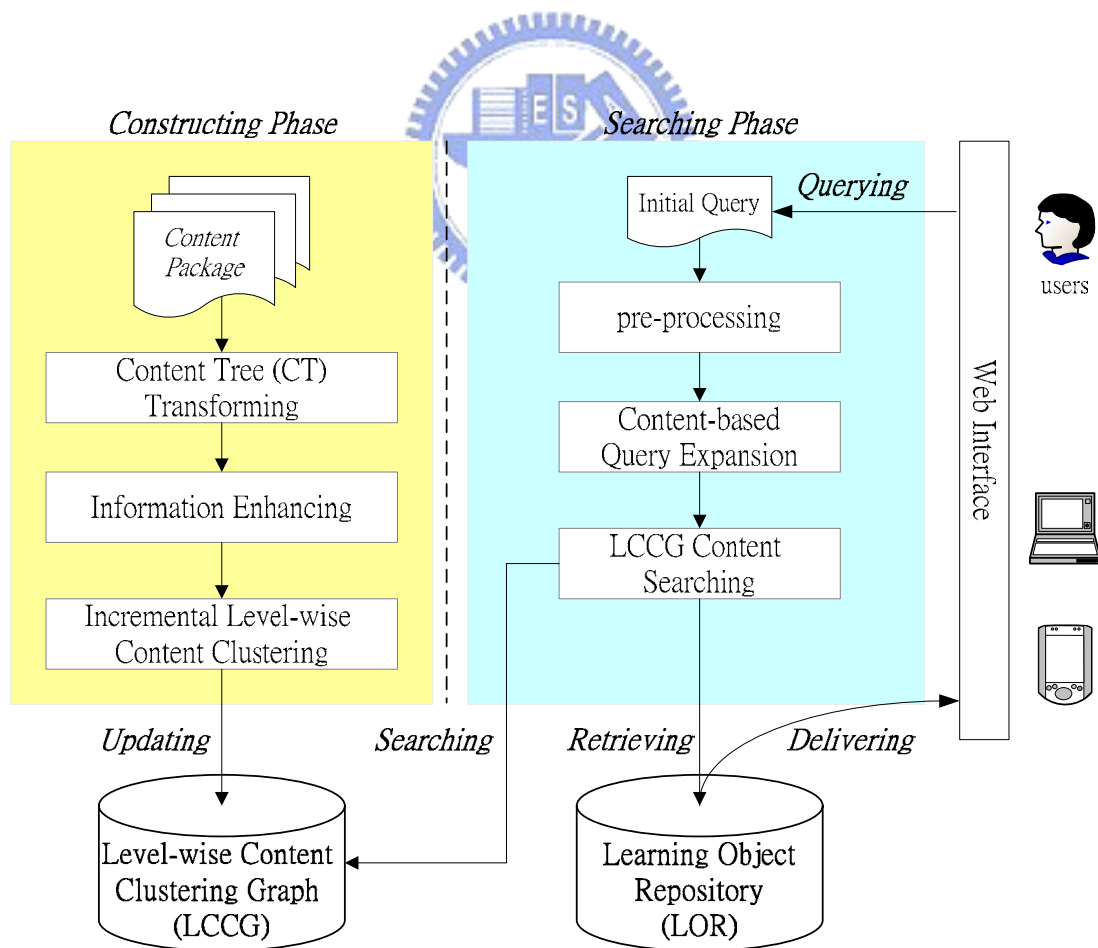


Figure 3.1: Level-wise Content Management Scheme (LCMS)

Chapter 4 Constructing Phase of LCMS

In this chapter, we describe the constructing phrase of LCMS, which includes 1) *Content Tree Transforming* module, 2) *Information Enhancing* module, and 3) *Level-wise Content Clustering* module, shown in the left part of Figure 3.1.

4.1 Content Tree Transforming Module

Because we want to create the relationships among leaning objects (LOs) according to the content structure of learning materials, the organization information in SCORM content package will be transformed into a tree-like representation, called Content Tree (CT), in this module. Here, we define a maximum depth δ for every CT. The formal definition of a CT is described as follows.

Definition 4.1: Content Tree (CT)

Content Tree (CT) = (N, E), where

- $N = \{ n_0, n_1, \dots, n_m \}$.
- $E = \{ \overrightarrow{n_i n_{i+1}} \mid 0 \leq i < \text{the depth of CT} \}$.

As shown in Figure 4.1, in CT, each node is called “*Content Node (CN)*” containing its metadata and original keywords/phrases information to denote the representative feature of learning contents within this node. **E** denotes the link edges from node n_i in upper level to n_{i+1} in immediate lower level.

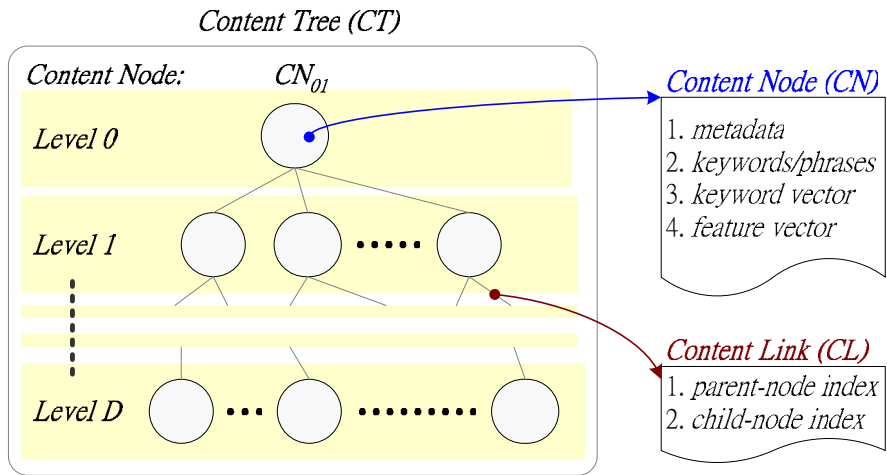


Figure 4.1: The Representation of Content Tree

Example 4.1: Content Tree (CT) Transformation

Given a SCORM content package shown in the left hand side of Figure 4.2, we parse the metadata to find the *keywords/phrases* in each CN node. Because the CN, “3.1”, is too long, so that its included child nodes, i.e., “3.1.1” and “3.1.2”, are merged into one CN, “3.1”, and the weight of each keywords/phrases is computed by averaging the number of times it appearing in “3.1”, “3.1.1”, and “3.1.2”. For example, the weight of “AI” for “3.1” is computed as $\text{avg}(1, \text{avg}(1, 0)) = 0.75$. Then, after applying Content Tree Transforming Module, the CT is shown in the right part of Figure 4.2.

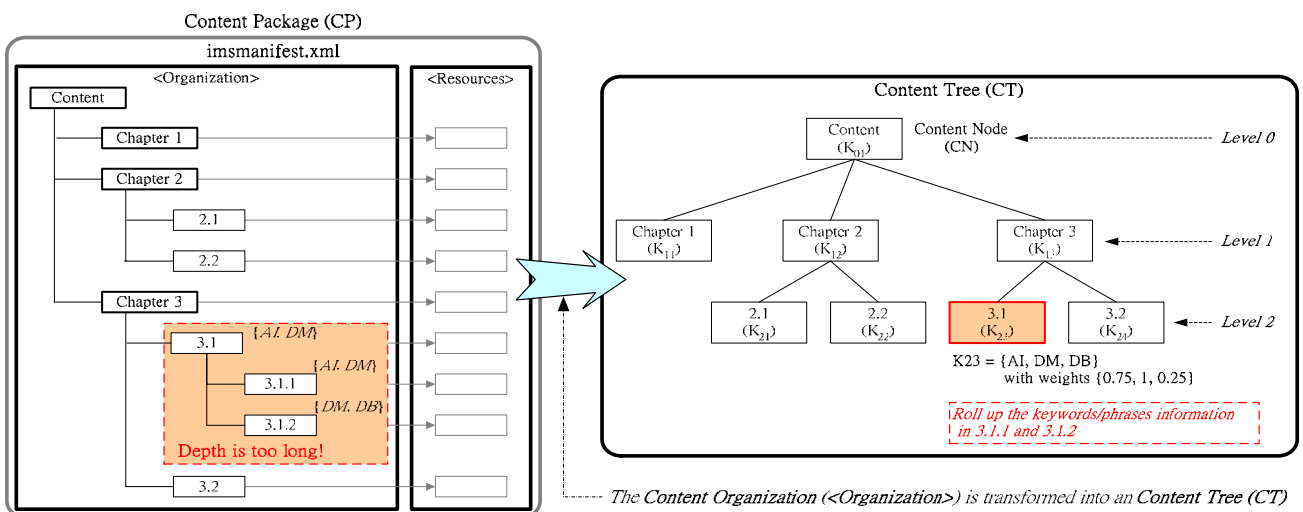


Figure 4.2: An Example of Content Tree Transforming

Algorithm 4.1: Content Package to Content Tree Algorithm (CP2CT-Alg)

Symbols Definition:

CP : denotes the SCORM content package.

CT : denotes the Content Tree transformed the CP.

CN : denotes the Content Node in CT.

CN_{leaf}: denotes the leaf node CN in CT.

D_{CT} : denotes the desired depth of CT.

D_{CN} : denotes the depth of a CN

Input : SCORM content package (CP)

Output : Content Tree (CT)

Step 1: For each element **<item>** in CP

1.1: Create a CN with keyword/phrase information.

1.2: Insert it into the corresponding level in **CT**.

Step 2: For each **CN_{leaf}** in CT

If the depth of **CN_{leaf}** > **D_{CT}**

Then its parent CN in depth = **D_{CT}** will merge the keywords/phrases of all included child nodes and run the rolling up process to assign the weight of those keywords/phrases.

Step 3: Content Tree (CT)

4.2 Information Enhancing Module

In general, it is a hard work for user to give learning materials an useful metadata, especially useful “keywords/phrases”. Therefore, we propose an information enhancement module to assist user to enhance the meta-information of learning materials automatically. This module consists of two processes: **1) Keyword/phrase Extraction Process** and **2) Feature Aggregation Process**. The former extracts additional useful keywords/phrases from other meta-information of a content node (CN). The latter aggregates the features of content nodes in a content tree (CT) according to its hierarchical relationships.

4.2.1 Keyword/phrase Extraction Process

Nowadays, more and more learning materials are designed as multimedia contents. Accordingly, it is difficult to extract meaningful semantics from multimedia resources. In SCORM, each learning object has plentiful metadata to describe itself. Thus we focus on the metadata of SCORM content package, like “title” and “description”, and want to find some useful keywords/phrases from them. These metadata contain plentiful information which can be extracted, but they often consist of a few sentences. So, traditional information retrieval techniques can not have a good performance here.

To solve the problem mentioned above, we propose a ***Keyword/phrase Extraction Algorithm (KE-Alg)*** to extract keyword/phrase from these short sentences. First, we use tagging techniques to indicate the candidate positions of interesting keyword/phrases. Then, we apply pattern matching technique to find useful patterns from those candidate phrases.

To find the potential keywords/phrases from the short context, we maintain sets of words and use them to indicate candidate positions where potential words/phrases may occur. For example: the phrase *after* the word “called” may be a key-phrase; the phrase *before* the word “are” may be a key-phrase; the word “this” will *not* be a part of key-phrases in general cases. These word-sets are stored in a database, called **Indication Sets (IS)**. At present, we just collect a **Stop-Word Set** to indicate the words which are not a part of key-phrases to break the sentences. Our *Stop-Word Set* includes punctuation marks, pronouns, articles, prepositions, and conjunctions in the English grammar. We still can collect more kinds of inference word sets to perform better prediction if it is necessary in the future.

Afterward, we use the **WordNet [WN]** to analyze the lexical features of the words in the candidate phrases. WordNet is a lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. It is developed by the Cognitive Science Laboratory at Princeton University. In WordNet, English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. And different relation-links have been maintained in the synonym sets. Presently, we just use WordNet (version 2.0) as a lexical analyzer here.

To extract useful keywords/phrases from the candidate phrases with lexical features, we have maintained another database, called **Pattern Base (PB)**. The patterns stored in *Pattern Base* are defined by domain experts. Each pattern consists of a sequence of lexical features or important words/phrases. Here are some examples: « *noun* + *noun* », « *adj.* + *adj.* + *noun* », « *adj.* + *noun* », « *noun* (if the word can only be a noun) », « *noun* + *noun* + “*scheme*” ». Every domain could have its own

interested patterns. These patterns will be used to find useful phrases, which may be a keyword/phrase of the corresponding domain. After comparing those candidate phrases by the whole *Pattern Base*, useful keywords/phrases will be extracted. Example 4.2 illustrates an example of the *Keywords/phrases Extraction Algorithm*. Those details are shown in Algorithm 4.2.

Example 4.2: Keyword/phrase Extraction

As shown in Figure 4.3, give a sentence as follows: “*challenges in applying artificial intelligence methodologies to military operations*”. We first use *Stop-Word Set* to partition it into several candidate phrases: {“*challenges*”, “*applying artificial intelligence methodologies*”, “*military operation*”}. By querying WordNet, we can get the lexical features of these candidate phrases are: {“*n/v*”, “*v+adj+n+n*”, “*n/adj+n*”}. Afterward, by matching with the important patterns stored in *Pattern Base*, we can find two interesting patterns “*adj+n*” and “*n/adj+n*” occurring in this sentence. Finally, we extract two key-phrases: “*artificial intelligence*” and “*military operation*”.

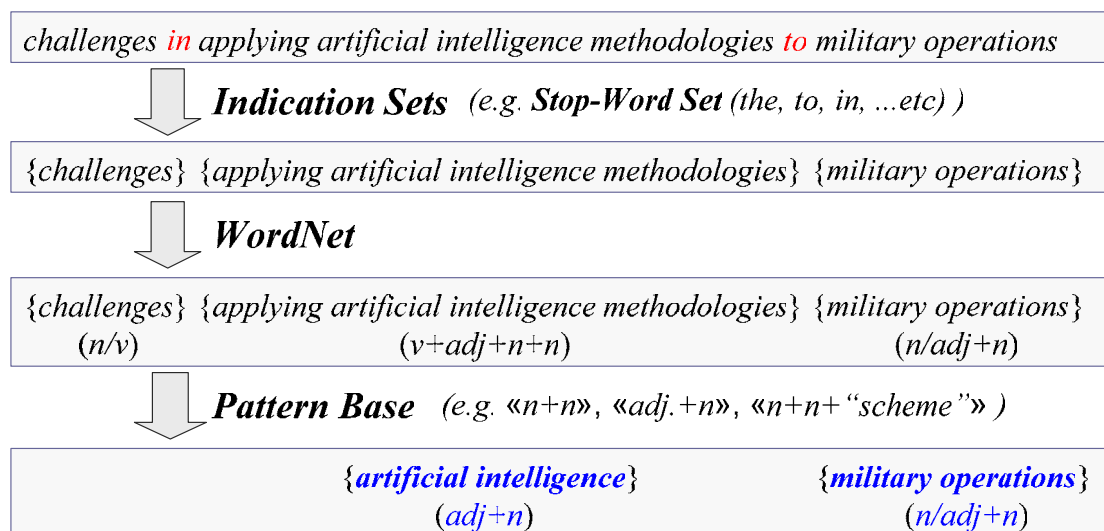


Figure 4.3: An Example of Keyword/phrase Extraction

Algorithm 4.2: Keyword/phrase Extraction Algorithm (KE-Alg)

Symbols Definition:

SWS: denotes a stop-word set; consists of punctuation marks, pronouns, articles, prepositions, and conjunctions in English grammar

PS : denotes a sentence

PC : denotes a candidate phrase

PK : denotes keyword/phrase

Input : a sentence

Output : a set of keyword/phrase (*PKs*) extracted from input sentence

Step 1: Break the input sentence into a set of *PCs* by *SWS*

Step 2: For each *PC* in this set

2.1: For each word in this *PC*

2.1.1: Find out the lexical feature of the word by querying **WordNet**.

2.2: Compare the lexical feature of this *PC* with **Pattern-Base**.

2.2.1: If there is any interesting pattern found in this *PC*,

mark the corresponding part as a *PK*.

Step 3: Return *PKs*

4.2.2 Feature Aggregation Process

In Section 4.2.1, additional useful keywords/phrases have been extracted to enhance the representative features of content nodes (CNs). In this section, we utilize the hierarchical relationship of a content tree (CT) to further enhance those features. Considering the nature of a CT: the nodes closer to the root will contain more general concepts which can cover all of its children nodes. For example, a learning content “data structure” must cover the concepts of “linked list”.

Before aggregating the representative features of a content tree (CT), we apply the **Vector Space Model (VSM)** approach [CK+92][RW86] to represent the keywords/phrases of a CN. Here, we encode each content node (CN) by the simple encoding method which uses single vector, called keyword vector (KV), to represent the keywords/phrases of the CN. Each dimension of the KV represents one keyword/phrase of the CN. And all representative keywords/phrases are maintained in a **Keyword/phrase Database** in the system.

Example 4.3: Keyword Vector (KV) Generation

As shown in Figure 4.4, the content node CN_A has a set of representative keywords/phrases: {“*e-learning*”, “*SCORM*”, “*learning object repository*”}. And we have a keyword/phrase database shown in the right part of Figure 4.4. Via a direct mapping, we can find the initial vector of CN_A is $\langle 1, 1, 0, 0, 1 \rangle$. Then, we normalize the initial vector and get the keyword vector of CN_A : $\langle 0.33, 0.33, 0, 0, 0.33 \rangle$

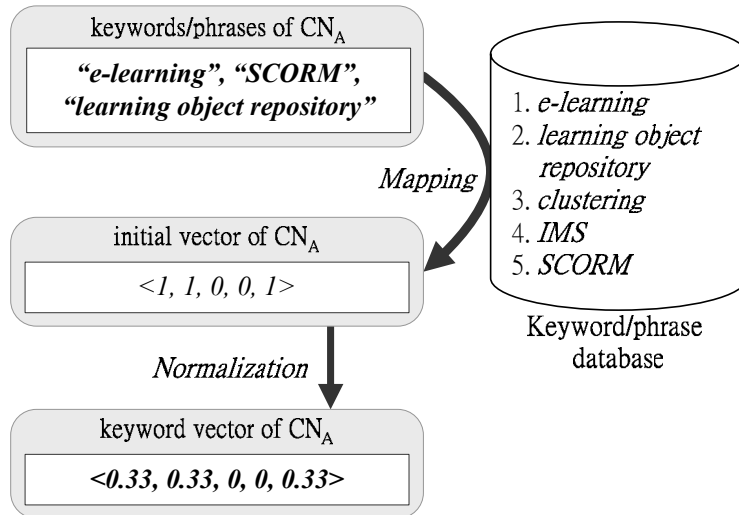


Figure 4.4: An Example of Keyword Vector Generation

After generating the keyword vectors (KVs) of content nodes (CNs), we compute the feature vector (FV) of each content node by aggregating its own keyword vector with the feature vectors of its children nodes. For the leaf node, we set its $FV = KV$; For the internal nodes, $FV = (1-\alpha) * KV + \alpha * \text{avg.}(FVs \text{ of its children})$, where α is a parameter used to define the intensity of the hierarchical relationship in a content tree (CT). The higher the α is, the more features are aggregated.

Example 4.4: Feature Aggregation

In Figure 4.5, content tree CT_A consists of three content nodes: CN_1 , CN_2 , and CN_3 . Now, we already have the KVs of these content nodes and want to calculate their feature vectors (FVs). For the leaf node CN_2 , $FV_{CN_2} = KV_{CN_2} = \langle 0.2, 0, 0.8, 0 \rangle$. Similarly, $FV_{CN_3} = KV_{CN_3} = \langle 0.4, 0, 0, 0.6 \rangle$. For the internal node CN_1 , according to the formula, $FV_{CN_1} = (1-\alpha) * KV_{CN_1} + \alpha * \text{avg.}(FV_{CN_2}, FV_{CN_3})$. Here we set the intensity parameter α as 0.5, so

$$\begin{aligned}
 FV_{CN_1} &= 0.5 * KV_{CN_1} + 0.5 * \text{avg.}(FV_{CN_2}, FV_{CN_3}) \\
 &= 0.5 * \langle 0.5, 0.5, 0, 0 \rangle + 0.5 * \text{avg.}(\langle 0.2, 0, 0.8, 0 \rangle, \langle 0.4, 0, 0, 0.6 \rangle) \\
 &= \langle 0.4, 0.25, 0.2, 0.15 \rangle
 \end{aligned}$$

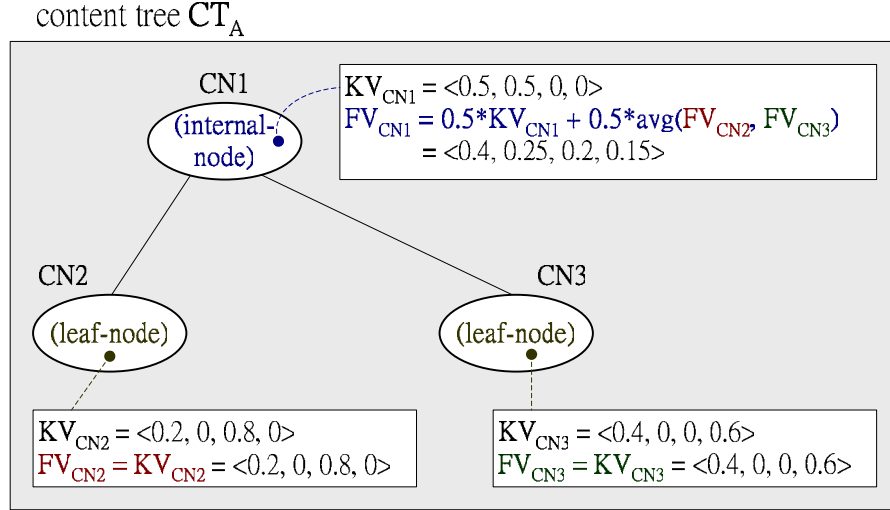


Figure 4.5: An Example of Feature Aggregation

Algorithm 4.3: Feature Aggregation Algorithm (FA-Alg)

Symbols Definition:

D : denotes the maximum depth of the content tree (CT)

$L_0 \sim L_{D-1}$: denote the levels of CT descending from the top level to the lowest level

KV : denotes the keyword vector of a content node (CN)

FV : denotes the feature vector of a CN

Input : a CT with keyword vectors

Output : a CT with feature vectors

Step 1: For $i = L_{D-1}$ to L_0

1.1: For each CN_j in L_i of this CT

1.1.1: If the CN_j is a leaf-node, $FV_{CN_j} = KV_{CN_j}$

Else, $FV_{CN_j} = (1-\alpha) KV_{CN_j} + \alpha * \text{avg.}(FVs \text{ of its child-nodes})$

Step 2: Return CT with feature vectors

4.3 Level-wise Content Clustering Module

After structure transforming and representative feature enhancing, we apply the clustering technique to create the relationships among content nodes (CNs) of content trees (CTs). In this thesis, we propose a Directed Acyclic Graph (DAG), called *Level-wise Content Clustering Graph (LCCG)*, to store the related information of each cluster. Based upon the LCCG, the desired learning content including general and specific LOs can be retrieved for users.

4.3.1 Level-wise Content Clustering Graph (LCCG)

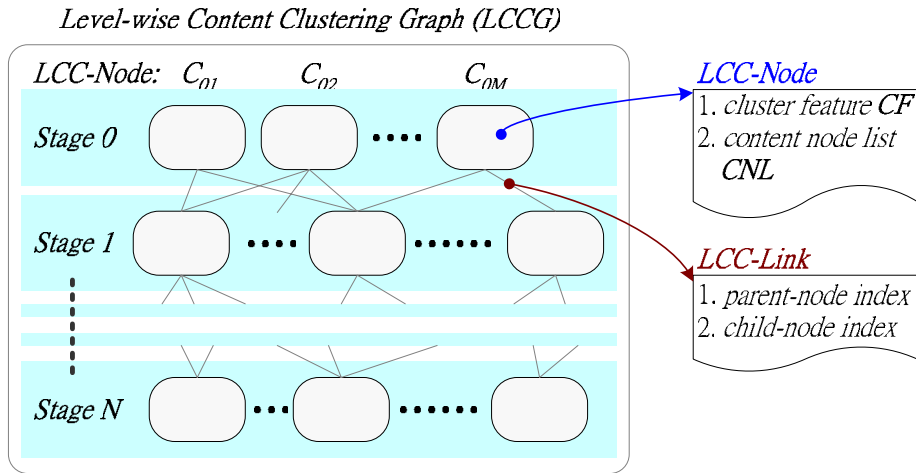


Figure 4.6: The Representation of Level-wise Content Clustering Graph

As shown in Figure 4.6, *LCCG* is a multi-stage graph with relationships information among learning objects, e.g., a Directed Acyclic Graph (DAG). Its definition is described in Definition 4.2:

Definition 4.2: Level-wise Content Clustering Graph (LCCG)

Level-wise Content Clustering Graph (LCCG) = (N, E), where

- $N = \{ (CF_0, CNL_0), (CF_1, CNL_1), \dots, (CF_m, CNL_m) \}$.

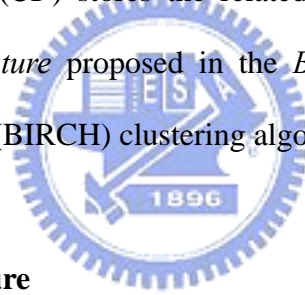
It stores the related information, *Cluster Feature (CF)* and *Content Node*

List (CNL), in a cluster, called **LCC-Node**. The **CNL** stores the indexes of learning objects included in this **LCC-Node**.

- $E = \{ \overrightarrow{n_i n_{i+1}} \mid 0 \leq i < \text{the depth of LCCG} \}$.

It denotes the link edge from node n_i in upper stage to n_{i+1} in immediate lower stage.

For the purpose of content clustering, the number of the stages of LCCG is equal to the maximum depth (δ) of CT, and each stage handles the clustering result of these CNs in the corresponding level of different CTs. That is, the top stage of LCCG stores the clustering results of the root nodes in the CTs, and so on. In addition, in LCCG, the **Cluster Feature (CF)** stores the related information of a cluster. It is similar with the *Cluster Feature* proposed in the *Balance Iterative Reducing and Clustering using Hierarchies (BIRCH)* clustering algorithm and defined as follows.



Definition 4.3: Cluster Feature

The Cluster Feature (CF) = (N, \overrightarrow{VS} , CS), where

- N: it denotes the number of the content nodes (CNs) in a cluster.
- $\overrightarrow{VS} = \sum_{i=1}^N \overrightarrow{FV}_i$. It denotes the sum of feature vectors (FVs) of CNs.
- $CS = \left| \sum_{i=1}^N \vec{V}_i / N \right| = \left| \overrightarrow{VS} / N \right|$. It denotes the average value of the feature vector sum in a cluster. The $||$ denotes the *Euclidean distance* of the feature vector. The $(\overrightarrow{VS} / N)$ can be seen as the **Cluster Center (CC)** of a cluster.

Moreover, during content clustering process, if a content node (CN) in a content tree (CT) with feature vector (\overrightarrow{FV}) is inserted into the cluster $CF_A = (N_A, \overrightarrow{VS}_A, CS_A)$,

the new $CF_A = (N_A + 1, \overline{VS}_A + \overline{FV}, \left| \frac{\overline{VS}_A + \overline{FV}}{N_A + 1} \right|)$. An example of Cluster Feature (CF) and Content Node List (CNL) is shown in Example 4.5.

Example 4.5: Cluster Feature (CF) and Content Node List (CNL)

Assume a cluster C_0 stores in the LCC-Node N_A with (CF_A, CNL_A) and contains four CNs: $CN_{01}, CN_{02}, CN_{03},$ and CN_{04} , which include four feature vectors, $\langle 3,3,2 \rangle, \langle 3,2,2 \rangle, \langle 2,3,2 \rangle$ and $\langle 4,4,2 \rangle$, respectively. Then, the $\overline{VS}_A = \langle 12,12,8 \rangle$, the $CC = \overline{VS}_A / N_A = \langle 3,3,2 \rangle$, and the $CS_A = |CC| = (9+9+4)^{1/2} = 4.69$. Thus, the $CF_A = (4, \langle 12,12,8 \rangle, 4.69)$, and $CNL_A = \{ CN_{01}, CN_{02}, CN_{03}, CN_{04} \}$

4.3.2 Incremental Level-wise Content Clustering Algorithm

Based upon the definition of LCCG, we propose an *Incremental Level-wise Content Clustering Algorithm*, called *ILCC-Alg*, to create the LCC-Graph according to the CTs transformed from learning objects. The *ILCC-Alg* includes two processes: 1) *Single Level Clustering Process*, 2) *Content Cluster Refining Process*, and 3) *Concept Relation Connection Process*. Figure 4.7 illustrates the flowchart of *ILCC-Alg*.

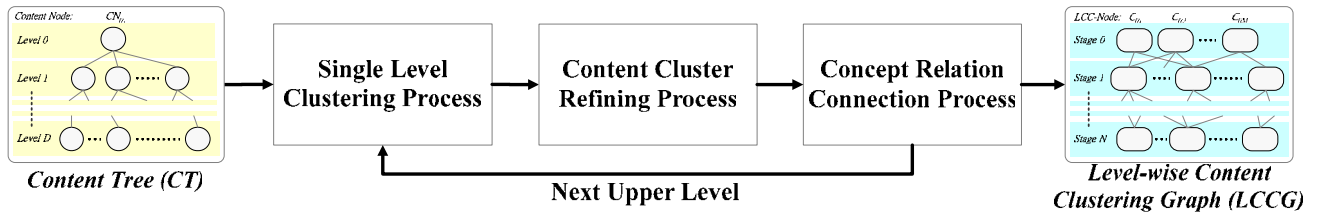


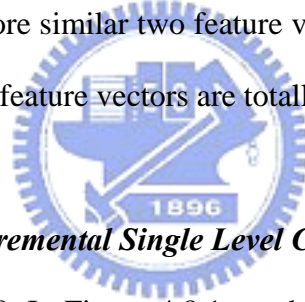
Figure 4.7: The Process of ILCC-Algorithm

(1) Single Level Clustering Process

In this process, the content nodes (CNs) of CT in each tree level can be clustered by different similarity threshold. The content clustering process is started from the lowest level to the top level in CT. All clustering results are stored in the LCCG. In addition, during content clustering process, the similarity measure between a CN and an LCC-Node is defined by the cosine function which is the most common for the document clustering. It means that, given a CN CN_A and an LCC-Node $LCCN_A$, the similarity measure is calculated by

$$sim(CN_A, LCCN_A) = \cos(FV_{CN_A}, FV_{LCCN_A}) = \frac{FV_{CN_A} \bullet FV_{LCCN_A}}{|FV_{CN_A}| |FV_{LCCN_A}|},$$

where FV_{CN_A} and FV_{LCCN_A} are the feature vectors of CN_A and $LCCN_A$ respectively. The larger the value is, the more similar two feature vectors are. And the cosine value will be equal to 1 if these two feature vectors are totally the same.



The basic concept of *Incremental Single Level Clustering Algorithm (ISLC-Alg)* is also described in Figure 4.8. In Figure 4.8.1, we have an existing clustering result and two new objects, CN_4 and CN_5 , needed to be clustered. First we compute the similarity between CN_4 and the existing clusters, $LCC-Node_1$ and $LCC-Node_2$. In this example, the similarities between them are all smaller than the similarity threshold. That means the concept of CN_4 is not similar with the concepts of existing clusters, so we treat CN_4 as a new cluster $LCC-Node_3$. Then we cluster the next new object, CN_5 . After computing and comparing the similarities between CN_5 and existing clusters, we find CN_5 is similar enough with $LCC-Node_2$, so we put CN_5 into $LCC-Node_2$ and update the feature of this cluster. The final result of this example is shown in Figure 4.8.4. Moreover, the detail of *ISLC-Alg* is shown in Algorithm 4.1.

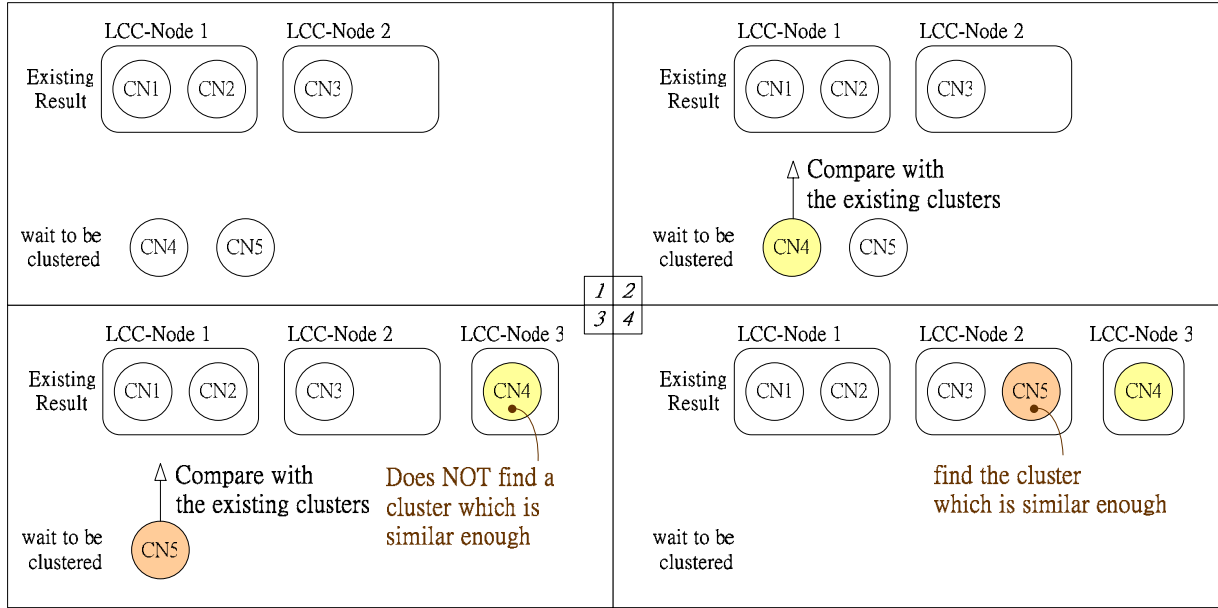


Figure 4.8: An Example of Incremental Single Level Clustering

Algorithm 4.4: Incremental Single Level Clustering Algorithm (ISLC-Alg)

Symbols Definition:

LN_{Set} : the existing LCC-Nodes (LN_S) in the same level (L)

CN_N : a new content node (CN) needed to be clustered

T_i : the similarity threshold of the level (L) for clustering process.

Input : LN_{Set} , CN_N and T_i .

Output : The set of LCC-Nodes storing the new clustering results.

Step 1: $\forall n_i \in LN_{Set}$, calculate the similarity $sim(n_i, CN_N)$

Step 2: Find the most similar one, n^* , for CN_N

2.1: If $sim(n^*, CN_N) > T_i$

Then insert CN_N into the cluster n^* and update its **CF** and **CL**

Else insert CN_N as a new cluster stored in a new LCC-Node.

Step 3: Return the set of the LCC-Nodes.

(2) Content Cluster Refining Process

Due to the ISLC-Alg algorithm runs the clustering process by inserting the content trees (CTs) incrementally, the content clustering results are influenced by the inputs order of CNs. In order to reduce the effect of input order, the Content Cluster Refining Process is necessary. Given the content clustering results of ISLC-Alg, Content Cluster Refining Process utilizes the cluster centers of original clusters as the inputs and runs the single level clustering process again for modifying the accuracy of original clusters. Moreover, the similarity of two clusters can be computed by the Similarity Measure as follows:

$$\text{Similarity} = \text{Cos} (CC_A, CC_B) = \frac{CC_A \bullet CC_B}{|CC_A| |CC_B|} = \frac{(\overrightarrow{VS}_A / N_A) \bullet (\overrightarrow{VS}_B / N_B)}{CS_A * CS_B}$$

After computing the similarity, if the two clusters have to be merged into a new cluster, the new **CF** of this new cluster is: $\text{CF}_{\text{new}} = (N_A + N_B, \overrightarrow{VS}_A + \overrightarrow{VS}_B, |(\overrightarrow{VS}_A + \overrightarrow{VS}_B) / (N_A + N_B)|)$.

(3) Concept Relation Connection Process

The concept relation connection process is used to create the links between LCC-Nodes in adjacent stages of LCCG. Based on the hierarchical relationships stores in content trees (CTs), we can find the relationships between more general subjects and more specific ones. Thus, after applying ISLC-Alg to two adjacent stages, we then apply *Concept Relation Connection Process* and create new LCC-Links

Figure 4.9 shows the basic concept of *Incremental Level-wise Content Clustering Algorithm (ILCC-Alg)*. Every time getting a new content tree (CT), we

apply *ISLC-Alg* from bottom to top, and update the semantic relation links between adjacent stages. Finally we can get a new clustering result. The algorithm of *ILCC-Alg* is shown in Algorithm 4.5.

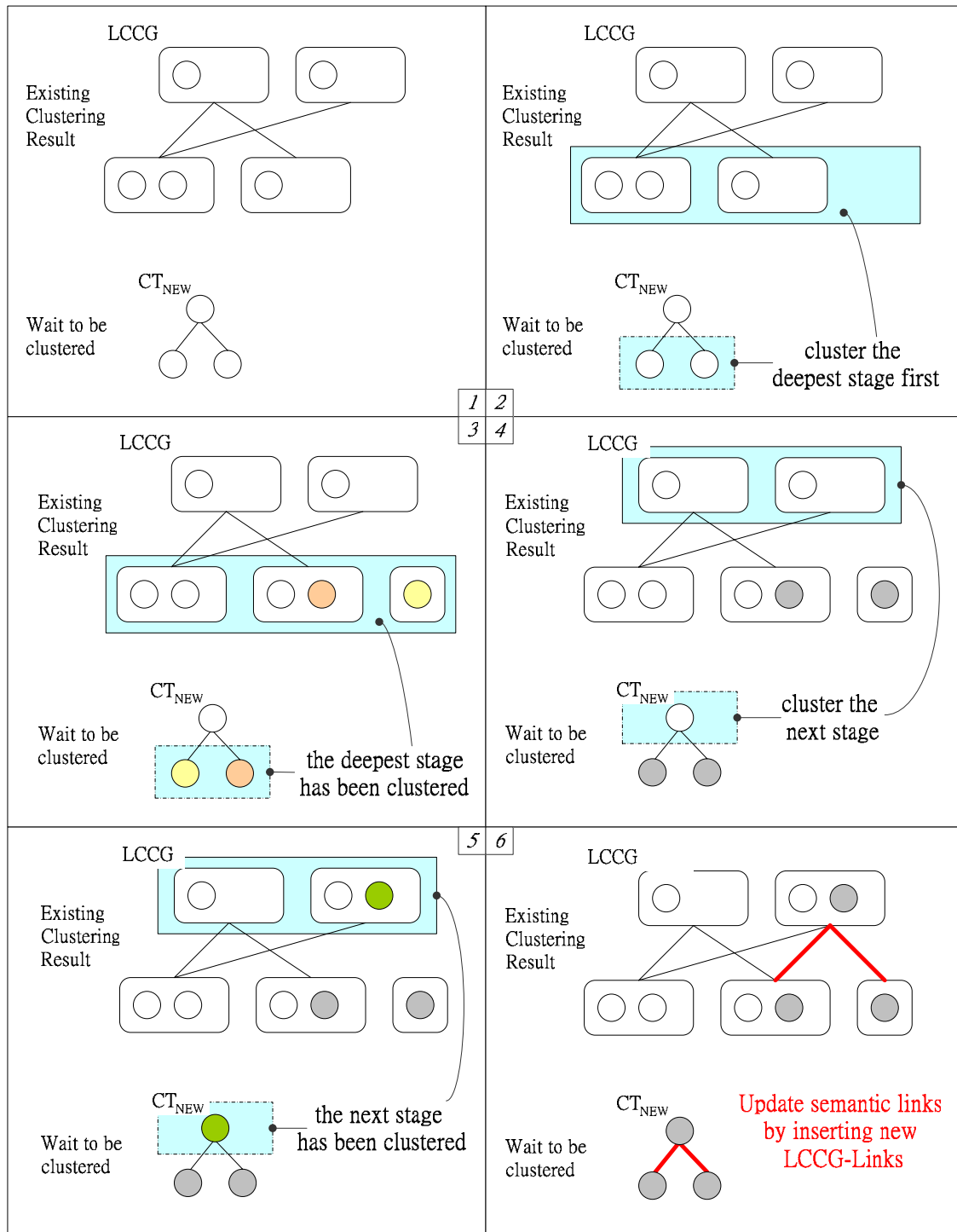


Figure 4.9: An Example of Incremental Level-wise Content Clustering

**Algorithm 4.5: Incremental Level-wise Content Clustering Algorithm
(ILCC-Alg)**

Symbols Definition:

- D** : denotes the maximum depth of the content tree (CT).
- $L_0 \sim L_{D-1}$** : denote the levels of CT descending from the top level to the lowest level.
- $S_0 \sim S_{D-1}$** : denote the stages of LCC-Graph.
- $T_0 \sim T_{D-1}$** : denote the similarity thresholds for clustering the content nodes (CNs) in the level $L_0 \sim L_{D-1}$ respectively.
- CT_N** : denotes a new CT with a maximum depth (**D**) needed to be clustered.
- CN_{Set}** : denotes the CNs in the content tree level (**L**).
- LG** : denotes the existing LCC-Graph
- LN_{Set}** : denotes the existing LCC-Nodes (LN_S) in the same level (**L**)

Input : LG, CT_N , $T_0 \sim T_{D-1}$

Output : LCCG which holds the clustering results in every content tree level.

Step 1: For $i = L_{D-1}$ to L_0 , do the following Step 2 to Step 4.

Step 2: Single Level Clustering:

2.1: $LN_{Set} =$ the $LN_S \in LG$ in L_i

2.2: $CN_{Set} =$ the $CN_S \in CT_N$ in L_i

2.2: For LN_{Set} and any $CN \in CN_{Set}$,

Run *Incremental Single Level Clustering Algorithm (ISLC-Alg)*

with threshold T_i .

Step 3: If $i < D-1$

3.1: Construct LCCG-Link between S_i and S_{i+1} .

Step 4: Return the new LCCG

Chapter 5 Searching Phase of LCMS

In this chapter, we describe the searching phase of LCMS, which includes 1) *Preprocessing* module, 2) *Content-based Query Expansion* module and 3) *LCCG Content Searching* module, shown in the right part of Figure 3.1.

5.1 Preprocessing Module

In this module, we translate user's query into a vector to represent the concepts user want to search. Here, we encode a query by the simple encoding method which uses a single vector, called query vector (QV), to represent the keywords/phrases in the user's query. If a keyword/phrase appears in the *Keyword/phrase Database* of the system, the corresponding position in the query vector will be set as "1". If the keyword/phrase does not appear in the *Keyword/phrase Database*, it will be ignored. And all the other positions in the query vector will be set as "0".

Example 5.1: Preprocessing: Query Vector Generator

As shown in Figure 5.1, the original query is: {"*e-learning*", "*LCMS*", "*learning object repository*"}. And we have a *Keyword/phrase Database* shown in the right part of Figure 5.1. Via a direct mapping, we can find the query vector is $\langle 1, 0, 0, 0, 1 \rangle$.

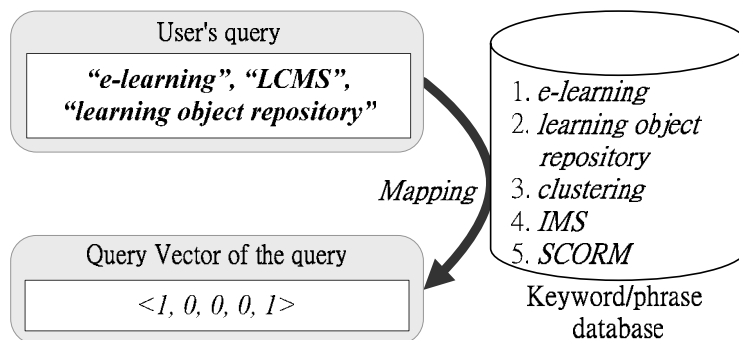


Figure 5.1: Preprocessing: Query Vector Generator

5.2 Content-based Query Expansion Module

In general, while users want to search desired learning contents, they usually make rough queries, or called short queries. Using this kind of queries, users will retrieve a lot of irrelevant results. Then, they need to browse many irrelevant item to learn “How to set an useful query in this system to get what I want?” by themselves. In most cases, systems use the relational feedback provided by users to refine the query and do another search, iteratively. It works but often takes time for users to browse a lot of non-interested items. In order to assist users efficiently find more specific content, we proposed a query expansion scheme, called *Content-based Query Expansion*, based on the multi-stage index of LOR, i.e., LCCG.

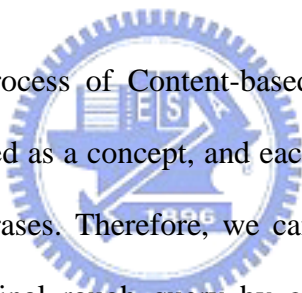


Figure 5.2 shows the process of Content-based Query Expansion. In LCCG, every LCC-Node can be treated as a concept, and each concept has its own feature: a set of weighted keywords/phrases. Therefore, we can search the LCCG and find a sub-graph related to the original rough query by computing the similarity of the feature vector stored in LCC-Nodes and the query vector. Then we integrate these related concepts with the original query by calculating the linear combination of them. After concept fusing, the expanded query could contain more concepts and perform a more specific search. Users can control an *expansion degree* to decide how much expansion s/he needs. Via this kind of query expansion, users can use rough query to find more specific content stored in the LOR in less iterations of query refinement. The algorithm of Content-based Query Expansion is described in Algorithm 5.1.

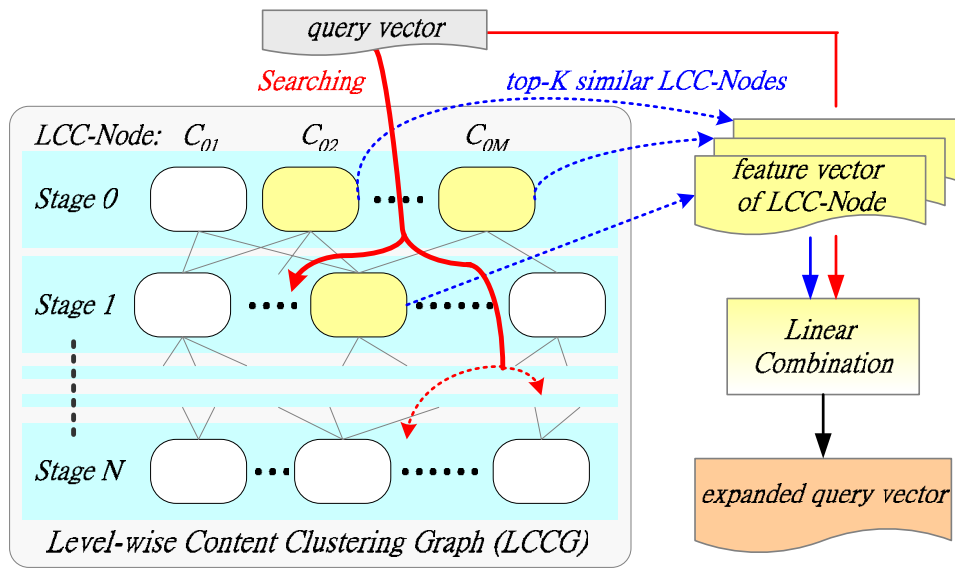


Figure 5.2: The Process of Content-based Query Expansion

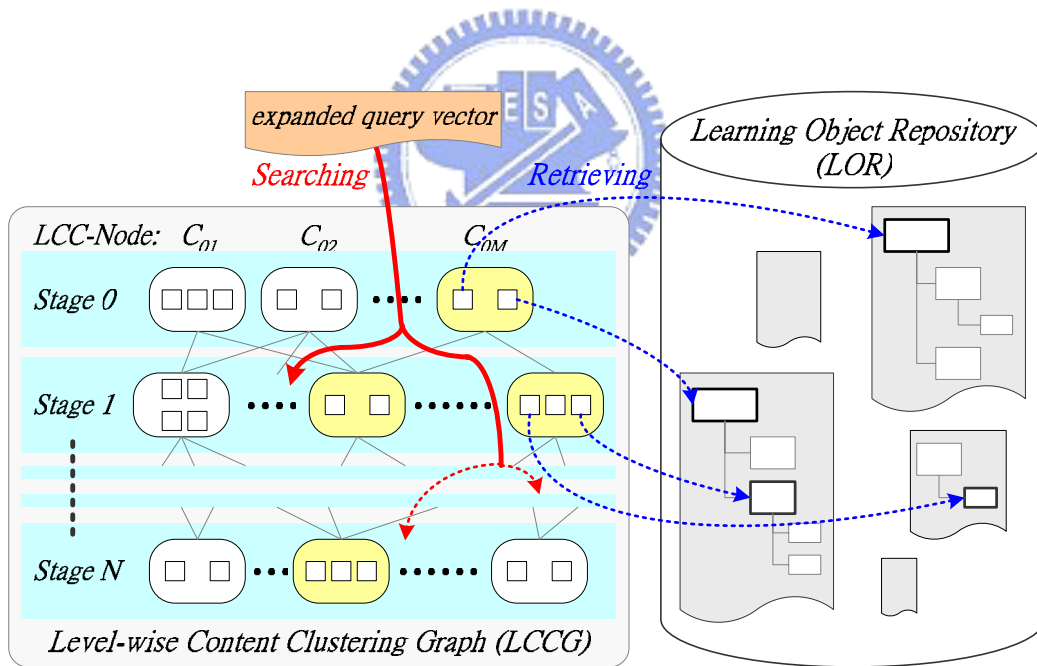


Figure 5.3: The Process of LCCG Content Searching

Algorithm 5.1: Content-based Query Expansion Algorithm (CQE-Alg)

Symbols Definition:

Q : denotes the query vector whose dimension is the same as the *feature vector* of content node (CN)

T_E : denotes the expansion threshold assigned by user

β : denotes the expansion parameter assigned by system administrator

S₀~S_{D-1}: denote the stage of an LCCG from the top stage to the lowest stage

ExpansionSet and **DataSet**: denote the sets of LCC-Nodes

Input : a query vector **Q**, expansion threshold **T_E**

Output : an expanded query vector **EQ**

Step 1: Initial the **ExpansionSet** = ϕ and **DataSet** = ϕ

Step 2: For each stage **S_i** ∈ LCCG,

repeatedly execute the following steps until **S_i** ≥ S_{DES}

2.1: **DataSet** = **DataSet** ∪ LCC-Nodes in stage **S_i** and **ExpansionSet** = ϕ

2.2: For each **N_j** ∈ **DataSet**,

If (the similarity between **N_j** and **Q**) ≥ **T_E**

Then insert **N_j** into **ExpansionSet**

2.3: **DataSet** = **ExpansionSet** //for searching more precise LCC-Nodes in next stage in LCCG

Step 3: **EQ** = (1-β)***Q** + β*avg(feature vectors of LCC-Nodes in **ExpansionSet**)

Step 4: return **EQ**

5.3 LCCG Content Searching Module

The process of LCCG Content Searching is shown in Figure 5.3. In LCCG, every LCC-Node contains several similar content nodes (CNs) in different content trees (CTs) transformed from content package of SCORM compliant learning materials. The content within LCC-Nodes in upper stage is more general than the content in lower stage. Therefore, based upon the LCCG, users can get their interesting learning contents which contain not only general concepts but also specific concepts. The interesting learning content can be retrieved by computing the similarity of cluster center (CC) stored in LCC-Nodes and the query vector. If the similarity of LCC-Node satisfies the query threshold users defined, the information of learning contents recorded in this LCC-Node and its included child LCC-Nodes are interested for users. Moreover, we also define the *Near Similarity Criterion* to decide when to stop the searching process. Therefore, if the similarity between the query and the LCC-Node in the higher stage satisfies the definition of *Near Similarity Criterion*, it is not necessary to search its included child LCC-Nodes which may be too specific to use for users. The *Near Similarity Criterion* is defined as follows:

Definition 5.1: Near Similarity Criterion

Assume that the similarity threshold T for clustering is less than the similarity threshold S for searching. Because similarity function is the *cosine* function, the threshold can be represented in the form of the angle. The angle of T is denoted as $\theta_T = \cos^{-1} T$ and the angle of S is denoted as $\theta_S = \cos^{-1} S$. When the angle between the query vector and the cluster center (CC) in LCC-Node is lower than $\theta_S - \theta_T$, we define that the LCC-Node is *near* similar for the query. The diagram of Near Similarity is shown in Figure.

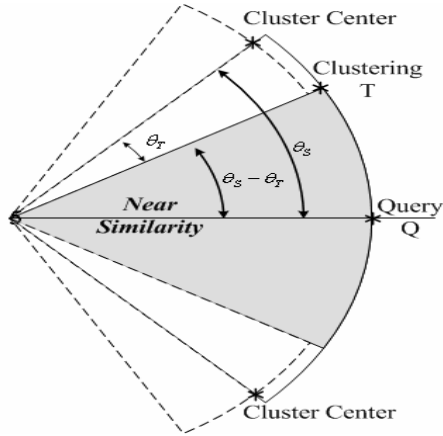


Figure 5.4: The Diagram of Near Similarity According to the Query Threshold Q and Clustering Threshold T

In other words, *Near Similarity Criterion* is that the similarity value between the query vector and the cluster center (CC) in LCC-Node is larger than $\text{Cos}(\theta_s - \theta_r)$, so that the *Near Similarity* can be defined again according to the similarity threshold **T** and **S**.

$$\begin{aligned} \text{Near Similarity} > \text{Cos}(\theta_s - \theta_r) &= \text{Cos}\theta_s \text{Cos}\theta_r + \text{Sin}\theta_s \text{Sin}\theta_r \\ &= \text{S} \times \text{T} + \left(\sqrt{1 - \text{S}^2}\right) \left(\sqrt{1 - \text{T}^2}\right) \end{aligned}$$

By the *Near Similarity Criterion*, the algorithm of the **LCCG Content Searching Algorithm (LCCG-CSAlg)** is proposed as shown in Algorithm 5.2.

Algorithm 5.2: LCCG Content Searching Algorithm (LCCG-CSAlg)

Symbols Definition:

Q : denotes the *query vector* whose dimension is the same as the *feature vector* of content node (CN)

D : denotes the number of the stage in an LCCG.

S₀~S_{D-1}: denote the stage of an LCCG from the top stage to the lowest stage.

ResultSet, **DataSet**, and **NearSimilaritySet**: denote the sets of LCC-Nodes.

Input: The query vector **Q**, search threshold **T** and the destination stage **S_{DES}** where $S_0 \leq S_{DES} \leq S_{D-1}$.

Output: the **ResultSet** contains the set of similar clusters stored in LCC-Nodes.

Step 1: Initiate the **DataSet** = ϕ and **NearSimilaritySet** = ϕ

Step 2: For each stage **S_i** ∈ LCCG,

repeatedly execute the following steps until **S_i** ≥ **S_{DES}**

2.1: **DataSet** = **DataSet** ∪ LCC-Nodes in stage **S_i**, and **ResultSet** = ϕ

2.2: For each **N_j** ∈ **DataSet**,

If **N_j** is near similar with **Q**

Then insert **N_j** into **NearSimilaritySet**.

Else If (the similarity between **N_j** and **Q**) ≥ **T**

Then insert **N_j** into **ResultSet** }

2.3: **DataSet** = **ResultSet**. //for searching more precise LCC-Nodes in next stage in LCCG

Step 3: Output the **ResultSet** = **ResultSet** ∪ **NearSimilaritySet**

Chapter 6 Implementation and Experimental Results

6.1 System Implementation

To evaluate the performance, we have implemented a web-based system, called *Learning Object Management System (LOMS)*. The operating system of our web server is FreeBSD4.9. Besides, we use PHP4 as the programming language and MySQL as the database to build up the whole system.

Figure 6.1 shows the configuration page of our LOMS. The upper part lists the parameters used in our *Level-wise Content Management Scheme (LCMS)*. The “maximum depth of a content tree” is used in *CP2CT-Alg* to decide the maximum depth of the content trees (CTs) transformed from SCORM content packages (CPs). Then the “clustering similarity thresholds” defines the clustering thresholds of each level in the *ILCC-Alg*. Besides, the “searching similarity thresholds” and “near similarity threshold” are used in the *LCCG-CSAlg* to traverse the LCCG and retrieve the desired learning objects. The lower part of this page provides the links to maintain the *Keyword/phrase Database*, *Stop-Word Set*, and *Pattern Base* of our system.

As shown in Figure 6.2, users can set the query words to search LCCG and retrieve the desired learning contents. Besides, they can also set other searching criterions about other SCORM metadata such as “version”, “status”, “language”, “difficulty”, etc. to do further restrictions. Then all searching results with hierarchical relationships are shown in Figure 6.3. By displaying the learning objects with their hierarchical relationships, users can know more clearly if that is what they want. Besides, users can search the relevant items by simply clicking the buttons in the left

side of this page or view the desired learning contents by selecting the hyper-links. As shown in Figure 6.4, a learning content can be found in the right side of the window, and the hierarchical structure of this learning content is listed in the left side. Therefore, user can easily browse the other parts of this learning contents without perform another search.

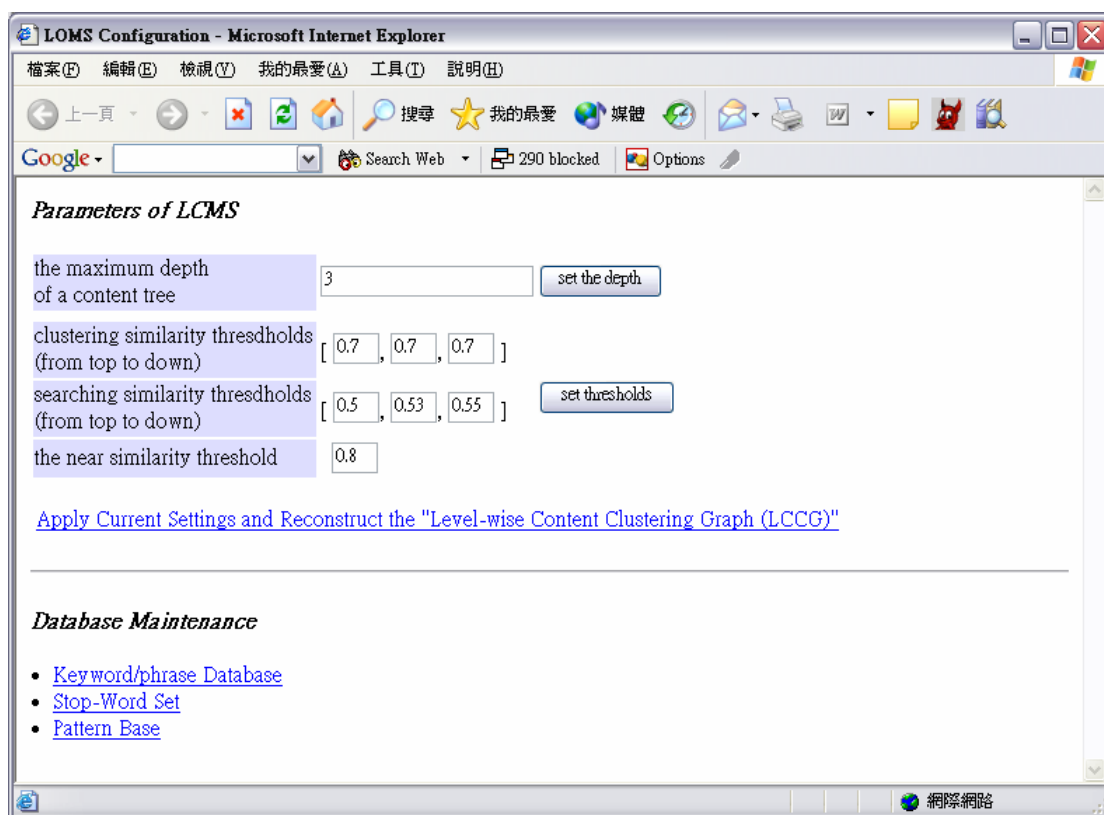


Figure 6.1: System Screenshot: LOMS configuration

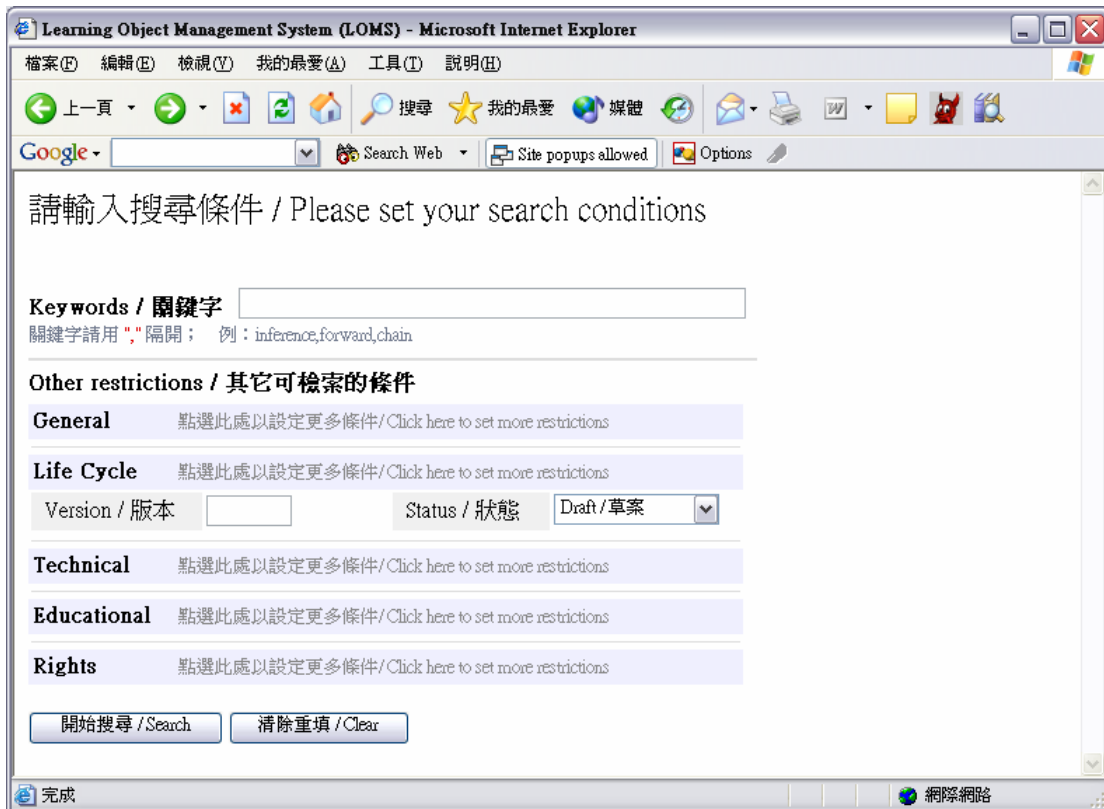


Figure 6.2: System Screenshot: Searching

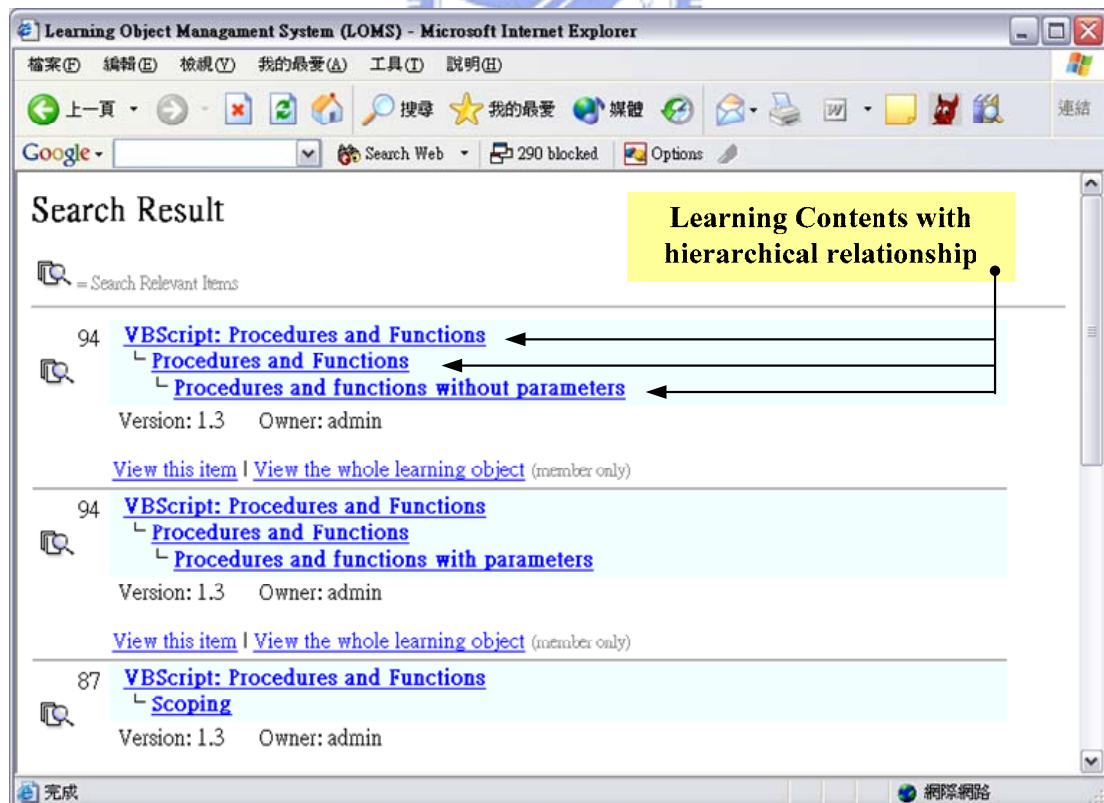


Figure 6.3: System Screenshot: Searching Results

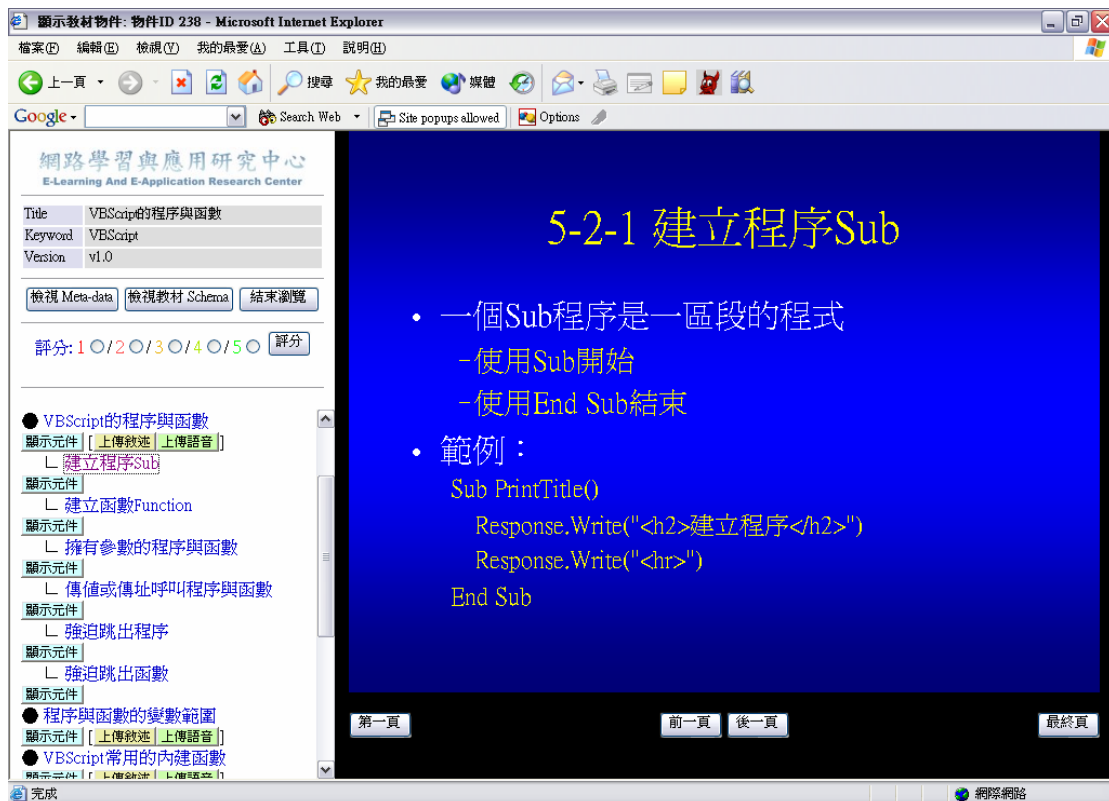


Figure 6.4: System Screenshot: Viewing Learning Objects



6.2 Experimental Results

In this section, we describe the experimental results about our LCMS.

(1) Synthetic Learning Materials Generation and Evaluation Criterion

Here, we use synthetic learning materials to evaluate the performance of our clustering algorithms. All synthetic learning materials are generated by three parameters: 1) **V**: The dimension of feature vectors in learning materials, 2) **D**: the depth of the content structure of learning materials, 3) **B**: the upper bound and lower bound of included sub-section for each section in learning materials.

In the Incremental Level-wise Content Clustering Algorithm (ILCC-Alg), the Incremental Single Level Clustering Algorithm (ISLC-Alg) can be seen as a kind of traditional clustering algorithms. To evaluate the performance, we compare the

performance of ILCC-Alg with ISLC-Alg which uses the leaf-nodes as input in content trees. The resulted cluster quality is evaluated by the **F-measure** [LA99] which combines the *precision* and *recall* from the information retrieval. The F-measure is formulated as follows:

$$F = \frac{2 \times P \times R}{P + R}$$

, where P and R are *precision* and *recall* respectively. The range of F-measure is $[0,1]$. The higher the F-measure is, the better the clustering result is.

(2) Experimental Results of Synthetic Learning materials

There are 500 synthetic learning materials with $V=15$, $D=3$, and $B = [5, 10]$ are generated. The clustering thresholds of ILCC-Alg and ISLC-Alg are 0.92. After clustering, there are 101, 104 and 2529 clusters generated from 500, 3664 and 27456 content nodes in the level L_0 , L_1 , and L_2 of content trees, respectively. Then, 30 queries generated randomly are used to compare the performance of two clustering algorithms. The **F-measure** of each query with threshold 0.85 is shown in Figure 6.5. Moreover, this experiment is run on AMD Athlon 1.13GHz processor with 512 MB DDR RAM under the Windows XP operating system. As shown in Figure 6.5, the differences of the F-measures between ILCC-Alg and ISLC-Alg are small in most cases. Moreover, in Figure 6.6, the searching time using LCCG-CSAlg in ILCC-Alg is far less than the time needed in ISLC-Alg. Figure 6.7 shows that the clustering with clustering refinement can improve the accuracy of LCCG-CSAlg search.

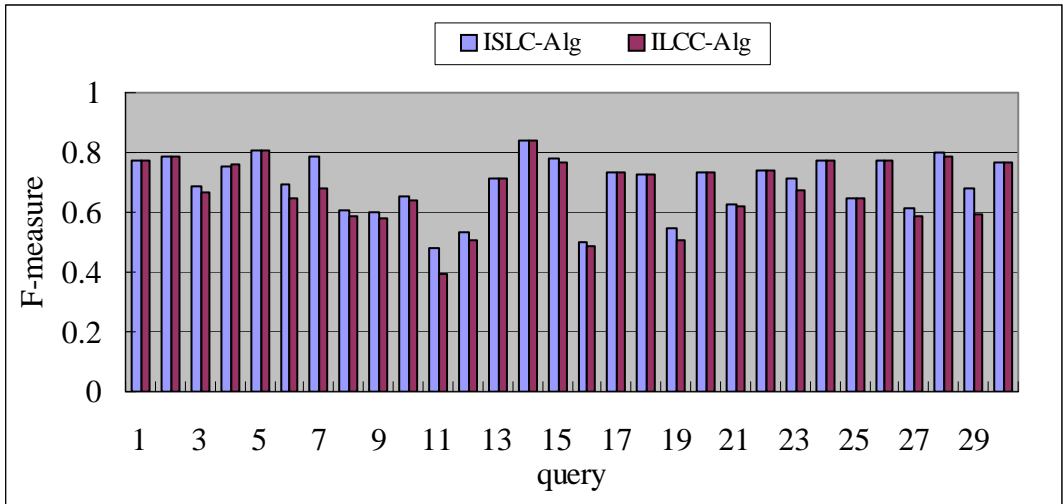


Figure 6.5: The F-measure of Each Query

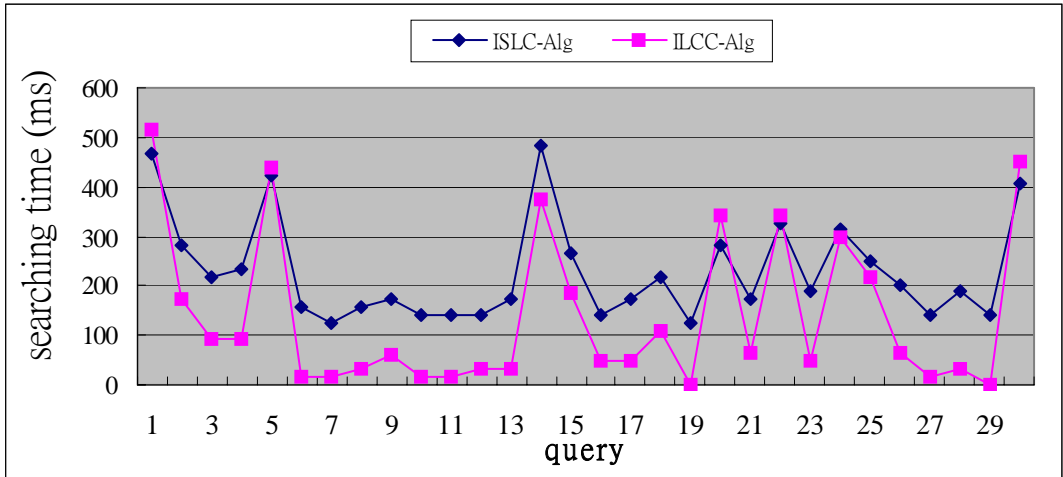


Figure 6.6: The Searching Time of Each Query

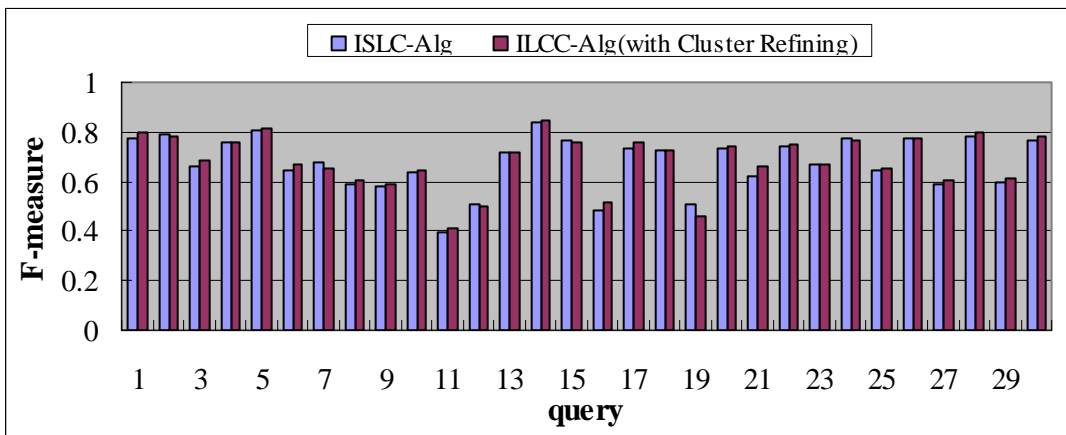


Figure 6.7: The Comparison of ISLC-Alg and ILCC-Alg with Cluster Refining

(3) Real Learning Materials Experiment

In order to evaluate the performance of our LCMS more practically, we also do two experiments using the real SCORM compliant learning materials. Here, we collect 100 articles with 5 specific topics: *concept learning*, *data mining*, *information retrieval*, *knowledge fusion*, and *intrusion detection*, where every topic contains 20 articles. Every article is transformed into SCORM compliant learning materials and then imported into our web-based system. In addition, 15 participants, who are graduate students of *Knowledge Discovery and Engineering Lab of NCTU*, used the system to query their desired learning materials.

To evaluate our Content-based Query Expansion Algorithm (CQE-Alg), we select several sub-topics contained in our collection and request participants to search them using at most two keywords/phrases with/without our query expansion function. In this experiments, every sub-topic is assigned to three or four participants to perform the search. And then we compare the precision and recall of those search results to analyze the performance. As shown in Figure 6.9 and Figure 6.10, after applying the CQE-Alg, because we can expand the initial query and find more learning objects in some related domains, the precision may decrease slightly in some cases while the recall can be significantly improved. Moreover, as shown in Figure 6.11, in most real cases, the F-measure can be improved in most cases after applying our CQE-Alg. Therefore, we can conclude that our query expansion scheme can help users find more desired learning objects without reducing the search precision too much.

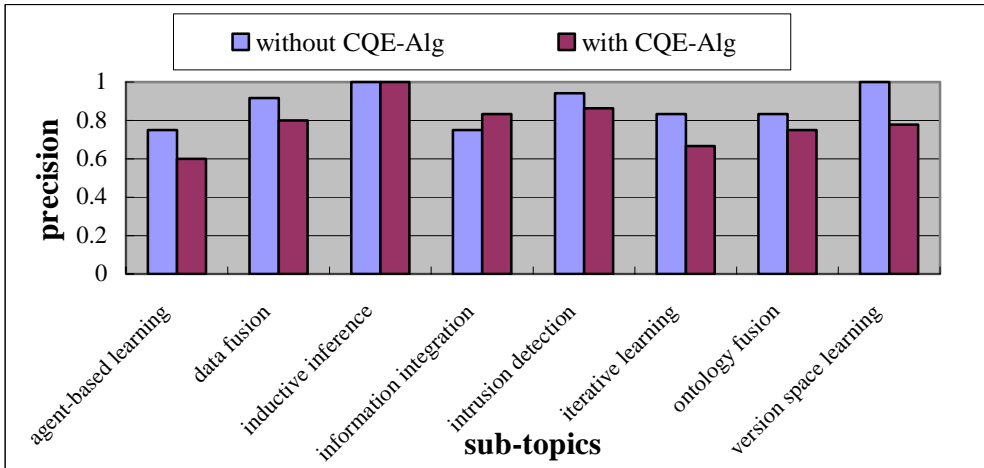


Figure 6.9: The precision with/without CQE-Alg

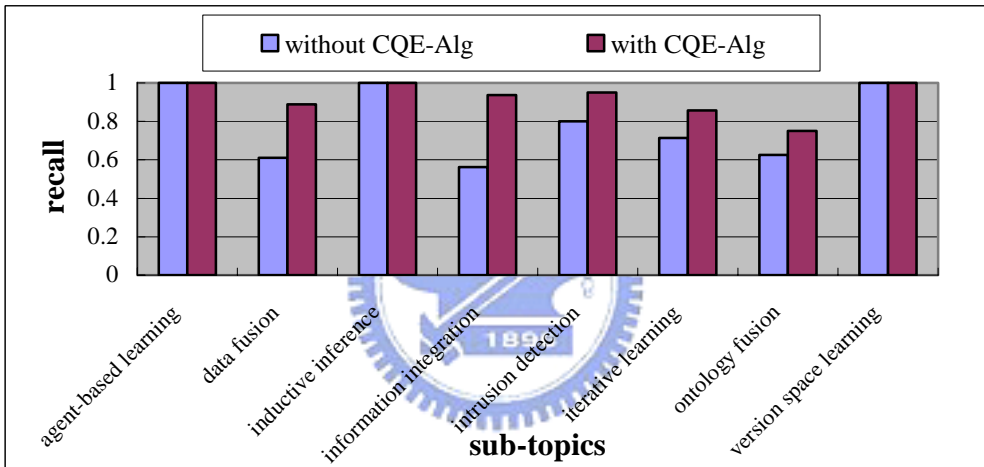


Figure 6.10: The recall with/without CQE-Alg

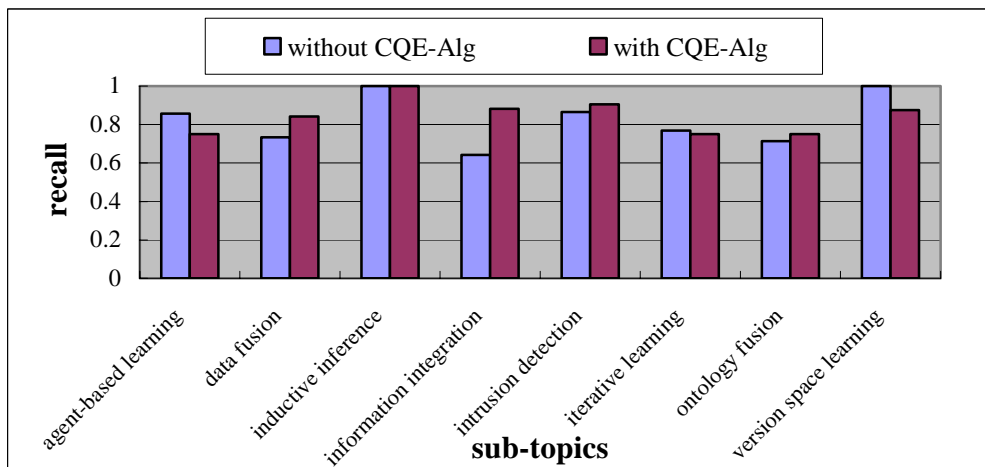


Figure 6.11: The F-measure with/withour CQE-Alg

Moreover, a questionnaire is used to evaluate the performance of our system for these participants. The questionnaire includes the following two questions: **1) Accuracy degree:** “Are these learning materials desired?”, **2) Relevance degree:** “Are the obtained learning materials with different topics related to your query?”. As shown in Figure 6.11, we can conclude that the LCMS scheme is workable and beneficial for users according to the results of questionnaire.

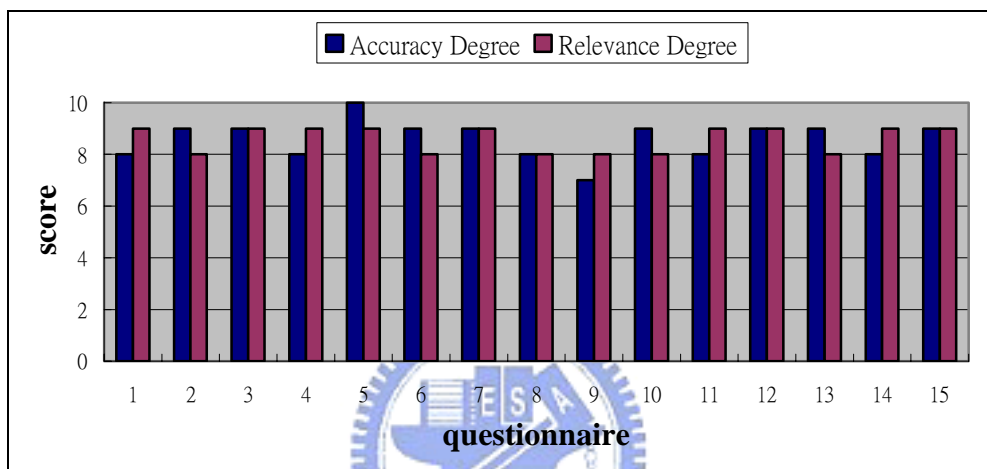


Figure 6.12: The Results of Accuracy and Relevance in Questionnaire (10 is the highest)

Chapter 7 Conclusion and Future Work

In this thesis, we propose a *Level-wise Content Management Scheme*, called *LCMS*, which includes two phases: *Constructing phase* and *Searching phase*. For representing each teaching materials, a tree-like structure, called *Content Tree (CT)*, is first transformed from the content structure of *SCORM Content Package* in the *Constructing phase*. And then, an information enhancing module, which includes the *Keyword/phrase Extraction Algorithm (KE-Alg)* and the *Feature Aggregation Algorithm (FA-Alg)*, is proposed to assist user in enhancing the meta-information of content trees. According to the *CTs*, the *Level-wise Content Clustering Algorithm (ILCC-Alg)* is then proposed to create a multistage graph with relationships among learning objects (LOs), called *Level-wise Content Clustering Graph (LCCG)*. Moreover, for incrementally updating the learning contents in *LOR*. The *Searching Phase* includes the *LCCG Content Searching Algorithm (LCCG-CSAlg)* to traverse the *LCCG* for retrieving desired learning content with both general and specific learning objects according to the query of users over the wire/wireless environment. Besides, the *Content-based Query Expansion Algorithm (CQE-Alg)* is proposed to assist users in refining their queries to retrieve more specific learning objects from a learning object repository.

For evaluating the performance, a web-based *Learning Object Management System*, called *LOMS*, has been implemented and several experiments also have been done. The experimental results show that our *LCMS* is efficient and workable to manage the *SCORM* compliant learning objects.

In the near future, more real-world experiments with learning materials in several domains will be implemented to analyze the performance and check if the proposed management scheme can meet the need of different domains. Besides, we will enhance the scheme of LCMS with scalability and flexibility for providing the web service based upon real SCORM learning materials. Furthermore, we are trying to construct a more sophisticated concept relation graph, even an ontology, to describe the whole learning materials in an e-learning system and provide the navigation guideline of a SCORM compliant learning object repository.



References

Websites

- [AICC] Aviation Industry CBT Committee (AICC) 2004, AICC - Aviation Industry CBT Committee. <http://www.aicc.org>
- [ARIADNE] Alliance for Remote Instructional and Authoring and Distribution Networks for Europe (ARIADNE) 2004, ARIADNE: Foundation for The European Knowledge Pool. <http://www.ariadne-eu.org>
- [CETIS] CETIS 2004, 'ADL to make a 'repository SCORM'', The Centre for Educational Technology Interoperability Standards. <http://www.cetis.ac.uk/content2/20040219153041>
- [IMS] Instructional Management System (IMS) 2004, IMS Global Learning Consortium. <http://www.imsproject.org/>
- [Jonse04] Jones, E.R., 2004, Dr. Ed's SCORM Course, <http://www.scormcourse.jcasolutions.com/index.php>
- [LSAL] LSAL 2003, 'CORDRA (Content Object Repository Discovery and Resolution/repository Architecture)', Learning Systems Architecture Laboratory: Carnegie Mellon LSAL. <http://www.lsal.cmu.edu/lsal/expertise/projects/cordra>
- [LTSC] IEEE Learning Technology Standards Committee (LTSC) 2004, IEEE LTSC | WG12. <http://ltsc.ieee.org/wg12/>
- [SCORM] Sharable Content Object Reference Model (SCORM) 2004, Advanced Distributed Learning. <http://www.adlnet.org/>
- [W3C] W3C (updated 9 Jun. 2004), World Wide Web Consortium. <http://www.w3.org>
- [WN] WordNet, <http://wordnet.princeton.edu/>
- [XML] eXtensible Markup Language (XML) (updated 26 Mar. 2004), Extensible Markup Language (XML). <http://www.w3c.org/xml/>

Articles

- [BL85] C. Buckley, A. F. Lewit, "*Optimizations of Inverted Vector Searches*", SIGIR '85, 1985, pp:97-110.

- [CK+92] D. R. Cutting, D. R. Karger, J. O. Pedersen, J. W. Tukey, “*Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections*”, Proceedings of the Fifteenth International Conference on Research and Development in Information Retrieval, 1992, pp. 318-329.
- [KC02] S.K. Ko and Y.C. Choy, “*A Structured Documents Retrieval Method supporting Attribute-based Structure Information*”, Proceedings of the 2002 ACM symposium on Applied computing, 2002, pp. 668-674.
- [KK01] S.W. Khor and M.S. Khan, “*Automatic Query Expansions for aiding Web Document Retrieval*”, Proceedings of the fourth Western Australian Workshop on Information Systems Research, 2001
- [KK02] R. Kondadadi, R. Kozma, “*A Modified Fuzzy ART for Soft Document Clustering*”, Proceedings of the 2002 International Joint Conference on Neural Networks, Vol. 3, 2002, pp:2545-2549.
- [KK04] M.S. Khan, S.W. Khor, “*Web Document Clustering using a Hybrid Neural Network*”, Journal of Applied Soft Computing, Vol. 4, Issue 4, Sept. 2004.
- [LA99] B. Larsen and C. Aone, “*Fast and Effective Text Mining Using Linear-Time Document Clustering*”, Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 16-22.
- [LM+00] H.V. Leong, D. MeLeod, A. Si, and S.M.T. Yau, “*On Supporting Weakly-Connected Browsing in a Mobile Web Environment*”, Proceedings of ICDCS2000, 2000, pp: 538-546.
- [MR04] F. Meziane, Y. Rezgui, “*A Document Management Methodology based on Similarity Contents*”, Journal of Information Science, Vol. 158, Jan. 2004.
- [RW86] V.V. Raghavan, and S.K.M. Wong, “*A Critical Analysis of Vector Space Model in Information Retrieval*”, Journal of the American Society for Information Science, 37, 1986, pp. 279-287.
- [SA04] S. Sakurai, A. Suyama, “*Rule Discovery from Textual Data based on Key Phrase Patterns*”, Proceedings of the 2004 ACM Symposium on Applied Computing, Mar. 2004.
- [SS+03] M. Song, I.Y. Song, X.H. Hu, “*KPSpotter: A Flexible Information Gain-based Keyphrase Extraction System*”, Proceedings of the fifth ACM International Workshop on Web Information and Data Management, Nov. 2003
- [VV+04] I. Varlamis, M. Vazirgiannis, M. Halkidi, Member, IEEE Computer Society,

Benjamin Nguyen, “*THESYS, a closer view on web content management enhanced with link semantics*”, IEEE Transaction on Knowledge and Data Engineering, Jun. 2004.

[WC+04] E.Y.C. Wong, A.T.S. Chan, and H.V. Leong, “*Efficient Management of XML Con-tents over Wireless Environment by Xstream*”, Proceedings of the 2004 ACM sym-posium on Applied computing, 2004, pp. 1122-1127.

[WL+03] C.Y. Wang, Y.C. Lei, P.C. Cheng, S.S. Tseng, “*A Level-wise Clustering Algorithm on Structured Documents*”, 2003.

[YL+99] S.M.T. Yau, H.V. Leong, D. MeLeod, and A. Si, “*On Multi-Resolution Document Transmission in A Mobile Web*”, the ACM SIGMOD record, Vol. 28, Issue 3, Sep. 1999, pp:37-42.

