Categorizing Questions According To A Navigation List For A Web-Based Self-Teaching System : AEGIS

Tsunenori Mine, Akira Suganuma, and Takayoshi Shoudai
Faculty of Information Science and Electrical Engineering, Kyushu University
6-1 Kasuga-kouen, Kasuga, Fukuoka 816-8580, Japan
{mine,suga}@is.kyushu-u.ac.jp, shoudai@i.kyushu-u.ac.jp

Abstract

With increasing access to the Internet and the wealth of material online, a Web-based self-teaching system has considerable educational value. Accordingly, we developed AEGIS (Automatic Exercise Generator based on the Intelligence of Students), which automatically generates questions whose difficulty level fits the achievement level of a student. However, it was implicitly assumed that all the questions were already categorized according to their subjects. In practice, this is not the case, but it is unreasonable (because of time and cost) to expect teachers to categorize each question into a suitable subject domain. Therefore, we need a method for categorizing questions automatically according to specified teaching concepts.

This paper presents an automatic question categorization mechanism according to both a list of teaching concepts, called a Navigation List (NaviList for short), and the meaning of questions. We define an XML tag called a CONCEPT tag, which indicates a concept in a question, and an ontology, which is a hierarchical cluster of concepts. The method uses the tags and the ontology to categorize questions, based on the similarity between each category in a NaviList pre-composed by a teacher and an ontological concept specified by a CONCEPT tag in a question.

1 Introduction

The World Wide Web is a widely accessible public source of information, which encourages exploration and self-help in the pursuit of knowledge, and provides a wide variety of multi-media contents and added-value services. In an educational domain, many teachers hold lectures using Web contents as a teaching material and even develop new lecture methods based on Web technologies. A Web-based self-teaching system, which enables a student to study something at his/her own pace, anytime and anywhere, has considerable educational benefits, not only for students who take part in a lecture, but also for other people who do not directly take it.

We have developed several kinds of Web-based self-teaching systems[1, 2, 3]. Through our experiences of teaching in classes and developing such systems, we recognized the necessity of methods both for evaluating students' achievement levels and for generating questions suitable for the students automatically.

We have consequently designed and developed an automatic evaluator of a student's achievement level called AEGIS(Automatic Exercise Generator based on the Intelligence of Students)[4]. AEGIS generates questions from tagged documents, presents them to students and marks their answers automatically. AEGIS however expects that the source documents for generating questions have already been categorized into a particular domain according to their subjects. So, AEGIS presents questions to students according to their achievement level, but not to the domain where the questions belong.

From the point of view of an educational policy, these questions should be tailored according to a concept or an aim of a lecture because each teacher teaches his/her students based on these concepts and aims. The questions should also be given to students in a suitable order for his/her lecture. For example, in the case of teaching a programming language, we may want to present the questions classified under the category of 'programming language specification' when focusing on the grammar of the language. On the other hand, we may want to give the questions classified under 'programming patterns' when focusing on program design. In both cases, the same questions can be given although the order of presentation may be different. However, it requires a long time and high cost for a teacher to categorize questions one by one into their suitable domains. It is therefore important to develop a method for categorizing questions automatically according to the teaching concept of a lecture.

This paper presents an automatic question categorization mechanism according to both a list of teacher's teaching concepts, called Navigation List (**NaviList** for short), and the meaning of questions. We define both an XML tag called a CONCEPT tag, which indicates a concept in a question, and a method, which categorizes questions based on the similarity between each category in a NaviList precomposed by a teacher and one or more keywords surrounded by the CONCEPT tags in a question. Such keywords give a context-sensitive representation of the conceptual meaning of a question. The similarity calculation is performed with an ontology, which is a hierarchical cluster of questions' concepts. Each question is categorized into its most similar category in a NaviList, provided that its similarity is over a predetermined threshold.

After a student selects a category, AEGIS can then choose the most suitable questions from the category for him/her, according to both his/her achievement level and the difficulty level of the question, just as before.

2 A Method for Mapping Questions with a Category in a NaviList

Our method maps a question onto a category item in a NaviList if the similarity between them is over a threshold, which is determined empirically. The similarity is calculated based on the concepts of a question and those of the category. The easiest way to calculate the similarity between them would be matching concept labels attached to a question with those to the category. Unfortunately, this method sometimes fails because it requires consistency in choosing and attaching concept labels to questions and categories. This is difficult to realize, especially when different people are making the associations. In any case, concept labels are not always the same although they may be connected with some relationship such as a-kind-of, synonym, a-part-of, and so forth. Concepts of a category in a NaviList usually subsume those of questions. Therefore the similarity calculation between concepts of both a NaviList category and a question requires an ontology related to them (Fig.1).

Our method defines an ontology as a hierarchical cluster that consists of common concepts of its sub clusters, recursively. The cluster is constructed by a simple linkage clustering method[7, 8]. The cluster's primitive cluster, called the base cluster, consists of both a main concept and its related concepts of questions and NaviList categories. The main concept can be regarded as a class and its related concepts as its instances. Although a created cluster has at most a few hierarchies, it is enough for our purpose because the purpose is not to calculate the exact or precise similarity between them, but just to find a connection between a question and a NaviList category.

When we construct the ontology, first, we apply a morphological analyzer[6] to teaching documents. Then, concept labels of a question are attached to the words of specific parts of speech such as a common noun, proper noun or undefined, for example an operator or a reserved word of a programming language.

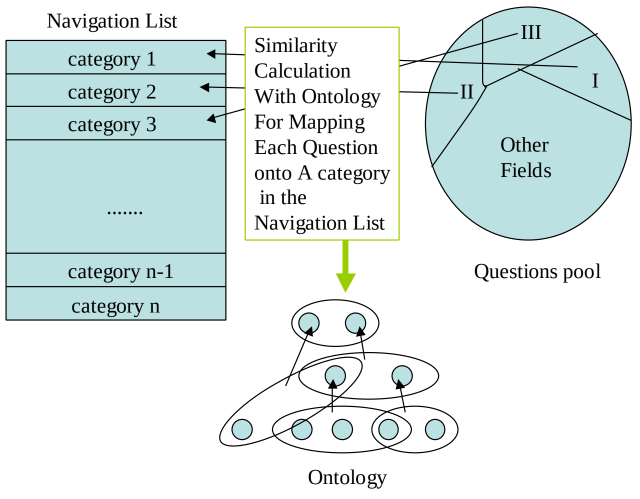


Figure 1. Mapping a concept of a question onto that of a category in a NaviList

The concept labels of a NaviList category are mostly chosen by NaviList creators (e.g. teachers) from the representative concepts in superior level sub-clusters. The representative concepts are selected by calculating the weight of their frequency in a question text, which is referred to as TF(text frequency), and inverse question text frequency, referred to as IDF(inverse document frequency).

A procedure to create an ontology is as follows:

1. Transform teaching documents(e.g. MS Power Point slides) into Web(XML or HTML) format documents.
2. Eliminate XML and HTML tags from the Web format documents to give plain text documents.
3. Apply a morphological analyzer to the plain text documents.
4. Reconnect keywords that have been segmented too much, with hand-written rules.
5. Extract salient keywords according to their TF.IDF weight.
6. Cluster the keywords by a simple linkage clustering method[7, 8].

Table 1 depicts a part of ontology automatically generated from lecture slides about the introduction to C programming.

Table 1. A part of an ontology created automatically from Power Point lecture slides about the introduction to C programming

A part of Ontology related to Text File	
Main Concept	Related Concepts
text file	character, line, unit, fprintf, random access, sequential, fgetc
fgetc	character, processing, text file, lecture, fgets, input, line, sscanf, unit, output, fprintf
fprintf	character, processing, text file, fgets, line, unit, fgetc
line	character, processing, text file, fgets, fprintf, fgetc
unit	text file, fprintf, fgetc
character	raw, text file, string, line, float, numerical value, concatenation, output, order, fprintf, comparison, letter, for, creating date, author, error level, fgetc, abcde, World

3 A Concept Tag and a Category of a NaviList

This section, first, gives explanations about a new tag, called a *CONCEPT* tag ⟨CONCEPT⟩. Categories in a NaviList are constructed with keywords in a teaching document. Such keywords are surrounded by concept tags and represent the conceptual meaning of the document. As mentioned in Section 2, we map these keywords onto categories in a NaviList.

The tag ⟨CONCEPT⟩ is defined by a DTD of XML (Fig.2). Tagging a teaching document is partially automated. First a morphological analyzer[6] is applied to the teaching document, and then tags are automatically attached to the document in a simple way based on past attachment processes. After that, using an authoring tool for XML, the system asks the user to confirm whether or not newly tagged areas are correct. The authoring tool can automatically attach concept tags to the areas with the same keyword to which the concept tag has been attached in the past.

We specify a question in teaching documents by tags which were formally defined by a DTD of XML[4]. These tags are also described in Fig.2, which include a tag ⟨QUESTION⟩ for specifying an area of a question, a tag ⟨DEL⟩ for specifying a solution, and a tag ⟨LABEL⟩ for a hint of some questions. The tag ⟨DEL⟩ has an attribute LEVEL for specifying a pair of initial minimum and maximum difficulties of the question's solution. For more detailed definitions and explanations, please see [4].

```
⟨!DOCTYPE  EXERCISE  [
⟨!ELEMENT  QUESTION  (#PCDATA | DEL | CONCEPT
                                         | LABEL )*⟩
⟨!ELEMENT  DEL       (#PCDATA)⟩
⟨!ELEMENT  CONCEPT   (#PCDATA)⟩
⟨!ELEMENT  LABEL     (#PCDATA)⟩
⟨!ATTLIST  QUESTION  SUBJECT  CDATA #IMPLIED⟩
⟨!ATTLIST  DEL       CAND     CDATA #IMPLIED
                     LEVEL    NMTOKENS #REQUIRED
                     GROUP    NMTOKEN #IMPLIED
                     REF      IDREF #IMPLIED⟩
⟨!ATTLIST  LABEL     NAME     ID #IMPLIED⟩
] ⟩
```

Figure 2. DTD of the tags defined for AEGIS

An example of a teaching document with ⟨CONCEPT⟩ tags, is described in Fig.3. The example has some words which are surrounded by ⟨CONCEPT⟩ tags. These ⟨CONCEPT⟩ tags specify important concepts for teaching the subject. The concepts are used to construct categories of a NaviList and an ontology, and also to map a question onto a category of the NaviList. The question specified by a tag ⟨QUESTION⟩ has to contain several keywords surrounded by a tag ⟨CONCEPT⟩, otherwise it is meaningless. Each question is classified into a category in a NaviList. Each category represents one of the important subjects in a teaching document.

Since we believe that the difficulty of categories' concepts in a NaviList should depend on the order of the categories in the NaviList, a question will be categorized into such a category any of whose concepts' difficulty is more than or equal to that of any concepts of a question.

4 The Construction and Functions of AEGIS

Once categorizing questions has been finished, AEGIS chooses questions from one of the categories. AEGIS is implemented in Java. It runs under JDK (ver. 1.3) and Tomcat (ver. 3.2). The system is based on the MVC (Model, View, Controller) model. The MVC model is an application architecture, which was firstly introduced at a Smalltalk programming environment. This model divides an application into three parts which are called Model (or logic), View (or presentation), and Controller (or communication, control). These three parts can be developed separately because they are functionally independent from each other. This model supports modular system development and ease of maintenance.

Our system AEGIS consists of two subsystems: the *Manager System* and the *User System*. The Manager System is used for teachers to manage questions. This system

In the previous section, we learned a ⟨CONCEPT⟩ program ⟨/CONCEPT⟩ for adding two integers and showing the answer on the display. In the similar way, for all basic ⟨CONCEPT⟩ arithmetic operations ⟨/CONCEPT⟩ including ⟨CONCEPT⟩ addition ⟨/CONCEPT⟩, ⟨CONCEPT⟩ subtraction ⟨/CONCEPT⟩, ⟨CONCEPT⟩ multiplication ⟨/CONCEPT⟩ , and ⟨CONCEPT⟩ division ⟨/CONCEPT⟩, we can make a Pascal program in the following way.
⟨QUESTIONSUBJECT = ”arithmeticoperations”⟩
This program computes the multiplication and division for two input integers and shows the answer.

```
program enzan;
var x,y:integer;
    seki,shou:integer;
begin
    ⟨CONCEPT⟩write ⟨/CONCEPT⟩('Input two integers : ');
    ⟨CONCEPT⟩readln ⟨/CONCEPT⟩(x,y);
    seki:=⟨DEL CAND=”x,xy,x×y,x mul y” LEVEL=”1 5”⟩x*y⟨/DEL⟩;
    shou:=⟨DEL CAND=”x/y,x÷y,xdivy,x mod y” LEVEL=”1 5”⟩x div y⟨/DEL⟩;
    ⟨CONCEPT⟩writeln ⟨/CONCEPT⟩('Seki:',seki);
    ⟨CONCEPT⟩writeln ⟨/CONCEPT⟩('Shou:',shou)
end.
```

⟨/QUESTION⟩
The 7th statement multiplies x by y, and the 8th statement divides x by y. We note that the answer of “div” is an integer.

Figure 3. Example of teaching documents with the tags

also supports teachers in creating a NaviList with an ontology generated from concepts of their teaching documents.

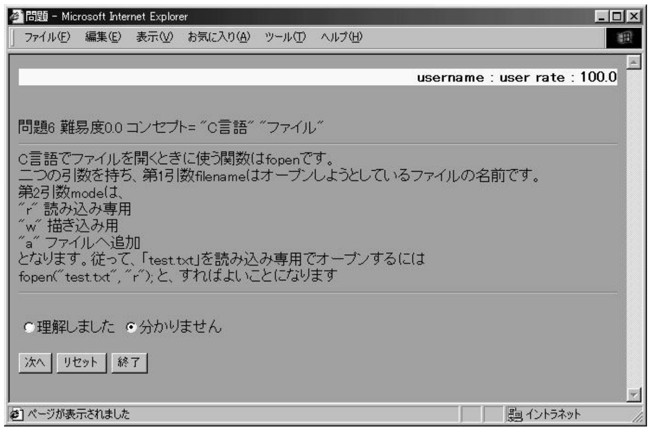
The other system, the User System, helps students to learn a subject. It has two function modes: *Learning Mode* and *Test Mode*. In Learning Mode, AEGIS shows part of a teaching text in its normal form (e.g. Power Point slide, Web pages, etc.), which is utilized for generating questions. It is useful for students to learn its contents and to confirm their understanding of those contents. Furthermore, from the point of view of teaching materials creator (e.g. a teacher), Learning Mode can be used for checking how many students understand the materials and then the appropriateness of the slides to describe questions suitable for the students. In Test Mode, AEGIS generates questions to let students try to answer. The questions AEGIS generates have various levels according to each student's achievement level. After submitting their answers, AEGIS marks those answers and returns the marked results to the student. The types of a question AEGIS generates are a multiple-choice question, fill-the-gap question, and error-correcting question. The difficulty of a question is strongly related to the type of the question. The marked results of a series of questions are stored into each student's profile, and it is used to evaluate the new achievement level of the student. At the same time, AEGIS evaluates the new difficulty level of questions. These two evaluations are dynamically computed at some predetermined intervals. We have evaluated the effectiveness of the updating mechanism of both levels by computer simulations [5, 9], and showed that the estimated difficulty level of a question gradually approaches the inherent one of the question and the estimated achievement level of a student does also his/her inherent one, provided that we assumed there are such inherent levels.

Fig.4 and Fig.5 depict a snapshot of Learning Mode and that of Test Mode, respectively, both of which generate a fill-the-gap question.

5 Concluding Remarks

AEGIS[4] is a Web-based self-teaching system, which searches teaching documents described in XML and generates three types of questions automatically. A question to be generated is based on both its difficulty level and the achievement level of a student who tries the question. In order to generate the questions automatically, the questions in a teaching document are specified by an XML tag, called a QUESTION tag. The user profiles of AEGIS keep his/her marked results and are utilized to make AEGIS generate a next question. This generating a question is managed by one of AEGIS's subsystems, called the *User System*. The effectiveness of this User System was reported by computer simulations[5, 9].

However, it had been implicitly assumed that all the questions were already categorized according to their subjects. This paper presented a new subsystem of AEGIS, called the *Manager System*, and a method for categorizing questions which AEGIS generates automatically, using a new XML tag called a CONCEPT tag. The Manager System categorizes these questions into the categories of a NaviList created by teachers, and assists the students in mastering their subject by navigating them through a lecture


問題 - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
次へ リセット 終了
ページが表示されました
イントラネット

username : user rate : 100.0

問題6 難易度0.0 コンセプト= ″C言語″ ″ファイル″

C言語でファイルを開くときに使う関数はfopenです。
二つの引数を持ち、第1引数filenameはオープンしようとしているファイルの名前です。
第2引数modeは、
″r″ 読み込み専用
″w″ 描き込み用
″a″ ファイルへ追加
となります。従って、「test.txt」を読み込み専用でオープンするには
fopen(″test.txt″, ″r″); と、すればよいことになります

理解しました [✓ 分かりません]

Figure 4. Learning Mode

course's contents, with continual assessment and feedback by asking and marking appropriate questions.

Teaching materials for AEGIS are intended for text documents written in XML or HTML. Recently many teachers use a presentation software for their lectures, for example Microsoft Power Point. The files of Power Point can be transformed into Web documents without any difficulties. They are good materials to generate questions. Although AEGIS can deal with the documents to create an ontology and to map a question onto a category in a NaviList, it does not surround appropriate keywords by ⟨CONCEPT⟩ tags yet automatically. We need to continuously investigate a method to solve the problem. We also have to evaluate the created ontology and our presented method to map a question onto a category in a NaviList.

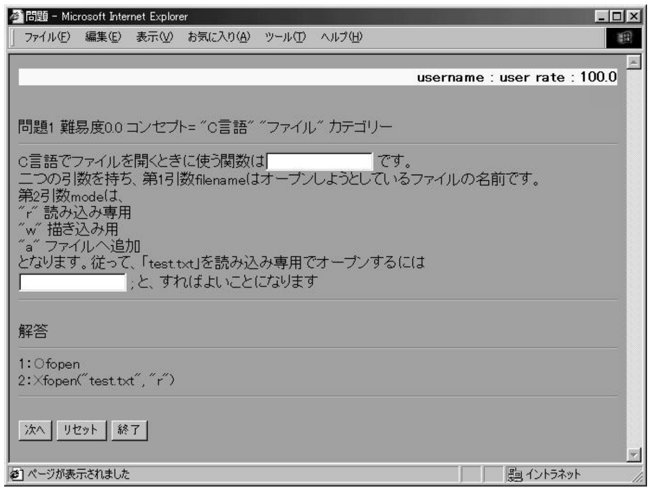
AEGIS is implemented as a Java application. The modular design supported by the MVC model has made it straightforward to extend AEGIS with the new subsystem reported here. The same modularity should support extension to AEGIS in its implementation as a multi-agent system using the KODAMA framework[10].

Acknowledgment

This research was partly supported by a Grant-in-Aid for Scientific Research on Priority Areas (2) from the Ministry of Education, Culture, Sports, Science and Technology, No. 14022239, 2002, and by a Grant for Special Academic Research P&P from Kyushu University, type C, 2001-2002.

References

[1] H. Sato, T. Mine, T. Shoudai, H. Arimura, and S. Hirokawa. On web visualizing how programs run for teaching 2300 students. In *ICCE(International Conference on Computers in Education)97 in Malaysia*, pages 952–954, 1997.


問題 - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
次へ リセット 終了
ページが表示されました
イントラネット

username : user rate : 100.0

問題1 難易度0.0 コンセプト= ″C言語″ ″ファイル″ カテゴリー

C言語でファイルを開くときに使う関数は[　　　　] です。
二つの引数を持ち、第1引数filenameはオープンしようとしているファイルの名前です。
第2引数modeは、
″r″ 読み込み専用
″w″ 描き込み用
″a″ ファイルへ追加
となります。従って、「test.txt」を読み込み専用でオープンするには
[　　　　] ; と、すればよいことになります

解答

1:○fopen
2:×fopen(″test.txt″, ″r″)

Figure 5. Test Mode

[2] T. Mine, D. Nagano, K. Baba, T. Shoudai, and S. Hirokawa. On-web visualizing a mechanism of a single chip computer for computer literacy courses. In *ICCE(International Conference on Computers in Education)98*, volume 2, pages 496–499, 1998.

[3] A. Suganuma, R. Fujimoto, and Y. Tsutsumi. An WWW-based supporting system realizing cooperative environment for classroom teaching. In *World Conference on the WWW and Internet*, pages 830–831, 2000.

[4] T. Mine, A. Suganuma, and T. Shoudai. The design and implementation of automatic exercise generator with tagged documents based on the intelligence of students:AEGIS. In *the ICCE/ICCAI 2000*, pages 651–658, 2000.

[5] A. Suganuma, T. Mine, and T. Shoudai. Automatic generating appropriate exercises based on dynamic evaluating both students' and questions' levels. In *ED-MEDIA 2002–World Conference on Educational Multimedia, Hypermedia & Telecommunications*, pages 1898–1903, 2002.

[6] Y. Matsumoto, A. Kitauchi, T. Yamashita, Y. Hirano, H. Matsuda, K. Takaoka, and M. Asahara. Morphological analysis system CHASEN version 2.2.8 manual. In *Technical report, Nara Institute of Science and Technology*, 2001.

[7] T. Mine, S. Lu, and M. Amamiya. Discovering relationships between topics of conferences by filtering, extracting and clustering. In *the 3rd International Workshop on Natural Language and Information Systems(NLIS2002)*, to appear, 2002.

[8] O. Zamir and O. Etzioni. Web document clustering:a feasibility demonstration. In *the 21th Intl. ACM SIGIR Conference*, pages 46–54, 1998.

[9] A. Suganuma, T. Mine, and T. Shoudai. AEGIS : Automatic exercise generator based on the intelligence of students with tagged documents. In *5th Joint Conference on Knowledge-Based Software Engineering(JCKBSE2002)*, to appear, 2002.

[10] G. Zhong, S. Amamiya, K. Takahashi, T. Mine, and M. Amamiya. The design and application of KODAMA system. In *IEICE Transactions on Information and Systems*, E85-D(4), pages 637–646, 2002.