

Structural Discovery of E-lessons

Azita Bahrami

*Department of Information Technology, Armstrong Atlantic State University,
11935 Abercorn Street Savannah, GA 31419, USA*

bahramaz@mail.armstrong.edu

Abstract

An e-lesson is comprised of a “body” and a “view”. The body is the actual content of the e-lesson and the assumption is that it is an html document. The view is the metadata section of this html document, and it is made up of a set of words and phrases that are either explicit or implicit and are referred to as “terms”. This paper proposes a methodology that is capable of discovering the physical and logical structure of a given e-lesson through the use of that e-lesson’s view and resulting in a major change concerning the granularity of data retrieval by the e-lesson’s retrieval system. The major change is that the proposed method allows for solely the requested specific segment of an e-lesson to be returned instead of the entire lesson. The methodology is based on a warehouse-oriented framework that has been suggested by the author in the past.

Keywords: e-lesson’s physical and logical structures, segmentation, explicit and implicit metadata, and e-lesson’s map.

1. Introduction

Several frameworks have been presented for finding the e-lessons of interest on the internet. The framework presented by Siqueira et al [1] uses the warehousing technology to find the e-lessons of interest. That framework assumes that the entire e-lesson is a single entity not providing a mechanism for isolating only the segments in which the e-lesson searcher is interested. For example, an e-lesson, λ , for subject S may be delivered by the system. The system assumes that the entire λ is about S even when S is only a subtopic within that lesson. In other words, though the general location of the subject is known, its exact location is not, and thus cannot be isolated and returned to the searcher.

To remedy this problem one needs to discover the *logical* structure of an e-lesson in terms of its components (i.e. subject/topics and sub-topics). Such a discovery enables a retrieval system to segment the

e-lesson and retrieve only those segments of the e-lesson that are directly related to the user’s needs.

Even though text segmentation has been implemented in all three types of natural language documents, image documents, and Web-based documents, this paper is interested only in the the last type. Presently, the majority of the work done on such documents tries to extract features from unstructured HTML documents [2, 3, 4, 5] and very few that try to discover the logical structure of an e-lesson. For instance, the work of Lilijohn et. el [6] is geared toward discovery of the logical structures within the body of an e-lesson. They use a Semantic Web based methodology to find the content for each component of the e-lesson. This method works based on a premise that the e-lesson is presented as a well-defined XML document, and thus every topic and sub-topic within the e-lesson can be located through the use of a rich ontology. The framework, however, does not address a solution for e-lessons that are not presented as a well-defined XML document.

This paper proposes a methodology that is able to find (a) the actual content of each component of a given e-lesson and (b) the logical structure of the entire e-lesson. The methodology is based on a warehouse-oriented framework that has previously been suggested by the author [7].

The rest of the paper is organized as follow: The foundation for the current research is explained in section two. The details of the proposed methodology are presented in section three, and the summary and future research are included in section four.

2 Foundation

The framework is basically an E-Lesson Retrieval System (ELeRS) that is able to retrieve existing e-lessons on public or private (with permission) networks for the purposes of (a) creating and maintaining a warehouse of e-lessons, (b) helping an e-lesson designer to use, reuse, and refer to the existing e-lessons, and (c) using OLAP technology to

manipulate the warehouse along the three dimensions of Subject, Audience, and Field of study (S, A, and F).

The ELeRS uses the descriptive words and phrases (in this paper, we refer to them as “terms”) in the metadata section (View) of an e-lesson as indices for finding the e-lessons of interest. Since e-lesson designers are not yet bound by a standard, they are free to apply varied descriptive words with the same meaning. As a result, there is a need for having an ontology in the ELeRS to recognize and reconcile those terms that are syntactically different but semantically the same. In addition, the ontology is used to establish the conceptual hierarchy of the terms listed in the e-lesson’s view. Furthermore, such a conceptual hierarchy of terms is integrated with a bigger conceptual hierarchy of the terms that are within the views of all other relevant e-lessons kept in the warehouse.

When a designer of an e-lesson issues a query to find all the e-lessons about subject S for audience A in the field F, the Controller component of the ELeRS receives the query, consults the “term ontology” and determines whether the subject/ topic S within the conceptual hierarchy is composed of sub-topics S1, S2, and S3, for example. Also, the term ontology determines whether the equivalent terms for S, for example, are S’ and S”, for S1 is S’1, for S2 are S’2 and S”2. As a result, the Controller modifies the designer’s query to include the subjects S’, S”, S1, S’1, S2, S’2, S”2, and S3 in addition to the subject S. The new query is implemented by the OLAP that finds all the e-lessons of interest in the warehouse and returns them to the user through Controller along with their views. In addition, the mapping between multiple copies of the terms and the e-lesson’s bodies is also created and delivered by the ELeRS.

Ordinarily, the data returned by the ELeRS represent the granulation at a broad level, the e-lesson itself, and not at a narrow level, a specific segment of the e-lesson. Finding component(s) of interest within a returned e-lesson is not a task of the ELeRS but is delegated to the user to be done manually. This is an arduous task for the user diminishing his/her enthusiasm for using the system. This paper intends to alleviate such hardship.

3 Methodology

An e-lesson is comprised of a “body” and a “view”. The body is the actual content of the e-lesson, which is assumed to be an html document. The view is the metadata section of the html document and is composed of a set of words and phrases. Both are examined below.

View. The *elements* in a view are referred to as “terms”, while their *associated concepts* are referred to as “conceptual hierarchy”. If there is a conceptual hierarchy among the terms of a given view, it is identified by the “term ontology”. (This ontology was introduced in [7]). A conceptual hierarchy is either an ordered graph (a tree) or a partially ordered graph (a lattice). When one traverses the conceptual hierarchy upward, the chance of physically finding the terms of interest in the body of the e-lesson goes down. That is because some terms are *implicit* rather than *explicit*.

The explicit terms are either *casual* or *core*. The differences between the two are as follow.

- a. In the body of the e-lesson, the appearance of a casual explicit term is less frequent than the appearance of a core explicit term.
- b. The casual explicit terms appear more in the introduction and the summary sections of the e-lesson, but the core casual explicit terms appear more in the body of the e-lesson and *between* the introduction and the summary.
- c. The casual explicit terms are located in a higher level than the core explicit terms within the conceptual hierarchy of the e-lesson.

Using the above mentioned criteria collectively, one can easily separate the core from the casual explicit terms.

Body. The e-lesson’s body has two different structures, *physical* and *logical*. The physical structure refers to the way in which the e-lesson’s text, such as paragraphs and headings, are organized. The logical structure refers to the relationships among the components of the e-lesson. Both structures are further explored in the following subsections.

3.1 Physical Structure

An e-lesson’s body is comprised of m paragraphs and n headings where (m and n ≥ 1). A paragraph is either formal or informal. A formal paragraph is a paragraph that the original designer/programmer has encased between the pairs of <p> and </p> tags. An informal paragraph, however, does not have paragraph tags. As a result, the following grammar is proposed for identification of the informal paragraphs. Note: a pair of curly braces is utilized to denote the non-terminal items in the grammar.

```
{Informal paragraph} → {Begin} {Middle} {End}
{Begin} → <br> [<br>]+ | </hi> | </p>
{Middle} → sentence [Sentence]+
{End} → <br> [<br>]+ | <p> | <br> <hi>
```

Each paragraph, whether formal or informal, is to be identified by an identification number (PID), a starting position (PStart), and an ending position

(PEnd). Paragraph identification number is constructed by the letter “P” followed by an order number (e. g. P_1, \dots, P_n). If the entire HTML source of an e-lesson is considered to be a long string of characters, then the positions of the first and the last characters of the paragraph are the values for the PStart and PEnd, respectively.

A heading is either a word or a phrase, formal or informal. A formal heading is surrounded by a pair of $\langle h_i \rangle \langle /h_i \rangle$ (where $1 \leq i \leq 8$). An informal heading is not surrounded by the heading tags but for which the following grammar is provided

```
{Informal heading} → {Part1} {Bold_title} {Part2} |
{Part1} {Regular_title} {Part3}
{Part1} → [ <br> ] + | <p>
{Bold_title} → <b> {Regular_title} </b>
{Regular_title} → a_word | a_phrase |
a_numbered_word | a_numbered_Phrase
{Part2} → {Part3} | , | : | . | - | --
{Part3} → [ <br> ] * | <br>
```

Each heading is given an identification number (HID), starting position (HStart) and an ending position (HEnd) in the document. The identification number is denoted by the letter “h” followed by the heading level that is determined by the heading tags (for formal headings: $HID = h_1$ through h_8).

Headings of any level (1-8) can be either numbered or unnumbered. A formal numbered heading is the one for which the original designer has provided both a pair of heading tags and a title number. Example: $\langle h_1 \rangle$ 1. Introduction $\langle /h_1 \rangle$. A formal unnumbered heading has only the heading tags ($\langle h_1 \rangle$ Introduction $\langle /h_1 \rangle$).

An informal heading may also be numbered (example: 1. Introduction) or unnumbered (example: Introduction). The proposed methodology requires the assignment of heading tags (i.e. levels) to the informal headings. This process is investigated in the next section.

3.1.1 Numbering of the Headings

We start with defining a notation that would later be used in this section. Let us have three headings numbered 3.1, 3.2, and 3.1.1. The notations $3.1 > 3.1.1$ and $3.1 = 3.2$ are used to express the fact that the heading with number 3.1 is in higher level than 3.1.2, and it is at the same level as the heading numbered 3.2. The same notations are used to express the levels among the formal tagged headings (e.g. $h_1 > h_2$).

The e-lesson’s body has a “homogenous” heading format if it satisfies one of the *homogeneity* conditions. These conditions are:

- All the headings have heading tags and numbered.
- All the headings have heading tags and unnumbered.
- All the headings lack heading tags but are numbered

- None of the headings are either numbered or have heading tags
- No headings exist.

Let us assume that we have four binary variables of *Homogeneity* (H), *tagged heading* (TH), *numbered heading* (NH), and *untagged-and-unnumbered heading* (UH). These variables have sixteen possible combinations displayed in Table 1. For each combination, the numbering of the headings is done by taking an action that is shown in the “Action” column of the Table 1. The details of each action are provided shortly.

Table 1. Possible combinations for types of headings.

H	TH	NH	UH	Action
0	0	0	0	Practically Impossible
0	0	0	1	Practically Impossible
0	0	1	0	Practically Impossible
0	0	1	1	A(numbered heading)
0	1	0	0	Practically Impossible
0	1	0	1	A(tagged heading):
0	1	1	0	B()
0	1	1	1	C()
1	0	0	0	D()
1	0	0	1	E()
1	0	1	0	Headings are re-numbered through outlining techniques.
1	0	1	1	Practically Impossible
1	1	0	0	Heading numbers are dictated by the heading tags.
1	1	0	1	Practically Impossible
1	1	1	0	Heading numbers are dictated by the heading tags.
1	1	1	1	Practically Impossible

When homogeneity value is “0” (i.e. “No”), there must be at least two types of heading format in the e-lesson’s body; otherwise, the combination would not make sense. Such combinations carry the value of “Practically Impossible” in their Action column. When homogeneity value is “1” (i.e. “Yes”), only the combinations that strictly satisfy the homogeneity conditions exist. The invalid combinations, also, carry the value of “Practically Impossible” in their Action column.

Action A: A (T)

Let N_i and N_j be the two headings of type T (either numbered or tagged heading).

If a set of unnumbered headings is placed above all the type T headings and the N_i is the first type T heading in the document, then:

- Use the Precedence Table (Table 2, described in Action E()) to assign initial numbers (IN) to every unnumbered heading in the set.
- Assign N_i to all unnumbered headings starting with the highest IN and, accordingly, adjusting the IN values of the other unnumbered headings in the set.

If a set of unnumbered headings is placed between the type T headings (N_i and N_j), then:

- Use the Precedence Table to assign initial

numbers (IN) to every unnumbered heading in the set.

If the level of $N_i > N_j$, then:

- Assign the level of N_j to all the unnumbered headings starting with the highest IN and, accordingly, adjusting the IN values of the other unnumbered headings in the set.

If $N_i = N_j$, then:

- Assign $N_i + 1$ to all the unnumbered headings starting with the highest IN and, accordingly, adjusting the IN values of the other unnumbered headings in the set.

If $N_i < N_j$, then:

- Assign $N_i - 1$ to all the unnumbered heading starting with the highest IN and, accordingly, adjusting the IN values of the other unnumbered headings in the set.

If a set of unnumbered headings is placed below all the type T headings and the N_i heading preceding it is the last type T headings in the document, then:

- Use the Precedence Table to assign initial numbers (IN) to every unnumbered heading in the set.
- Assign N_i to all the unnumbered headings starting with the highest IN and adjusting the IN values of the other unnumbered headings in the set.

End of Action A.

Action B: B()

Assign a heading tag to each numbered heading in accordance with the section level number accompanying the heading. For example, the numbered heading of “1 α ”, “1.1 β ”, and “2 μ ”, would be tagged `<h1> 1 α </h1>`, `<h2> 1.1 β </h2>`, and `<h1> 2 μ </h1>`, respectively. If there is a missing number in the sequence, empty headings are created for them, which then are inserted in the body.

End of Action B.

Action C: C()

Apply Action B to resolve the problem resulted from the combination of numbered headings and tagged headings. Apply Action A(tagged heading) to resolve the problem caused by the combination of unnumbered headings and tagged headings.

End of Action C.

Action D: D()

Let the title of the e-lesson that is requested from the retrieval system be α . Add a virtual heading of `<h1> α </h1>` at the beginning of the e-lesson’s body.

End of Action D.

Action E: E()

A level is assigned to the informal heading using the *precedence* table (Table 2). This table is composed of three precedence sub-tables for the three cases when all the headings are one of (a) bold, (b) regular, or (c) a mixture of bold and regular. For a given precedence sub-table, a heading that follows the format given in the first and last rows of the table has

the highest level (i.e. level 1) and lowest level, respectively.

End of Action E.

Table 2: The Precedence Table for the informal headings without numbering: (a) All the headings are bold, (b) None of the headings are bold, and (c) Some of the headings are bold and the others are not.

<code>
 [
]⁺ {bold title}
 [
]⁺</code>
<code>
 [
]⁺ {bold title}
</code>
<code>
 {bold title}
</code>
<code>
 [
]⁺ {bold title} (, : . - --)</code>
<code>
 {bold title} (, : . - --)</code>
a
<code>
 [
]⁺ {regular title}
 [
]⁺</code>
<code>
 [
]⁺ {regular title}
</code>
<code>
 {regular title}
</code>
b
<code>
 [
]⁺ {bold title}
 [
]⁺</code>
<code>
 [
]⁺ {regular title}
 [
]⁺</code>
<code>
 [
]⁺ {bold title}
</code>
<code>
 [
]⁺ {regular title}
</code>
<code>
 {bold title}
</code>
<code>
 {regular title}
</code>
<code>
 [
]⁺ {bold title} (, : . - --)</code>
<code>
 {bold title} (, : . - --)</code>
c

3.2 Logical Structure

The logical structure of the e-lesson’s body represents the relationship among the segments. To discover the logical structure of a given e-lesson, first the segments for each explicit term in an e-lesson’s view are identified and then the relationships among the segments are determined.

3.2.1 Segmentation

This is a process that takes an e-lesson’s body and divides it into a number of segments in such a way that each segment is considered as the content of one of the core explicit terms in the e-lesson’s view. To complete this process, the following three steps are taken:

- Building a look-up table to capture the physical structure of the e-lesson.
- Building a map of the e-lesson using the look-up table and the view.
- Identifying the beginning and ending of each segment of the e-lesson by using its map.

In the remaining part of this section, four algorithms are presented for implementing the above three steps.

Each algorithm is followed by an example for a better grasp of the concepts.

The following algorithm is in charge of building a look-up table to capture the physical structure of the e-lesson.

Algorithm LOOK-UP

Given: The entire html source of an e-lesson as a long string of characters, E, and an empty look-up table, L, with three columns of Type, Start, and Finish.

Objective: Building a look-up table for the e-lesson.

Step1- Repeat while (a formal or an informal paragraph exist)

- Identify the beginning and ending of the paragraph in E, assign a paragraph number to the paragraph, and insert (PID, PStart, PEnd) as a record into L.

Step2- Repeat while (a formal or an informal heading exist)

- Identify the beginning and ending of the heading in E, assign a heading number to the heading, and insert (HID, HStart, HEnd) as a record into L.

Step 3- End;

Example. Suppose the e-lesson given in Figure 1 (in the form of a skeleton), has the view of $T = \{t1, t2, t3, t4, t5, t6, t7, t8\}$. The formal and informal paragraphs AND headings are shown in this figure. The terms t7 and t8 are implicit because they cannot be found in the body. As a result, they are ignored. The look-up table produced by the execution of the algorithm LOOK-UP (for the e-lesson of Figure 1) is shown in Table 3.

The following algorithm would build the map for the e-lesson by using the look-up table and the view of the e-lesson's body.

Algorithm MAP

Given: The entire html source of an e-lesson as a long string of characters, E, the set of core explicit terms in the view of the e-lesson, $T = \{t_1, \dots, t_n\}$, the Look-up table, L, and an empty map, M, which is a table with four columns of Term, Type, Start, and Finish.

Objective: Building the map of the e-lesson.

Step 1- Repeat steps 2 and 3 for every t_i in T;

Step 2- Find all the occurrences of t_i in E.

Step 3- Repeat for each occurrence of t_i ;

- Use the Starting position of the Occurrence (SO) in E to find an entry in the Look-up Table so that $\text{entry.Start} \leq \text{SO} \leq \text{entry.Finish}$;
- Insert record $\langle t_i, \text{Type}, \text{Start}, \text{Finish} \rangle$ into M;

Step 4- Insert into M the records of L-M.

// For these records, the Type column in M would be empty.

Step 5- Sort M by attribute Start in ascending order;

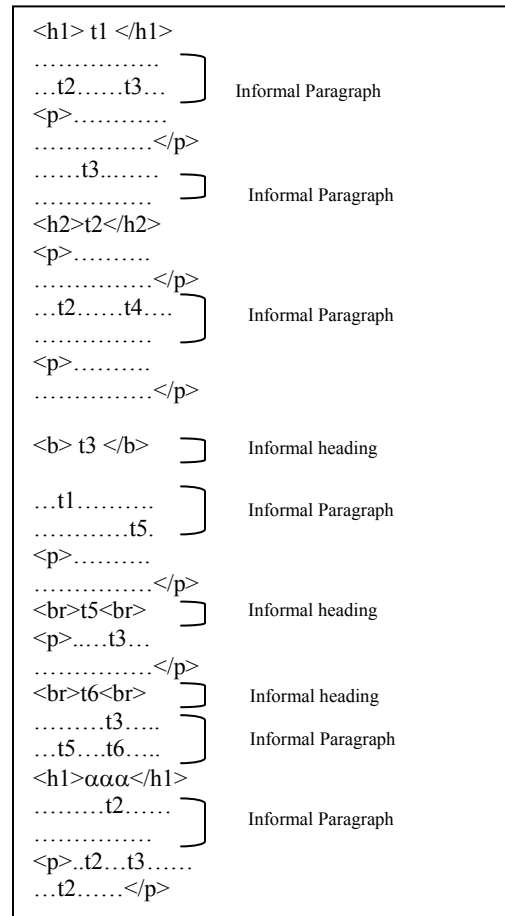


Figure 1: The skeleton of an e-lesson

Table 3. The Look-up table for the e-lesson in Figure 1.

Type	Start	Finish
h1	5	20
h2	800	810
h2	2000	2020
h3	3000	3040
h3	4000	4050
h1	5000	5030
P1	25	200
P2	205	598
P3	600	798
P4	825	1500
P5	1503	1998
P6	2022	2512
P7	2515	2998
P8	3043	3998
P9	4053	4998
P10	5033	6000
P11	6003	8741

Step 6- For two rows (i and j) in M,

Case 1: If $((\text{term}_i = \text{term}_j) \& (\text{type}_i \neq \text{type}_j))$, then: Delete one of the rows.

Case 2: If $((\text{term}_i \neq \text{term}_j) \& (\text{type}_i = \text{type}_j))$ and both are paragraphs) & (there are two other rows, m and n, within M in which $(\text{term}_i = \text{term}_m) \&$

(type_m = "h*") & (term_j = term_n) & (type_n = "h*")), then: Delete both rows i and j.

Case 3: If ((term_i ≠ term_j) & (type_i = type_j and both are paragraphs) & (there is another row, m within M in which (term_i = term_m) & (type_m = "h*")), then: Delete the row i.

Step 7- For any two rows i and j in M for which (term_i = term_j) & (type_i = "h*") & (type_j ≠ "h*") Delete row j.

Step 8- End;

The final M contents are shown in Table 4.

Table 4. The Map, M, of the e-lesson in Figure 1.

#	Term	Type	Start	Finish
1	t1	h1	5	20
2	-	P2	205	598
3	t2	h2	800	810
4	-	P4	825	1500
5	t4	P5	1503	1998
6	t3	h2	2000	2020
7	-	P7	2515	2998
8	t5	h3	3000	3040
9	t6	h3	4000	4050
10	-	h1	5000	5030

Table 5. The AllSegments Table, G, an outcome of the PARTITION algorithm for e-lesson of Figure 1.

#	Term	Type	Begin	End
1	t1	h1	5	4999
2	-	h1	5000	8741
3	t2	h2	800	1999
4	t3	h2	2000	4999
5	t5	h3	3000	3999
6	t6	h3	4000	4999

The following two algorithms identify the beginning and ending of each segment of the e-lesson using its map. The first algorithm is a recursive algorithm that partitions the e-lesson's body based on the headings' levels. The second algorithm actually delivers the starting and ending points for each component of the E-lesson.

Algorithm PARTITION (i, M, last)

Given: The map table (M) and the AllSegments table (G), with four columns named Term, Type, Begin, and End that carry term's name, term's type, the beginning and ending positions of the term's segment. The variable *i* is an index and the variable *last* initially has the order number of the last character in the e-lesson's body.

Objective: Partitioning the E-lesson based on headings' levels.

Step1. Table M1 includes all the rows in M for which Type = h_i;

If M1 ≠ ∅, then:

Repeat for j = 1 to |M1|;

Build a new record, R, for G using row_j of M1: R.Term = Term_j; R.Type = Type_j;

R.Begin = Start_j;

If j < |M1| Then R.End = Start_{j+1} - 1

Else R.End = Last;

Insert R into G;

// partitioning within each heading

K = all the rows in M1 for which

(M1.start > R.Begin) &

(M1.Finish ≤ R.End) & (M.Type = "h*");

If (k ≠ ∅), then:

Call PARTITION(i+1, K, R.end);

Step 2. End;

The AllSegments Table produced by the execution of the algorithm PARTITION for the e-lesson of Figure 1 is shown in Table 5.

Algorithm SEGMENTATION

Given: The Map Table, M, and the AllSegments Table, G.

Objective: Determining the starting and ending points for the components of the E-lesson.

Step1. If for row i of M there is a row k in G so that (Term_i = Term_k), then:

If for rows i and i+1 in M, (Term_{i+1} = ∅) &

(Start_{i+1} ≥ Begin_k) & (Finish_{i+1} ≤ End_k): then

Delete row i + 1 from M.

Step2- Repeat for row i = 1 to |M|

If for row i of M there is a row k in G so that

term_i = term_k, then:

Delete row i from M.

Step 3- G = G ∪ M;

Step 4- Repeat for row i = 1 to |G| of G

If (row_i.Term = ∅), then:

Assign a unique system-generated term value to the row.

Step 5- End;

The final outcome of the Segmentation algorithm is illustrated in Tables 6. Each row of the table is one segment.

Table 6. AllSegments Table (G): The segmentations produced for e-lesson of the Figure 1.

#	Term	Type	Begin	End
1	t1	h1	5	4999
2	Sys1	h1	5000	8741
3	t2	h2	800	1999
4	t3	h2	2000	4999
5	t5	h3	3000	3999
6	t6	h3	4000	4999
7	t4	P5	1503	1998

3.2.2 Segments' Relationships

First, the terms *encompassing* and *encompassed* are defined and an algorithm for delivering the logical structure of the e-lesson is discussed. If s_i and s_j are the segments of the terms t_i and t_j, respectively, and if

s_j is completely inside s_i , then s_i is an *encompassing* segment and s_j is an *encompassed* segment. The terms t_i and t_j are also referred to as encompassing and encompassed terms, respectively.

Algorithm STRUCTURE

Given: The AllSegments Table, G, and an empty logical structure Table, LS in which each row has six attributes of encompassing term (ITerm), encompassing term starting position (IStart), encompassing term ending position (IEnd), encompassed term (DTerm), encompassed term starting position (DStart), and encompassed term ending position (DEnd).

Objective: Identifying the logical structure of the e-lesson.

Step 1- Sort G by attribute Begin in ascending order;
 Step 2- Repeat for $i= 1$ to $|G|-1$;
 Repeat for $j = i+1$ to $|G|$;
 If (for rows i and j in G, $\text{Begin}_i < \text{Begin}_j$ & $\text{End}_i \geq \text{End}_j$), then:
 If (there is a row, k , in LS in which $\text{DTerm}_k = \text{Term}_j$), then:
 Remove row k from LS;
 Append $\langle \text{Term}_i, \text{Begin}_i, \text{End}_i, \text{Term}_j, \text{Begin}_j, \text{End}_j \rangle$ to LS;

Step 3- Remove those rows from LS for which ITerm is unique and DTerm is empty;

Step 4- End;

The logical structure for the e-lesson of Figure 1 produced by the above algorithm is shown in Table 7 and displayed as a tree in Figure 2.

Table 7. The Logical Structure Table (LS) produced for e-lesson of the Figure 1.

ITerm	IStart	IEnd	DTerm	DStart	Dend
t1	5	4999	t2	800	1999
t1	5	4999	t3	2000	4999
t2	800	1999	t4	1503	1998
t3	2000	4999	t5	3000	3999
t3	2000	4999	t6	4000	4999

4. Summary and Future Research

A number of algorithms are proposed that collectively are able to identify the physical and logical structures of a given e-lesson presented in an HTML document. The value of the proposed methodology lies in the fact that the granularity of a retrieved e-lesson changes from e-lesson itself to its components (subject/topics and sub-topics). Providing such a capability would directly affect both areas of Active Teaching and Semantic Web.

This research is in progress and intends to (a) determine the effectiveness of the proposed methodology, (b) use the proposed methodology in modeling of an Active Teaching environment, and (c)

explore the use of the proposed methodology in Semantic Web.

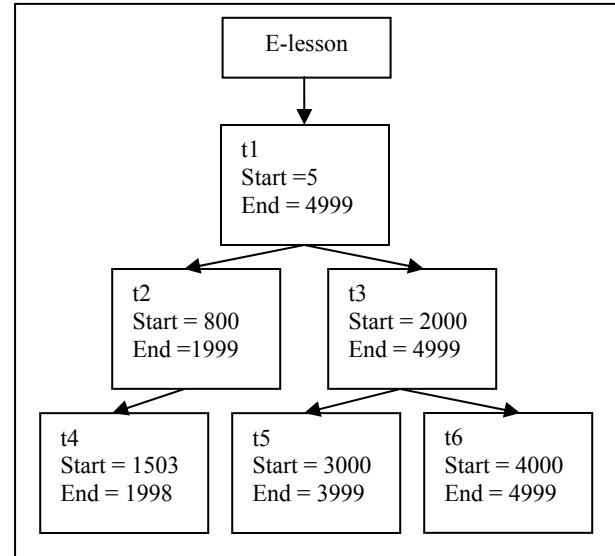


Figure 2: The tree representation of the logical structure for the e-lesson of Figure 1.

5. References

1. Siqueira S. W. M., Braz M. H. L. B., and Melo R. N., "E-Learning Content Warehouse Architecture", Proc. of the Intl. Conf. of WWW/Internet, pp. 739-742, Nov. 2002, Lisbon, Portugal.
2. Crescenzi V., Mecca G., and Meriardo P., "RoadRunner: Towards Automatic Data Extraction from Large Web sites", Proc of the Very Large Database Conf., 2001.
3. Embley D., Jiang S., and Ng Y., "Record-Boundary Discovery in Web Documents. Proc. of the ACM SIGMOD conf, 1999.
4. Knoblock C. A., Lerman K., Minton S., and Muslea I., "Accurately and Reliably Extracting Data from the Web: A machine Learning Approach. IEEE Data Engineering Bulletin, 23 (4):33-41, 2000.
5. Eugene Agichtein and Venkatesh Ganti, "Mining Reference Tables for Automatic Text Segmentation, Proc. of the 10th ACM SIGKDD Intl Conf. on knowledge Discovery and Data Mining. (KDD'04), August 2004, Seattle, WA.
6. Ljilijana Stojanovic, Steffen Staab, and Rudi Studer, "eLearning based on semantic web", The Proc. of WebNet2001-World Conf. on the WWW and Internet, Orlando, Florida, 2001.
7. Azita Bahrami, "A framework for development and management of e-lessons in e-learning", Proc. of the 2005 Intl. Conf. on Web Information Systems and Technologies (WEBIST'05), Miami, Florida, May 2005, pp. 504-509.