엔지니어링 문서의 문장 자동 계층정의 방법론

A Methodology for Automatic Hierarchy Definition of Sentences in Engineering Documents

박 상 일* 김 봉 근** 김 경 환*** 이 상 호†
Park, Sang Il Kim, Bong-Geun Kim, Kyeong-Hwan Lee, Sang-Ho
(논문접수일 : 2009년 6월 5일 ; 심사종료일 : 2009년 7월 30일)

요 지

본 논문은 엔지니어링 문서에서 각 제목의 머리기호가 그 문서의 논리적 계층 구조를 표현한다는 점을 이용하여 문서 내 각 제목의 계층을 자동으로 분류하는 방법론을 제시하였다. 제시한 방법론은 일반 텍스트 문서에서 세부 제목을 추출하는 방법과 추출된 제목의 계층을 정의하는 방법으로 구성된다. 문서의 세부 제목은 문장의 맨 앞에 위치한 머리기호의 형태를 미리 정의된 머리기호 그룹과 비교하여 추출하며, 추출된 제목의 계층은 머리기호 형태의 변화에 따라 각 제목간의 상대적 위치를 파악함으로써 정한다. 제시된 방법론을 이용하여 일반 텍스트 문서를 세부 제목에 따라 구조화된 XML 문서로 변환하는 시범 모듈을 개발하였으며, 20개의 엔지니어링 문서를 대상으로 그 성능을 분석하였다.

핵심용어 : 엔지니어링 문서, 자동 계층정의, XML 문서

Abstract

This paper proposes a methodology for automatic hierarchy classification of subtitles in a engineering document by the a fact that heading symbols of subtitles represent a hierarchical structure of the document. The proposed methodology is composed of two methods: extracting subtitles from plan text document and determining hierarchical structure of the subtitles. The subtitles in a document is extracted by comparing heading symbol patterns with predefined heading symbol groups, and the depth levels of the subtitles are determined by analyzing relative location of subtitles according to change of the heading symbol patterns. A prototype module, which can transform a plain text document into a structured XML document in accordance with a hierarchical structure of subtitles, is developed based on the proposed methodology, and the performance of the module is analyzed with 20 engineering documents.

Keywords *: engineering documents, automatic hierarchy definition, XML document*

1. 서 론

전자적으로 기록된 정보는 크게 구조화된 정보(structured information), 준구조화된 정보(semi-structured information) 및 비구조화 정보(unstructured information)의 3가지 형태로 나눌 수 있다. 구조화된 정보란 정형화된 정보모델에 따라 기록된 정보를 의미하는 것으로 표준화된 스키마 또는 정보모델에 의해 생성되며, 데이터베이스에 저장되어 있는 정보나 IFC(Industry Foundation Classes) 모델에 의해 생성된 빌딩의 모델 등이 이에 해당한다. 준구조화된 정보란 정형화된 스키마는 없지만 데이터의 내용과 더불어 형식이나 의미를 지니는 부가적인 정보가 포함된 정보로서 웹문서나 스키마가 없는 XML 문서가 이에 속한다. 그리고 비구조화 정보란 특정한 형식이 없이 저장된 정보를 의미한다. 구조 및 수리계산서, 공사시방서 등 현재 건설 실무에서 엔지니어링 문서로 활용되고 있는 대부분의 문서들은 비 구조화된 정보형식에 속

† 책임저자, 종신회원・연세대학교 사회환경시스템공학부 교수
Tel: 02-2123-2808 ; Fax: 02-364-5300
E-mail: lee@yonsei.ac.kr
* 학생회원・연세대학교 토목공학과 박사과정
** 정회원・연세대학교 토목공학과 박사과정
*** 연세대학교 토목공학과 석사과정

• 이 논문에 대한 토론을 2009년 10월 31일까지 본 학회에 보내주시면 2009년 12월호에 그 결과를 게재하겠습니다.

하며, 이들이 건설 프로젝트 수행과정에서 생산되는 정보의 상당부분을 차지하고 있다(Caldas 등, 2003; Zhu 등, 2001). 따라서 이러한 비구조화 문서의 효율적인 처리를 위하여 많은 연구가 진행되었는데, 대부분 대량으로 생산된 문서를 특정한 분류체계에 따라 문서 종류를 분류하는 내용에 초점이 맞추어져 있다. 예로서, McKechnie 등(2001)는 건설분야 문서의 서지정보기록을 지원하기 위한 기법을 제시하였고, Meziane 등(2003) 및 Rezgui(2006)은 문서에서 색인 정보를 추출하고 건설분야 온톨로지와 맵핑하여 문서를 관리하는 방안을 제시하였다. 그러나 Liu 등(2006)이 언급한 바와 같이 엔지니어링 분야의 경우 동일한 특징을 가지는 여러 문서의 내용을 제공받기 보다는 문서에 포함되어 있는 내용 중 사용자가 원하는 일부분에 대한 여러 정보를 필요로 하는 경우가 많기 때문에 문서 내에서 특정 정보를 추출하거나 효과적인 탐색이 고려되지 않은 위의 사례들을 실무에서 유용하게 사용하기에는 한계가 있다.

이러한 한계점을 극복하고자 Kosala 등(2006) 및 Liu 등(2006)는 문서 내에서 사용자 중심의 의미있는 정보 추출을 위한 연구를 수행하였다. 그러나 이들 연구에서 대상으로 하는 문서는 이미 마크업(mark-up)이 되어있는 준구조화된 문서의 특성을 지니고 있다. 따라서 엔지니어링 문서의 내용을 컴퓨터가 인식 가능하도록 의미적 정보를 체계화한다면 지식기반 시스템구축에 있어 그 활용성을 높일 수 있을 것이다. 본 연구는 이러한 노력의 일환으로서 효율적으로 비구조화 문서정보를 준구조화된 문서로 변환하고자 실무에서 작성된 문서의 텍스트 정보로부터 세부 제목들을 추출하고 각 제목의 계층적 구조를 분류하기 위한 방법론을 제시하였다.

2. 명시적 의미구조에 따른 XML 문서 변환 흐름

Wang 등(2005)은 문서의 의미적 구조(semantic structure)를 명시적 의미구조(apparent semantic structure)와 내적 의미구조(latent semantic structure)의 두 부분으로 분류하였다. 명시적 의미구조는 문서 내에서 문서작성자가 기입한 세부 제목의 구조를 의미하며, 내적 의미구조는 문서의 단락들이 내포하고 있는 의미에 따른 구조를 뜻한다. 따라서 명시적 의미구조의 경우 작성자에 의해 단락간의 순서가 유지되는 반면 내적 의미구조는 단락이 가지는 의미적 유사성 관계에 기반하여 구축되기 때문에 원래의 단락 순서가 무시된다.

한편, 엔지니어링 문서는 업무를 수행함에 따라 필요한 정보나 생산된 다양한 정보를 기록하고 전달하는 것을 목적으로 작성되기 때문에 일반적인 문서보다는 비교적 체계화된 구조를 이루고 있다. 일예로서, 토목분야의 강교 구조계산서의 경우 세부 제목으로 서술되는 것들은 '행위 이름', '부위 이름' 그리고 '변수 이름'으로 크게 3가지의 종류로 나눌 수 있다. '행위 이름'은 구조계산을 수행하는데 수반되는 세부 행위를 나타낸 것으로서 '슬래브 설계', '주형 설계', '이음부 설계', '단면 검토'와 같은 것들이 포함되며, 주로 '부위 이름'과 함께 사용되거나 '부위 이름' 하위에서 반복적으로 나타난다. '부위 이름'은 해당 구조물을 물리적으로 이루는 요소들이나 공간 또는 타입을 지칭하는 것으로서 '켄틸레버부', '제1 지간 중앙부', 'splice-1'과 같은 요소들을 나타내며, 이들은 적어도 한번이상 반복적으로 나타난다. 마지막으로 '변수 이름'은 액티비티를 수행하는데 있어 주어진 조건을 설명하거나 수행 이후 최종적인 결과를 설명할 때 주로 나타나며, '교량제원', '사용재료', '고정하중', '활하중' 등이 이에 포함되고, 이들은 동일한 타입의 부위에 해당되는 경우 '부위 이름' 이하에 반복적으로 나타난다.

일반적으로 대규모의 종방향 구조를 가지고 있는 엔지니어링 문서는 매우 많은 세부 제목들을 가지고 있으며, 문서 작성자는 각 세부 제목들 사이의 관계를 체계적으로 독자에게 전달하기 위해 머리기호를 이용하여 세부 제목의 구조를 명시적으로 나타낸다. 본 연구는 이러한 머리기호를 이용하여 엔지니어링 문서에서 세부 제목들을 추출하고, 추출된 세부 제목들을 이용하여 명시적 의미구조에 따라 구조화된 XML 문서를 생성하는 방법을 제시하였다. 전체에 대한 과정은 그림 1과 같다.

그림 1은 텍스트 문서를 대상으로 준구조화된 문서인 XML 스키마 파일로의 생성과정을 나타낸 것이다. 그림 1에 나타낸 바와 같이 문서의 세부제목의 구조는 사전에 미리 생성해 놓은 머리기호 그룹을 활용하여 문장에서의 머리기호를 분리해내는 과정과 분리한 머리기호를 바탕으로 문장 간 상호 비교를 통하여 문장의 상대적 계층을 정의하는 두 가지 과정을 통해 추출된다. 본 연구에서는 문서가 가지고 있는 문장의 인식을 통해 머리기호를 분리해내기 때문에 텍스트 파일을 입력파일로 사용하였으며, 이는 실무에서 사용되는

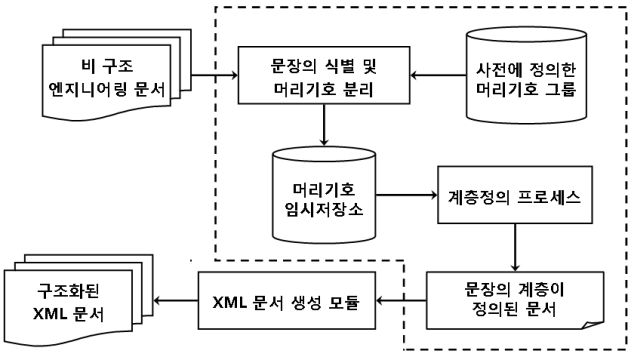


그림 1 세부 제목에 따른 준구조화된 XML 문서 생성과정

대부분의 상용 문서작성 프로그램에서 쉽게 변환될 수 있다. 출력파일은 준구조화된 문서의 대표적인 형태인 XML 문서 파일로 생성된다.

3. 텍스트 정보의 분석 및 계층 정의 방법

3.1 머리기호의 분리 방법

엔지니어링 문서를 분석(parsing)하기 위해 먼저 주어진 문자열의 구성요소를 정의하였으며, n번째 문장의 문자열의 집합 M^n이라 할 때, M^n의 구성요소를 Backus-Naur Form으로 표현하면 식 (1)과 같다.

$$< M^n >::=< H^n >< C^n > | < C^n > \quad (1)$$

여기서, H^n은 머리기호에 대한 문자열 집합을 나타내고, C^n은 내용에 대한 문자열 집합으로 공백을 포함할 수 있다. 본 연구에서는 H^n과 C^n과의 연결은 공백으로 되어있는 것으로 가정한다. 식 (1)에서 나타낸 바와 같이 H^n이 만약 문장 내에 존재 한다면 항상 문장의 제일 앞에 나와야 한다. 앞서 서술한대로 문장의 계층정의는 머리기호를 통해 이루어지며, 머리기호를 인식하는 방법은 식 (2)와 같다.

$$\exists H^n \in \forall ID^T \Leftrightarrow ID(H^n) = T \quad (2)$$

여기서, ID^T는 그룹 id값 T를 갖는 머리기호 집합을 의미하며, $ID(H^n)$은 머리기호 H^n의 그룹 id를 의미한다. 머리기호 그룹은 문서에 자주 나오는 머리기호들을 저장하고 있는 단어(원소)들의 집합으로 아라비아 숫자, 로마 숫자, 알파벳, 한글 자음, 원문자, 괄호문자, 기타 특수문자 등이 포함되어 있으며, 각 그룹마다 고유의 그룹 id를 가지고 있다. 표 1은 본 연구에서 머리기호의 분리를 위해 사용한 머리기호 그룹의 사례를 나타낸 것이다. 표 1에서 그룹 id 1에서 41까지는 여러 머리기호들이 하나의 머리기호 그룹을 구성하고 있으며, 그룹 id 42에서 53은 각각 하나의 머리기호가 하나의 머리기호 그룹을 구성하고 있다.

문장의 계층을 정의할 수 있음은 문장에서 머리기호를 분리해 낼 수 있다는 것과 같은 의미를 지니는 것으로, 이를 위해서는 문장의 앞부분에 표 1에서 제시한 머리기호를 포함하고 있어야 하고, 그렇지 않은 경우에 해당하는 C^n는 계층을 정의하는 항목에서 제외된다.

3.2 머리기호 분리의 예외사항 처리

하나의 문장이 식 (1)에서 언급한 바와 같이 H^nC^n의 형태로 구성이 되어 있는 경우에는 머리기호의 분리에 큰 문제가 없지만, C^n의 형태로 구성이 되어 있는 경우에는 집합 C^n의 앞부분에 머리기호 그룹의 그룹 id 2에 해당하는 원소를 포함하고 있는 경우가 생긴다. 즉, 문장의 첫 머리에 "1.3", "2.6"과 같이 소수(decimal)와 공백이 함께 나오게 되면 이는 머리기호 그룹의 숫자 그룹 2와 같은 형태를 갖게 된다. 소수가 첫머리를 갖는 문장의 경우에는 머리기호의 분리과정을 거치지 않아야 하므로, 본 연구에서는 식 (3)을 활용하여 머리기호의 분리가 이루어지지 않도록 하였다.

표 1 머리기호 그룹 및 그룹 id

그룹명	그룹 id(T)	머리기호(ID^T)
숫자 그룹1	1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, …
숫자 그룹2	2	1.1, 1.2, …, 2.1, 2.2, …, 3.1, 3.2, …
숫자 그룹3	3	1.1.1, 1.1.2, …, 1.2.1, 1.2.2, …, 2.1.1, 2.1.2, …
숫자 그룹4	4	1.1.1.1, 1.1.1.2, …, 1.1.2.1, 1.1.2.2, …, 1.2.1.1, 1.2.1.2, …
원문자 그룹1	5	①, ②, ③, ④, ⑤, ⑥, ⑦, ⑧, ⑨, ⑩, ⑪, ⑫, ⑬, ⑭, ⑮
원문자 그룹2	6	㉮, ㉯, ㉰, ㉱, ㉲, ㉳, ㉴, ㉵, ㉶, ㉷, ㉸, ㉹, ㉺, ㉻
원문자 그룹3	7	ⓐ, ⓑ, ⓒ, ⓓ, ⓔ, ⓕ, ⓖ, ⓗ, ⓘ, ⓙ, ⓚ, ⓛ, ⓜ, ⓝ, ⓞ, ⓟ, ⓠ, ⓡ, ⓢ, ⓣ, ⓤ, ⓥ, ⓦ, ⓧ, ⓨ, ⓩ
로마숫자 그룹1	8	Ⅰ, Ⅱ, Ⅲ, Ⅳ, Ⅴ, Ⅵ, Ⅶ, Ⅷ, Ⅸ, Ⅹ, Ⅺ, Ⅻ
로마숫자 그룹2	9	ⅰ, ⅱ, ⅲ, ⅳ, ⅴ, ⅵ, ⅶ, ⅷ, ⅸ, ⅹ, ⅺ, ⅻ
한글기호 그룹	10	가, 나, 다, 라, 마, 바, 사, 아, 자, 차, 카, 타, 파, 하
한글자음 그룹	11	ㄱ, ㄴ, ㄷ, ㄹ, ㅁ, ㅂ, ㅅ, ㅇ, ㅈ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎ
알파벳 그룹1	12	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
알파벳 그룹2	13	a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
괄호 그룹1	14-27	각 그룹의 머리기호 끝에 기호 ")"를 붙인 그룹들 예) ID14 = {1), 2), 3), …}, ID24 = {ㄱ), ㄴ), ㄷ), …}
괄호 그룹2	28-41	괄호그룹1의 머리기호 앞에 기호 "("를 붙인 그룹들 예) ID28 = {(1), (2), (3), …}, ID38 = {(ㄱ), (ㄴ), (ㄷ), …}
기타 그룹	42-53	{-}, {*}, {◁}, {◀}, {▷}, {▶}, {◉}, {◈}, {◎}, {•}, {○}, {□}는 각각 하나의 그룹

$$\exists ID(\widehat{H^n}) = 2,\ H^n = \begin{cases} \widehat{H^n}, & \text{if } ID(H^{n-1}) = 1 \text{ or } 2 \text{ and } \widehat{H^n} - H^{n-1} = 0.1 \\ \varnothing, & otherwise \end{cases} \quad (3)$$

여기서, $\widehat{H^n}$은 n번째 문장의 첫머리 부분이 머리기호 그룹의 원소인 경우에 임시로 분리한 머리기호로, 임시로 분리한 머리기호의 그룹 id가 2인 경우에만 식 (3)의 과정을 거치게 된다. 이때의 머리기호 H^n은 $n-1$번째 문장의 머리기호(H^{n-1})의 그룹 id가 1 또는 2이고, n번째 문장에서 임시로 저장한 머리기호($\widehat{H^n}$)와 $n-1$번째 문장의 머리기호(H^{n-1})의 차가 0.1인 경우에만, 머리기호로써 문장에서 분리가 되고, 그렇지 않은 경우에는 문장의 첫머리는 문장에서 분리되지 않는다.

3.3 머리기호 분리를 위한 전체 프로세스

그림 2는 3.1절과 3.2절에서 설명한 내용을 바탕으로 머리기호 분리를 위한 전체 프로세스를 나타낸 것으로, 문장의 머리기호 요소 분리와 머리기호 그룹 id 부여는 사전정의 머리기호 그룹 원소와의 비교를 통해 수행되며, 예외처리 과정을 거쳐 분리한 머리기호(H^n)와 그에 해당하는 머리기호 그룹 id가 저장되어 후속 프로세스인 문장의 자동 계층정의의 입력파일로 활용된다.

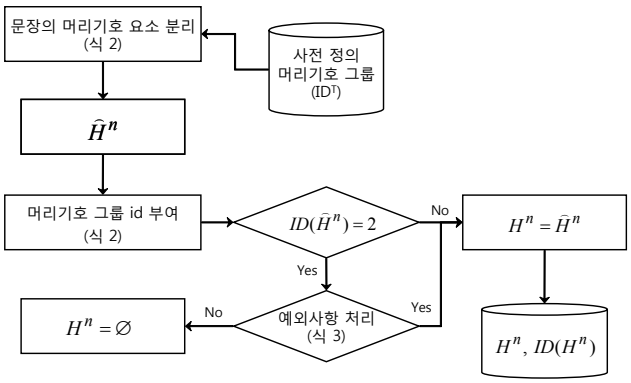
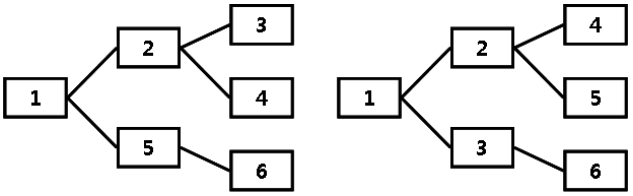


그림 2 머리기호 분리를 위한 전체 흐름도

3.4 문장의 계층 정의 방법

앞 절에서 제시한 방법을 활용하여 문장에서 분리한 머리기호는 그 자체만으로는 문서 내에서 계층을 식별 할 수는 없다. 일반적으로 문서를 작성할 때 로마 숫자(표 1의 로마 숫자 그룹 1)나 아라비아 숫자(표 1의 숫자 그룹 1)를 제일 처음에 사용하여 이것을 포함한 문장을 제일 상위 계층 혹은 두 번째 계층이라고 간주할 수도 있지만, 이는 문서 작성자



(a) 깊이 우선방식 (b) 너비 우선방식

그림 3 트리 노드의 선형 순서정렬 방식

에 의해 결정되는 요소로 머리기호 그룹이 문서 내에서 절대적인 계층을 갖지는 않는다. 따라서 본 연구에서는 머리기호를 활용한 문장의 상대적인 위치를 파악하여 계층을 정의하는 방법을 제시하였다.

그림 3(a)는 트리 노드의 선형 순서정렬방식 중 깊이 우선방식(depth-first order)을 나타내는 것이며, 그림 3(b)는 너비 우선방식(breadth-first order)을 나타낸다. 너비 우선방식은 각 요소의 번호를 지정하는데 있어 같은 계층을 가지는 요소의 번호 지정이 끝난 후 하위 계층의 번호를 지정하는 방식인 반면에, 깊이 우선방식은 하나의 요소와 직접적으로 관련이 있는 하위 계층 요소의 번호를 먼저 지정을 한다. 일반적으로 문서의 작성 순서는 깊이 우선방식의 작성 순서와 같다.

그림 3(a)에서 번호가 매겨져 있는 각각의 요소를 문장에서의 머리기호라고 한다면, 요소 1(1)은 문서에서 제일 상위계층을 나타내는 문장의 머리기호이고, 요소 2(2)와 요소 5(5)는 두 번째 계층, 요소 3(3)과 요소 4(4)와 요소 6(6)은 세 번째 계층을 나타내는 문장의 머리기호로 볼 수 있다. 만약 문서의 작성자가 문서 작성 시 실수없이 같은 계층을 가지고 있는 문장은 같은 머리기호 그룹에 속한 머리기호를 사용한다면, 요소 2(2)와 요소 5(5)는 같은 머리기호 그룹 id를 가지고 있을 것이고, 요소 3(3)과 요소 4(4)와 요소 6(6) 역시 같은 머리기호 그룹 id를 가지고 있을 것이다. 그리고 요소 1(1)과 요소 2(2)와 요소 3(3)의 머리기호 그룹 id는 모두 다른 값을 가지고 있을 것이다. 즉, 문서 내에서 문장이 어떠한 위치에 배치되어 있느냐와 관계없이 어떤 머리기호 그룹 id를 가지고 있느냐에 따라서 문서의 계층을 파악할 수 있다. 그림 4는 트리 노드의 선형 순서정렬 방식 중 깊이 우선방식을 응용하여 문서 내에서 문장의 계층을 자동으로 정의할 수 있는 프로세스를 나타낸 것이다.

그림 4에서 $Lv(x)$는 변수 x의 문서 내에서의 계층 등급을 나타내는 것으로, 문서의 제일 상위 계층의 등급을 1로 지정하고, 계층이 한 단계씩 낮아질 때마다 등급의 숫자는 한 단계씩 증가한다. $Lv(ID(H^n))$의 의미는 n번째 문장의 머리기호 H^n을 가지고 있는 문장 머리기호 그룹 id의 계층

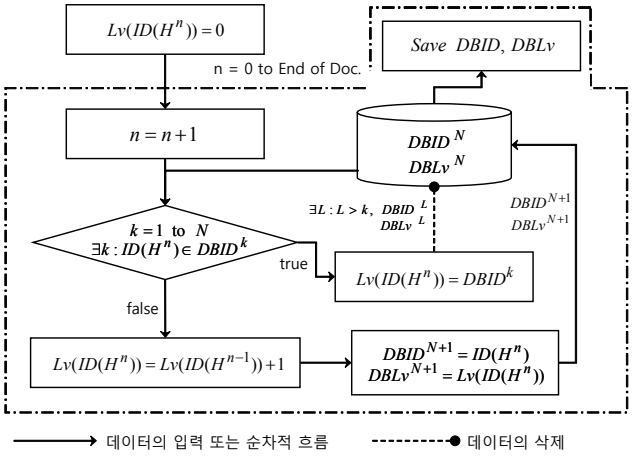


그림 4 문장의 자동 계층정의 프로세스

문장 번호		
①	1. 설계기준	ID("1")=1, Lv=1
②	1) 교량제원	ID("1)")=14, Lv=2
③	i) 교량등급	ID("i)")=22, Lv=3
④	ii) 교량형식	ID("ii)")=22, Lv=3
	:	:
	:	:
⑤	3. 바닥판 설계	ID("3")=1, Lv=1
⑥	3.1 캔틸레버부	ID("3.1")=2, Lv=2
⑦	1) 우측 캔틸레버부	ID("1)")=14, Lv=3
	:	:
	:	:

그림 5 문서 내 문장 계층정의의 예시

을 뜻하는 것으로, 계층 등급의 초기값을 0으로 선정하였다. 계층 등급의 초기화 이외의 나머지 프로세스는 문장마다 문서의 끝까지 반복적으로 진행된다. *DBID*와 *DBLv*는 문장의 계층정의 시 참조하기 위해여 임시로 저장해 놓은 데이터 테이블로 각각 머리기호 그룹 id값과 그때 해당하는 문장에서의 계층 등급을 의미한다. *N*은 임시 저장테이블에 저장되어 있는 *DBID*와 *DBLv*의 개수를 나타내며, 문서에서 제일 첫 문장을 분석할 경우에는 참조할 수 있는 데이터 값이 없고, 한 번의 루프를 수행해야 $DBID^1$과 $DBLv^1$이 생성된다. 문장에서 분리한 머리기호를 활용하여 문장의 계층을 정의하기 위해서는 분리한 머리기호의 그룹 id와 임시 저장테이블에 저장되어 있는 값의 비교를 통해 이루어진다. 만약 분리한 머리기호의 그룹 id가 임시 저장테이블에 저장되어 있지 않다면, 해당 문장은 문서에서 처음 출현한 경우로 간주되고, 해당 머리기호 그룹 id에 새로운 계층 등급이 부여된다. 새로운 계층 등급은 바로 앞 문장에서 정의된 계층 등급보다 한 단계 아래의 등급을 부여받게 되며, 새로 생성된 머리기호 그룹 id와 그룹 id에 해당하는 계층 등급의 정보는 임시 테이블에 저장되어 문장의 계층 등급 정보를 제공한다. 만약 분리한 머리기호 그룹 id와 임시 저장테이블에 저장되어 있는 값이 일치하면, 해당 머리기호 그룹 id는 저장되어 있는 값과 같은 계층 등급을 부여받는다. 그림 5는 실제 엔지니어링 문서에 포함되어 있는 내용의 일부를 사례로 나타낸 것이다.

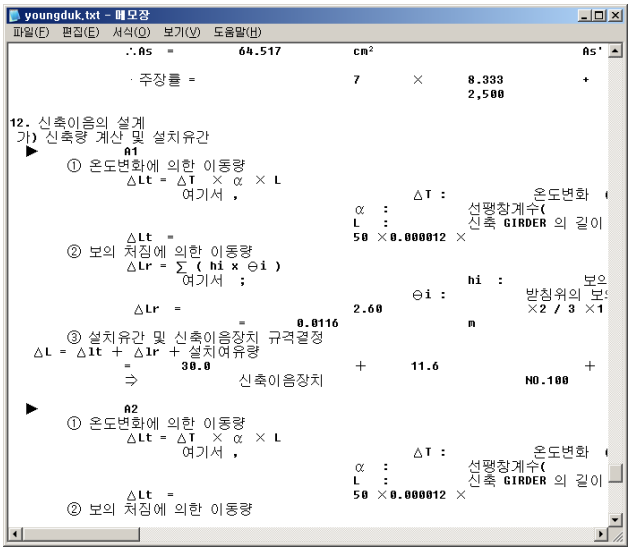
그림 5에서 문장 ①에서 ③까지는 문장의 머리기호와 임시 저장테이블에 저장된 값이 일치되지 않는 경우로, 문장의 등급은 바로 위 문장보다 한 단계 아래 등급을 부여받는다. 문장 ④와 ⑤는 문장의 머리기호가 임시 저장테이블에 일치하는 값이 있는 경우로서, 두 문장은 임시 저장테이블에 저장되어 있는 문장의 계층과 같은 등급을 부여받는다. 문장 ⑥의 경우는 또다시 새로운 머리기호가 출현한 경우로, 이때에는 문장 ⑤보다 한 단계 아래 등급의 계층 등급을 부여받는다. 하지만, 앞서 설명한 프로세스를 따른다면 문장 ⑦의 경우는 문장 ②의 머리기호 그룹 id 및 문장 계층 등급이 임시 저장테이블에 존재하기 때문에 문장 ⑦은 계층 등급 2를 부여받는 잘못된 결과를 생성한다. 이는 머리기호와 문장의 계층 등급 간의 상호 절대적 매칭의 문제점을 드러내는 것으로, 문장 ⑤에서 ⑦이 문장 ①에서 ④와 같은 머리기호 그룹 id 1, 14, 22의 순서가 아닌 이상 정확한 결과를 생성할 수 없다.

본 연구에서는 이러한 결과를 방지하기 위하여 데이터 삭제 프로세스를 추가하였다. 그림 4의 점선으로 구성된 부분이 데이터 삭제의 흐름을 나타낸 것으로, 문장에서 분리한 머리기호와 임시 저장테이블에 같은 값이 존재한다면, 해당 머리기호의 계층 등급을 정의해준 후 정의해준 계층 등급보다 등급이 낮은 경우의 머리기호 그룹 id와 계층 등급을 삭제하도록 하였다. 그림 5에서 문장 ⑤가 이에 해당하는 경우로서 문장 ⑤의 계층 등급을 1로 정의해준 후 그 보다 낮은 등급을 나타내고 있는 문장 ②, ③, ④의 머리기호 그룹 id 및 계층 정보를 임시 저장테이블에서 삭제한다. 이와 같은 프로세스를 거치게 되면 문장 ⑥과 ⑦의 머리기호는 이전에 임시 저장테이블에 저장되어 있지 않기 때문에 바로 위 문장보다 한 단계 낮은 등급의 계층을 부여받게 되는 프로세스를 거치게 된다. 그 결과 문장 ⑥은 계층 등급 2가 되고, 문장 ⑦은 계층 등급 3으로 정의되어 올바른 결과를 생성하게 된다.

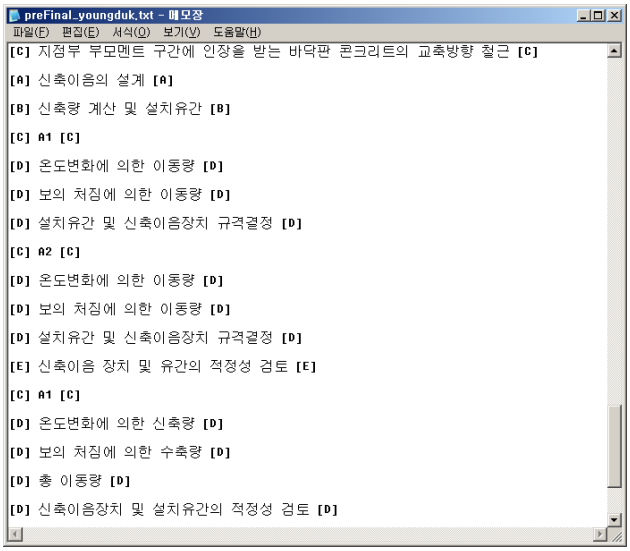
4. 시범 응용 모듈의 구현 및 성능 평가

4.1 문서의 자동 계층정의 모듈의 구현

본 연구에서는 앞서 제시한 방법을 바탕으로 Microsoft의



(a) 텍스트 형식의 교량 구조계산서



(b) 계층 정보를 담고 있는 구조계산서

(c) 구조계산서의 XML 스키마 문서화

그림 6 교량 구조계산서의 준구조화된 정보문서로의 변환 과정

.NET Framework 2.0 기반의 Visual C#을 이용하여 텍스트 기반 엔지니어링 문서를 문장의 계층 정보를 포함하고 있는 문서로의 변환 모듈을 개발하였다. 그림 6은 토목 분야의 대표적인 엔지니어링 문서인 교량의 구조계산서를 준구조화된 정보 문서인 XML 스키마 문서로 변환한 모습을 나타낸 것이다.

그림 6(a)는 문서의 자동 계층정의를 위해 입력하는 파일로 토목분야의 대표적인 엔지니어링 문서인 텍스트 형식의 교량 구조계산서의 일부를 나타낸 그림이며, 그림 6(b)는 입력 파일 문서의 문장에 대한 계층정의 결과를 나타낸 것이

다. 그림 6(b)에서 문장의 계층정보는 문장의 양쪽에 〔A〕, 〔B〕, 〔C〕 등의 태그 형태로 나타내며, 문서에서 서술부인 구조계산을 위한 실제 데이터는 문장의 계층과 무관한 것으로 결과물 출력에서는 제외하였다. 그림 6(c)는 그림 6(b)를 XML 스키마 파일로 변환한 모습을 나타낸다.

4.2 구현 모듈의 성능평가

본 연구에서 개발한 모듈의 성능은 실무에서 작성된 구조계산서 10개, 연구보고서 5개 및 학위논문 5개의 총 20개 실험문서를 대상으로 평가하였다. 본 논문에서 제시된 방법론에 따른 분석 단위인 문장의 수는 총 319,339개이다. 제안된 방법론의 특성에 따라 상대적 계층정의에 대한 정확도(GAA)와 절대적 계층정의에 대한 정확도(PAA)를 이용하여 개발된 모듈의 성능을 평가하였으며, 각 정확도의 산정 방법은 정보검색 분야에서 주로 활용되는 식 (4) 및 (5)에 나타난 바와 같은 Recall 방법론을 이용하였다. Recall 방법론은 TP(true positive) 및 FN(false negative)를 이용한 측정 방법론이다. 본 연구에서 TP는 분리해야 하는 머리기호 및 제목을 정확히 분리하여 인식한 경우를 의미하고, FN은 분리해야 하는 항목을 분리하지 못한 경우를 의미한다.

$$\text{GAA(Generalized Accuracy for Application module)} = \text{Correct}^{G} / (TP + FN) \quad (4)$$

$$\text{PAA(Precise Accuracy for Application module)} = \text{Correct}^{P} / (TP + FN) \quad (5)$$

식 (4)에서 Correct^{G}는 TP에서 상대적 계층정의가 올바로 이루어진 것의 개수를 나타내고, 식 (5)에서 Correct^{P}는 TP에서 절대적 계층정의가 올바로 이루어진 것의 개수를 나타낸다.

본 연구에서 수행된 실험에서 전체 평균은 GAA가 98.0%, PAA가 94.8%를 나타내어 높은 정확도를 가지고 있음을 알 수 있다. 그림 7은 각 실험대상 문서에 따른 정확도의 분포를 나타낸 것이다. 그림 7에 나타낸 바와 같이 GAA의 경우 구조계산서가 96.9%, 연구보고서가 98.6%, 학위논문이 99.1%로서 높은 정확도를 보여주고 있다. PAA의 경우 연구보고서와 학위논문의 경우 모두 97%이상 높을 값을 나타냈으나 구조계산서의 경우 90.7%로 GAA보다 약 6.2%의 감소를 보였다. 오류는 개조식 형태의 구문을 세부 제목으로 인식하는 것이었으며, 그 외 문서작성자의 머리기호 표기 오류에 의한 것이 주된 원인이었다. 특히 실무에서 작성된 구조계산서의 경우 연구보고서나 학위 논문에 비하여 머리기호

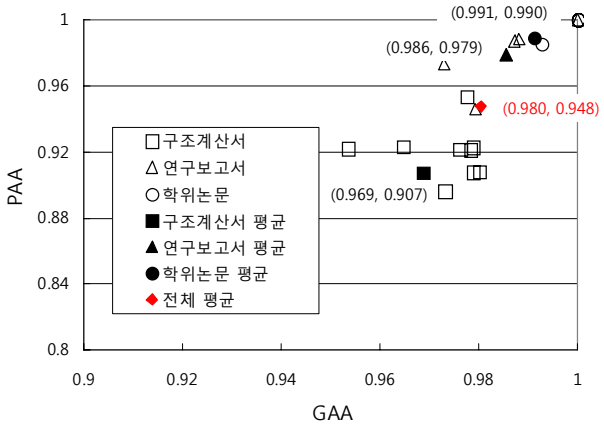


그림 7 실험 대상 문서별 정확도 분포

의 사용이 체계화되지 않았던 것이 절대적 계층정의 관점에서의 정확도를 크게 감소시키는 원인으로 분석되었다.

5. 결 론

본 연구에서는 복잡한 엔지니어링 문서에서 세부 제목을 나타내기 위해 사용되는 머리기호를 이용하여 문서의 명시적 의미구조를 자동으로 정의할 수 있는 방법론을 제시하였다. 제시된 방법론을 기반으로 텍스트 정보를 명시적 의미구조에 따라 XML 문서로 생성하는 문서변환 시범모듈을 개발하였으며, 여러 엔지니어링 회사에서 작성된 구조계산서 10개, 연구보고서 5개 및 학위논문 5개를 대상으로 그 성능을 검증하였다. 본 연구 수행에 따라 얻은 결론은 다음과 같다.

본 연구에서 제시한 방법론은 엔지니어링 실무에서 다양한 표현 방식에 의해 작성되고 있는 엔지니어링 문서의 텍스트 정보를 신속하고 효율적으로 준구조화된 XML 문서로 변환하는데 사용될 수 있음을 확인하였다. 특히 실무 문서에서 나타나는 다양한 물리적 포맷의 특성과 머리기호에 상관없이 텍스트 정보로부터 문서가 가지는 고유의 계층구조를 추출할 수 있었다. 또한 총 20개의 수집된 문서를 대상으로 한 응용모듈의 성능 평가에서 GAA 및 PAA에 대한 전체 평균이 각각 98.0% 및 94.8%로 매우 높은 정확도를 가지고 있음을 검증하였다.

감사의 글

본 연구는 건설교통부에서 실시한 건설핵심기술연구개발사업(교량설계핵심기술연구단)의 연구비 지원에 의해 수행되었으며, 2009년도 교육인적자원부 BK21 사업의 일환인 연세대학교 사회환경시스템공학부 미래사회기반시설 산학연공동사업단의 부분적인 지원을 받았음.

참 고 문 헌

Caldas, C.H., Soibelman, L. (2003) Automating hierarchical document classification for construction management information systems, *Automation in Construction*, 12(4), pp.395~406.

Kosala, R., Blockeel, H., Bruynooghe, M., Bussche, H.V. (2006) Information extraction from structured documents using k-testable tree automaton inference, *Data & Knowledge Engineering*, 58(2), pp.129~158.

Liu, S., McMahon, C.A., Darlington, M.J., Culley, S.J., Wild, P.J. (2006) A computational framework for retrieval of document fragments based on decomposition schemes in engineering information management, *Advanced Engineering Informatics*, 20(1), pp.401~413.

McKechnie, J., Shaaban, S., Lockley, S. (2001) Computer assisted processing of large unstructured document sets: a case study in the construction industry, *Proceedings of the 2001 ACM Symposium on Document engineering*, pp.11~17, Atlanta, Georgia, USA.

Meziane, F., Rezgui, Y. (2003) A document management methodology based on similarity contents, *Information Sciences*, 158, pp.15~36.

Rezgui, Y. (2006) Ontology-centered knowledge management using information retrieval techniques, *Journal of Computing in Civil Engineering*, 20(4), pp.261~270.

Wang, Z., Wang, Y., Gao K. (2005) A new model of document structure analysis, *Lecture Notes in Computer Science*, 3614, pp.658~666.

Zhu, Y., Issa, R.R. A., Cox, R.F. (2001) Web-based construction document processing via malleable frame, *Journal of Computing in Civil Engineering*, 15(3), pp.157~169.