Construction of a Dynamic Document Using Context-Free Pieces

Kenji Hanakawa
Department of Electrical Engineering
and Computer Science
Osaka Prefectural College of Technology
26-12 Saiwaicho, Neyagawashi 630-01, Japan
hanakawa@ecs.osaka-pct.ac.jp

Hideaki Takeda Toyoaki Nishida
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-01, Japan
{takeda, nishida}@is.aist-nara.ac.jp

Abstract

In this paper, we propose a system for shared writing and reading of documents suitable for computer network environment which is based on pieces of documents. Pieces of documents are statements about particular facts such as "The earth has a spherical surface". In this system, writers can create document pieces and store them into a common repository. Readers then can select document pieces from it. However, the selected documents are disjointed and hard to read. In order to make them easy to understand, it is necessary to organize them according to semantic models on which human writing is based. We introduce two semantic models: Top-down Model and Context-strengthen Model. We show how to create document pieces and how to construct a document from them with these models.

1. Introduction

Information, which is mainly presented in natural language, can be categorized into two types. One is information about only facts, such as encyclopedias, technical manuals and textbooks. The other is information including works of fiction or subjective opinions, such as novels, poems, tales and stories. We call the former type "document", and study techniques for effective creation, storing, distribution and using it.

Usually, all documents are written for expected readers by one or a small number of authors. Not only conventional books, but also electrical web pages and on-line manuals are written in this way. We call these documents "static documents".

It is impossible for an author to write a document which matches every reader's requirements because they can be various. Readers who can not get all necessary information from one book need to read other books. If the author could write documents for some variety of readers, these documents are ideal but would be expensive and not practical.

Recently, a large quantity of information is required for our activities. Systems that can supply necessary documents effectively are needed. However, a simple combination of the static documents and computer technologies is not useful because it can cause easily redundancy and contradiction.

For instance, suppose skilled software engineers. They usually do not read a programming reference manual from the first page to the last page. As they already know most part of the programming language, they only refer to specific parts of the manuals. If they cannot get all the information from a single manual, they will read other manuals. As a result, they can get disjointed fragments from some set of manuals. It is difficult to merge such fragments because they can easily contain some statements about the same subject which contradict each other.

Most information consists of facts, which are universal truths or propositions accepted by all member of the community. "The earth has a spherical surface" is an example of fact. Textbooks and science or technology documents contain these facts. Writing them must be the common task of the community. However, in the conventional book-style authoring systems, writing is an activity of one or a small number of authors. Computing technology such as the Internet provides media between writers and readers, but not among writers.

We have two hypotheses:

- A document consists of context-free pieces. I.e., the pieces can be understood independently without knowing the context.
- A document can be constructed according to syntactic and semantic models, and it is then easy to read.

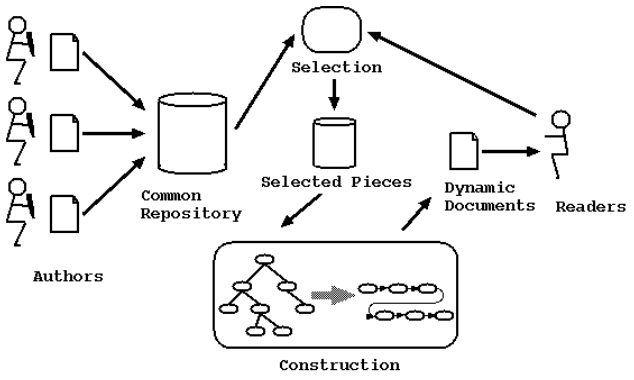

Figure 1. The dynamic document system

Based on these hypotheses, we propose a dynamic document system which consists of common repositories and document constructors.

A dynamic document is a non-static document created dynamically in response to readers' requirement. Figure 1 outlines the concept of a dynamic document and shows how the system works. Writers create document pieces and store them into the common repository. When a reader inputs her/his search requirements, the system selects the relevant pieces from the repository and constructs them into one document.

2. The dynamic document system

2.1. The common repository

The purpose of the dynamic document system is world wide information sharing. The common repository is a large scale distributed database system which is connected with users by network. It's similar to an digital library, but "book" which is a basic unit in libraries is not used here.

The necessary conditions of the common repository are as follows:

1. All pieces are true.

2. All pieces are context independent.

3. All pieces are unique.

If there are two or more pieces that are contradictory to each other, the authors need to discuss to find out which is true and put it in a repository.

The common repository is taken as a set of facts in which all elements are true. Consequently all elements of any subsets extracted from the set are true. Therefore, when a user selects parts from the common repository by any searching technologies, all selected document pieces are true. They are also unique, because elements of the set are unique. A repository can be constructed from various types of pieces. For example, section and paragraph can be the units of a piece.

2.2. The dynamic document constructor

A user access the common repository using a client system which inputs user's keywords and outputs a dynamic documents. The client system contains the dynamic document constructor.

The dynamic document constructor is a computer program which constructs dynamic documents. It is based on the basic assumption that documents written by human beings with syntactic and semantic constraints are easy to read. The constructor uses models of human writing: the Top-down Model and the Context-strengthen Model. Details of these models will be discussed later in this paper.

3. Creation of context-free pieces

This study is based on the hypotheses that a document consists of context-free pieces. In this section, we discuss methods for extracting these document pieces.

Generally speaking, it is difficult to understand the meaning of isolated parts extracted from documents, because the loss of neighboring parts causes the loss of context. There are two common phenomena involved with contexts in the technical document: referring and ellipsis. A part of a document can include pronouns such as "it" and "this". Usually pronouns refer to preceding words or noun phrases, and meaning of the pronouns are equal to meaning of preceding words or noun phrases. Documents include ambiguous words or noun phrases caused by ellipsis which are disambiguated by adding the preceding words or noun phrases.

Document pieces are generated using the following four processes.

1. Dividing: Divide a document into parts.

2. Completion: Add or substitute words to be understood independently.

3. Storing: Gather parts or paragraphs from more than one documents and store them in one repository.

4. Unifying: Unify parts that have the same information.

1. Bash is the shell, or command language interpreter, that will appear in the GNU operating system.
2. The name (of Bash) is an acronym for the "Bourne-Again SHell", a pun on Steve Bourne, the author of the direct ancestor of the current UNIX-shell /bin/sh.
3. Bash is an sh-compatible shell that incorporates useful features from the Korn shell and the C shell.
4. It $\longrightarrow$ Bash is ultimately intended to be a conformant implementation of the IEEE POSIX Shell and Utilities specification.

Figure 2. Avoiding of context dependency

At present these processes can be done only by human beings and not by computers. However, these processes are not tedious tasks for the original author.

We have experimented this method using a technical manual of Bash, the GNU shell. The manual is called "Bash's article.txt". Figure 2 is a typical example of applying the completion process to make pieces from the manual. In the figure, the words in parentheses are added words, words on the left-hand side of an arrow are removed words, and words on the right side of an arrow are substituted words. The subject of the second sentence, *The name*, is unidentified, and *of Bash* is added next to it. The subject of the fourth sentence, *It*, is also unidentified, and *Bash* is substituted for *It*.

Substitution and adding some words or noun phrases can preserve original meaning and make the sentence context free. This result shows that documents can be divide into context free document pieces.

Our study has shown that sentences in the manual could be translated into context-free one, except programming examples and explanations. As there are so many reference connections between them, we could not transform them into document pieces. However, we could make a document piece from a group of programming examples and explanations.

4. Construction of dynamic documents

Documents written by humans have contextual connections between parts. On the other hand, a set of document pieces which have no connections is hard to understand. If a computer program constructs a document which has the same structure of documents written by humans, it can be easily understood. We think that the structure is based on semantic models.

In the following sections, we will discuss two semantic models. One is the Top-down Model, in which relation between the former sentence and the latter sentence is considered as one of the eight relations. This model is used for arranging sentences in paragraphs. The other is the Context-strengthen Model in which the relation between the upper element and the lower element is considered as the relation between the stronger and the weaker context. This model is used for creating the structure of a document. We will explain each model in the following sections.

4.1. The Top-down Model

We have examined paragraphs and have identified the following 8 categories of connection between two succeeding : concrete, feature, attribute, example, element, item, association and sequence.

These categorical connections, except sequence, are abstracted as connections between a superior concept and an inferior concept and the former sentence is the superior sentence. We call the constraint on these categories "Top-down Model".

We have used the following steps for arranging sentences which are extracted from some paragraphs in Bash's article.

1. divide an original paragraph into sentences.
2. identify connections between the sentences.
3. construct a tree using the connections.
4. flatten the tree structure to a list of sentences using the Depth First Search algorithm.

We have generated some sequences of sentences and confirmed all of them are readable and one of them is equal to the original paragraphs.

The document construction based on The Top-Down Model needs ability of identification of the top-down connection between sentences. As we have not developed this ability, we can not implement the automatic document construction.

4.2. The Context-strengthen Model

A sentence is considered the most basic unit of document pieces. However, it is too small to manage practically. As the Top-down Model needs understanding of natural language statements, currently it's difficult to make a dynamic document constructor with this model. We propose another semantic model with which a machine can sort paragraphs and create sections, subsections, paragraphs and so on.

Documents are constructed of elements of several levels, such as sections, subsections and items. Therefore, documents are considered as tree structures where elements are represented as nodes. Relations between

Table 1. Paragraphs and their concepts

subject of paragraph	concepts included in paragraph			
scanf	function	input	with format	stdio
fscanf	function	input	with format	stream
printf	function	output	with format	stdio
fprintf	function	output	with format	stream
gets	function	input	without format	stdio
fgets	function	input	without format	stream
puts	function	output	without format	stdio
fputs	function	output	without format	stream

upper level elements and lower level elements are represented as links. The meaning of "element" used here is similar to "non-terminal symbol" on natural language processing.

Each element has its own context. In this section, the word "context" has a slightly different meaning from the previous sections. Context is a constraint that limits the elements which can be put under it. Context is also considered as the ability given by a set of concepts. So, a context can be represented by a set of nouns (or noun phrases) which is basically equal to the heading of the elements. However, sometimes some of them are omitted.

We define the order relations of contexts C1, C2 which are subsets of U which consists of all the concepts involved with selected document pieces.

- *if* $C_1 = C_2 \cup \{x\}$,
  $x \in U,\ x \notin C_2$,
  then C_1 *is stronger than* C_2.

- *if* $C_1 = C_0 \cup \{x\},\ C_2 = C_0 \cup \{y\}$,
  $x \in U,\ x \notin C_0$,
  $y \in U,\ y \notin C_0$,
  and y *is an abstracted concept from* x,
  then C_1 *is stronger than* C_2.

The following rules can be found by observing real documents. If an element E2 contain another element E1, the context of E1 must be stronger than the context of E2. A leaf of the tree that is a selected document piece has context of concepts which are represented by keywords included in them.

We will show applying the Context-strengthen Model to a C programming textbook. When paragraphs which explain input-output functions of C are selected, each paragraph has a super class concept such as *function* and its attribute concepts such as *input*, *with format* and *stdio* as shown in Table 1. Then, elements of the document can be created as combination of these concepts. Connecting these elements according to the rule, tree structures can be constructed.

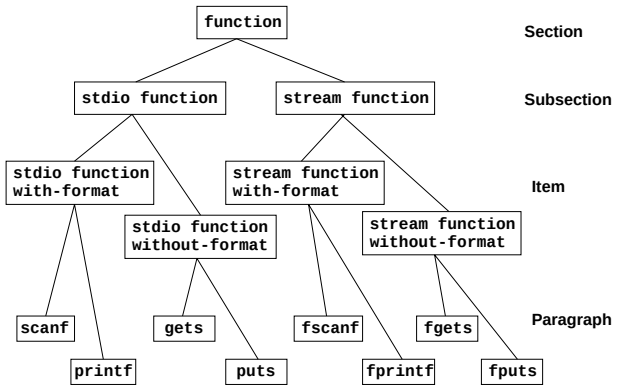

Figure 3. An example applying the Context-strengthen Model

Figure 3 shows an example of the tree structure. Rectangles used in the figure represent elements of the document and labels represent context sets.

The document has one section *function* and two subsections *stdio function* and *stream function*. These subsections have concept sets added *stdio* or *stream* to *function*. By adding concepts, context of the subsections become stronger than context of the section. The document also has four items and eight paragraphs. The elements of item level and paragraph level also have stronger concepts than their superior elements.

If some elements have concept sets including unique words, definition of abstract relations among these words are needed. We use abstraction graph that consist of abstraction links which are directed links representing abstraction connections. An abstraction connection is a connection between an instance and a class or between a sub-class and a super-class or between a concept and an attribute of it.

For example, if *converter* is substituted for {*input*, *function*, *with format*} and *formatter* is substituted for {*output*, *function*, *with format*}, we need to define connections of meaning of these words. Figure 4 shows an example of abstraction graph in that *function* is a super-class of both *converter* and *formatter*, *with format* is attribute of both, *input* is an attribute of *converter* and *output* is an attribute of *formatter*.

Figure 5 shows another example of tree structure. Both of the tree structures in Figure 3 and 5 are reasonable and documents based on them are easy to read.

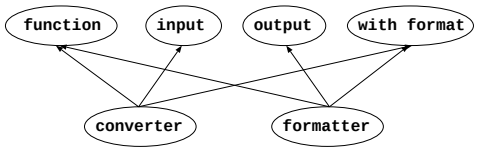

Figure 4. An abstraction graph

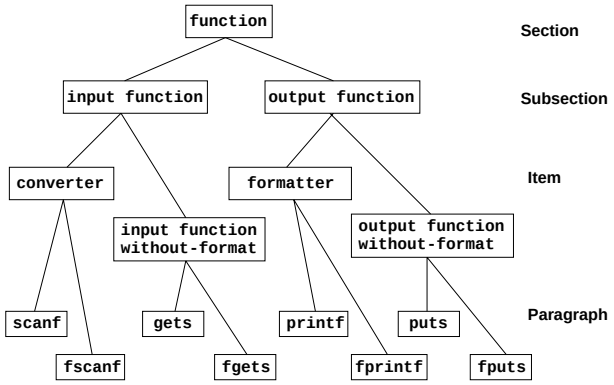

Figure 5. Another example applying the Context-strengthen Model

5. Related work

The aim of this study is computer-aided knowledge sharing. This theme has been researched by many researchers for example see [1]. However, knowledge in their research is machine readable knowledge. Therefore they are represented in logical expressions. Knowledge in this study is human readable and represented in natural language.

For sharing of human readable knowledge, digital libraries have been developed. However, they are the static document systems and have problems we pointed in section 1.

A schema for stories has studied [3]. It is also based on syntactic and semantic models. Our approach of modeling natural language text is similar to it. However, the research with stories aims to understand meaningful connections between sentences of stories. As connections between sentences of documents has functions of structuring but does not have functions of adding meaning to sentences, we can divide it into pieces. Our model is simpler and easier to implement than the model for stories.

The abstraction graph uses a technique named "weak information structure" that reduce the cost of knowledge developing by allowing knowledge representing without rigorousness. We have developed some systems using this technique [2]. Their aim is to share information among persons in a group. The weakness of information structure causes flexible knowledge developing and using by a group.

In this study, we unify two types of abstraction link: between sub-class(instance) and super-class(class) and between object and its attribute. Unified links are also weak structures which have lack of information. However, they are sufficient for the dynamic document constructor. We think the method can reduce the cost of developing abstraction graphs.

6. Conclusion

In this paper, we proposed a dynamic document system that can create a document dynamically and only contains the information which a user needs. This system will reduce the redundancy and the contradiction that a static document has. A dynamic document system consists of a common repository which contains context-free document pieces. A context-free document piece is expression in natural language which is understandable without context. We showed that context-free document pieces can be made by splitting a static document.

In order to construct a dynamic document, we proposed two semantic models. The first models is Top-down Model based on connection between two succeeding sentences. We showed a technical document can be constructed by eight categories of connection.

The second model is Context-strengthen Model which considers a document as a tree structure of sections, subsections, and so on. Connected nodes of the tree structure have relation of order of context strength. With this model, a computer can sort paragraphs and create sections, subsections and items.

We are implementing the Context-strengthen Model, and will apply it to an interactive generator of HTML textbooks which can fit various learners. Using the system, we will confirm documents generated with the model are easy to read.

References

[1] R. V. Guha. Representation of defaults in Cyc. In *Proc. AAAI-90*, pages 608–614, 1990.

[2] H. Maeda, M.Kajihara, H. Adachi, A. Sawada, H. Takeda, and T. Nishida. A system for community information sharing and its evaluation at an international conference. In *Conference on Knowledge-Based Intelligent Electronic Systems (KES'98)*, pages 405–410, 1998.

[3] D. Rumelhart. Notes on a schema for stories. In *Representation and Understanding: Studies in Cognitive Science*, pages 211–236. Academic Press, New York, 1975.