データベースシステム 129－5
（２００３．１．２３）

動的な文書構築のための文書部品ファイルの選択・組織化ツール

花川賢治† 天笠俊之‡ 波多野賢治‡ 植村俊亮‡

† 大阪府立工業高等専門学校 電子情報工学科 ‡ 奈良先端科学技術大学院大学 情報科学科

我々は，個々の読者に適合する文書をデータベースに蓄えられた多数の小さな文書部品から動的に構築する研究を行ってきた．生成される動的な文書は，読者が必要とする情報を過不足なく含むだけでなく，個々の読者にとって参照しやすい構造を持っていることが望ましい．しかしながら，妥当な文書構造を規定する制約が小さいため，多様な文書構造が生成可能であり，その中から個々の読者が望む構造をシステムが自動で選択することは困難であると考えられる．そこで，本発表では，ユーザが簡単な文字列で「どのような文書部品が必要か」だけでなく,「どのように検索結果の文書部品を構造化するか」も指定する方法を提案し，様々な分野の文書に応用可能な汎用性の高い文書部品ファイルの選択・組織化ツールを紹介する．この方法では，ユーザは述語と AND/OR 演算子から構成される論理的な表現を入力する．システムは，AND 結合を階層，OR 結合を枝分かれに展開することにより，文書の木構造を構築する．

A Document Piece File Selecting and Organizing Tool for Dynamic Document Construction

Kenji Hanakawa† Toshiyuki Amagasa‡ Kenji Hatano‡ Shunsuke Uemura‡

†Department of Electrical Engineering and Computer Science, Osaka Prefectural College of Technology

‡Graduate School of Information Science, Nara Institute of Science and Technology

We have studied an approach of dynamically constructing documents which are suitable for particular readers. Constructed documents are required not only to contain all the information needed by the readers, but also to have structures which are easy to understand for the readers. As constraints of document structures are weak, many structures can be constructed by computers and it is hard to select the document structure the user want from them. In this study, we introduce a method for specifying both what document pieces are needed and how to organize the selected document pieces into a document, and show a generalized tool used for selecting and organizing document piece files. In this method, a user enters a logical representation which consists of predicates and AND/OR operators, and the system constructs a document tree structure by translating AND conjunctions into layers and OR disjunctions into branches.

1 はじめに

現在，出版，放送，インターネットなどを使って取得できる情報量が非常に増大しているが，そのことが利益ではなく不利益として働くことが少なくない．その原因の一つとして情報流通における冗長性が考えられる．すなわち，同じ内容の情報が異なる表現で複数の場所に出現するため，みかけの情報量が増大し，我々の情報処理の負荷が増大しているのである．この問題は，インターネット上の情報流通においても顕著であるが，従来からある「本」による情報流通においても存在していた問題である．

この問題の根本的な原因は情報流通の単位が「本」のように大きいことにある．読者が必要とする情報が過不足なく記述されている本は，ある意味において理想的な本と考えることができる．しかしながら，多様な読者全員が満足するようなただ一つの理

想的な本を書くことは不可能である．そのため，多くの種類の本が著述される．すなわち著者により情報の冗長性が生産される．それにもかかわらず読者は 1 冊の本で情報要求を満足させることができないので，重複した内容を含む複数の本を参照する．その結果，読者は情報の冗長性に直面する．

我々は，情報流通のおける冗長性の問題を解決するためには，情報流通の単位をもっと小さくする必要があると考え，動的文書の概念を提案した．動的文書とは，コンピュータシステムにより，個々のユーザに合わせて文書部品から生成される文書のことを指す．我々は過去において，いくつかの動的文書システムを試作したが，システムの設計が応用に強く依存していたため，複数の応用システムを開発する上で効率が悪かった．

そこで，本研究では，動的文書システムの作成を飛躍的に効率化する汎用のツールを開発する．このツールを使って作成される動的文書システムは，自動的に文書を構築する機能は持たない．その代わりに，非常に簡単な述語という表現を入力して文書構造を指定することができる．ここでの述語とは，原始述語，AND 演算子，OR 演算子から構成される検索条件と文書構造を同時に示すことができる文字列である．また，一つの文書部品がファイルシステム上の一つのファイルに対応していることを前提とすることで，このツールに，ある種のファイルユーティリティとしての外観を持たせている．すなわち，ファイルを検索するのと同様の操作で文書部品を検索し，ファイルシステム上のディレクトリ階層を航行するのと同様の操作で，主メモリ上に構築された動的文書の木構造を航行することができる．

本研究とよく似た概念および研究は，著者が調べた範囲ではみつからなかった．少し類似した概念としては，DTD，情報検索の検索結果のクラスタリング，軸づけ検索が考えられる．DTD も文書構造を定義するという目的で使われる．しかし，DTD は文書構造を構文的，スキーマ的に定義し，本研究の文書構造の述語表現が意味論的，インスタンス的に定義するのとは対照的である．

情報の検索と組織化が統合されているという意味で，情報検索の検索結果のクラスタリング (たとえば [1]) は本研究のツールと類似した機能をもつ．しかしその組織化はシステムの自動的な処理によるもので，本研究のツールが，ユーザの意図に基づき検索結果を組織化するのとは対照的である．

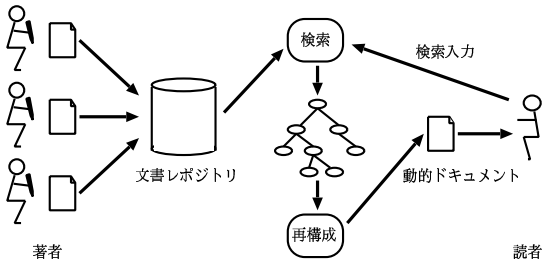


図 1: 動的文書システムの構成

金田は「軸づけ検索」と呼ばれるユニークな文書の検索と組織化の手法 [2] を提案し，電子百科事典とニュースデータベースに応用した．この手法では，検索文字列に加えてあらかじめ決められたなかから軸をひとつ選択すると，システムは検索結果である文書要素を軸に沿って並べて表示する．軸づけ検索は，ユーザの意図に基づき検索結果を組織化する点が本研究のツールと類似している．しかし，以下のような問題点がある．本研究ではそれらをクリアしている．

1. 軸が限定される．
2. 検索と組織化の処理が特定の応用システムに依存する．
3. 検索と組織化のそれぞれのために入力が必要である．

以下，2 節では本研究の先行研究である動的文書システムと文書モデルについて概要を述べる．3 節では文脈構造による文書構造を述語を使って表現する方法について述べる．4 節では述語による表現から文書構造を構築する処理について述べる．5 節では本ツールの設計と実装について述べる．6 節では本ツールを使った応用について述べ，最後に 7 節でまとめを述べる．

2　動的文書システムと文書モデル

ここでは，これまで研究してきた動的文書システムと動的文書を構築するのに用いる文書モデルについて述べる．

動的文書システムは，ネットワーク上で共有される文書レポジトリに自然言語で記述した情報の断

片である文書部品を多数格納し，ユーザから要求があったときに，必要な文書部品を取出し一つの文書を構成し提示する (図 1)．文書レポジトリに格納される文書部品は多くの専門家が真であることを合意したものに限り，冗長性がないように管理される．

動的文書システムにおける文書部品には，文脈非依存性が要求される．ここでの文脈非依存性とは，外部環境とは無関係に必ず記述の解釈が一意であることを指す．情報が普遍的，客観的な事実であれば，文書部品を文脈非依存にすることができると考えられる．[3] では実際に人間が書いた文書をパラグラフ単位で分解し，省略と指示照応に関係した部分を補足するという軽微な修正で文脈非依存な記述に書き換えることができることを示した．

読者は有用な文書部品を文書レポジトリから取り出して読むが，ばらばらの文書部品を読むことは苦痛を伴うので，人間が書く文書に近い文書を文書部品から構築する機能が必要になる．コンピュータに文書部品を人間にとって読みやすくなるように構成させるためには，実際に人間が書いた文書に内在する構造をモデル化したものに従わせることが必要になる．そこで，文書モデルとして代表的と考えられる以下の 3 種類について調べた [4].

1. 概念構造
   文書の構造が概念間の意味的な関係を表現する．すなわち，文書の要素 (章，節，パラグラフなど) に対応する概念間の関係と文書の要素間の関係が一致するタイプの文書にみられる構造である．たとえば，多くの植物図鑑は種の系統樹と同じ構造を持つ.

2. 計算構造
   計算構造は，具体的な問題解決を行うときの構造である．たとえば旅行ガイドブックでは，推奨旅行プランが示されていて，それにそって観光地の情報が示されていることが多い．あるいは，教科書には例題が載っていて，解決のプロセスにそって既に学習した知識が説明されることが多い．計算構造に従い文書を構成することにより，コンピュータの持つ問題解決能力と情報提示の能力を統合でき，利用者は問題解決と同時に有用な情報を取得することができる.

3. 文脈構造
   文書要素が解釈可能か否かおよび解釈の内容は，文書中の置かれる場所 (環境) に依存する．その文書要素が置かれた場所が持っている情報が文脈であるとする．文書要素のそれぞれは文脈を持ち，その文脈は文書要素の中身を読んで理解するのに使われると考える．それと同時に，文脈は文書要素が置かれるときの制約としての働きも持つ．つまり，ある文脈を持っている文書要素に置くことのできる文書部品は，それの解釈にその文脈を使用するものに限られる．二つの文脈があったときに，一方に置くことができる文書部品の集合と他方に置くことができる文書部品の集合の間に包含関係が成立するときには，両者の間には文脈の強弱の関係が存在する．文書の構造のすべての親子関係のあるノード間において，親ノードが持つ文脈は子ノードが持つ文脈よりも強いという条件が保たれている．文書部品の集合が与えられたとき，そこから動的文書を構成するためには，この制約を満足する木構造を生成すればよい.

ただし，すべての文書が 3 種類のいずれかの構造を持つのではない．これら以外の構造を持つ文書も存在すると考えられる．また，複数の構造が混在した文書もあると考えられる．本研究では，文脈構造に焦点を置く．文脈構造が現れる代表的な文書としては，旅行やレストランのガイドブック，パンフレット，商品カタログなどがある.

3 述語による文脈構造の表現

ここでは，文脈構造に従う文書構造を原始述語と AND/OR 演算子を用いて論理的に表現するアプローチについて述べる.

一般に，与えられた文書部品から生成できる文書構造は一意ではなく，むしろ人間が妥当とみなす文書構造は多数考えられる．読者はそれらの間で優劣を評価することが可能である．また，その評価は読者と利用状況に強く依存すると考えられる．したがって，特定のユーザが望ましいと評価する文書構造をコンピュータが自動的に生成することは容易でないと考えられる．そこで，本研究では，ユーザが非常に小さな労力で文書構造を表現し，それに従いコンピュータが動的文書を構築する方法を開発した．この方法では，文書構造の表現が文書部品の検索条

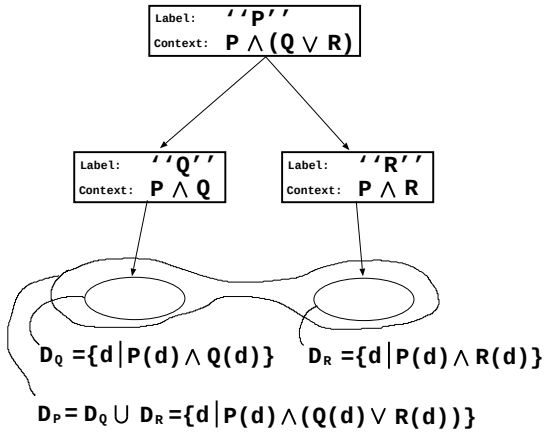


図 2: 文脈構造を満足する文書構造の例

件の表現と統合されているので，ユーザの入力に要する負荷は，単に検索条件だけを入力するばあいよりも少ししか増えないと考えられる．

文脈構造では，文書構造を構成するノードは，それぞれが格納する文書部品の条件である文脈を持つと考える．文脈は文書部品を与えたときに真か偽を返す述語で表現することができる．

ここでは以下のものを述語とする．この表現は，否定あるいは含意の演算子は含まない．

1. 論理演算子を含まない基本となる原始述語
   文書部品与えると真か偽を返す．
2. 二つの述語を AND 演算子で結合したもの
   両方の述語が真を返すとき真を返す．
3. 二つの述語を OR 演算子で結合したもの
   一方でも真を返すと真を返す．

ここで，複合的な述語が与えられたときに文脈構造を満たすような木構造を構成することについて考える．述語 P, Q, R があり，文書部品の検索条件として $P \wedge (Q \vee R)$ が与えられたとする．このとき，文脈構造を満たすような木構造として，文脈 $P \wedge (Q \vee R)$，$P \wedge Q$，$P \wedge R$ から構成される木構造が考えられる (図 2)．子ノードは，それぞれが文書部品の集合として D_Q と D_R を持つ．親ノードは，間接的に $D_Q \cup D_R$ を持つ．

ノードには文脈を表わすラベルを付加するが，実際の文脈を表わすと冗長になるので，簡潔に表わすため，子ノードのラベルからは親ノードのラベルと重複する部分を除き，親ノードのラベルからは子ノードのラベルと重複する部分を除くようにする．

ノードは子ノードを持つノードと文書部品を持つノードのどちらかで，子ノードと文書部品の両方を持つノードは存在しない．また，根ノードが持つ文脈は，ユーザが検索条件として与えた述語と一致する．

ここでは，単純な例を用いて説明したが，一般的に，与えられた述語の構造を調べて，AND 演算子を階層，OR 演算子を枝分かれに展開することで文脈構造を満足する文書構造が構築できると考えられる．

4 述語による表現から文書の木構造を構成する処理

次に，述語による表現から文書の木構造を構成する処理について述べる．この処理は，以下の 4 ステップから成る．

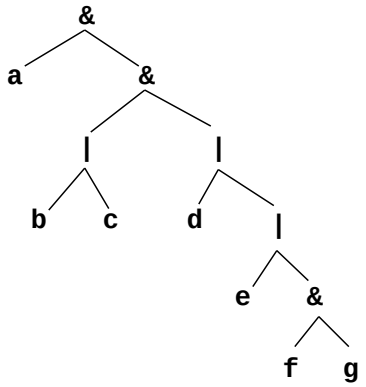
(1) 述語を表わす文字列から構文木を生成する．(2) 構文木から文書木を生成する．(3) 文書木の葉が持つ文脈にマッチする文書部品を検索する．(4) 文書部品を持たないノードを削除する．

最初に実行する述語表わす文字列から構文木を生成する処理は，一般的なプログラミング言語の構文解析と同様の処理である．本研究での述語は AND 演算子 (&) と OR 演算子 (|) と基本述語を表わす文字列から構成され，BNF で構文を表わすと以下のようになる．

```
predicate ::= primitive_predicate
        | predicate '&' predicate
        | predicate '|' predicate
        | ( predicate )
```

その次に，構文木から文書木を生成する．文書木のノードのデータ構造は，2 個のリストから構成される．ひとつは述語を格納する文脈リスト，もうひとつは部分木のポインタを格納する部分木リストである．

構文木から文書木を生成する手続きは，構文木を深さ優先探索するのと同時に再帰的に呼び出す．この手続きは，訪問した構文木のノードの種類により以下に示す 3 種類があり，いずれも引数として解析木の部分木が与えられ，文書木の部分木を返す．



a&(b|c)&(d|e|f&g)

図 3: 述語の構文解析の例

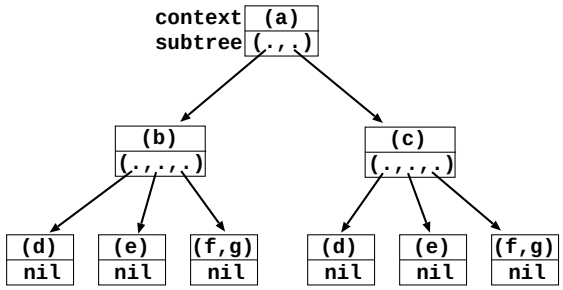


図 4: 文書木構築の例

1. 訪問したノードが primitive_predicate のばあい
   (a) 新しい文書木のノードを生成する.
   (b) 新しく生成したノードの文脈リストに primitive_predicate を入れる.
   (c) 新しく生成したノードの部分木リストに nil(要素が空であることを意味する) を入れる.
2. 訪問したノードが OR 演算子 (|) のばあい
   (a) 新しい文書木のノードを生成する.
   (b) この手続きを左側のオペランドについて実行する.
   (c) この手続きを右側のオペランドについて実行する.
   (d) 新しく生成したノードの文脈リストに nil を入れる.
   (e) 新しく生成したノードの部分木リストに表 1 に従い要素をセットする.
3. 訪問したノードが AND 演算子 (&) のばあい
   (a) この手続きを左側のオペランドについて実行する.
   (b) この手続きを右側のオペランドについて実行する.
   (c) 左側のオペランドから生成された部分木のすべての葉ノードの部分木リストに表 2 に従い要素をセットする.

表 1, 2 で使われている単語の説明を以下に示す.

left (right): 左 (右) オペランドから生成された文書木
leaf: 文脈リストが nil でなく部分木リストが nil である文書木ノード
with_context: 文脈リストが nil でなく部分木リストも nil でない文書木ノード
without_context: 文脈リストが nil であり部分木リストが nil でない文書木ノード
concatenate: 二つのリストを引数に与えるとそれらを連結した一つのリストを返す関数
push: 第一引数で与えられたリストの後尾に第二引数で与えられた要素を追加する関数
unshift: 第一引数で与えられたリストの先頭に第二引数で与えられた要素を挿入する関数
list: 引数で与えられた要素を格納したリストを返す関数
leaf_of_left: 左オペランドから生成された文書木の葉
context: 文脈リスト
subtree: 部分木リスト

構文解析の例と文書木の構築の例を図 3 と図 4 に示す.

上に述べた処理で生成された文書木のすべての葉について, それぞれの葉が持つ文脈を満たす文書部品を検索する. 検索でみつかった文書部品はそれぞれの葉に格納する.

検索を行った結果, 1 件も該当する文書部品がみつからなかった葉は文書木から削除する. さらに, 葉を持たないノード, 子ノードを持たないノードも文書木から削除する.

5 文書部品ファイルの選択・組織化ツールの設計と実装

5.1 原始述語

本システムの原始述語の構文と意味を表 3 に示す.

表 3 の中で使った比較演算子, 文字列照合演算子, サイズ表記, 日時表記は以下のような文字列である.

比較演算子: “==”,“<”,“>”,“<=”,“>=” のいずれか.

文字列照合演算子: “==” または “=~”. 後者を用いた

表 1: OR ノードから部分木リストを生成する方法

左オペランドから生成された文書木のタイプ	右オペランドから生成された文書木のタイプ	新しく生成したノードの部分木リスト
without_context	without_context	concatenate (left.subtree, right.subtree)
without_context	leaf or with_context	push (left.subtree, right)
leaf or with_context	without_context	unshift (right.subtree, left)
leaf or with_context	leaf or with_context	list (left, right)

表 2: AND ノードから文脈リストと部分木リストを生成する方法

右オペランドから生成された文書木のタイプ	新しく生成したノードの文脈リスト	新しく生成したノードの部分木リスト
leaf	push (leaf_of_left.context, right.context)	nil
without_context	leaf_of_left.context	right.subtree
with_context	push (leaf_of_left.context, right.context)	right.subtree

ときには正規表現の照合になる.
サイズ表記: 数字の列の後にキロバイトを意味する k，メガバイトを意味する m，バイトを意味する c のどれかを付けた文字列.
日時表記: 時刻を表わす年から始まる文字列. または，月，日，時，分を表わす文字列.

5.2 システム構成

本ツールのシステムの構成を図 5 に示す. 各処理について今回採用した実装方法を簡単に述べる. 今回の実装には Java と Swing を用いた.

述語の入力　述語自体は文字列なので，テキストエディタの機能を用い入力することができるが，入力を容易にするために，JButton，JList，JTextField，JDialog 等を使い，GUI による原始述語の入力機能を持たせた.

述語の構文解析　論理式の構文解析には，Carnegie Mellon University の Human-Computer Interaction Institute に所属する Scott E. Hudson が開発した CUP Parser Generator for Java を用いた. このツールは，JAVA で書かれた YACC で，テキストで書かれた構文規則を読み込むと JAVA で書かれた構文解析プログラムを出力する.

字句解析には，Princeton University の Elliot Berk が開発した JLex: A Lexical Analyzer Generator for Java(TM) を用いた. このツールは入力ストリームから，文字列を読み込み，終端記号を出力するプログラムである.

文書木の生成　前節で述べた処理を行う.

ファイル検索　ファイル検索は，指定されたディレクトリ下にあるすべてのファイルを深さ優先探索により得る. この処理は UNIX の find コマンドの処理とよく似ている. 次にファイルそれぞれについて文書木の葉の条件との照合を行い，成功したばあいに葉が持つリストに照合に成功したファイルのパスを追加する.

文書木の表示　マイクロソフト Windows のエクスプローラのように木構造の階層が一つのウインドウに表示される形式と，マッキントシュのファインダのようにディレクトリが一つのウインドウに対応する形式の 2 種類の文書木の表示機能がある. 前者には JTree，後者には JFrame を用いた.

6 応用

本ツールが実際に有用であると思われる 2 種類の応用例について調べた.

表 3: 原始述語

述語の種類	構文	意味
サイズ	size 比較演算子 サイズ表記	ファイルのサイズとサイズ表記を比較する
名前	name 文字列照合演算子 文字列	ファイル名と文字列を照合する
パス	path 文字列照合演算子 文字列	ファイルパスと文字列を照合する
作成日時	time 比較演算子 日時表記	ファイルの作成日時と日時表記を比較する
部分文字列	grep 文字列	ファイルに文字列 (正規表現) が含まれていると真
種類	type 文字列照合演算子 文字列	UNIX の file コマンドの出力と同様の種類名と文字列を照合する
コメント	comment 文字列照合演算子 文字列	画像，音声ファイルなどに含まれるコメントテキストと文字列を照合する
Perl スクリプト	Perl スクリプト	ファイルのパスを変数$_にセットしてスクリプトを実行し，最後の評価値が 0 または空文字列でなければ真を返す

メイルの分類システム　このシステムは inbox ディレクトリにあるメッセージファイルを検索する．それぞれのメッセージファイルは，“Subject:”，“From:”，“Date:” などで始まる行を含むヘッダとボディから構成される．

メイルの分類には部分文字列照合の原始述語が有効である．たとえば，11 月と 12 月に taro と hanako から送られてきたメイルを検索し，4 種類の組み合わせで分類したいばあいには，以下の述語を入力するとよい．

```
(grep '^Date:.*Nov'|grep '^Date:.*Dec')&
(grep '^From:.*taro@'|grep '^From:.*hanako@')
```

年と月の階層でメッセージファイルを分類したいときには，以下の述語を入力するとよい．

```
(grep '^Date:.*1999'|grep '^Date:.*2000') &
(grep '^Date:.*Jan'|grep '^Date:.*Feb'| ...
|grep '^Date:.*Nov'|grep '^Date:.*Dec')
```

この入力は煩雑であるが，上の述語の後ろの部分をたとえば MONTH というような名前で登録しておき，ユーザが MONTH を入力すると自動的に 12 個の月名から成る OR 節に置換するようなマクロ機能を用意すると入力の手間は小さくなる．

複数のフォルダを作成して普段からメッセージファイルを分類して入れておく方法と比較して，このシステムは，普段の手動による分類が不要で，利用時の目的に応じて様々な分類ができることにメリットがあると思われる．

オンラインレストランガイド　このシステムは，http://gourmet.yahoo.co.jp/gourmet/restaurant にあるようなレストランごとの HTML ファイルを検索する．HTML ファイルは統一された形式を持ち，それぞれのファイルは以下の項目を含む．

- レストランの名前
- 住所
- 最寄りの地下鉄の駅
- レストランの紹介記事
- 広い料理の種類 (洋食，和食，エスニック...)
- 狭い料理の種類 (フランス料理，イタリア料理，寿司...)
- 電話番号

レストランの HTML ファイルはメイルのメッセージファイルよりも以下の二つの理由で，本システムで直接利用するのに困難がある．まず，このような HTML ファイルには自明である情報が抜けている場合がある．レストランを紹介した HTML ファイルのばあい，料理の分類が記述されていないことがある．たとえば，イタリアレストランの HTML ファイルに「イタリア」という文字列がどこにも現れないことがある．もう一つの問題として，メイルのメッセージファイルのヘッダのように特定の文字列 (たとえば Subject:) で始まる規則がないので，文字列が意図とは異なる場所でマッチしてしまう可能性がある．そのためメイルの場合よりも検索の厳密性が低くなる．HTML のタグを利用して厳密な照合をすることは可能であるが，述語がかなり複雑になる．前者の問題は深刻なので，料理の種類の情報を個々のファイルに挿入する必要がある．

この応用でも部分文字列照合の原始述語が有効である．大阪の北区と中央区のフランス料理とイタリ

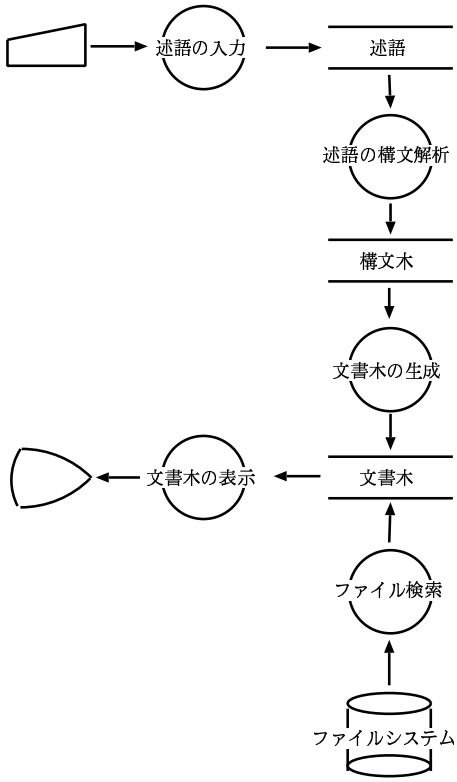


図 5: ツールのシステム構成

ア料理のレストランを検索するばあいには，以下の述語を入力するとよい．

```
(grep '大阪市北区'|grep '大阪市中央区')&
(grep 'フランス料理'|grep 'イタリア料理')
```

上の例では，既存のテキストファイルをそのまま文書部品として利用したが，動的に文書を構築する目的で，よりファイルの粒度を小さくすることで，より詳細に必要な情報と組織化の方法を指定することができる．レストランについての文書部品としては，店内を撮影した写真，地図，メニューなどを独立した文書部品ファイルとすることが考えられる．さらに，深さ優先探索により文書構造に置かれた文書部品ファイルを 1 次元化し，HTML のような形式の一つのファイルに変換してブラウザで閲覧したり，印刷する機能が有用である．

7 おわりに

動的文書システムを効率良く開発することを目的とした文書部品ファイルの選択・組織化ツールを紹介した．

動的文書の構造をコンピュータに組み込まれた処理により自動構築することが困難であるので，本研究では，動的文書の構造をユーザの意図に基づき構築するアプローチを採用した．そのため，ユーザは文書部品の検索条件と検索結果の文書部品から生成される動的文書の構造の両方を指定する必要性が生じたが，その負担を低減する工夫として，述語の論理的な構造により動的文書の構造を表現する方法を提案した．また，ツールの汎用性を高めるために，一つの文書部品を一つのファイルに記述することを前提とすることで，文書部品の検索がファイルの検索の機能で実現できるようにした．その結果，応用に依存したコンパイラ言語を使った作業が必要なくなり，動的文書システムの開発の効率は飛躍的に向上すると期待できる．

ツールに追加すべき機能としては，述語に名前をつけて定義するマクロ機能と文書木から閲覧，印刷に適した 1 次元のファイルを生成する機能がある．これらの機能の実装を早急に行いたい．

今後は，このツールを使って多くの応用システムを開発し，実際に利用することで，動的文書の有用性を確認していきたい．

謝辞

本研究の一部は，文部科学省科学技術研究費補助金（課題番号: 12680417, 13780236, 14019064, 14780325），ならびに科学技術振興事業団 (JST) の戦略的基礎研究推進事業 (CREST)「高度メディア社会の生活情報技術」プログラムの支援によるものである．ここに記して謝意を示す．

参考文献

[1] http://vivisimo.com (2002).

[2] Y. Kanada, Axis-specified Search: A Fine-grained Full-text Search Method for Gathering and Structuring Excerpts, Proc. 3rd ACM Conf. on digital Libraries, 108–117 (1998).

[3] K. Hanakawa, H. Takeda, T. Nishida, Construction of a Dynamic Document Using Context-Free Pieces, Proc. SEKE'99, 212–216 (1999).

[4] K. Hanakawa: Interactive Construction of Textbooks, International Conference on Information Society in the 21st Century, pp.584–588 (2000).