A method of constructing a retrieval environment for language resources using full-text retrieval system "Himawari"

『日本語科学』21(2007年4月) 111-123 [研究ノート]

全文検索システム『ひまわり』を利用した言語資料検索環境の構築手法

山口 昌也
(国立国語研究所)

キーワード
全文検索システム，言語資料，XML

要 旨

現在，新聞・小説などのテキストデータベースや言語研究用に構築されたコーパスなどの言語資料が利用できるようになっている。しかし，言語資料を検索・閲覧するための手段が提供されることは少なく，言語資料が有効に活用されていないという問題がある。本稿の目的は，言語資料を有効に活用するため，全文検索システム『ひまわり』を用いて，言語資料の検索環境を構築する方法を示すことである。特に，検索環境構築時の実際的な事柄（文字コードなど）にも配慮し，既存の言語資料をどのような形式に整形すれば，どのような検索環境が構築できるのかを，実例に基づいて説明する。本稿では，まず，『ひまわり』の機能概要，および，検索能力を説明したのち，それに基づいて，(1)生テキストに近い言語資料，(2)形態素情報が付与された言語資料，(3)画像データと関連づけられた言語資料，の3種類の言語資料に対する検索環境を構築する。

1. はじめに

近年，計算機で処理することのできる，電子化された言語資料が増加している。90年代から徐々に利用が可能になりはじめた新聞，小説などのテキストデータベース（例：毎日新聞テキストデータベース（毎日新聞社），「青空文庫」[1]）に加えて，言語研究用に作成されたコーパス（例：『太陽コーパス』（田中他 2005），「日本語話し言葉コーパス」（前川 2004），「BTSによる多言語話し言葉コーパス」（宇佐美 2005）など）が次々公開されるようになってきた。さらに，組織的に構築されるだけでなく，個人の研究者が自分の作成した言語資料をWeb上で公開する試み[2]も行われるなど，より広い範囲で言語資料の構築が行われている。その一方で，言語資料を研究用に利用するための手段が同時に提供されることは少なく，言語資料が有効に活用されていないという問題がある。

そこで，本稿では，言語資料の有効活用のために，筆者らが開発した全文検索システム『ひまわり』を利用して，言語資料の検索環境を構築する方法を示す。特に，検索環境構築時の実際的な事柄（文字コードなど）にも配慮し，既存の言語資料をどのような形式に整形すれば，どのような検索環境が構築できるのかを実例に基づいて説明する。実例として取り上げる言語資料は，次の三つである。

基　本：生テキストに近い言語資料

応用１：形態素情報が付与された言語資料

応用２：画像データと関連づけられた言語資料

本稿の構成は，次のようになっている。まず２節で，全文検索システム『ひまわり』の特徴，検索能力について説明する。次に，３節では，検索環境を構築する一般的な手順について説明した後，上記三つの言語資料の検索環境を構築する。最後に，４節でまとめを行う。なお，『ひまわり』のシステム構成や検索アルゴリズムなど技術的な側面については，本稿では簡単に言及するにとどめる。詳細については，山口・田中(2005)を参照していただきたい。

2. 全文検索システム『ひまわり』の機能

2.1. 概要

『ひまわり』は，言語研究用に設計された全文検索システムである。『ひまわり』は，「言語研究用」ということに利用目的を定め，コンピュータに不慣れな利用者でも，さまざまな種類の言語資料を容易に検索・閲覧できるように設計されている。

図１は，芥川龍之介の「蜘蛛の糸」と「猿蟹合戦」(「青空文庫」に収録されているテキストを変換して利用）を対象に，文字列「から」を全文検索した例である。検索結果は，表形式で表示され，１行が一つの検索結果である（図１では，総計37個の検索結果のうち，25番目から37番目を表示している)。検索結果には，前後文脈，著者，タイトルなど，検索文字列に付随するさまざまな情報を同時に表示することができる。図１を用いつつ，『ひまわり』の特徴を箇条書きする。

- 多様な形式の文書に対する全文検索
  - 言語資料には，新聞，小説，談話資料，辞書などさまざまな種類があり，個々に記述形式を定義する必要がある。『ひまわり』は，XMLで記述された言語資料を全文検索することができるため，さまざまな文書形式の言語資料に対応することができる。例えば，『ひまわり』は，記述形式の大きく異なる，総合雑誌コーパスの『太陽コーパス』と類語辞典の『分類語彙表』の両方を検索対象とすることができる。
  - 全文検索だけでなく，全文検索された文字列にタグづけされている言語研究用の「付随情報」(例：図１の検索結果中の「タイトル」や「著者」）を抽出するとともに，それらの情報を検索結果の絞り込みに利用することができる（例：「タイトル」が「蜘蛛の糸」の結果だけに絞り込む)。
- 言語研究に適した形式での検索結果の表示
  - 従来から言語研究で用いられてきたKWIC形式で検索結果を表示することができる(図１)。また，検索結果の各列ごとに，ソート，絞り込み条件を指定できる。
  - 言語資料中の任意の要素をHTMLブラウザに表示することが可能である（図１の「蜘蛛の糸」全文表示がその例である)。これにより，KWICよりも広範囲な文脈の表示や原資料に近い体裁での表示が可能である。

- 利用の簡便さ
  - GUIを用いることにより，言語資料の内部構造に関する知識を意識しないで検索条件を指定することができる。
  - Windows, Linux, Macなど多くの環境で動作する[3]。また，無料で利用できる[4]。

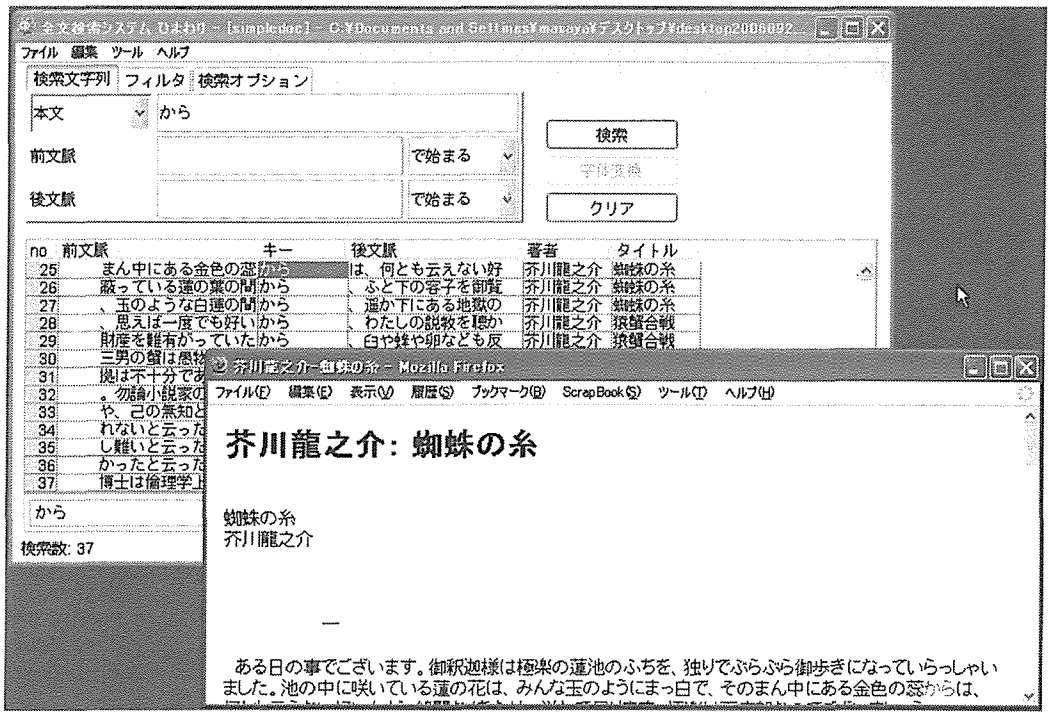


図1 『ひまわり』の検索実行例

2.2.『ひまわり』の検索機能

『ひまわり』は，XML文書を対象として，指定された要素の要素内容（XMLのタグでマークアップされている内容），および，要素属性に対して，全文検索を行うことができる。索引づけとしてSuffix Array方式（山下 2000）を用いることにより，高速な全文検索を実現している。この後の3節では，『ひまわり』の全文検索機能と全文検索結果に付随する情報の検索機能，さらに，言語資料の閲覧機能について説明する。

2.2.1. 全文検索機能

まず，要素内容を全文検索する能力について示すことにする。ここでは，図2を検索対象のXML文書とする[5]。このXML文書では，作品全体をarticle要素，本文をbody要素，コメントをcomment要素として記述している。『ひまわり』は，全文検索対象の要素を指定することができるので，body要素を全文検索対象とし，検索文字列を「蜘蛛」とすると，body要素冒頭の

「蜘蛛」(L1)だけが検索結果となり，comment要素に含まれる「蜘蛛」を検索結果から除外できる。

なお，文字列の照合時には，XMLのタグは無視される。したがって，図2中のL1の「蜘蛛」は，ルビを表すrubyタグで囲われているが，「蜘蛛の糸」を検索文字列として検索を行っても，正しく検索される。

```
<article title="蜘蛛の糸" author="芥川龍之介">
<body>
<ruby t="くも">蜘蛛</ruby>の糸　....（L1）
芥川龍之介

ある日の事でございます。御釈迦様は極楽の蓮池のふちを，独りでぶらぶら御歩きになって...
</body>
<comment>
この文章は，芥川龍之介の「<ruby t="くものいと">蜘蛛の糸</ruby>」から引用 ...（L2）
</comment></article>
```

図2　XML文書の例（芥川龍之介の「蜘蛛の糸」から引用）

次に，要素属性に対する全文検索について見てみよう。検索する際には，検索対象の要素と属性を指定する。例えば，図2のXML文書のルビに対して全文検索を行いたい場合は，ruby要素のt属性に対して全文検索を行えばよい。要素内容と同様に，指定された要素属性の任意の文字列を検索することができるが，文字列の照合は，指定した属性内だけで行われる。照合に成功した場合は，照合に成功した属性値だけでなく，要素内容も検索結果として返す。例えば，ruby要素のt属性に対して，検索文字列「くも」を全文検索すると，L1の「蜘蛛」とL2の「蜘蛛の糸」が検索結果として得られる[6]。

2.2.2. 付随情報の検索機能

『ひまわり』は全文検索で照合に成功した場合，その文字列（以後，結果文字列）に付随する情報を検索することができる。具体的には，次の要素と属性である。

- 前後文脈：結果文字列の前後文脈を抽出することができる。この機能により，KWICの表示を実現している。抽出する前後文脈の長さはユーザが指定できる。
- 要素の属性：結果文字列をマークアップしているタグの属性を取得することができる。例えば，図2で「蜘蛛」を全文検索し，L1の「蜘蛛」が検索されたとき，この「蜘蛛」を要素内容として持つrubyタグとarticleタグの属性を取得することができる。これにより，「蜘蛛」に付与されているルビや「蜘蛛」が出現した資料のタイトル，著者を抽出できる。
- 親要素：結果文字列が含まれる親要素全体を検索することができる。例えば，L1の「蜘

蛛」の例で言えば，親要素である body 要素を取り出すことができる。書物全体を閲覧する機能は，この機能を用いて実現されている。

- 前後の要素：結果文字列の親要素と同名の要素の前後要素を検索することができる。この検索機能は，結果文字列に対する前後の形態素を検索する場合などに有効である。

2.2.3. 検索結果の閲覧

『ひまわり』では，検索結果の閲覧は，(1)KWIC による閲覧，(2)外部プログラムによる閲覧，といった 2 種類の方法を用いることができる。方法(1)は，図 1 で示したとおりである。

方法(2)は，検索結果を外部プログラムに渡して，閲覧する方法である。外部プログラムに渡すことができる情報は，(a)検索結果の任意のフィールド，もしくは，(b)検索文字列の任意の親要素である。(a)を用いることにより，著者データベースを閲覧したり，音声や画像などのテキストデータ以外のデータを閲覧することも可能である。また，(b)を利用することにより，1 作品全体をブラウザで閲覧することなどが可能になる。特に，ブラウザを利用した閲覧方法では，XSLT スタイルシートを適用することにより，言語資料を原資料の紙面に近い形に変換するなど，好みの形式で言語資料を閲覧することができる。

3. 言語資料検索環境の作成

本節では，2 節で示した『ひまわり』の検索能力を利用して，言語資料検索環境を作成する方法を実例とともに示す。まず，一般的な言語資料検索環境の構築手順について述べた後，三種類の言語資料検索環境の作成方法を説明する。

3.1. 一般的な手順

『ひまわり』を用いた言語資料検索環境を構築するには，次の三つのことをする必要がある。

(1) 既存の言語資料の整形（『ひまわり』用 XML データの作成）
(2) 閲覧用スタイルシートの作成
(3)『ひまわり』の設定

本稿では，(1)に焦点を当てて説明する。(2)の「閲覧用スタイルシートの作成」は，図 1 で示したように，言語資料をブラウザで表示する際の表示形式を定めるものである。スタイルシートの作成は，XSL と CSS に基づいた一般的な XML 関連技術なので，その説明は，専門の書物にゆずる。また，(3)の「『ひまわり』の設定」については，『ひまわり』ホームページで公開している資料を参照していただきたい。

「既存の言語資料の整形」の手順は，以下のとおりである。すでに述べたように，『ひまわり』は XML 形式の言語資料を検索対象とするが，個人の研究者が人手でマークアップすることを考慮して説明する。具体的には，直接 XML 形式で記述するのではなく，容易に入力できる独自タグで研究用の情報を付与し，最後に XML 形式に変換する。

- 言語資料の電子化

- 付加情報の記述
- 『ひまわり』用 XML データへの変換

3.1.1. 言語資料の電子化

ここでは，まず，『ひまわり』用に言語資料を整形する際の要件について述べることにする。『ひまわり』用の電子化ファイルは，文字コードとして，次の二つの条件を満たす必要がある。

- 文字符号化方式： UTF-16 （Little Endian, Byte Order Mark 付き）
- 改行文字： ラインフィード(LF)[7]

UTF-16は Unicode の文字集合を格納することができる。したがって，現在のところ，実用上，最も多くの文字を扱える文字符号化方式の一つである。古典語の言語資料のように，Unicode の文字集合に含まれない文字が資料内に存在する場合は，一定の包摂基準を用いて Unicode の文字集合にマッピングするか，外字として扱う（例えば，＝に置き換え，注釈を加えるなど。『太陽コーパス』では，外字専用のタグを設けている（田中他 2005））必要がある。

3.1.2. 付加情報の記述

言語資料には，多くの場合，本文だけが含まれているのではなく，研究に役立つさまざまな情報が付加（以後，「付加情報」）されている。例えば，書誌情報や，誤字・脱字に対する言語資料作成者による注記などである。図1の「タイトル」「著者」は付加情報を抽出した結果である。

本文に対する付加情報は，XML のタグか，独自の形式の簡易タグで記述する。いずれの形式でも，付与する場合は，次の二つの要件を満たす必要がある。

(1) 付加情報と本文とを明確に区別して記述すること

(2) 検索機能に適合した記述を行うこと

要件(1)は，『ひまわり』の仕様上，全文検索の対象となる本文は一つであることから，生じるものである。例えば，原資料の本文に対する修正情報を付加する場合，修正前・後，いずれかの文字列を本文とし，他方を本文に対するタグの属性として記述する。次の例は，修正後の文字列を本文として記述した例である。

コーパス<注記 種類="本文修正"原文="が">を</注記>作成する

簡易タグを使用して付加情報を記述する場合は，タグの解釈に曖昧性が発生しない方法で記述することが重要である。次の例は，カッコでルビを記述した例であるが，二つの問題がある。

昨日蜘蛛（くも）の話を聞いた。

一つは，ルビがどこまでかかるか明示されておらず，「蜘蛛」だけにルビがつくのか，「昨日蜘蛛」まで付くのか，単純な機械処理では判断できないことである。もう一つは，カッコが本文で使われうる文字である場合，それがタグを表すのか，本文を表すのかが判断できなくなることである。よく行われている解決法としては，「昨日/蜘蛛(くも)」のように，(1)「/」でルビの範囲を明確にし，(2)タグを ASCII 文字（いわゆる「半角文字」），本文を非 ASCII 文字（いわゆる「全角文字」）として，付加情報であることを明確にする，という方法がある。

一方，要件(2)は，2.2.2節で示した付随要素の検索機能に対応した形式で付加情報を記述するということである。次の例は，本文中のタイトル部分にtitleタグを付与した例である。この例では，articleタグのtitle属性は，本来，冗長である。しかし，『ひまわり』の検索機能では，article要素の要素内容のtitle要素を取得できないので，「要素の属性」（2.2.2節参照）の取得機能を利用するために，article要素属性としてもタイトルを記述しておく。

```
<article title="蜘蛛の糸">
<title> 蜘蛛の糸 </title>
```

3.1.3.『ひまわり』用ＸＭＬデータへの変換

前節で示したような独自タグや『ひまわり』の検索機能に適合したXMLデータを作成するには，形式の変換処理を行う必要がある。独自形式のタグについては，Perlなどのスクリプト言語を用いて変換するのが一般的であろう。また，すでにXML形式で記述してあるデータについては，XSLTを利用すれば，容易に『ひまわり』用のXML文書に変換することができる[8]。

なお，『ひまわり』用データ作成支援ツール『えだまめ』を用いれば，著者名，資料名，ルビ，注記など，言語資料としての基本的な情報を付与した『ひまわり』用XMLデータをXMLの知識なしで作成することができる。また，専用の閲覧用のスタイルシートや『ひまわり』の設定ファイルが同梱されているので，自分で作成する必要はない[9]。

3.2. 生テキストに近い言語資料の場合

3.2.1. 概略

本節では，『ひまわり』用のXMLデータの基本的な構造を示すために，付与されるタグが少なく，生のテキストに近い言語資料の場合を説明する。ここでは，図3（芥川龍之介の「蜘蛛の糸」から引用）を基に説明する。

```
<article title="蜘蛛の糸" author="芥川龍之介">
<ruby t="くも">蜘蛛</ruby>の糸<l no="1" />
芥川龍之介<l no="2" />

ある日の事でございます。御釈迦様は極楽の蓮池のふちを、独りでぶらぶら<l no="3" />御歩きになっていらっしゃいました。池の中に咲いている蓮の花は、みんな玉<l no="4" />のようにまっ白で、そのまん中にある金色の蕊からは、何とも云えない好い匂<l no="5" />が、絶間なくあたりへ溢れて居ります。極楽は丁度朝なのでございましょう。<l no="6" />
やがて御釈迦様はその池のふちに御佇みになって、水の面を蔽っている蓮の<l no="7" />葉の間から、ふと下の容子を御覧<p no="1" />になりました。この極楽の蓮池の下は、丁度<l no="8" />
</article>
```

図3　XML文書の例

図3の例には，図2で説明したarticleタグ，rubyタグの他に，行とページの情報を記述するために次の二つのタグを付与している[10]。

p要素：ページ区切りを表す要素である。この要素は，範囲を持たない空要素として記述される。ページ番号は，no属性で記述する。

l要素：行区切りを表す要素である。この要素もp要素と同様，空要素として記述され，行番号を表すno属性を持つ。

行やページの情報は，原資料を参照する際に有用な情報であるが，電子化する際には，注意が必要である。具体的には，物理的な改行やページの情報は実際に改行するのではなく，図3のように，タグで記述しなければならない。なぜならば，『ひまわり』は，タグを無視して文字列の照合を行うが，改行文字を一つの文字と認識するからである。したがって，例えば，「蓮の」の後で改行されていると，「蓮の葉」を検索した場合，文字列照合ができなくなる。図1のように，原資料に近い形式での表示を実現する際には，lやpのタグを解釈して，改行するように表示する。

その一方で，段落末などの論理的な改行は，図3の「<l no="6" />」後の改行のように，紙面をそのまま反映する形で，改行を電子化すると，誤った照合を防ぐことができる。例えば，図3の言語資料に対して，「。やがて」という文字列を全文検索した場合，照合に失敗する。

3.2.2. 複数作品の一括検索

図3のXML文書には，単一の作品しか入っていない。しかし，通常の利用においては，複数の作品を一括して検索することが一般的であろう。『ひまわり』で一括検索を実現する方法の一つは，一つのXML文書に複数の作品をまとめて記述しておく方法である。図4に例を示す。この例のように，1作品を一つのarticle要素とし，article要素を作品分だけ列挙すればよい。

```
<corpus name="芥川龍之介作品集">
<article title="蜘蛛の糸" author="芥川龍之介">
  ある日の事でございます。御釈迦様は極楽の蓮池のふちを、独りでぶらぶら御歩き
                    : (中略)
とも云えない好い匂が、絶間なくあたりへ溢れて居ります。極楽ももう午に近くなったのでございましょう。
</article>
<article title="猿蟹合戦" author="芥川龍之介">
  蟹の握り飯を奪った猿はとうとう蟹に仇を取られた。蟹は臼、蜂、卵と共に、怨敵の猿を殺 ...
                    : (中略)
  とにかく猿と戦ったが最後、蟹は必ず天下のために殺されることだけは事実である。...
</article></corpus>
```

図4　複数の文書を一括検索するためのXML文書の例

3.3. 形態素情報が付与された言語資料の場合

この節では，形態素情報が付与された言語資料の扱い方を説明する。この種の言語資料は，基本的に，生のテキストに形態素情報を付与したものとして扱うことができる。実際の例として，『日本語話し言葉コーパス』中のXMLデータ（講演ID:S05F1600から引用）を『ひまわり』用の形式に変換した結果を図5に示す。この中で，talk要素は一つの講演を，su要素は形態論情報である「短単位」を表す。su要素の中のe，t，p，x属性は，それぞれ見出し，代表表記，品詞，品詞細分類を表す。

図5では，見やすさのために，一つのsu要素ごとに改行されているが，実際のXMLデータでは，改行や空白は入れず，生のテキストにsuタグが付与された状態になる。したがって，talk要素に対して全文検索を実行すれば，形態素情報を考慮しないで全文検索を実行することになる。

それに付け加え，形態素情報を考慮した検索も可能である。この場合，su要素を対象として，全文検索を行い，文字列照合の範囲をsu要素内に限定する。この機能を利用することにより，例えば，su要素に対して，格助詞の「で」と認定されている形態素だけを抽出できるようになる。さらに，2.2.2節で示した前後要素に対する検索を行えば，名詞に格助詞の「で」が連接する用例を検索することも可能である。

```
<talk speaker="0001" id="S05F1600">
  <su e="テーマ" t="テーマ" p="名詞">テーマ</su>
  <su e="イマ" t="今" p="名詞">今</su>
  <su e="マデ" t="まで" p="助詞" x="副助詞">まで</su>
  <su e="ノ" t="の" p="助詞" x="格助詞">の</su>
  <su e="ジンセイ" t="人生" p="名詞">人生</su>
  <su e="デ" t="で" p="助詞" x="格助詞">で</su>
</talk>
```

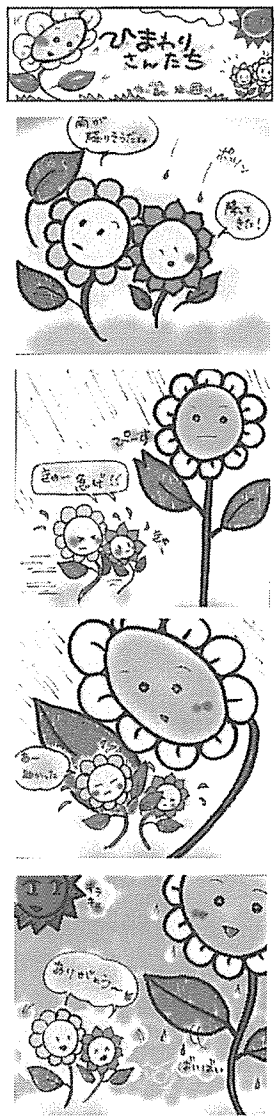
図5　形態素情報を含んだ言語資料の例

3.4. 画像データと関連づけられたテキストの場合

漫画や翻刻した言語資料には，電子化されたテキストと原資料のスキャン画像とが対応づけられていることがある。このような資料は，文字化された部分を検索するだけでなく，検索された文字列と対応する原資料自体が閲覧の対象となる。例えば，主としてセリフのみが文字として表現されている漫画では，セリフの分析には画像はかかせない。また，翻刻された言語資料は，付随する原資料がテキストの分析や理解を補助するであろう。

ここでは，四コマ漫画を『ひまわり』用の言語資料に変換する例を示す。図6（左）が原資料，（右）がそれをXMLとして記述した結果である。XMLデータは，各コマと対応している。このXMLデータは，一つの言語資料を表すcorpus要素を含めて，四つの要素で構成される。

manga 要素：一つの四コマ漫画を表す要素である。漫画のタイトルを表す title 属性，著者を表す author 属性，さらに，タイトル部分の画像ファイルを記述しておく fig 属性を持つ。
scene 要素：漫画の一コマに対応する要素であり，manga 要素の中に出現順に列挙される。図 6 の場合，四コマ漫画なので，四つの scene 要素がある。この要素が持つ属性には，対応するコマの画像ファイル名を記述するための fig 属性，コマの番号を表す no 属性がある。
sound 要素：漫画中で発せられている「音」を表す要素である。この要素には，「音」の発生源と種類を記述するための属性として，それぞれ source, type 属性を用意している。例えば，図 6




```
<corpus name=" 漫画コーパス">
<manga title=" ひまわりさんたち"
       author=" 画:桐生りか, 原作:山口昌也"
       fig="title.png">

   <scene fig="1.png" no="1">
       <sound type="speech" source=" 陽くん">
          雨が降りそうだね
       </sound>
       <sound type="speech" source=" 葵ちゃん">
          降ってきた！
       </sound>
       <sound type="onomatopoeia" source=" 雨">
          ポツン
       </sound>
   </scene>

   <scene fig="2.png" no="2">
       <sound type="speech" source=" 陽くん,葵ちゃん">
          きゃー急げ！！
       </sound>
       <sound type="speech" source=" 葵ちゃん">
          きゃー
       </sound>
       <sound type="speech" source=" ひまわりお母さん">
          ぴーす
       </sound>
   </scene>

   <scene fig="3.png" no="3">
       <sound type="speech" source=" 陽くん,葵ちゃん">
          あー助かった
       </sound>
   </scene>

   <scene fig="4.png" no="4">
       <sound type="speech" source=" 陽くん,葵ちゃん">
          ありがとう〜
       </sound>
       <sound type="speech" source=" ひまわりお母さん">
          ばいばい
       </sound>
       <sound type="onomatopoeia" source=" おひさま">
          キラキラ
       </sound>
   </scene>
</manga>
</corpus>
```

図 6　四コマ漫画の例

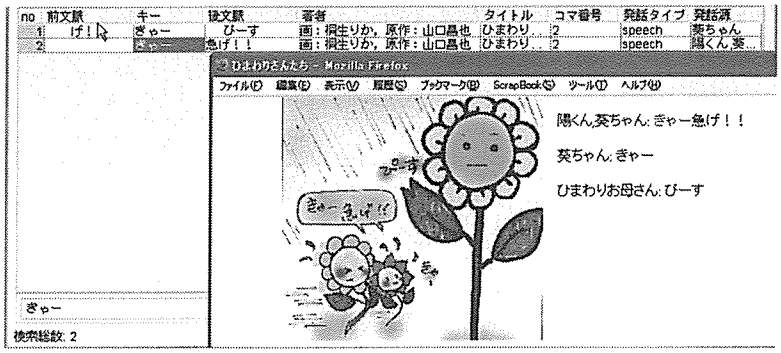


図7　四コマ漫画の検索例

のtype属性は，onomatopoeia（オノマトペ）やspeech（登場人物の発話）などの値をとる。

図6のXMLデータを『ひまわり』で検索すると，図7のようになる。この図のとおり，title, author, type, source属性として付与した内容が，それぞれ「タイトル」「著者」「発話タイプ」「発話源」として検索されていることがわかる。また，scene要素のfig要素は，検索文字列に対応するコマの画像として表示される。

4. おわりに

本稿では，検索環境が未整備の言語資料が存在する現状をふまえ，全文検索システム『ひまわり』を用いて，言語資料検索環境を構築する方法を示した。具体的には，『ひまわり』の検索能力を概説し，『ひまわり』用のXMLデータを作る一般的な手順を示した。また，検索環境の構築例として，三種類の言語資料に対する『ひまわり』用のXMLデータを実例として示した。

注

1　http://www.aozora.gr.jp

2　例：日本の19世紀小説に関する研究資料サイト「ふみくら」
（千葉大 高木 元氏，http://www.fumikura.net）

3　これら以外のOSでも，Javaの実行環境JRE ver.1.4.2以上が動作する環境で利用することができる。

4　『ひまわり』のホームページは，http://www.kokken.go.jp/lrc から「全文検索システム『ひまわり』」のリンク先を参照のこと。

5　芥川龍之介の「蜘蛛の糸」から引用。ただし，ルビは説明用に適宜付加している。

6　厳密に「くも」だけを検索したければ，正規表現で「^くも¥」とすればよい。

7　Windows環境の場合は，キャリッジ・リターン（CR）とLFの2文字で改行を表すので，LFへの変換が必要である。詳しくは，『ひまわり』ホームページの「簡単な検索用データの作成方法」を参照のこと。

8　『ひまわり』のホームページでは，各種資料を『ひまわり』用のXMLデータに変換する具体

的な方法を紹介している。例えば，CSV 形式の『分類語彙表』（国立国語研究所2004）の変換には Perl スクリプトを，XML 形式の『日本語話し言葉コーパス』の変換には，XSLT を用いている。

9 『えだまめ』は http://www.kokken.go.jp/lrc にて，無償公開している。

10 行番号，および，ページ番号は，説明のための便宜的なものであり，実際とは異なる。

参考文献

宇佐美まゆみ(2005)『BTS による多言語話し言葉コーパス』

国立国語研究所(2004)『分類語彙表 増補改訂版』，大日本図書

田中牧郎，山口昌也，吉田谷幸宏，小木曽智信，近藤明日子(国立国語研究所 編)(2005)『太陽コーパス 雑誌『太陽』日本語データベース』，博文館新社

毎日新聞社(1991-2005)『毎日新聞テキストデータベース』

前川喜久雄(2004)「「日本語話し言葉コーパス」の概要」『日本語科学』15，111-133，国書刊行会

山口昌也，田中牧郎(2005)「構造化された言語資料に対する全文検索システムの設計と実現」『自然言語処理』12(4)，55-77，言語処理学会

山下達夫(2000)「用語解説 Suffix Array」『人工知能学会』15(6)，1142，人工知能学会

謝　辞

四コマ漫画を描いてくださった桐生りかさんに感謝いたします。また，本研究の一部は，博報『ことばと文化・教育』研究助成を受け，実施されました。

（投稿受理日：2006年10月12日）

（最終原稿受理日：2007年 2 月 5 日）

山口　昌也（やまぐち　まさや）

国立国語研究所研究開発部門

190-8561　東京都立川市緑町10-2

masaya@kokken.go.jp

Japanese Linguistics 21 (April, 2007) 111–123 〔Note〕

A method of constructing a retrieval environment for language resources using full-text retrieval system "Himawari"

YAMAGUCHI Masaya
The National Institute for Japanese Language

Keywords
full-text retrieval system, language resource, XML

Abstract

Recently, many language resources, for example, text database of newspapers and novels, have become available for language study. However, these resources often do not have their own retrieval systems. This situation makes it difficult for users to utilize these resources for language study. In this paper, I propose a method of constructing a retrieval and browsing environment for existing language resources, using the full-text retrieval system "Himawari" that can retrieve contents and arguments in XML document under a user-specified condition. This paper describes a practical procedure that converts existing language resources into the XML documents that "Himawari" is able to handle. In the introduction, the functions and the retrieval ability of "Himawari" will be expounded. Then, the retrieval environment will be constructed for three kinds of language resources: 1) a nearly raw text, 2) a text with morphological information, 3) a text related to images (a four-frame comic strip) – i.e., these resources are converted to XML documents. Finally, the usefulness of the environment is demonstrated by showing the efficiency of the tagged information appeared in the search results.