DEWS2007 B7-3

A-doc ファイルのアタッチの機能を持つ専用ブラウザの試作

佐藤　裕紀† 　遠山　元道††

† †† 慶應義塾大学理工学部情報工学科　〒 223–8522 神奈川県横浜市港北区日吉 3–14–1
E-mail: †hiroki@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　A-doc とは、Web における結合可能な情報資源としての辞書型の文書である。A-doc の各エントリは結合キーと対応する文書を持ち、本研究では HTML ドキュメントと A-doc エントリの結合 (アタッチ) を行う専用ブラウザを試作した。アタッチには (1) 作者指定 (2) ユーザ指定 (3) 自動の 3 つのモードがある。

キーワード　A－ｄｏｃ, ウェブ, 情報資源, 結合

Trial manufacture of a browser which has a function of attachment A-doc file

Hiroki SATOU† and Motomichi TOYAMA††

† ††Department of Information and Computer Science, Faculty of Science and Technology,
Keio University
Hiyoshi3–14–1, Kouhoku-ku, Yokohama-shi, Kanagawa, 223–8522 Japan
E-mail: †hiroki@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

Abstract　A-doc is a dictionary-modeled document as attachable information resource in the Web. Each entry of A-doc has join-keys and a document which correspond the entry. In this study, We produce a browser experimentally which attaches a HTML document to A-doc entries. In Attachment there are three modes: (1) author designated (2) user designated (3) automatic mode.

Key words　A-doc, WEB, information resource, attach

1. はじめに

現在の Web 技術では、Web 文書の作成者がハイパーリンクを生成しておくことで、他の文書を参照することができる。しかし、ハイパーリンクを張っていない事柄に対しても情報を求めたい場合、Web 文書の読み手の意図で、直接他のファイルへの参照はできない。検索エンジンによって Web 文書の検索を行う方法もあるが、検索を何回も繰り返さなければならなくなる場合もある。例えば、外国語で書かれた Web 文書を閲覧している際、分からない単語が出現する度に辞書で調べなければならなかったり、Web 文書中で著名人の名前を見て検索エンジンでプロフィールや画像などを 1 つ 1 つ調べたりするといったことが挙げられる。

ここで関係データモデルではテーブルにおけるタプルの選択の他に、「結合」という操作ができる。これを Web の世界に応用することで、従来の「Web 文書の検索」ではなく、HTML ファイルの読み手の意図での「Web 文書の結合」という操作が可能になるのではないかと著者らは考えた。

そこで、Web における結合可能な情報資源として、当研究室では、“A-doc” と呼ばれる「辞書型の文書」を考案した。詳細は後述するが、A-doc とは辞書でいう「見出し語」・「項目」に相当するものが格納されたエントリの集合の XML 文書であり、各エントリ内には、一つ以上の「見出し語」と HTML で記述される「項目」が内在されている。今後の展望としては、将来的には様々な企業・団体・個人が、保持しているデータをもとに独自に A-doc を作成し、一般に公開するようになることを当研究室では望んでいる。また、企業の「製品情報」や団体の「名簿」といった情報を、従来の Web ページに代わり、A-doc によって提供することもできる。そして、ユーザーが使用したい A-doc の URL をブックマークのように登録し、必要に応じて参照すればよい。

こういったシステムが誕生すれば、例えば、「英語の HTML 文書」とある出版社が提供している「英和辞典 A-doc」を結合することにより、HTML 文書内の個々の単語から、その意味が掲載されている Web 文書を直接参照できるようになるといった、ユーザーの意図で複数の事柄から、他の文書に対して並列に直接参照することが可能になる。 また、例えば、「和英辞典」、「和仏辞典」、「和独辞典」を用意し、1 つの日本語の文字

列に対する、各外国語の訳を参照するといったように、1 つの事柄に対して複数の項目の参照も可能になる。その際は、ユーザーの意図に応じて A-doc を選択的に利用すればよい。また、結合操作があることから、Web 文書の正規化も可能となり、近年の情報の爆発の原因となっている冗長な文書情報の排除が行えるのではないかとも考えられる。

ここで、本研究では、HTML 文書と、A-doc のエントリとの結合を行う専用ブラウザを試作することを目的とする。Web 文書の結合を可能にする。アタッチの方式に、作者指定アタッチ、ユーザ指定アタッチ、フルアタッチの 3 つの方式を考察し、試作を通じて、用途などについて考察する。

ここで、以下に本論文の構成を述べる。第 2 章では主に A-doc の詳細について説明する。第 3 章では 3 つのアタッチ方式について説明する。第 4 章では専用ブラウザの詳細を述べる。第 5 章では本研究について動作例を示す。第 6 章では関連研究について紹介する。第 7 章では結論を述べる。

2. A-doc の仕様

2.1 概　　要

A-doc とは、辞書で言う、「見出し語」と「項目」に相当するものが格納されたエントリの集合であり、XML で記述される。各エントリ内には、一つ以上の見出し語と、それに対応する項目が内在されており、この見出し語を結合キーとして HTML 文書との結合を行う。ここで、項目の内容は、現時点では HTML 形式の文書を想定しているが、将来的には画像、音楽、動画データ等も格納できるようにしたい。

A-doc の記述方式には大きく分けて 2 種類あり、項目が実体として存在する静的 (スタティック) 方式と、項目の一部がデータベースに格納されている動的 (ダイナミック) 方式の 2 種類がある。ただし、後者については本論文ではこれ以上言及しない。更に、静的方式では標準型 (スタンダードタイプ) と分離型 (セパレートタイプ) の二つのタイプが存在する。

2.2 標準型 (スタンダードタイプ)

これは、エントリ内の “doc” ノードに、HTML 形式のドキュメントが直接埋め込まれた形式をとる。また、HTML のドキュメントに共通部分がある場合は、“html_preamble” ノード、“html_trailer” ノードにそれぞれ共通のヘッダー、フッターを格納し、共通でない部分のみを “doc” ノードに格納してもよい。実際の記述例は図 1 のようになる。なお、この文書は英単語に対する日本語訳が掲載された、「英和辞典 A-doc」想定している。

まず、“adoc” タグの “type” 属性の値を “static-standard” にする。これによって、スタンダードタイプの A-doc であることを宣言する。なお、規格外の値が格納されている場合にも、スタンダードタイプの A-doc と見なされる。また、“doc” ノード (必要ならば “html_preamble” ノード、“html_trailer” ノード) に HTML 形式のドキュメントを格納する際は、コメントアウトの形式にするか HTML エンコーディングを施した形式 (HTML タグを、“<”、“>” で記述する) にするかのいずれかが可能である。ただし、コメントアウトの記述を含む場合は後者の方式で記述せざるを得ない。これは、内部の HTML タグが XML タグの一部と見なされるのを防ぐためである。

また、スタンダードタイプの A-doc は、ファイルサイズが大きくなる傾向にあるので、大量のエントリを含むような形式には向かない。

```
<?xml version="1.0"
 encoding="Shift_JIS" standalone="yes" ?>
<adoc type="static-standard">
 <header />
 <body>
  <entry>
   <kw>senate</kw>
   <kw>Senate</kw>
   <doc>sen・ate n. (the 〜) (古ローマの) 元老院; 議会; (普通 the S-) (米・カナダ・豪・仏の) 上院; (大学の) 評議員会, 理事会.
   </doc>
  </entry>
  <entry>
   …
  </entry>
 </body>
</adoc>
```

図 1　英和辞典 A-doc のスタンダードタイプによる記述例

2.3 分離型 (セパレートタイプ)

これは、スタンダードタイプと違い、エントリ内の “doc” ノードに、項目の URI を格納する形式をとる。記述例は図 2 のようになる。まず、“adoc” タグの “type” 属性の値を “static-separate” にする。これによって、セパレートタイプの A-doc であることを宣言する。そして、“doc” ノードには、HTML 文書への URI を格納する。

2.4 アタッチ

専用ブラウザにより、A-doc のエントリ内の見出し語を結合キーとみなして、エントリと HTML 文書の結合を行う操作を、著者らは「アタッチ (attach)」と呼んでいる。技術面では、「はてなダイアリー」[1] や「ウィキペディア」[2] といった、「辞書型」の文書へ自動リンクシステムを参考にし、既存の HTML の技術である「ハイパーリンク」に変換することによって実現する。更に、「アタッチ」にはいくつかの方式があるが、詳細次節には述べる。

3. アタッチにおける 3 つの動作モード

3.1 作者指定アタッチ

作者指定アタッチとは、どの A-doc を参照したいか、どんなエントリが欲しいかといったアタッチ方式を HTML 文書の作成者・編集者が、予め指定するアタッチ方式を言う。

まず、A-doc への参照方法としては、“anc” というタグを導入する。構造としては、一つ以上の “kw” ノードに格納され

```
<?xml version="1.0"
 encoding="Shift_JIS" standalone="yes" ?>
<adoc type="static-separate">
 <header />
 <body>
  <entry>
   <kw>senate</kw>
   <kw>Senate</kw>
   <doc>senate.html</doc>
  </entry>
  <entry>
   <kw>committee</kw>
   <kw>Committee</kw>
   <doc>committee.html</doc>
  </entry>
  <entry>
   ...
  </entry>
 </body>
</adoc>
```

図 2　英和辞典 A-doc のセパレートタイプによる記述例

る結合キー (見出し語)、一つ以上の “adoc” ノードに格納される A-doc への URI、そして、“word” ノードに格納されるハイパーリンクに変換された際に表示される文字列を内在している。これを HTML 文書内のアタッチを行いたい箇所に埋め込んでおく。そして、A-doc 内のエントリと anc タグ内のお互いの結合キー同士を比較してエントリのアタッチを行うのである。記述例は図 3 のようになる。

```
<anc>
 <kw>イチロー</kw>
 <kw>鈴木</kw>
 <adoc>webtest/example2/talent.adoc</adoc>
 <adoc>webtest/example2/mlb.adoc</adoc>
 <word>イチロー</word>
</anc>
```

図 3　“anc” タグ記述例

また、アタッチアルゴリズムは図 4 の通りである。これは、A-doc 内のエントリと、anc タグ内のキー集合の中で一つでも一致するものがあれば、そのエントリのアタッチを行い、これを “anc” タグ、指定された A-doc がある限り続けるということを示している。なお、指定された A-doc にアクセスできない場合、エラーメッセージを表示させ、ユーザーに注意を喚起するようにしているが、その他の細かいエラー対策は省略した。

また、アタッチされた “anc” ノードは、“a” ノードに変換している。例えば、図 3 の変換後の記述は、
 イチロー
のようになる。URL は、プロトコル名の後に、作者指定アタッチを示す接頭語 “ADAAdoc(Author Designated Attached A-doc)” が続き、その後にシステム内でアタッチされたエントリを表す番号が続いたものになっている。ブラウザからアタッチされた文字列をハイパーリンクで参照するようにすれば、URL のエントリ番号を元に、1 つないし複数のエントリを表示できる。

入力：HTML 文書
出力：HTML 文書、配列 E

（1） HTML 文書から “anc” ノードを 1 つ取得
　もしなければ計算終了
（2） “anc” ノードから “adoc” のエレメントを 1 つ取得
　もしなければ (12) へ
（3） (2) のエレメント (A-doc の URL) を元に A-doc データを取得
　もし接続に失敗したら、配列 E へ “No Data” を格納し (11) へ
（4） (3) の A-doc データから、エントリ 1 つを取得
　もしなければ (11) へ
（5） (4) のエントリからキーを 1 つ取得
　もしなければ (10) へ
（6） “anc” ノードからキーを 1 つ取得
　もしなければ (8) へ
（7） (5)、(6) のキーを比較
　もし一致したら、エントリの “doc” ノードを元に辞書型の文書情報を取得し配列 E へ格納、“anc” ノードの最初のキーへ移動し (9) へ
　一致しなければ、“anc” ノードの次のキーへ移動し (6) へ
（8） “anc” ノードの最初のキーへ移動
（9） エントリの次のキーへ移動し (5) へ
（10） A-doc データの次のエントリへ移動し (4) へ
（11） “anc” ノードの次の “adoc” エレメントへ移動し (2) へ
（12） “anc” ノードをハイパーリンクに変換
（13） HTML 文書の次の “anc” ノードへ移動し (1) へ

図 4　作者指定アタッチのアルゴリズム

3.2　フルアタッチ

3.2.1　概　　要

フルアタッチとは、HTML 文書内でアタッチできる文字列を検索し、それらのほぼ全てに自動でアタッチを行うアタッチ方式を言う。前述の作者指定アタッチと違い、HTML 文書の作成者・編集者が指定する事柄はなく、後述するが、予めユーザーがブラウザに A-doc の URL を登録しておき、その中からどれを使用するかを選択するだけである。

3.2.2　アタッチアルゴリズム

アタッチアルゴリズムは図 5 の通りである。これは、エントリ内のキーの文字列を HTML 文書内で探し、見つかったらそ

の文字列にアタッチをしていくという操作を、A-doc が指定されている限り続けるということを示している。

また、作者指定アタッチと同様に、アタッチされた文字列を “a” タグで囲うようにすることでハイパーリンクに変換する。変換後の記述例は

 イチロー

のようになる。URL は、プロトコルの後に、フルアタッチを示す接頭語 “FAAdoc(Full Attached A-doc)” が続き、システム内でのアタッチされたエントリを表す番号が続いたものになっている。

ただし、このアルゴリズムでは、何らかのタグや “a” タグに囲まれている文字列に対してはアタッチできないようにしている。前者は明らかであるが、後者の場合は、HTML 文書の作成者・編集者の意図に反する可能性が高く、ユーザーにとっても有益な情報であると考えにくいからとした。そのため、先に見つかったエントリのアタッチが優先され、後からアタッチされるはずのエントリはアタッチできなくなる。

そこで、先にアタッチされた文字列に含まれるような文字列に対しても多重にアタッチできるように図 5 の 5 行目の処理を改良した。改良された部分は図 6 の通りである。

入力：HTML 文書
出力：HTML 文書、配列 E

（1） ブラウザから A-doc の URL を 1 つ取得
　もしなければ計算終了
（2） (1) の URL を元に A-doc データを取得
　もし接続に失敗したら、(8) へ
（3） (2) の A-doc データから、エントリ 1 つを取得
　もしなければ (8) へ
（4） (3) のエントリからキーを 1 つ取得
　もしなければ (7) へ
（5） キーと一致する文字列を HTML 文書内で探す
　もし見つかった文字列がどの “a” ノード内部にないか、どのタグの内部にもなければ、文字列をハイパーリンクに変換、エントリの “doc” ノードを元に項目を取得し配列 E へ格納
(これを該当する文字列が文書内に存在する限り繰り返す)
（6） エントリの次のキーへ移動し (4) へ
（7） A-doc データの次のエントリへ移動し (3) へ
（8） ブラウザの次の A-doc の URL へ移動し (1) へ

図 5　フルアタッチのアルゴリズム

ただし、このアルゴリズムでも、もともとの HTML 文書の作成者が書いたリンク文字列（また、作者指定アタッチ）にはアタッチできない。

3.3　ユーザー指定アタッチ

ユーザー指定アタッチとは、ブラウザのユーザーがアタッチ方法を指定するアタッチ方式を言う。ユーザーがブラウザから文字列を選択し、それを結合キーと見なしてアタッチを行う。フルアタッチでは、ブラウザ上ではつながった文字列として見えていても、実際はタグ等で文字列が切れてしまい、アタッチ

(5) キーと一致する文字列を HTML 文書内で探す
　もし見つかった文字列がどの “a” ノード内部にないか、どのタグの内部にもなければ、文字列をハイパーリンクに変換、エントリの “doc” ノードを元に辞書型の文書情報を取得し配列 E へ格納
　上記以外で、もし見つかった文字列が、上記によるハイパーリンクの “a” ノード内部にあるならば、その “a” タグの URL を変更、エントリの “doc” ノードを元に辞書型の文書情報を取得し配列 E へ格納
(これを該当する文字列が文書内に存在する限り繰り返す)

図 6　改良したフルアタッチのアルゴリズム

ができなくなる可能性あるが、このアタッチ方式では、その問題を気にしないでアタッチが出来るため、先述の二つのアタッチ方式の補助という位置づけである。

また、アタッチアルゴリズムは図 7 の通りである。これは、ブラウザから選択された文字列と一致する結合キーがエントリにあれば、そのエントリのアタッチを行うという操作を、A-doc が指定されている限り続けるということを示している。

入力：HTML 文書から選択された文字列 S
出力：配列 E

（1） ブラウザから A-doc の URL を 1 つ取得
　もしなければ計算終了
（2） (1) の URL を元に A-doc データを取得
　もし接続に失敗したら、(8) へ
（3） (2) の A-doc データから、エントリ 1 つを取得
　もしなければ (8) へ
（4） (3) のエントリからキーを 1 つ取得
　もしなければ (7) へ
（5） S と (4) のキーを比較
　もし一致したら、エントリの “doc” ノードを元に辞書型の文書情報を取得し配列 E へ格納し、(7) へ
（6） エントリの次のキーへ移動し (4) へ
（7） A-doc データの次のエントリへ移動し (3) へ
（8） ブラウザの次の A-doc の URL へ移動し (1) へ

図 7　ユーザー指定アタッチのアルゴリズム

4.　専用ブラウザの実装

4.1　概　　要

今までの議論を踏まえ、専用ブラウザを試作した。概観は図 8 のようになる。ここで、作者指定アタッチはサーバーサイドのスクリプト言語などを用いれば実装が可能であり、ユーザー指定アタッチも Firefox [3] などのプラグインとしての実装も可能であると考えられるが、あえてブラウザという形態を取った理由としては、A-doc システムの初めての試作という点で、様々な可能性などを検討するためである。

以下にアタッチ関連の機能を簡単に紹介する。 まず、URL

バーなどのツールバーがある段の下に、登録された A-doc とチェックボックスが表示されている。また、その段の下にアタッチ方式の切り替え用のラジオボタンがあり、その右にアタッチボタンがついている。フルアタッチかユーザー指定アタッチをラジオボタンで選択し、アタッチを行う際はアタッチボタンを押せばよい。また、必要に応じて A-doc の URL をブラウザに登録し、フルアタッチ、ユーザー指定アタッチの際に使用する。

図 8　専用ブラウザの概観

4.2　システム構成

本システムの大まかな構成を図 9 示す。本システムは、大きく分けると、ブラウザの本体部、作者指定アタッチ部、フルアタッチ部、ユーザー指定アタッチ部の 4 つに分かれている。専用ブラウザで HTML 文書の要求をすると、取得されたデータが自動的に「作者指定アタッチ部」を通され、“anc” タグを解析して、その結果がブラウザの HTML 表示部へ渡される。また同時に “anc” ノードからアタッチ方法を読み取りエントリを取得し、ブラウザ本体部へ渡す。また、フルアタッチ、ユーザー指定アタッチを実行すると、それぞれのアタッチ部が起動するようになっている。以下に、ブラウザ本体部と 3 つのアタッチ部について、個々に説明していくことにする。

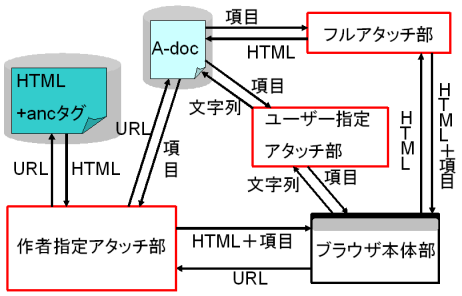

図 9　大まかなシステム構成

作者指定アタッチ部では、“anc” タグ解析部で “anc” ノードを読み取り、その部分をハイパーリンクへ変換し、ブラウザの表示部へ解析結果の HTML 文書データを渡して表示させる。また、読み取った “anc” ノードからアタッチ方法を解析し、そのデータをアタッチ接続部 (アタッチ本体部) へ渡す。それをもとに A-doc へ接続、取得 A-doc データを文字列型に変換し、A-doc を解析し、アタッチされたエントリがあるならばそれをブラウザ本体部の内部変数へ渡し、アタッチされたエントリの表示要求があったときにそのデータを参照させる。

フルアタッチ部では、現在ブラウザに表示されている HTML 文書データと、アタッチしたい A-doc の URL を渡し、作者指定アタッチ部と同様に A-doc へ接続、取得データを文字列型に変換し、A-doc を解析して、HTML 文書データのアタッチされる文字列をハイパーリンクに変換してブラウザの表示部へ渡して表示させる。また、アタッチされたエントリがあるならばそれをブラウザ本体部の内部変数へ渡し、アタッチされたエントリの表示要求があったときにそのデータを参照させる。

ユーザー指定アタッチ部では、現在ブラウザに表示されている HTML 文書から選択された文字列と、アタッチしたい A-doc の URL を渡し、A-doc へ接続、取得データを文字列型に変換し、A-doc を解析し、アタッチされるエントリがあるならばそれをブラウザ表示部へ渡し、そのまま表示させる。

5.　動作例と応用

5.1　事 前 準 備

以下のような 3 つの A-doc、talent.adoc(タレント名鑑が記載されていると想定、図 10)、mlb.adoc(メジャーリーグベースボールの選手データが記載されていると想定、図 11)、english-japanese.adoc(英和辞典が記載されていると想定、図 12) を用意し、予めブラウザにこの優先順位で登録しておく。なお、3 つとも全てスタティック・セパレートタイプの記述をしているが、これは簡略化のためであり、もちろん、スタティック・スタンダードタイプのものでも結果は基本的に同じになる。また、“doc” ノードに格納されている項目の中身については省略する。

5.2　作者指定アタッチ

作者指定アタッチの動作例を示すために、図 13 のような HTML 文書を用意した。専用ブラウザでこの文書を閲覧すると、「イチロー」の文字列と「田村正和」の文字列にリンクが張られている (図 14)。

ここで、「イチロー」の部分を展開すると (図 15)、“talent.adoc” からのエントリと、“mlb.adoc” からのエントリがアタッチされている。これに関しては、“talent.adoc”、また、“mlb.adoc” と HTML 文書の 1 つ目の “anc” ノード内の「イチロー」という一致した結合キーがあったためアタッチされたのである。

また、図は省略するが、「田村正和」の部分を展開すると、“talent.adoc” からのエントリが表示される。また、「指定された A-doc が見つからない」という旨のエラーメッセージが表示されている。これは、“talent.adoc” のエントリと 2 つ目の “anc” ノード内で、「田村正和」という一致した結合キーがあったためアタッチされたのである。しかし、“furuhata.adoc” が存在しないためエラーとなる。また、エントリのアタッチする確率を高めるためには、“anc” ノード、エントリに、項目に関

```
<?xml version="1.0"
 encoding="Shift_JIS" standalone="yes" ?>
<adoc type="static-separate">
 <header />
 <body>
  <entry>
   <kw>田村正和</kw>
   <doc>masakazu_tamura.html</doc>
  </entry>
  <entry>
   <kw>イチロー</kw>
   <kw>鈴木一郎</kw>
   <doc>ichiro.html</doc>
  </entry>
 </body>
</adoc>
```

図 10　talent.adoc

```
<?xml version="1.0"
 encoding="Shift_JIS" standalone="yes" ?>
<adoc type="static-separate">
 <header />
 <body>
  <entry>
   <kw>イチロー</kw>
   <kw>鈴木一郎</kw>
   <kw>Ichiro</kw>
   <doc>ichiro_suzuki.html</doc>
  </entry>
  <entry>
   <kw>松井秀喜</kw>
   <kw>マツイ</kw>
   <kw>matsui</kw>
   <kw>ゴジラ</kw>
   <doc>matsui.html</doc>
  </entry>
 </body>
</adoc>
```

図 11　mlb.adoc

する「呼び名」をできるだけ多く含めばよい。

5.3　フルアタッチ

フルアタッチの動作例を示すために、図 16 のような HTML 文書を用意した。これに対して、英和辞典 A-doc(“english-japanese.adoc”) を選択して、優先型フルアタッチを実行すると、いくつかの単語にリンクが張られる (図 17)。このうち、“Senate” の文字列を展開すると、“english-japanese.adoc” 中

```
<?xml version="1.0"
 encoding="Shift_JIS" standalone="yes" ?>
 <adoc type="static-separate">
 <header />
  <body>
  <entry>
   <kw>senate</kw>
   <kw>Senate</kw>
   <doc>senate.html</doc>
  </entry>
  <entry>
   <kw>committee</kw>
   <kw>Committee</kw>
   <doc>committee.html</doc>
  </entry>
  <entry>
   <kw>repudiate</kw>
   <kw>Repudiate</kw>
   <doc>repudiate.html</doc>
  </entry>
   ...
 </body>
</adoc>
```

図 12　english-japanese.adoc

の “Senate” に対応する項目がアタッチされていることがわかる (図は省略)。

“Senate” に関しては、“english-japanese.adoc” のエントリに “Senate” という結合キーがあったためアタッチされた。他のアタッチされた文字列に関しても同様である。しかし、“WASHINGTON”、“Relation” といった文字列に対してはアタッチされていない。これは、こういった単語が “english-japanese.adoc” に格納されていないからである。

5.4　ユーザー指定アタッチ

フルアタッチの例で示した HTML 文書中の “Senate” という文字列をカーソルで選択し、更に英和辞典 A-doc(“english-japanese.adoc”) を選択した状態で、ユーザー指定アタッチを実行すると、フルアタッチを行った例と同じエントリがアタッチされ、表示される。

これは、“english-japanese.adoc” のエントリに “Senate” という結合キーがあったためアタッチされたのである。ここで、もし “Senate committee”、“enate”、といった、A-doc のエントリ内の結合キーにとって過不足のある文字列を選択して実行していたら、何もアタッチされない。

5.5　応 用 例 1

ある日本語の HTML 文書を用意する。また、詳細は省略するが、英和辞典 A-doc と同様の形式で、和英辞典 A-doc、和仏辞典 A-doc、和独辞典 A-doc を用意し、また、この優先順位で

```
<html>
 <head><title>応用例</title></head>
 <body>
  <anc>
   <kw>イチロー</kw>
   <kw>鈴木</kw>
   <adoc>webtest/example2/talent.adoc</adoc>
   <adoc>webtest/example2/mlb.adoc</adoc>
   <word>イチロー</word>
  </anc>
  と
  <anc>
   <kw>田村正和</kw>
   <kw>古畑任三郎</kw>
   <adoc>webtest/example2/furuhata.adoc</adoc>
   <adoc>webtest/example2/talent.adoc</adoc>
   <word>田村正和</word>
  </anc>
  はドラマ共演した事がある。
 </body>
</html>
```

図 13　HTML 文書例

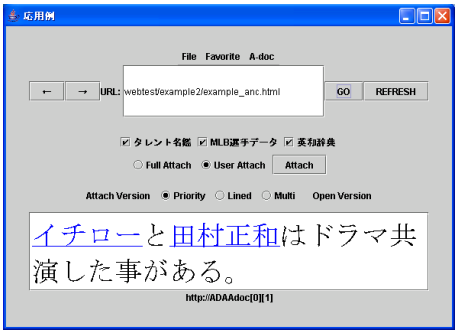

図 14　作者指定アタッチの実行結果 1

ブラウザに登録しておく。この 3 つの A-doc を選択した状態で多重型フルアタッチを実行すると、それぞれのエントリ内に見出し語が存在する単語にリンクが張られている (図 18)。また、このうち、「今日」という文字列を展開すると、この文字列に対する外国語訳

の項目がアタッチされていることが分かる (図 19)。ここで、もし優先型フルアタッチを実行していたら、優先順位の影響で和英辞典 A-doc のみがアタッチされ、他の 2 つのエントリはアタッチされないことになる。

5.6　応 用 例 2

知りたいと思う情報が複数ある場合、ただ、それらを列挙しただけの HTML ファイルを用意し、それに対してアタッチさせることで検索エンジンによって検索を行う代わりに結合され

図 15　作者指定アタッチの実行結果 2

```
<html>
 <head><title>例</title></head>
 <body>
  <h1>
   Senate committee repudiates Bush on Iraq
  </h1>
  <p> WASHINGTON - In a calculated snub of
      President Bush, the Democratic-controlled
      Senate Foreign Relations Committee
      dismissed plans for a troop buildup in Iraq
      on Wednesday as "notin the national
      interest" of the United States.</p>
 </body>
</html>
```

図 16　HTML 文書例 2

図 17　フルアタッチの実行結果 1

た Web 情報が得られる。例えば、知りたい英単語を羅列し、フルアタッチを行うことで, 並列に英単語の意味が求められる。このような利用例は、本システムが従来の検索エンジンに代わる、新たな Web 情報の探索支援ツールと成り得る可能性を示している。

5.7　その他の事例・今後の展望

多重型フルアタッチでは、先にアタッチされた文字列に含まれるような文字列に対しても多重アタッチが可能なため、関連

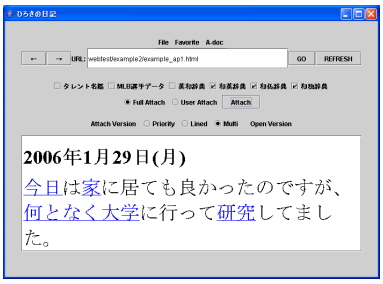

図 18　応用例による実行結果 1

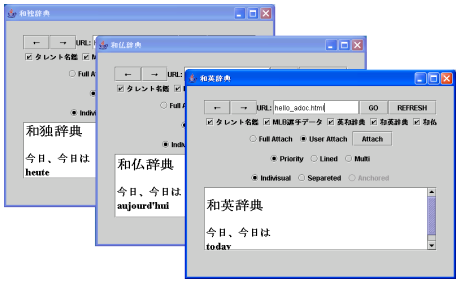

図 19　応用例による実行結果 2

の低いエントリがアタッチされてしまう可能性がある。例えば、「東大阪大」という文字列に先にアタッチされて、その後「東大」、「大阪」、「阪大」などといったキーが現れた場合、これらのエントリもアタッチされてしまうことになる。こういった事例は稀であると言えるが、その可能性は最初にアタッチされた文字列が長ければ長いほど高くなるとも考えられる。

また、ブラウザ上ではつながった文字列として見えていても、実際はタグ等で文字列が切れていて、つながった文字列として認識できない場合がある。例えば、「東大阪大」という文字列に対し、先に「東大」というキーでアタッチされた場合、文字列は “ 東大 阪大” のようになってしまい、「東大阪大」、「大阪」などと言ったキーが現れても、もはやアタッチができなくなる。つまり、エントリやキーの出現順位に依存するのである。実際にこういった事例はどの程度の頻度で起こりうるかはいずれ測定したい。また、この問題に対する解法も今後の課題としたい。

また、現在のアタッチアルゴリズムは、ブラウザ側とエントリ側の結合キーの集合の中で、共通部分があればアタッチするという単純なものである。そのため、誤字脱字によりアタッチできなくなったり、それを防ぐために記述する結合キーが増加してしまうといったことにつながる。また、アタッチする際に、キーやエントリの優先順位によって、結果が変化してしまうという問題もある。

今後は、こういった問題に対し、現在のアタッチ条件に緩急をつけ、よりユーザーの意図に合ったエントリのアタッチを目指したい。

5.8　そ　の　他

以下に、今回動作例の参考にした Web サイトを載せる。

- ウィキペディア [2]
- 西日本電信電話株式会社
http://www.ntt-west.co.jp/news/0501/050128c.html
- Go Mariners! シアトル・マリナーズ完全ガイド
http://www.junglecity.com/seattlepi/roster.htm
- goo 辞書 http://dictionary.goo.ne.jp/index.html
- 古畑任三郎ファイナル
http://wwwz.fujitv.co.jp/furuhata/index2.html
- Examiner.com http://www.examiner.com/a-526601 Senate_Committee_Repudiates_Bush_on_Iraq.html

6.　関 連 研 究

「はてなダイアリー」[1] とは、あるブログサービスであり、記事を投稿すると「はてなダイアリーキーワード」と呼ばれる辞書型の文書へのリンクが自動的に張られる。はてなダイアリーキーワードでは、各見出し語に対してその辞書型の文書が掲載されている。ブログに記事を投稿した際、見出し語をキーと見なし、文書内から存在するキーワード文字列を探し、見つかったらその文字列に対して、はてなダイアリーキーワードへのハイパーリンクを張る。

「ウィキペディア (Wikipedia)」[2] とは、インターネット上で作成、公開されているオープンコンテント方式の百科事典のようなものである。 ウィキペディアでも、各見出し語に対してその辞書型の記事が掲載されている。また、記事の編集者の意図により、本文中で他の記事に参照したい文字列を “[[”、“]]” という記号で囲むことによって、囲まれた文字列と一致する記事を検索し、そのような記事が存在すればリンクが張られる。

SRS(Sequence Retrieval System) [4] は、データベースをフラットファイルのままインデックスをつけることによりに高速に検索するためのシステムで、WWW からの検索時には、他のデータベースへのハイパーリンクが可能になる。

7.　ま　と　め

本研究において、HTML 文書と、辞書型の文書 “A-doc” のエントリとの結合 (アタッチ) を行う専用ブラウザを試作した。それによって、既存の辞書型の文書の参照方式とは違い、より柔軟な参照ができるようになった。

また、動作例を用いて問題点を検討した。現在のアルゴリズムでは、アタッチされる結果は、エントリやキー、A-doc のアクセス順位に依存することが分かっている。

今後はアタッチアルゴリズムを改善し、結果がアクセス順位に依存しないようにすること、また、結合キーの比較の際の条件緩和、A-doc 構造の整備などについて検討してゆきたい。

文　　献

[1] はてなダイアリー http://d.hatena.ne.jp/
[2] フリー百科事典『ウィキペディア（Wikipedia）』
http://ja.wikipedia.org/wiki/%E3%83%A1%E3%82%A4%E3%83%B3%E3%83%9A%E3%83%BC%E3%82%B8
[3] Mozilla Japan - Firefox
http://www.mozilla-japan.org/products/firefox/
[4] Thure Etzold and P. Argos, “an indexing and retrieval tool for flat file data libraries”, *Computer Applications in the Biosciences*, Vol.9, No.1, pp. 49-57, 1993