Wiki 的手法に基づく構造化データの編集について

守岡 知彦

京都大学 人文科学研究所

人文情報学的なデータベースの構築においてしばしば直面する問題のひとつに形式化の問題がある。この問題を解決するための良い方法のひとつは、データベースをリファクタリングすることだと考えられるが、これはあまり容易なことではないといえる。そこで、本論文では、構造化データの形式化を Wiki 的な手法で試行錯誤しながら進めるための枠組や方法について議論するとともに、著者が開発している構造化データを Wiki 的な手法に基づき編集するための WWW アプリケーションである EgT について述べる。

Editing Structured Data based on Wiki-way

MORIOKA, Tomohiko

Institute for Research in Humanities, Kyoto University

In development of databases about informatics in humanities, formalization is a one of the most important and difficult problems. Refactoring is a good solution, but it is not easy to apply large scale databases. In this paper, we discuss about how we can design and modify format of structured data with trial and error, and describe about EgTwhich is a Web application to edit structured data based on Wiki-way.

1 はじめに

人文情報学的なデータベースの構築においてしばしば直面する問題のひとつに形式化の問題がある。データを提供する人文学研究者とそれをモデル化して計算機に載せる情報学研究者との意思疎通や相互理解の不備、あるいは、そもそもの研究対象の性質に由来する原理的な問題などから、一般的な関係データベースでは扱いにくいようなデータになってしまったり、あるいは、不備や例外、解釈の間違い等を見つけた結果、スキーマを作り直したくなることも少なくない。形式化されたデータや実際のシステムを見てその問題点を潰して行くというのはこの種のデータベースを実現する上で有効な方法だといえるが、巨大なデータベースをリファクタリングするのはあまり容易なことではないといえる。より良いモデルや形式を探る上で試行錯誤を繰り返すことは重要であるが、現状では形式化やシステム開発に不慣れな人にとってハードルが高いのが現状である。そこで利用者が容易に編集しやすい Wiki 的な方法で、構造を持っている（ことが判っている）がきっちりとした形式化がなされていないようなデータの編集やその形式化の支援を行うことを考える。

2 データベースの修正

データを修正するという場合、

1. 誤った内容の修正
2. 分節化・符号化・データ表現等の規準の修正

の２つの場合があるといえる。

1 は、通常、既存の形式の範囲内で行える修正であるといえ、特に問題はない。しかしながら、内容の修正を行っている内に、データ形式の変更を迫られる場合がある。例えば、漢字の部首・画数のデータを、部首と（部首内）画数という２つの欄（素性）で表現していたとする。この時、しんにょう（「辶」「辶」「辶」）やくさかんむり（「艸」「艹」「艹」「艹」）などの幾つかの部品を持つ漢字の場合、どの字体を想定するかによって画数は異なり得る。また、部首の取り方が複数

ありうる文字もあり、その場合、どの部首を用いたかによって部首内画数は異なり得る。そうすると、このあたりをより詳細に記述したいと思うならば、字体や字形、部首や画数の選び方・数え方の規準、出典等の付加情報や、部首と画数の依存関係を記述可能な形にデータ形式を修正する必要が生じる。大量にデータを入力する際には、しばしば、定義されたデータ形式に合わせて、厳密には正しくない取りあえずの値が入力されるが、データの校正の際に、判断に迷うような事例に遭遇することは、文字に限らず良くあることではないかと思われる。これが 2 のケースである。

こうした時の判断はそれ自体重要な知見であるといえ、その際に想定された選択肢やシナリオ、議論や判断、意思決定までの課程、最終的な結論や新たに定義された規準・概念等は全てオントロジーの構成要素たる知識であり、記録されていることが望ましい (できれば、なるべく機械可読な形で)。すなわち、データ形式自体を情報システムにおける一級オブジェクト (first class object) としてデータ化の対象とすることは人文情報学的な研究において重要な意味を持つ情報を収集する上で有益であると考えられる。

一方、現在の一般的なデータベース・システムにおいて、データベースの形式を定義するスキーマは容易に編集可能な対象とはなっていない。特に、既に大量のデータが蓄積されている場合、事実上不可能であることが少なくない。このために、実際にデータベースを実装する前の設計作業に大きな負担がかかってしまうといえる。また、この設計作業においてコンピューターの支援が十分に受けられず、実際のシステム (ないしは、そのプロトタイプ) を見ないまま議論することになるために、人文学研究者と情報学研究者のイメージの齟齬を生じやすいのではないかと考えられる。また、この議論の過程自体を記録し、再利用することも容易ではない。

逆にいえば、こうした形式化に関わるような作業を含むデータ化のプロトタイピング作業をコンピューターによって支援することによって、人文情報学的なデータベース構築における困難さの幾つかを解消することができるのではないかと考えられる訳である。

ところで、異なる分野の研究者の対話を支援するという観点で考えた場合、この『形式化』は情報科学的な意味での狭義の形式のみならず、情報の可視化を含むような問題、例えば、model-view-control の設計や、内部表現と入力用表現と表示用の形式の対応関係や切り分け、複数の概念やパターン・コンテナ等への分解といったことの支援を、処理結果 (画面イメージ) を見ながら修正できるような仕組みを提供することが重要なのではないかと思われる。

3 知識表現の Wiki 的な編集

WWW 上で複数人が共同で試行錯誤しながら情報編集するためのツールとしては、Wiki が広く使われている。一般的な Wiki は人間が読むためのテキストを対象にしており、章やパラグラフ、リストや表、リンクといった、HTML のサブセットに相当するような、簡単な構造のみを備えていることが多い。

一方、近年、セマンティック **Web** と呼ばれる、『賢い』機械処理を可能とするために、メタデータやオントロジー等を付与しようとする流れがある。Wiki においても、こうしたことを Wiki 的手法によって実現しようとする試みがあり、ここでは機械可読な知識表現を生成することを目的とした Wiki を『セマンティック Wiki』と呼ぶこととする。

セマンティック Wiki は大別すると、

1. 人間が読むための通常の Wiki テキストの中に知識表現を書くためのタグを埋め込む方法
2. 機械可読な知識表現を主体とする方法

に分かれる。[1]

前者の例として **Semantic MediaWiki** [2] を挙げる。これは Wikipedia などで使われている **MediaWiki** [3] の拡張であり、通常の MediaWiki のテキスト中に [[述語::目的語]] という記法で、主語 (subject)・述語 (predicate)・目的語 (object) の 3 項関係 (triple) を記述することが出来る。この場合、知識表現用のタグが埋め込まれた部分は機械可読であるものの、全体としては人間が読むためのテキストとなってしまい、テキスト全体の形式化自体を支援する訳ではない。一方、後者の場合、自由度や表現力の点ではもっとも強力であるが、機械可読性な形式を直接書く場合、初心者にとってはハードルが高いものになりがちだという問題がある。

さまざまな種類の資料を機械の都合に合わせることなく適切に表現するためには自由度や表現力が十分に

備わっている必要があるが、その結果、難易度が高くなるのは望ましくない。

ところで、共同作業でデータを編集するということはその複数人でデータを共有するということでもある。その時、形式や内容に関してある程度の合意は必要であるが、細かい部分での意見の差異は許容できた方が良いといえる。また、学説上の判断が分かれるような部分も共存できることが望ましい。

4 CHISE-Wiki

機械可読なデータのための Wiki の一種として、著者は **CHISE-Wiki** [4] というものを提案している。これは CHISE の文字オントロジー [5] を対象とした Wiki であるが、機械可読な知識表現を主体とする Wiki の一種になっており、データの種類や形式等の自由度と編集のしやすさの両立を計っている。これは『メタ編集』と呼ぶ手法によって実現されている。

CHISE-Wiki の編集対象となる文字は Chaon モデル [5] に基づく素性の集合であり、素性値は素性名毎に異なった型（形式）をとり得、その意味も素性名毎に異なり得る。こうした素性の性質は、文字と同様に、素性の集合で表現される。例えば、素性の型や表示用の形式、編集用の形式、名前や説明、他の素性との関係等は素性の素性として表現される。よって、文字（文字の素性）を編集するのと同様に素性（素性の素性）を編集することで、文字（文字の素性）を編集する環境の振舞を変えることができる。このような手法を『メタ編集』と呼ぶ。

メタ編集が行っていることは多くのデータベース・システムで行っているスキーマの定義に他ならない訳であるが、一般的なデータベース・システムでは型（データ形式）を変えるのはそれほど容易ではない。ナイーブに型を変えるとデータの意味が変わってしまうからである。CHISE-Wiki では内部表現を S 式にしており、オブジェクト自身が自らの型情報を持っているので、型定義や書式の変更は実データに影響しない。このため、メタ編集に基づく漸進的な形式化が行いやすい訳である。

一方、名前や説明、コメントといった主に人間が読むための情報は一般的な Wiki テキストのように記述すれば良い訳であるが、CHISE-Wiki では書式の定義において同様の記法を用いるようにしている。一般的な Wiki テキスト中に、整数を 10 進、16 進、区点表示するためのタグ、あるいは、リストをタブ区切り、コンマ区切り、S 式で表示するためのタグといった、各種表現形式で表示するためのタグを埋め込むことで、さまざまな表示を実現するとともに、編集時には指定した表現形式に従った編集モードを用意することで、素性の型や表現形式に従った編集を実現している。

CHISE-Wiki の編集対象は文字に限定されているが、この手法は素性の集合で表せるようなものであれば文字でなくても適用可能であるといえ、ここでは、CHISE-Wiki を一般化して、文字以外のさまざまな種類の情報を対象としたものを実現することにする。

5 EgT

素性の集合からなる構造を持った機械可読な情報を編集するための Wiki として現在 **EgT** (Editing Structured (or Semantic) Things) を開発中である。これは CHISE-Wiki のコードを拡張し、CHISE 文字オントロジーの基盤である Concord [6] の機能を利用して、CHISE-Wiki を含む形で文字以外の情報も扱えるようにしたものとなっている。

5.1 オブジェクト

EgT における Wiki 頁で表現されるものは、CHISE-Wiki と同様に、素性の集合からなるオブジェクトである。但し、CHISE-Wiki ではオブジェクトとして文字だけが存在したが、EgT ではそれを一般化したものとなっており、文字以外のさまざまなジャンルのオブジェクトが利用できる（図 1）。

5.2 オブジェクトの URI 中での表現

一般的な Wiki では Wiki 頁はユニークな ID を持つ。[1] 一方、Concord や CHISE の枠組ではオブジェクトは素性の集合で表現されており、オブジェクトに対するユニークで特権的な ID を除去しようとしている。そのため、URI によってどのように Wiki 頁（= オブジェクト）を表現するかが問題となるが、原理的

[1] エイリアスが利用可能なものもあるが

図 1: オブジェクトの表示例（本の書誌データ）

には Chaon モデルに基づき素性の集合によって表現すれば良いといえる。[2] また、現実的には、ID 素性を利用することで高速な名前解決が実現できる。いずれにせよ、あるオブジェクトに対応する Wiki 頁が唯一のユニークな URI を持つことを諦めるだけで、オブジェクトに対するユニークな ID なしに、ID 素性名と素性値の組（あるいは、素性名と素性値の組の集合）によって Wiki 頁を指定することは可能である。

URI 中におけるオブジェクトの表現は、具体的には、「ID 素性名:素性値」という形式によって表現することにする。この素性名には CHISE-Wiki と同様の URI 用の特殊文字の変換規則を用いる。[4] 例えば、例示オブジェクト素性 *=foo* は「*rep.foo*」、抽象オブジェクト素性 *=>foo* は「*foo*」のように表現される。素性値の表現形式も CHISE-Wiki と同様であり、

10 進整数 $[+-]^{?}[0\text{-}9]^{+}$ [例] 1234, -3, +25

16 進整数 "0x" $[0\text{-}9A\text{-}Fa\text{-}f]^{+}$; [例] 0x77E5, 0x4e16

10 進浮動小数点小数 ($[0\text{-}9]^{+}$ | $[0\text{-}9]^{*}$ "." $[0\text{-}9]^{+}$)("e" $[+-]^{?}$ $[0\text{-}9]^{+}$)$^{?}$; [例] .1, 3.14, 5e-3, 4.1e23

シンボル $[A\text{-}Za\text{-}z\$\_*@]$ $[0\text{-}9A\text{-}Za\text{-}z\$\_*@+-/]^{*}$

但し、ここで、

- [...] は ... で指定された文字のどれか
- $exp^{?}$ は *exp* が省略可
- exp^{*} は *exp* の 0 個以上の連なり
- exp^{+} は *exp* の 1 個以上の連なり

を表す

といった表現を用いる。

5.3 URI における関係素性名の表現

HTML, XML では <, > をエスケープする必要があるので、%<16 進>記法とは別に、CHISE-Wiki と同様の関係素性の URI 中における表現形式を設けることにする：

`from.`*foo*	関係素性 *<-foo* の URI 中での表現
`to.`*foo*	関係素性 *->foo* の URI 中での表現

[2] 入れ子状になる場合、XRI を用いれば良い。

5.4 ジャンル

Concord ではオブジェクトの種類を表す『ジャンル』(genre) という概念があり、文字は character ジャンル、素性は feature ジャンルに属し、それ以外のさまざまなジャンルを定義することができる。各ジャンルのオブジェクトは同様な素性の集合となっており、ジャンルをまたがってオブジェクトを参照することもできる。

ジャンルは URL 上では、対象となるオブジェクトを指し示すクエリー (query) 中のパラメーターのキー部として表現されている。CHISE-Wiki では対象となる文字オブジェクトを

prefix/view.cgi?char=*object*

[例] http://chise.org/chisewiki/view.cgi?char=字

のように URL で表現していたが、EgT ではこれを拡張して、

prefix/view.cgi?*genre*=*object*

[例] http://chise.org/est/view.cgi?era@ruimoku=rep.ruimoku-era-code:11210

のように表現する。

5.5 素性オブジェクト

EgT では素性は feature ジャンルのオブジェクトとして扱われており、他のオブジェクトと同様に素性の集合からなり、他のオブジェクトと同様に対応する頁でその定義を閲覧・編集することが可能である。

あるオブジェクトの定義で用いられている素性名や素性値の表示や編集用の形式は、その素性の定義に依存しており、その素性オブジェクトの素性を編集することで振舞を変えることができる。これを『メタ編集』と呼ぶ。

5.5.1 素性値の書式指定

あるオブジェクトに関する Wiki 頁において、ある素性の値を表示する際、その素性の素性 value-format の値で設定された書式指定に従うこととする。

5.5.2 素性名の書式指定

あるオブジェクトに関する Wiki 頁において、ある素性の名前を表示する際、その素性の素性 name の値を表示用の名前として用いることとする。

name の値が存在しない場合、その既定値として、素性名から機械的に整形した文字列を用いることとする。

5.5.3 素性対の書式指定

あるオブジェクトに関する Wiki 頁において、ある素性の情報を表示する際、その素性の素性 format の値で設定された書式指定に従うこととする。

素性の形式は、5.5.1 節で述べたリスト形式を用いる。すなわち、単純な文字列のリストを書くことも可能であるが、link 関数を用いてハイパーリンクを表現することも可能である。但し、組み込みタグ関数は 5.5.1 節で述べたものに加え、表示用の素性名を返す関数 name も利用可能となる。

素性の素性 format の値が設定されていない場合、その既定値として、

```
((name) ":" (value))
```

を用いる。ここで、(value) の返り値は、その素性の値を素性の素性 value-format の値に基づき整形したものとなる。value に対して属性 :format を指定すれば、その値が素性の素性 value-format の値の代わりに書式指定として用いられる。例えば、(value (:format HEX)) と書けば、素性値を 16 進数にしたものが返る。

5.6 ジャンル固有の情報

5.6.1 オブジェクトの表示名

書式指定としてリスト等を指定した場合、Wiki 頁中で各オブジェクトは対応する Wiki 頁へのリンクとなるが、そのリンクを表す文字列はオブジェクトのジャンルの素性 object-representation-format の値によって決定される。この値の形式は書式指定 (XML 抽象構文木) であり、その既定値は name である。すなわち、特に指定がない限り、オブジェクトの name 素性の値がオブジェクトを代表する表示 (アイコン的なもの) として用いられる。

5.6.2 ジャンル固有の書式指定

書式指定をジャンル毎にカスタマイズしたい場合がある。例えば、文字ジャンルでは素性 name の値を <*name*> にしたいが、通常はそのまま表示したいという場合、素性 name の素性 value-format の値をジャンル毎に変える必要がある。

このため、階層的素性名方式 [7] に基づき、ジャンルを示すドメイン $genre=*genre* を用いる。例えば、上記の例の場合、素性 name の素性 value-format の代わりに value-format@$genre=character に文字ジャンル固有の設定を行うことで解決できる。

6 内部表現と表示

6.1 書式指定と内部表現

Wiki 頁の内部表現としては Wiki 記法そのものを用いることが考えられるが、EgT では CHISE-Wiki と同様に、内部表現として XML の構文木に相当する S 式による汎用的な表現を用いる。

この内部表現ではノードを文字列、もしくは、

(タグ名 属性リスト 子ノード$_1$ 子ノード$_2$ ...)

という形式で表現する。ここで、この後者の形式を『XML 抽象構文木』と呼ぶことにする。

この XML 抽象構文木のタグ名はシンボルである。そして、属性リストの属性名は :*foo* という形式を用いることとする。また、子ノード$_n$ は文字列、もしくは、XML 抽象構文木をとるものとし、子ノードを持たないことも認めることとする。例えば、

(link (:ref "http://cvs.m17n.org/") "これはリンクです")

は

```
<link ref="http://cvs.m17n.org/">これはリンクです</link>
```

に相当する。また、

(link (:ref ("http://www.unicode.org/cgi-bin/GetUnihanData.pl?codepoint="
(HEX))) "U+" (HEX))

という風に、属性リストの属性値として子ノード列と同様なリストも認めることとする。なお、XML にする場合、

```
<link
ref="http://www.unicode.org/
cgi-bin/GetUnihanData.pl?
codepoint=<HEX/>">これはリンクです</link>
```

とは書けないので、

```
<link
><ref>http://www.unicode.org/
cgi-bin/GetUnihanData.pl?
codepoint=<HEX
/></ref>これはリンクです</link>
```

と書くこととする。

実際には、通常、この XML 抽象構文木そのものではなく、前述のノード（XML 抽象構文木、または、文字列）のリストが素性値、あるいは、素性の素性（素性属性）の値として格納される。例えば、

```
<link><ref><name-url/></ref>=U</link
>+<link><ref><unihan-url/></ref
><HEX/></link>(<decimal/>)
<prev-char/> <next-char/>
```

は

((link (:ref ((name-url))) "=U")
"+" (link (:ref ((unihan-url))) (HEX))
"(" (decimal) ")" (prev-char) " " (next-char))

と表現される。

6.2 表示の仕組み

EgT の表示は、

1. 『簡約』
2. 『表示』

の２つのフェーズによって実現されている。

『簡約』というのは、書式指定の XML 抽象構文木中の関数のように、素性値等によって動的に決定され

るような部分を実行時の環境に基づいて決定することであり、Lisp 言語等での eval (evaluation;『評価』) に相当するものである。『簡約』の結果、実際の素性値が埋め込まれた XML 抽象構文木が返る。

『表示』というのは、『簡約』の結果得られた XML 抽象構文木を最終出力の形式に変換することで、Lisp 言語等での print (『表示』) 相当するものである。出力形式としては、現在の所、HTML とプレインテキストが実装されている。

この『簡約』と『表示』という2つの過程を繋げることにより、最終的な頁の表示が実現される。

7 想定される利用法

ある程度構造化されているが形式が完全には揃っていないようなデータや、形式が完全には把握できてないようなデータは、一般的な関係データベースにとって苦手なタイプのデータだといえる。こうした場合、通常、把握できている構造に限ってスキーマを定義して関係データベース化するか、構造に関する情報の利用を諦めて全文データベース化する訳であるが、どちらの場合も取り敢えずの検索は実現できたとしても、形式化作業自体は支援できない。こうした問題は十分に形式化・モデル化・ドキュメント化されていない状況で入力作業を行った場合や、長期間にわたって続いたデータベース・プロジェクトなどでしばしば見られる現象であるが、こうした状況を解決するには多大な努力を必要としたり、あるいは、メンテナンスのための労力が追い付かず、プロジェクトが破綻することもあった。EgT が狙っているのはこうした分野における支援である。

EgT で想定している利用法は次のようなものである:

もし、ある程度構造化されたデータが存在していれば、とりあえず、素性の集合からなるオブジェクトとして Concord に載せる。例えば、表形式になっていれば、行をオブジェクト、列を素性と看做し、列に適当な素性名 (なるべくユニークな名前) を付ける。Concord に載せることができたら、EgT で開くことができるので、素性オブジェクトに名前を付けたり、書式指定をしたりして、見やすい形の表示形式を作っていく。また、素性値が何かのコードや ID 等になっていた場合、そのコードや ID を ID 素性としたオブジェクトを定義し、コードの代わりにそのオブジェクトを値とする素性を設ける。また、そのオブジェクトに名前 (name 素性等) を付ける。そうすると、EgT の画面上ではオブジェクトへのリンクとなり、そのリンクではオブジェクトに付けた名前が表示されることになる。

このようにして、常にプロトタイプ画面を見ながら、データの形式化やデータの修正を行うことができる訳である。

8 今後の課題

8.1 版管理

2 節で述べたように、データベースの形式化作業の変遷に関する情報は非常に重要な情報であり、これをなるべく機械可読な形で蓄積することは重要であるといえる。このためにはメタ編集の過程を記録する必要があり、版管理機構を導入する必要があるといえる。

8.2 プログラマブル化

データベースのリファクタリングを行う場合、テストケースやデータを書き換えるプログラム等が書けると便利である。このためには、条件分岐やイタレーター等の制御構文のための関数を導入することが考えられる。こうした制御用関数は柔軟な書式指定を実現する上でも有用であろう。

制御用関数を導入すれば、オブジェクトのメソッドが書けることになり、EgT をオブジェクト指向言語として利用することができるかも知れない。[3] こうしたプログラマブル化を行えば、実際には素性が存在しなくても、計算によって素性値を返すようなメソッドを書くことができ、素性の抽象化が可能となる。即ち、プログラマブル化は、実際の素性名や素性値の形式といったデータの実装とデータの利用者に対するインターフェースを分離し、データ抽象を進める上でも有用であると考えられる。

[3] セキュリティー上の問題を引き起こし得るので、利用可能な権限の適切な制御が必要であろう。

8.3 素性の変更・除去支援

漸進的なデータの改良を行う場合、素性の名前や形式をいきなり変えるのではなく、見掛け上の名前を付けたり、別名を付けたり、同じ情報を持つ別形式の別素性を追加する方が望ましいといえるが、ある程度データの整理が進んだり、あるいは、逆に、あまり作業が進んでない段階では、物理的な名前の付け替えを行った方が良い場合もある。Concord / EgT では素性はそれぞれ別のファイルとなっているので、素性名の付け替え自体は容易である。ただ、どこかのオブジェクトで、書き換え対象となる素性名を参照している場合、参照が切れてしまい問題である。

この問題を解決するためには、

1. 各オブジェクトを走査して、書き換え対象となる素性名を書き換えて回る
2. 別名を付ける
3. 改名の履歴を記録しておく

といった方法が考えられる。

また、使われていない素性やオブジェクトを検出して削除するガーベージコレクターを実現することも考えられる。この際、良く使われるデータをすぐに参照可能な場所に置き、あまり使われないデータは版管理システムにおける履歴として実現することで、良く使われるデータのセットをコンパクトにする世代型ガーベージコレクター化も有用であるかも知れない。同様に、プログラマブル化と併用し、メソッドプログラムとして実現した抽象素性の実行結果を素性値としてキャッシュするようにすれば、頻繁に利用されるデータ形式を考慮した形に自動的にマスターデータを書き換えるという手法も使えるかも知れない。

9 おわりに

構造データを Wiki 的に編集するためのツールである **EgT** について述べた。

Wiki は WWW 上で利用者が容易にコンテンツを編集することができるという点で優れており、この性質は情報をリファクタリングして行く上で有効であると考えられる。しかしながら、一般的な Wiki は基本的に人間が読むためのテキストが主体であり、構造を持った機械可読な情報が主体の場合に、自由度や表現力、拡張可能性を損なわずに簡単に編集できるようにすることはあまり容易ではない。そこで、CHISE-Wiki [4] を拡張し、一般化した EgT の開発を行った。EgT では CHISE-Wiki と同様に『メタ編集』という手法が利用可能であり、これによって機械可読性や拡張可能性と人間にとっての編集のしやすさや可読性の両立を図っている。

EgT はまだまだ開発途上のシステムであり、EgT が取り扱っている問題は現状では非常に限られたものである。データの形式化やレガシーデータの問題にはもっとさまざまな側面からのアプローチが必要であると思われる。もし、この論文によって、この問題に興味を持ち取り組む方が増えたならば幸いである。

参考文献

[1] 河本健作，北村泰彦：Semantic Wiki による RDF 自動生成，第 10 回セマンティックウェブとオントロジー研究会，人工知能学会 (2005). SIG-SWO-A501-02.

[2] : Semantic MediaWiki, `http://semantic-mediawiki.org/`.

[3] : MediaWiki, `http://www.mediawiki.org/`.

[4] 守岡知彦：CHISE のセマンティック Wiki 化の試み，情処研報， Vol. 2010-CH-87, No. 8, pp. 1–8 (2010).

[5] Morioka, T.: CHISE: Character Processing based on Character Ontology, *Large-scale Knowledge Resources (LKR2008)*, LNAI, No. 4938, pp. 148–162 (2008).

[6] 守岡知彦：Concord: プロトタイプ方式のオブジェクト指向データベースの試み，Linux Conference 抄録集， Vol. 4 (2006).

[7] 守岡知彦：文字オントロジーに基づく文字処理について，情処研報， Vol. 2006, No. 112, pp. 25–32 (2006). 2006-CH-72.