人工知能学会研究会資料
SIG-SWO-A801-06

汎用オントロジー構築における日本語 Wikipedia の適用可能性

Applying Japanese Wikipedia for Building up a General Ontology

桜井 慎弥[*1] 手島拓也[*1] 石川雅之[*1] 森田 武史[*1] 和泉 憲明[*2] 山口 高平[*1]
Shinya Sakurai Takuya Tejima Masayuki Ishikawa Takeshi Morita Noriaki Izumi Takahira Yamaguchi

[*1] 慶應義塾大学 Keio University
[*2] 産業技術総合研究所 National Institute of Advanced Industrial Science and Technology

本研究では，日本語 Wikipedia から汎用オントロジーを構築する手法を提案し，実際に構築したオントロジーの品質を評価することによって日本語 Wikipedia の汎用オントロジー構築における適用可能性を検証する．日本語 Wikipedia から汎用オントロジーを構築するための手法として，クラス階層構築についてはカテゴリ階層に対する文字列照合，インスタンスの収集については一覧記事に対するスクレイピング，プロパティ定義およびクラスの同定については Infobox と記事の所属カテゴリの分析法を提案する．

1. はじめに

オントロジーの中でも幅広い分野の一般的知識を記述した汎用（言語）オントロジーは，現在では英語版としては WordNet[1]，日本語版としては EDR 電子化辞書[2]がよく知られており，セマンティック Web 研究における貢献度は非常に高い．しかし，これらのオントロジーは膨大な時間とコストをかけて人手で構築されているため，固有名詞も含め，日々生まれ出る新しい語彙定義への即時対応が難しいのが現状である．

そこで本研究では，即時更新性，語彙網羅性に優れたオンライン百科事典 Wikipedia[3]から汎用オントロジーを構築することを目的とする．多言語対応の Wikipedia の中でも，本研究では日本語 Wikipedia の半構造化された情報資源に着目し，これをオントロジーに変換する．本稿では，日本語 Wikipedia のカテゴリ階層に対する文字列照合を行うことによってオントロジーのクラス階層を構築し，一覧記事に対するスクレイピングを行うことによってインスタンスを収集する手法を提案する．また，Infobox を利用してプロパティを定義し，さらに定義域および値域を定義することによって，Wikipedia という集合知の中で大衆の合意を得たクラスの同定を試みる．ケーススタディとして実際に Wikipedia のデータを利用してオントロジーを構築し，構築されたオントロジーの品質についての評価をすることによって，日本語 Wikipedia の汎用オントロジー構築への適用可能性を検証する．

以下，2 節では関連研究として Wikipedia からの大規模オントロジー構築を行っている主な研究を紹介する．3 節では日本語 Wikipedia から汎用オントロジーを構築するための提案手法を述べる．4 節では提案した手法を実際に日本語 Wikipedia に適用し，汎用オントロジーの構築を行う．5 節で構築した汎用オントロジーの評価を行って日本語 Wikipedia の汎用オントロジーへの適用可能性を検証し，6 節で本研究の総括と今後の課題を述べる．

2. 関連研究

Wikipedia からオントロジーを構築する主な研究を紹介する．

[4]は，Wikipedia の半構造化情報を RDF に変換することによって，大規模なデータベースを構築した．リソースとしては主に，英語 Wikipedia の Infobox や外部リンク，所属カテゴリといった半構造情報を利用している．しかし，抽出した情報は特にフィルタリングされておらず，Infobox から抽出した情報に関しては不適切な情報も大量に含まれてしまっている．また，RDF の構築にとどまっており，階層構造は定義されていない．

[5]は，*Conceptual Category* と呼ばれるカテゴリをクラスとして利用し，WordNet を拡張している．*Conceptual Category* とは英語 Wikipedia のカテゴリであり，“American singers of German origin”カテゴリのように，カテゴリ名の *head* 部分である“singers”が複数形になっているカテゴリのことである．このようにして特定された *Conceptual Category* に属する記事をインスタンスとしてクラスに付加している．インスタンスに関しては，*BORNINYEAR* や *LOCATEDIN* といった *Relation* を用いてメタデータを記述し，非階層構造も構築している．[5]は大規模なインスタンスの収集を可能にしているが，その手法は英語 Wikipedia に特化した手法である．本研究では，日本語 Wikipedia に対応した手法を提案し，インスタンス収集を行っている．また，[5]では，Infobox を分析し，頻繁に利用されている 170 の属性を同定している．さらに，同定した属性から，手動でプロパティ及びその定義域と値域の定義を行っている．例えば，Born や Birthday という属性については，手動で定義した *BIRTHDATE* プロパティ（定義域は person，値域は timeInterval）と対応づけることにより，適切なファクトの抽出を行っている．本研究では，Infobox から抽出した項目（属性）を直接プロパティとして定義するため，[5]のように表記は異なるが同じ意味を表す属性を，同一プロパティとして定義することには対応できていない．しかしながら，Infobox を有する記事が属するカテゴリをクラスと仮定して，自動的にプロパティの定義域を定義することを試みている．さらに，プロパティの定義域を定義することによって，Wikipedia という集合知として大衆の合意を得たクラスの同定を試みる．

[6]では，英語 Wikipedia カテゴリ階層からの is-a 関係の抽出が試みられている．手法としては，カテゴリリンクに六つのメソッドを適用することによって関係を抽出している．メソッドの中でも主要なものである *Syntax-based methods* は，簡単な文字列照合によるものである．本研究では，日本語 Wikipedia に対して *Syntax-based methods* と同様の手法の適用を試みるとともに，前方文字列照合部除去という手法を取り入れることによって，is-a 関係の規模の拡大を行っている．

連絡先：桜井慎弥，山口高平，慶應義塾大学理工学部
〒223-8522 神奈川県横浜市港北区日吉 3-14-1
Tel: 045-566-1614，E-mail: {s_saku,yamaguti}@ae.keio.ac.jp

3. 日本語 Wikipedia からの汎用オントロジー構築

3.1 汎用オントロジー構築支援手法

Wikipedia からの大規模オントロジー構築については，クラス階層については主に[6]での英語 Wikipedia のカテゴリ階層からの簡単な文字列照合による抽出，インスタンスは[5]の英語に特化した手法での収集が行われているのが現状である．これらを踏まえ本研究では，クラス階層については日本語 Wikipedia のカテゴリ階層に対して[6]の *Syntax-based methods* の適用を試みるとともに，カテゴリ階層に新たな文字列照合を適用することによって規模を拡張し語彙の網羅性を高める．インスタンスについては日本語 Wikipedia に適用可能な手法として一覧記事へのスクレイピングによって大規模な収集を行う．さらに Infobox の項目名からプロパティを定義し，そのプロパティの主語が属するカテゴリを分析することによって Wikipedia という集合知におけるクラスの同定を試みる．図 1 は，日本語 Wikipedia からの汎用オントロジーの構築の概念図である．

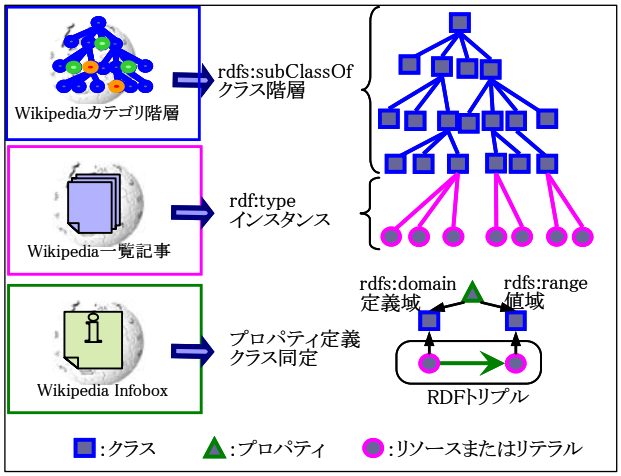

図 1　汎用オントロジーの構築の概念図

3.2 カテゴリ階層

カテゴリ階層とは，記事の分類を目的としたカテゴリが階層化されたものである．サブカテゴリは，あるカテゴリに属する記事が増加してくるとそのカテゴリの下位に作成され，カテゴリの記事の再分類に使用される．サブカテゴリは増加した記事を細分化するために作成されるという性質から，その名称は上位カテゴリの名称を含む複合語で形成される場合が多い．例えば「原子力ー原子力発電所」や「ソフトウェアーフリーソフトウェア」といった階層である．前者は単に関連の深いキーワードを示す is-a 関係としては不適切な関係であるが，後者はオントロジーの is-a 関係に相当する関係となっている．カテゴリ階層からは，この複合語に着目した文字列ベースでの抽出手法を用いることによって，is-a 関係を抽出できる可能性が高い．

3.3 一覧記事

一覧記事とは，物事のリストが記述された記事である．例えば，「言語の一覧」には世界の言語のリストが記述されている．文章表現を工夫したり細かな事実を確認したりする必要がないために，一覧記事の執筆者は非常に多いと考えられる．このため情報量は豊富であり，かつ記述形式がある程度統一されている．したがって，一覧記事から大規模なインスタンスを収集することが可能であると考えられる．

3.4 Infobox

Infobox は、テーブルを利用して Wikipedia の記事の属性（Wikipedia では主に“項目”と呼ばれている．）と属性値を整理して表示しているもので，記事の中にしばしば掲載されている．ここで使用される項目が，ドメインごとにある程度フォーマット化されているということが大きな特徴である．例えば「Java」の記事に掲載されている Infobox には“開発者”や“プラットフォーム”などの項目とそれぞれに対応する値が記述されており，この“開発者”や“プラットフォーム”という項目は，“プログラミング言語”Infobox の Template ページ[*1] で定められている．この，Infobox を有する記事－項目－値という三つ組は RDF トリプルと捉えることができ，主語と目的語である記事が属するカテゴリを調べることで，プロパティの定義域と値域を定義できる可能性があると考えられる．

3.5 クラス階層構築

本実験ではクラス階層構築手法として，カテゴリ階層の複合語に対する後方文字列照合と前方文字列照合部除去の2通りの文字列照合を提案する．

(1) 後方文字列照合

後方文字列照合とはカテゴリ階層を構成する親カテゴリ名と子カテゴリ名とを比較し，子カテゴリ名が“任意の文字列＋親カテゴリ名”となっているものを抽出する手法である．図 2 の例では“空港”という文字列の後方文字列照合により，“日本の空港”is-a“空港”という関係を得ることができる．この手法は，[6]で既に実践されている手法である．また，この手法は 1 世代の親－子カテゴリリンクだけではなく，N 世代離れたカテゴリリンクにまで適用することが可能である．例としては，“心理学”－“精神医学”－“分析心理学”といった親－孫のリンクからは“分析心理学”is-a“心理学”というリンクを抽出できる．

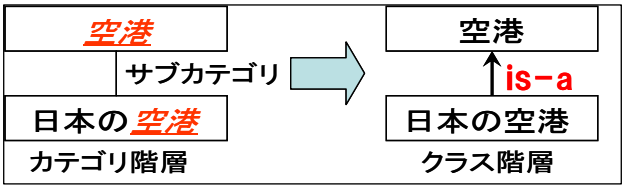

図 2　後方文字列 照合

(2) 前方文字列照合部除去

前方文字列照合部除去とは親カテゴリ名と子カテゴリ名とを比較し，親カテゴリ名と子カテゴリ名で“任意の文字列＋の”という部分が先頭から一致しているものを抽出，照合部を除去する手法である．図 3 の例では“日本の”という前方文字列照合部を除去することにより，“ゴルファー”is-a“スポーツ選手”という関係を得ることができる．この手法は，文字列の重複に依存しない is-a 関係を取得できる点が大きな利点である．この手法も N 世代離れたカテゴリリンクにまで適用することが可能である．

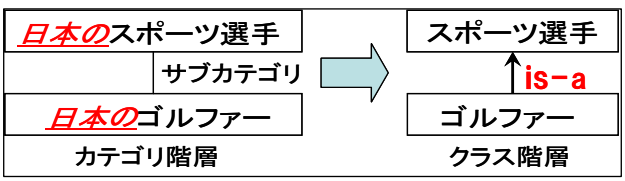

図 3　前方文字列照合部除去

[*1] http://ja.wikipedia.org/wiki/Template:Infobox_プログラミング言語

3.6 インスタンス収集

一覧記事はインスタンスを収集するためには不要な情報を含んでいる. しかし一覧記事は記述形式がある程度統一されているため, 含まれる不要情報もパターン化することができ, 除去することも十分可能である. 本実験では一覧記事から不要な情報を取り除くための手法としてスクレイピングを行うことによって, インスタンスを収集する.

3.7 プロパティ定義およびクラスの同定

図 4 のように, Infobox から生成される RDF トリプルの中から同じ項目名を持つものを収集し, これをプロパティとする. 図4のクラス, プロパティ,リソースまたはリテラルの記号は, 図 1 のものと対応している. それぞれのプロパティごとに, 主語となっている記事が属するカテゴリ(図 4 の A,B,C)の出現回数を算出する. 出現回数の多いカテゴリ(図 4 の A)をそのプロパティの定義域クラスと定義することによって, Wikipedia という集合知において大衆の合意を得たクラスの同定を試みる. Infobox のプロパティに対する主語は記事であるために必ず所属カテゴリを調べることが可能だが, 目的語部分は他記事へのリンクがなされているリソースである場合と, リンクが貼られていないリテラルである場合がある. したがって, 目的語に関しては属するカテゴリの出現回数を正しく算出することが難しいため, 本研究では主語を利用してクラス同定を行う.

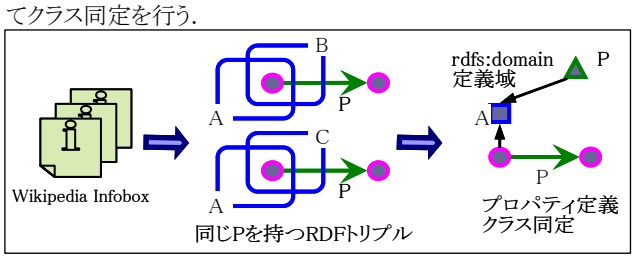

図 4　プロパティ定義およびクラスの同定

4. 実装

Wikipedia の全記事, 内部リンク, カテゴリリンクなどはフリーでダウンロードすることができるため(これらのデータはダンプデータ[*2]と呼ばれる), これを利用してクラス階層の構築とインスタンスの収集を行う.

4.1 クラス階層構築

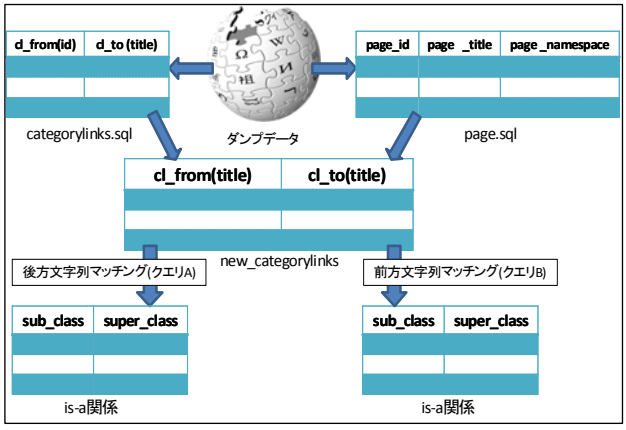

図 5　クラス階層抽出

図 5 に具体的なクラス階層抽出の手法を示す. ダンプデータの categorylinks.sql で表わされるテーブルには, それぞれ *cl_from* と *cl_to* のカラムに全記事とそれが所属するカテゴリの対応が表わされている. しかしこれはカテゴリ同士のリンク以外にもカテゴリと属する記事とのリンク関係も含んでおり, さらに記事のカラムは実際の記事名を表す文字列ではなく id で表わされている. ここでダンプデータの page.sql も利用する. page.sql は全記事の id と記事名と記事の namespace の対応を表している. namespace とは記事の属性を表すもので, カテゴリ記事の namespace の数は 14 である. categorylinks.sql と page.sql のテーブルを結合させ, 親カテゴリと子カテゴリの対応を表したテーブル *new_categorylinks* を生成する. このテーブルに図 6 に示した後方文字列照合を行うクエリ A, 前方文字列照合部除去を行うクエリ B を実行し, is-a 関係を抽出する.

・クエリA

```
SELECT super_class,sub_class FROM `new_categorylinks`
 WHERE sub_class REGEXP CONCAT('.*',super_class,'$')
```

・クエリB

```
SELECT
 TRIM(leading CONCAT(SUBSTRING_INDEX(super_class,'の',1),'の') FROM super_class),
 TRIM(leading CONCAT(SUBSTRING_INDEX(super_class,'の',1),'の') FROM sub_class)
 FROM `new_categorylinks`
  WHERE SUBSTRING_INDEX(super_class,'の',1)=SUBSTRING_INDEX (sub_class,'の',1)
   AND super_class LIKE '%の%'
```

図 6　文字列照合を行うクエリ

4.2 インスタンス収集

ダンプデータの pages-articles.xml は全記事の xml テキストファイルであり, 図 7 のようになっている.

以下, スクレイピングの具体的な内容を説明する.

(1)　大まかな不要情報の除去

図 7 の a の page タグ title タグを利用して, 一覧記事のテキスト以外を除去し, title タグ部分も除去する. 一覧記事では d のように‘＊’または‘＃’から始まる行(以下, ‘＊’行と呼ぶ)にインスタンスが記述されており, c のように‘=’で囲まれた部分にはインスタンスを分類する単語が記述されている(筆者はこれを目次見出しと呼ぶ). この c, d を残し, b の‘＊’や‘=’以外から始まる行を除去する. 図 7 中の“[[ ]]”は, Wikipedia の内部リンクを表している.

一覧記事の中には, ‘＊’や‘＃’を利用した箇条書きによる記述ではなく, テーブル形式でインスタンスを列挙している記事がある. 例としては“内閣総理大臣の一覧”がある. この記述形式は多数存在し, インスタンスがどのように列挙されているかのパターン化が難しい. 本実験では、テーブル形式でインスタンスが記述されている一覧記事からの抽出は行わない.

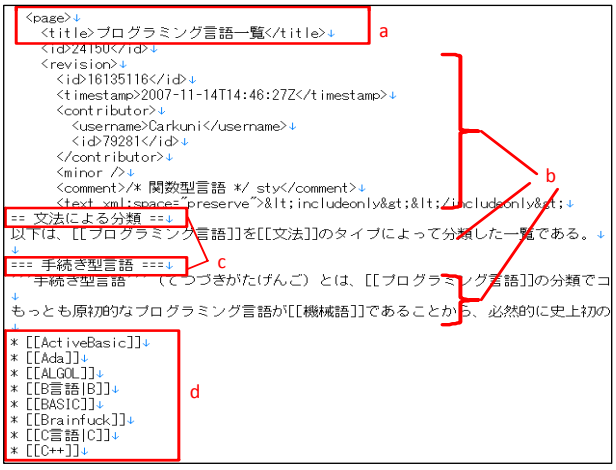

図 7　一覧記事ソーステキストの一部

[*2] http://download.wikimedia.org/jawiki/

以下からは図 8 に示される例のような，(1)で抽出した‘＊’行に存在する，正しいインスタンスを含んではいない行を除去していく．以下(2)～(6)は，図 8 の例に対応している．

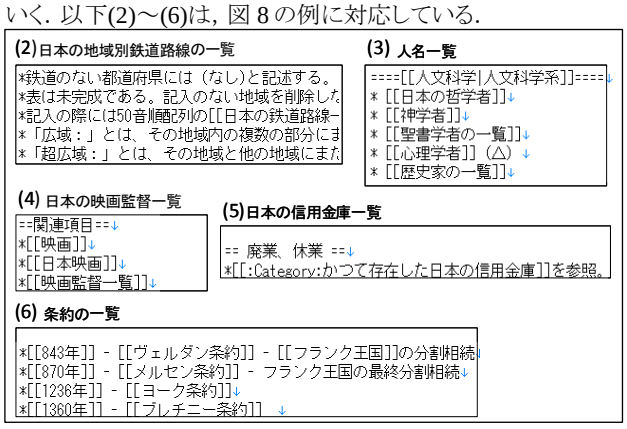

図 8　一覧記事の不要な情報の例

(2)　一覧記事の説明に使用される‘＊’行を除去

一覧記事の多くは，インスタンスを列挙する前に一覧の内容の説明を加えたり，類似した一覧記事へのリンクを紹介したりしている．図 8 のように，それらの記述の中で箇条書きを用いる場合も，‘＊’が使用されていることになる．この行の中には記事名に対するインスタンスは記述されていないため，除去する必要がある．このような‘＊’行は，タイトルと一つ目の目次見出しの間にある場合がほとんどであるため，その位置にある‘＊’行を除去すればよい．

(3)　「＊～一覧」と同じ目次見出しの下位にある‘＊’行を除去

図 8 を見ると，“日本の哲学者”は“人名”のインスタンスではないといったように，‘＊’行がタイトルに対するインスタンスになっていないことがわかる．一覧記事中には閲覧者の利便性向上のために関連の高いページへのリンクが列挙されている部分があり，これが‘＊’行として記述されてしまっている場合がある．このような‘＊’行の特徴は，同じ目次見出しに属する‘＊’行のどれかに“～一覧”という文字列を含んでいることである．図 8 でも，“聖書学者の一覧”という‘＊’行が同じ目次見出し以下に含まれていることがわかる．このように“～一覧”という文字列を含む‘＊’行を特定し、それが属する目次見出しに属する行をすべて除去する．

(4)　不要な目次見出し下位の‘＊’行を除去

ある特定の目次見出しに属する‘＊’行は，タイトルに対するインスタンスとしては誤りである．そのような目次見出しの条件は“関連”、“外部”、“備考”、“参考”、“related”、“凡例”、“カテゴリ”、“出典”、“特記事項”を含むことである．これらのキーワードを含む目次見出しに属する‘＊’行はすべて除去する．

(5)　不要な‘＊’行を除去

抽出した‘＊’行の中に未だ存在する不適な行で，行単位でパターン化できるものも存在する．不適な行のパターンとしては，「“＊[[Wikipedia:”から始まる」，「“＊[[:Category:”から始まる」，「“＃REDIRECT”から始まる」，「“＊人名　あ行”のように、五十音のインデックスをもつ」，「“＊[[～一覧]]”」，「“＊関連項目”」がある．このパターンに当てはまる‘＊’行を除去する．

(6)　不要な年号記述部分を除去

スクレイピング(7)で述べるが，インスタンスは‘＊’や‘＃’の直後に配置されていることが収集可能な条件である．図 8 のような年号の記述は除去し，そのような配置に修正する．

(7)　‘＊’行からのインスタンスの抽出

(6)までのスクレイピングで，ほぼ全ての‘＊’行の中には適切なインスタンスが記述されているという状態になった．最後に，‘＊’行の中でどの部分がインスタンスを表す文字列であるかを特定する六つのパターンを作成し，これに従ってインスタンス以外の部分をスクレイピングし，最終的に記事名で表されるクラスとインスタンスの残し，インスタンスを収集する．

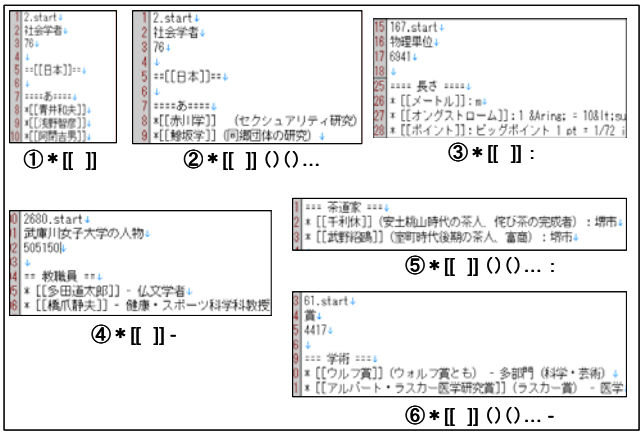

図 9　‘＊’行中でインスタンス箇所を特定するパターン

一覧記事では‘＊’や‘＃’の直後にインスタンスが配置されている行が圧倒的に多い．逆に‘＊’や‘＃’の直後ではない箇所にインスタンスが配置されている場合，その箇所の特定は難しい．また，‘＊’や‘＃’の直後といっても何文字目までがインスタンスの1語を表しているかの特定も難しい．このため，六つのパターンでは，基本的に‘＊’や‘＃’の直後のリンク記号“[[ ]]”に着目してインスタンス文字列を特定し，抽出を行う．

4.3 プロパティ定義およびクラス同定

まず 4.2 節と同様，ダンプデータの pages-articles.xml から Infobox 部を抽出する．記事名は前述のように title タグ利用して抽出する．Infobox 部は図 10 のように“{{Infobox (Template 名) }}”または“{{基本情報 (Template 名) }}”という形式を持った中括弧で括られており，項目名と値は“|(項目名)=(値)”と記述されている．筆者らはこの Template 名を“InfoboxType”と呼ぶ．記事，項目，値を RDF トリプルとして三つ組の形式で抽出してデータベースに格納し，これと 4.1 節の categorylinks.sql と page.sql のテーブルを利用して，プロパティごとに主語となっている記事が属するカテゴリの出現回数が多いものを検索する．出現回数が多いカテゴリは，Wikipedia という集合知の中で大衆の合意を得たクラスとして認めることができ，さらに対応するプロパティの定義域となり得ると考えることができる．

```
{{Infobox プログラミング言語
|名前 = Java
|パラダイム = [[オブジェクト指向プログラミング|オブジェクト指向]]
|開発者 = [[サン・マイクロシステムズ]]、[[Java Community Process]]
|登場時期 = 1990年代前半
|最新リリース = ([[Java Platform, Standard Edition|SE]]) Java SE 6 /
|型付け = 強い[[静的型付け]]
|プラットフォーム = [[Solaris]]、[[Linux]]、[[Microsoft Windows|Windows]]、ほか多数
|処理系 = 多数 ([[コンパイラ]]、[[Java仮想マシン|仮想マシン]])
|影響を受けた言語 = [[Objective-C]]、[[Smalltalk]]、[[C++]]、[[Eiffel]]、[[C Sharp|C#]]
|影響を与えた言語 = [[C Sharp|C#]]、[[D言語|D]]
|ウェブサイト = [http://java.sun.com/ java.sun.com]
}}
```

図 10　ソーステキスト中の Infobox 部

5. ケーススタディ

5.1 実験方法

2007 年 11 月現在のダンプデータをダウンロードして，オントロジーの構築を行った．データベースには MySQL，スクレイピングを行うプログラムの実装言語には Java を使用した．

(1) クラス階層構築

カテゴリ階層を構成しているリンクの数は 87,126 個であった．

後方文字列照合については 1 世代の親子関係だけではなく，2 世代のカテゴリリンクまで検索の対象を広げて適用し，クラス階層の抽出を行った．前方文字列照合部除去については，“日本の官公庁”－“法務省”－“日本の検察官”から抽出される“官公庁”－“検察官”のように，中間カテゴリを経由すると抽出精度がかなり落ちてしまうため，2世代以上離れたリンクからの抽出は行わない．クラス階層の評価方法は，抽出した全リンクからのランダム標本抽出によるリンクの正解率の区間推定を行う．正解の判断は，下位概念が上位概念の性質を継承しているか否かという点を基準とした．正解率の 95%信頼区間の算出式として，有限修正を加えた以下の式①を利用する．

$$\left[\ \hat{p}-1.96\sqrt{\left(1-\frac{n}{N}\right)\frac{\hat{p}(1-\hat{p})}{n-1}}\ ,\ \hat{p}+1.96\sqrt{\left(1-\frac{n}{N}\right)\frac{\hat{p}(1-\hat{p})}{n-1}}\ \right] \quad \cdots①$$

式①において，N は母数，n は標本数，$\hat{p}$は真の正解率の推定量であり，正解の標本数を総標本数で割ったものである．

(2) インスタンス収集

Wikipedia の一覧記事数は約 8,300 ページであった．

ダウンロードした xml テキストファイルに対してスクレイピングを行い，インスタンスの抽出を行った．インスタンスの評価方法もクラス階層の評価と同様に抽出したリンクから 1,000 個の標本抽出を行い，リンクの正解率の区間推定を行う．標本数に対して母数が非常に大きい場合は，使用する式は 5.1 節(1)の式①から有限修正部を除いたものとする．

(3) プロパティ定義およびクラス同定

Infobox データに関しては 2008 年 5 月現在のダンプデータをダウンロードして抽出し，これを利用して“Released”，“開発元”の二つのプロパティについて，主語となっている記事が属するカテゴリの出現回数をそれぞれ求めた．

5.2 実験結果

(1) クラス階層構築

後方文字列照合によって 4,671 個，前方文字列照合部除去によって 2,521 個，計 7,192 個の is-a 関係を抽出した．この is-a 関係を構成する概念数は 6,672 個であった．抽出した 7,192 個の母集団の中から 1,000 個の標本を抽出し，正誤を判定した．その結果から式①を利用して真の正解率の 95%信頼区間を算出すると，91.2±1.63%という結果が得られた．表 1 および表 2 にそれぞれ後方文字列照合，前方文字列照合部除去で抽出されたリンクの例を提示する．表 3 は誤りの例とその内容を表している．

本研究では，抽出した個々のリンクから階層構造の構築を試みたため，階層構造が一つのルートノードに集約されず，浮いてしまった多数のルートノードが存在するという結果となった．ルートとなっている各クラスから全てのリーフのクラスへのパスを調べたところ，抽出したパスの本数は 54,208 本であり，構造全体の階層の深さの平均は約 6.1741 で，分散は約 3.65 であった．

図 11 は，それぞれのルートノードからは，どれほどの深さのパスが何本存在するかを表したグラフである．

表 1　後方文字列照合で抽出した is-a 関係の例

親クラス	子クラス
高等学校	通信制高等学校
高速道路	各国の高速道路
高速鉄道	台湾高速鉄道
魚介料理	日本の魚介料理
魚類	軟骨魚類
鳥類	絶滅鳥類

表 2　前方文字列照合部除去で抽出した is-a 関係の例

親クラス	子クラス
食品メーカー	製パン業者
武器	刀剣
麺料理	焼きそば
齧歯類	ハムスター

表 3　is-a 関係の誤りの例

親クラス	子クラス	間違いの内容
グローバリゼーション	反グローバリゼーション	反･非などを含む
文庫	富士見ミステリー文庫	クラスーインスタンス
地理	建築物	抽象的な語が親
教育	コミュニティ･カレッジ	抽象的な語が親
教育の歴史	旧制教育機関	抽象的な語が親
文化	アニメ作品	抽象的な語が親
歴史	政治	抽象的な語が親
社会	事件	抽象的な語が親
経済	国立銀行	抽象的な語が親

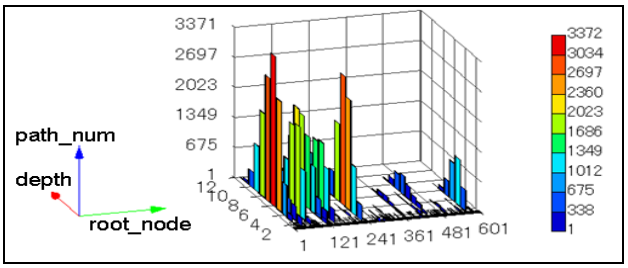

図 11　オントロジーの全体像

(2) インスタンス収集

ダンプデータから得られた一覧記事のソースには，722,753 行の‘＊’行が存在していた．4.2 節 (1)～(5)のスクレイピングを行った結果，‘＊’行は 639,349 行にまでスクレイピングされた．ここからさらに 4.2 節(7)のスクレイピングを行った結果，最終的に得られたインスタンスは 331,535 個，一覧記事の記事名から生成されたクラス数は 2,265 個であった．標本抽出をして正誤判定をした結果，正解率の 95%信頼区間は，96.9±1.1%であった．表 4 に抽出されたリンクの例を示す．表 5 は，インスタンスを多く持つクラスを表したものである．また誤りにはどのようなものがあったかを，表 6 に示す．

表 4　抽出したインスタンスの例

クラス	インスタンス
国会議員	松田竹千代
作曲家	本田雅人
映画スタッフ	高村倉太郎
国鉄･JRの車両形式	ヤ230
陸上競技選手	山田宏臣
神社	仙台東照宮
東北地方の道路	八戸南環状道路

表 5　インスタンスを多く持つクラスの例

クラス	インスタンス数
東京大学の人物	3901
日本の峠	3131
日本の漫画家	2472
アメリカ海軍駆逐艦	2144
日本の声優	2125
愛知県出身の人物	2001
人名 い	1768

表 6　インスタンスの誤りの例

クラス	インスタンス
言語	:en:Ngumba language
國學院大學の人物	伊藤誠（経済学者）
世界の民族衣裳	台湾
国際競技連盟	バイアスロン
スポーツ競技	サッカー

(3) プロパティ定義およびクラス同定

ダンプデータから得られた Infobox は 50,620 個であった. プロパティの種類は 1,287 種類, InfoboxType 数は 636 個であった. プロパティ, InfoboxType として用いられた回数が多いものをそれぞれ表 7 および表 8 に示す. また, 表 7 中に登場する“Released”プロパティと, さらに“開発元”プロパティについて, その主語となっていた記事が所属するカテゴリとして出現回数が多かったものをそれぞれ表 9 および表 10 に示す.

表 7　使用された回数が多いプロパティ

プロパティ	使用された回数
Name	16204
Background	13212
genre	11576
Label	11310
Artist	10515
Released	10500
Type	10482
Thisalbum	8938

表 8　使用された回数が多い InfoboxType

InfoboxType	使用された回数
Album	11258
会社	8757
Film	3766
Musician	2870
animanga/Footer	1770
baseball	1763
animanga/Header	1726
road	1395

表 9　“Released”プロパティの定義域候補

カテゴリ	出現回数
シングル関連のスタブ項目	1478
アルバム関連のスタブ項目	1059
コマーシャルソング	291
オリコンシングルチャート1位獲得作品	283
2007年のシングル	233
テレビドラマ主題歌	212
アニメソング	201

表 10　“開発元”プロパティの定義域候補

カテゴリ	出現回数
コンピュータ関連のスタブ項目	130
フリーソフトウェア	94
オープンソース	82
ソフトウェア関連のスタブ	79
フリーウェア	61
Mac_OS_Xのソフトウェア	54
プレイステーション2用ソフト	45

5.3 考察

(1) クラス階層構築

全体的な正解率という点ではかなり良い数値を得られたと思われる. 汎用オントロジーとしての階層の規模としてはまだ小さいものの, 後方文字列照合では複合語からなる is-a 関係を抽出できており, 前方文字列照合部除去では文字列に依存しない is-a 関係を抽出ができていることがわかる.

次に誤りの内容について考察する. 表 3 の一つ目の誤りにように, 子クラスが親クラスと照合していても「反」や「非」などの否定語が子クラスの先頭にきている関係は is-a 関係としては誤りになってしまう. 次に二つ目の誤りは, インスタンスを表してしまっている誤りである. Wikipedia では, 有名なものであれば明らかにインスタンスであるものでもカテゴリ化され, クラスのように扱われる傾向があり, 後方文字列照合ではこのような誤抽出をしてしまう可能性が高くなっていた. 表 3 は,「抽象的な語が親クラスとなっている」という要因の誤りが大部分を占めていたということも示している. ここでいう「抽象的な語」とは, Wikipedia カテゴリ階層上層部のカテゴリ名を名称の一部に持つようなカテゴリである. 日本語 Wikipedia のカテゴリ階層の最上層部は, 上位オントロジーと呼ばれるもののように概念の厳密な分類がなされているわけではなく, “主要カテゴリ”と呼ばれる“科学”, “学問”, “技術”, “自然”, “社会”, “地理”, “人間”, “文化”, “歴史”, “総記”の 10 種類がルートカテゴリとなっている.「抽象的な語」による誤りを多く抽出してしまった理由は, Wikipedia では主要カテゴリが分類の基幹となっているため, 上位部分以外の階層の中でも複合語の形でこの抽象的な語による分類が何度も登場し, それが前方照合部除去の条件に適合したためだと考えられる.

オントロジーの全体像である図 11 を見ると, あるクラスの部分だけ急激に階層が深くなっていることがわかる. 急激に深くなっているルート概念は, “文化と歴史”, “地理”, “事物”, “社会”, “歴史”, “文化”といった, 抽象的な概念である. このような単語以外で僅かながら深い階層が見られるのは, “機器”, “法”, “スポーツ組織”, “人物”であった. これら以外は深さ 2～4 程度の平坦な階層しか構築されていない. Wikipedia 主要カテゴリとなっているような抽象的な語が誤りの多くを生み出していることが明らかになっていたが, 加えて図 11 から, そのような語からしか深い階層構造を築くことができていないということもわかる. このことから, カテゴリ階層から文字列照合を利用してクラス階層を構築する場合, 上位部分の整備が極めて重要であるといえる. 上位オントロジーの整備によって今回構築したクラス階層がさらによりよいものになる可能性がある.

最後に, 抽出した is-a 関係から生成されたルートノードからリーフノードまでのパス（以下 Wiki-path と呼ぶ）を, 日本語語彙大系[7]と EDR の 2 種類の既存のオントロジーと比較した結果について述べる. 表 11 は, 三つのオントロジーのパスの比較例を表したものである. W は Wiki-path, N は日本語語彙大系, E は EDR 電子化辞書を表している.

まず具体物について述べる. 具体物はインスタンスが身の回りに多く存在し, イメージもしやすいので, Wikipedia には多くの記事が存在する. Wiki-path もこれに影響を受け, 多くのパスが存在する. これまでの考察からも明らかになっていたように, 構築したオントロジーは他の概念体系辞書と比較して上位概念の統制がとれていないということが表 11 の①～③からもわかる.

Wiki-path には具体物の中でも人物に関するパスが多く存在する. Wiki-path の人物に関するパスは, 他のオントロジーと比較すると数多くの特徴を持つ. まず表 11①の Wiki-Path は“人物”のすぐ下位の概念に“音楽家”が来ているが, EDR や日本語語彙大系には“人”や“人間”から“音楽家”に至るまでに多くの中間概念を介しているということである. 次に, Wiki-path に登場するクラスは, 多くの固有表現を持つクラスがほとんどであるということである. 表 11①では“音楽家”というクラスが生成されているが, 音楽を鑑賞する側の“聴衆”というクラスは生成されていない. 他の例を挙げると, “君主”や“医者”というクラスは存在するが“市民”や“患者”というクラスは生成されていないということである. これは一見欠点に感じられるかもしれないが, Population を多く持つ概念をクラスとして定義するオントロジーを構築する際には Wikipedia は有用であると考えることができる. また, 人物に関する Wiki-path で, ある程度の深さを持つものは, 職業による細かな分類がなされているものがほとんどであるという特徴も見られた. Wikipedia カテゴリ階層では人物を分類する

表 11　Wiki-path と他のオントロジーのパスとの比較

① 具体物 人間	W N E	事物－人物－音楽家－演奏家－ギタリスト－ジャズ・ギタリスト 名詞－具体－主体－人－人_職業_地位_役割－人_職業－人_専門的技術的職業－芸術家－音楽家 概念－物事－もの－具体物－生命体－人間－職業,肩書,役割で限定した人間－役割で捉えた人間 －職業で捉えた人間－芸術に身を投じている人－音楽における役割で捉えた人間－歌手－音楽家
② 具体物 人工物	W N E	文化と歴史－出来事－政治－行政－軍事－兵器－航空兵器－空対空ミサイル－イギリスの空対空ミサイル “兵器”の類はなし 概念－もの－ごともの－具体物－静物－機能で捉えた具体物－器具－戦争に使う道具－投げつけたり投下したりして爆発させる弾丸状の兵器－ミサイルという飛行兵器
③ 具体物 生物	W N E	事物－生物－動物－哺乳類－馬－競走馬 なし 概念－ものごと－もの－具体物－生命体－動物－脊椎動物に属する動物－馬－競馬における,役割や評価で捉えた馬 －競走馬
④ 具体物 地形	W N E	地理－地形－山－火山－海底火山 名詞－具体－場所－自然－地勢－地形－陸地－山 概念－位置－場所－地域－地形でみた場所－陸地－山－海底火山
⑤ 抽象物 時間	W N E	“過去”は未定義 名詞－抽象－抽象的関係－時間－非暦日－現在_過去_未来－過去 概念－時－時間点－時期－現在,過去,未来－過去

際には職業を基準に分類される場合が多く，性質属性（善人、悪人、変人･･･）などに関しては細かな分類がなされないという性質があるためと考えられる．職業による分類という点について他のオントロジーと比較すると，Wiki-path にはデザイナーからはさらに衣装デザイナー，グラフィックデザイナー，インテリアデザイナー，スキー選手からはさらにジャンプ選手，クロスカントリー選手，フリースタイル選手などに分類され，このような細かな分類は他のオントロジーには存在しない．

具体物の人工物に関してもおもしろい特徴を得ることができた．人工物といっても範囲が広いが，その中でも自動車や航空機といった輸送機器は乗り物として人気が高く、多くの Wikipedia 編集者を持つために記事が充実し，その結果カテゴリの階層も分類が細部にまでわたり易くなる．輸送機器以外にも "兵器"など，熱心な愛好家の多い分野では Wikipedia カテゴリからは細かい階層を取得できるということが表 11②からもわかる．日本語語彙大系では“兵器”という概念は定義されておらず，EDR よりも細かな分類が Wiki-path では実現されている．

この他には具体物の生物の分野では，日本語語彙大系は動物分類に関しては、獣・鳥・爬虫類・両生類・魚介類という分け方までだが，Wiki-path では表 11③のようにさらに細かな分類がなされている．地形の分野では，Wiki-path では固有表現が少ない田んぼ、空き地などは定義されていないものの，多くの固有表現を持つ地形（海、川、峠、滝）などはカバーされている．

具体物には Wiki-path ではさまざまな特徴が見られた．しかし抽象物は，Wikipedia で編集対象になりづらい大衆の関心の低い概念が多く，抽象物はカテゴリ化もされづらい．抽象物の記事も存在はするが，Wikipedia カテゴリの作成のされ方をみると，抽象物を的確に分類しようという考え方が重視されていない．したがって Wiki-path では抽象物はほとんど定義されていない．しかし今後は抽象的概念の定義についての議論が活発になり，多様なカテゴリによる分類が行われていくことも十分に考えることができる．

(2)　インスタンス収集

抽出したリンクの正解率を見ると，全体の抽出の精度はかなりの高さを持っていたことがわかる．表 5 を見ると，人物のインスタンス数が圧倒的に多いことがわかる．これは Wikipedia 一覧記事が人物のコンテンツを特に多く持つということをよく反映している結果である．しかし表 4 の例や，表 5 の “日本の峠”や “アメリカ海軍駆逐艦”など，分野にとらわれない抽出もできている．

誤りは，スクレイピングのルールが不足していることによるものが大部分を占めていた．表 6 の一つ目，二つ目の誤りはそれぞれ，Wikipedia の言語リンクを表す“:（言語コード）:”という記述を除去するルール，“( )”の注釈を除去するルールが不足していたために起こった．三つ目，四つ目の誤りは，‘＊’や‘＃’の行の中のどの部分がインスタンスそのものを表しているかを特定するためのパターンが不足していたために起こってしまった誤りである．ルールの不足以外にも，表 6 の五つ目の誤りのようにクラスーサブクラス関係を表してしまっているものもあった．

```
4139.start↓
千葉県の神社↓
856145↓
↓
== [[下総国]] ==↓
↓
=== [[香取市]] ===↓
*[[香取神宮]]「かとりじんぐう」香取市香取1697番地 '''官幣大社''' [[延喜式神名帳
*[[大戸神社]]「おおとじんじゃ」大戸521番地 '''県社'''↓
*[[木内大神（香取市）|木内大神]]「きのうちだいじん」木内1166番地 郷社↓
*[[豊玉姫神社（香取市）|豊玉姫神社]]「とよたまひめじんじゃ」貝塚117-1 郷社↓
*[[山倉大神]]「やまくらだいじん」山倉2347-1↓
*[[愛宕神社（香取市府馬）|愛宕神社]]（府馬） 「あたごじんじゃ」1971番地 郷社↓
873.start↓
アフリカの野鳥↓
101801↓
↓
== [[ダチョウ目（Sibley）|ダチョウ目]] [[w:Struthioniformes|Struthioniformes]] ==↓
↓
=== [[ダチョウ科（Sibley）|ダチョウ科]] [[w:Struthionidae|Struthionidae]] ===↓
# ''Struthio camelus'', [[w:Ostrich]], [[ダチョウ]]↓
# ''Struthio molybdophanes'', [[w:Somali Ostrich]], [[ソマリアダチョウ]]↓
464.start↓
クラシック音楽の作曲家（五十音順）↓
40206↓
↓
===配列のルール===↓
↓
==ア行==↓
*アースキン - [[トマス・アースキン]]（Thomas Alexander Erskine）↓
*アーネル - [[リチャード・アーネル]]（Richard Arnell）↓
*アーノルド - [[マルコム・アーノルド]]（Malcolm Henry Arnold; 1921-2006; [[イギリス]]）↓
*アーベル - [[カール・フリードリヒ・アーベル]]（Karl Friedrich Abel）↓
```

図 12　インスタンスを収集できなかった‘＊’行の例

4.2 節(1)～(5)のスクレイピングで，約 640,000 行の‘＊’行を特定できていたにもかかわらず，最終的に収集できたインスタンスは約 330,000 にとどまってしまっている．これは 4.2 節(7)のスクレイピングのルール不足によるものであるが，それでは，4.2 節(7)のルール以外にもどのようなパターンがあったかを検証していく．図 12 がその一例を表している．図 12 の一つ目の千葉県の神社の一覧記事では，約 3,000 個のインスタンスを収集し損ねていた．これは‘＊’の直後に内部リンクが付加されたインスタンスが配置されているものの，抽出ルールに当てはまらなかった例である．

“アフリカの野鳥”では，‘＃’の直後には学名が配置してあり，かつ“[[　]]”で括られているわけではなく強調文字を表す“''　''”で括られている．本来最も抽出したかったはずの野鳥の日本語名は，コンマ区切りの最後尾に配置されてしまっている．“～の野鳥”という名称の一覧記事からは，少なくとも 15,000 個のインスタンスを抽出し損ねていた．

クラシック音楽の作曲家の一覧記事では，‘＊’の直後には作曲家のファミリーネームが配置されており，ハイフンの後に続けて内部リンクを付加された日本語のフルネームが記述されている．

このように、一覧ページの記述には一般的な枠には収まらない様々なパターンが他にも存在する。より大規模に、正確に情報を抽出するのであれば、このような細かなパターンにも対応していかなければならない.

(3) プロパティ定義およびクラス同定

表 9 および 10 に多く登場する“～スタブ項目”カテゴリは, 内容的に未成長の記事が所属する, Wikipedia のメンテナンスに利用されるカテゴリである. このカテゴリは, カテゴリをクラスとみなすという観点では Wikipedia の記事に存在する典型的な不要カテゴリであり, これは容易に除去することが可能である. 表 9 では主に楽曲を表すカテゴリ名が, 表 10 ではソフトウェアを表すカテゴリが上位に来ていることがわかる. これから“Released”プロパティの定義域は“コマーシャルソング”や“作品”, “シングル”などと定義することが可能となり, 同様に“開発元”プロパティの定義域は“フリーソフトウェア”などと定義することが可能となる. 現時点ではこれらのプロパティのみの分析に留まっておりデータが出揃ってはいないが, 全てのプロパティについて主語の所属カテゴリの出現回数を調べることによって, 多くのプロパティの定義およびクラスの同定をすることが可能であると考えられる.

また, Template が存在しているということを考えると, InfoboxType は社会的に認知されたものであると言える. これはプロパティの定義域の定義以外にも, クラスを同定するための判断材料になる可能性が高い.

6. おわりに

本稿では, 汎用オントロジーを構築するために日本語 Wikipedia のカテゴリ階層からクラス階層を構築し, 一覧記事からインスタンスを収集する手法を提案した. さらに Infobox の項目をプロパティとし, その主語が属するカテゴリを定義域クラスとすることによってクラスの同定を試みた. 文字列照合では特に前方文字列照合部除去で文字列に依存しない is-a 関係を抽出することができ, 一覧記事からは日本語に対応した手法で大量のインスタンスを収集することに成功した. またその正解率も共に 90%を超える精度の高さであった. しかし質の高い汎用オントロジー構築のためには大きな課題が残されている. 複合語に着目した文字列照合は「抽象的な語」による多くの is-a 関係の誤りを生み, 文字列を利用した全自動構築には限界が見えたともいえる結果となった. この問題を解決するため, 今後は既存の上位オントロジー[8]を利用し, さらに人間とのインタラクションを取り入れながら半自動的に Wikipedia から汎用オントロジーを構築していく予定である. インスタンス収集に関しては, スクレイピングのさらなるパターン整理が必要になってくることに加え, 一覧タイトルから生成されたクラスをクラス階層に融合させるための手法も提案していかなければならない. また, 一覧記事以外にもインスタンス収集に利用可能なリソースを探していく必要もある. プロパティ定義およびクラスの同定に関しては, 未だ完全なデータ算出はできていないものの, Infobox を利用してプロパティを定義し,その主語が所属するカテゴリを利用して社会的に認知されているクラスを定義域として同定することが可能であるという一例を示すことができた. 今後は, 全てのプロパティについて主語の所属カテゴリの分析を続けていく予定である. また, 出現回数の多いクラスの, さらに一段階上位のカテゴリも視野に入れて分析することによって, より適切にプロパティの定義域を定義していくことに加え, InfoboxType との関連性の調査をすることも検討している.

本研究で構築したクラス階層とインスタンスに関しては, [9]で OWL ファイルと CSV ファイルを公開している.

参考文献

[1] WordNet : http://wordnet.princeton.edu/

[2] EDR 電子化辞書 : http://www2.nict.go.jp/r/r312/EDR/J_index.html

[3] 日本語 Wikipedia : http://ja.wikipedia.org/wiki/

[4] Sören Auer,Christian Bizer,Georgi Kobilarov,Jens Lehmann1, Richard Cyganiak,Zachary Ives: DBpedia: A Nucleus for a Web of Open Data, idelberg , ISWC/ASWC 2007, pp.722-735, LNCS 4825(2007)

[5] Fabian M. Suchanek, Gjergji Kasneci and Gerhard Weikum "YAGO - A Large Ontology from Wikipedia and WordNet", Elsevier Journal of Web Semantics (to appear)

[6] Simone Paolo Ponzetto,Michael Strube: Deriving a Large Scale Taxonomy fromWikipedia, Proc. AAAI-2007 , AAAI Press,pp.1440-1447,2007.

[7] 日本語語彙大系: http://www.kecl.ntt.co.jp/mtg/resources/GoiTaikei/

[8] 武田 英明: 上位オントロジー(<特集>開発されたオントロジー), 人工知能学会誌, Vol.19, No.2, pp. 172-178, 2004

[9] WikiOnt プロジェクト : http://www.yamaguti.comp.ae.keio.ac.jp/mmm/WikiOnt/index.html