XPathを用いた日本語文からの複合要素検索

吉橋 健治
東京工業大学社会理工学研究科
knj4484@ryu.titech.ac.jp
阿辺川 武
東京工業大学総合理工学研究科
abekawa@lr.pi.titech.ac.jp
仁科 喜久子
東京工業大学留学生センター
knishina@ryu.titech.ac.jp
奥村 学
東京工業大学精密工学研究所
oku@pi.titech.ac.jp

1 はじめに

現在，公開されている日本語学習支援システムの多くは，形態素解析で得られた単語をもとに辞書を検索することで，語の意味表示機能を実現している．“リーディング チュウ太”[1]では，この手法が用いられている [1].

しかし，形態素解析と辞書項目は整合性が取れていないものも多数存在し，複雑な辞書項目に関しては構文解析も考慮しなければ辞書項目の検索ができない場合もある．そこで，そのような不整合を解消し，辞書項目をより効率的に活用するための方法が必要となる．

本稿では，特に，辞書項目のうち，形態素解析で複数の形態素に分割されてしまうものを複合要素と呼び，研究の対象とする．表 1 に複合要素の例を示す.

表 1: 複合要素の例

文法項目	はおろか，なければならない
慣用句，イディオム	顔が広い
ことわざ	猿も木から落ちる
複合名詞	民主主義，自然言語

我々は，複合要素検索の問題を解決するために，構文解析の結果を XML の一種である GDA[2] に変換し，辞書項目を XPath[3] で登録しておく方法を提案した [4]. また，構文解析器に CaboCha[5]，単語辞書に EDR 日本語単語辞書 [6] を用いて実装し，現在，Web 上で公開している．[2].

本稿では，実装時における効果的な XPath の作成方法，検索方法について述べる．また，その XPath で複合要素を含む例文を検索した場合の精度についての評価実験の結果，実験で明らかになった問題点およびその解決方法について述べる．

2 XPathを用いた複合要素検索の原理

複合要素の XPath 検索に使われる XML 技術および検索の原理に関して具体例を交えて説明する．

検索対象テキストは構文解析し，GDA に変換しておく．GDA とは，文書の意味的，語用論的構造を計算機が自動的に認識することを可能にする XML のタグ集合である．

GDA は XML 形式で構成されているので，データの検索には既存の XML を扱う種々の手法が使用できる．そこで複合要素の検索には伊藤らの手法 [7] と同様に XML のデータ構造を検索する形式の一つである XPath を用いる．XPath は，XML データを表す木構造をたどり，ある条件を満たす要素や属性を検索する記述方法で，W3C により規定されている仕様である．「自然言語」,「顔が広い」に対する XPath を表 2 に示す．また「顔が広い」の XPath がマッチする GDA の例を図 1 に示す.

GDA の例の中で orig，prn，pos の三つの属性が形態素の原形，読み，品詞を表している[3]．XPath における parent，following-sibling，descendant-or-self は GDA で表される木構造の中で，それぞれ，親ノード，兄弟ノードのうち自分より後ろのもの，自分を含めた子孫ノードを表している．したがって，*[@orig=”顔”][following-sibling::*[@orig=”が”]] の部分によって，原形が「顔」である形態素および後続する形態素「が」を指定している．[parent::*[following-sibling::*[descendant-or-self::*[@orig=”広い”]]]] の部

[1]http://language.tiu.ac.jp/
[2]http://hinoki.ryu.titech.ac.jp/
[3]orig 属性は GDA の定義に従うものではなく，我々が独自に追加した属性である

表 2: XPath の例

自然言語	//*[@orig=”自然”][following-sibling::*[@orig=”言語”]]
顔が広い	//*[@orig=”顔”][following-sibling::*[@orig=”が”]] [parent::*[following-sibling::*[descendant-or-self::*[@orig=”広い”]]]]

```
<su>
  <adp>
    <np orig="彼" prn="カレ" pos="名詞">彼</np>
    <ad orig="は" prn="ハ" pos="助詞">は</ad>
  </adp>
  <adp>
    <np orig="顔" prn="カオ" pos="名詞">顔</np>
    <ad orig="が" prn="ガ" pos="助詞">が</ad>
  </adp>
  <adp orig="あまり" prn="アマリ" pos="副詞">あまり</adp>
  <v>
    <ajp orig="広い" prn="ヒロク" pos="形容詞">広く</ajp>
    <v orig="な い" prn="ナ イ" pos="助 動 詞">ない</v>
  </v>
</su>
```

図 1: 「顔が広い」の XPath がマッチする GDA の例

分によって,「顔が」が係り得る位置にあり，原形が「広い」である形態素を指定している．この例文に対して，文字列検索した場合,「広い」が活用しているために検索できず，形態素解析結果を用いた検索でも，構成要素が「あまり」の前後に分離しているので検索できない．しかし，GDA と XPath を用いることで文字列検索や形態素解析結果を使った検索より柔軟な検索条件が指定でき，正しく検索出来るようになる．

3 効果的な XPath の作成法および検索法

辞書項目に対応する XPath を登録し，文中の複合要素を検索するための基本的な手順は以下の通りである．

1. 全辞書項目の見出し語を形態素解析し，複数の形態素に分割される項目を抽出する
2. 抽出された項目を構文解析し，それから得られる GDA の構造に対応する XPath を作成する
3. 入力文を構文解析し，GDA に変換したものに対して，XPath 検索する

効率の良い検索を行うためには，各段階において考慮しなければならない問題点がある．

a) 形態素解析，構文解析は，元の辞書項目単独の場合と文中では解析結果が違う

b) GDA には読み，原形，品詞，係り受けなどの情報が含まれており，一つの GDA の構造に対してマッチする XPath は何通りも存在する

c) XPath を用いた検索は時間がかかるので，入力文に対してすべての複合要素の XPath 検索をするのは現実的ではない

a の問題に関しては，辞書項目を含んだ例文を用意することが出来るならば，例文の解析結果をもとにした方がよい．左右の接続コストが変化し，単独で解析した場合より，実際の文に近い解析結果を得る可能性が高いからである．例文を用意できない場合は次善の策として，辞書の品詞を用いて前後にその品詞に対する典型的な表現パターン（名詞に対して [助詞+動詞] など）を付け加えることが考えられる．特に，辞書項目単独で形態素解析をした場合のみ形態素が分割されてしまうものは XPath の登録数を無意味に増やし検索時間を伸ばしてしまうので注意が必要である．

b の問題に関しては，品詞情報は XPath に取り入れないほうが良い．形態素解析で得られる品詞は，複合要素の置かれている文脈によって変化してしまうものもあるからである．例えば,「叩き肉」という辞書項目を含む二つの文 1)「少し叩き肉を食べた」2)「少しの叩き肉を食べた」を形態素解析すると,「叩き」の部分の形態素が，1 では動詞，2 では名詞という解析結果になり，品詞を含む XPath にとって都合が悪い．

c の問題に関しては，入力文に対して全ての XPath 検索を行うのではなく，あらかじめ検索を行う XPath を絞り込んでおくとよい．最も良い方法は，複合要素を構成する形態素の中で，登録した XPath 全体の中で頻度の低い形態素をインデックスとして登録しておき，この形態素が入力文に現れる XPath だけ検索を実行する方法である．

4 評価

XPath 検索で複合要素が実際に検出されるかどうかを定量的に評価すること，および問題点を洗い出し解決策を立てることを目的とし実験を行った．

4.1 精度

EDR 日本語コーパスから 100 文，小説（我輩は猫である [8]）から 100 文をランダムに抽出したものを

検索対象とし，評価実験を行った．XPath 検索，文字列検索で検索された複合要素に対して，人手で正誤の判断をし，適合率，再現率を求めた．文字列検索は，XPath を登録した全ての辞書項目に関して見出語をそのまま検索パターンとして検索した．

まず，対象としたテキストに含まれていた複合要素の数を表 3 に示す．

表 3: 複合要素の出現数

	複合要素数		複合要素が
	延べ	異なり	含まれていた文
小説	255	180	90
EDR コーパス	240	98	84

複合要素が含まれていた文は EDR コーパス，小説全体で 9 割近くあり，一つの文に対して，平均で約 2.5 個の複合要素が含まれていたことになる．形態素と見出し語のマッチングでは辞書を引くことが不可能だった辞書項目が多数存在することを示している．また，EDR コーパス，小説でほぼ同じ回数，複合要素が出現していることから複合要素の出現頻度は，テキストの種類によらず，ほぼ同程度だと考えられる．これらの結果から複合要素検索の必要性が分かる．

次に，XPath 検索，文字列検索の精度を表 4，表 5 にそれぞれ示す．

表 4: XPath 検索の精度

	適合率	再現率	F 値
小説	0.61	0.76	0.68
EDR コーパス	0.68	0.80	0.73

表 5: 文字列検索の精度

	適合率	再現率	F 値
小説	0.65	0.75	0.70
EDR コーパス	0.61	0.72	0.66

どちらか一方だけの検索方法のみで検索された複合要素の比率を表 6 にしめす．

これらの結果から，文字列検索では検出できなかった複合要素を XPath 検索で検出できるようになったこと，および XPath 検索で検出できないものを文字列検索で補うことが可能であることが分かる．

4.2 XPath 検索の失敗例とその対策

EDR 日本語コーパスには形態素ごとの概念 ID も付与されている．この概念 ID と XPath を作った項目の概念 ID とを照応することで，複合要素が含まれる例文を抽出することができる．これらの文に対して XPath 検索をすることで自動的に検索の失敗例を収集することができる．

表 6: 一方の方式のみ検索が成功したものの比率

	XPath 検索のみ	文字列検索のみ
小説	0.26	0.25
EDR コーパス	0.27	0.23

4.2.1 形態素解析に起因する検索失敗例

XPath 検索の結果のうち，形態素解析に起因する失敗例を 500 個収集した．失敗要因を分類したものを表 7 に示し，それぞれのパターンについて解決策を論じる．

失敗パターン a，b： 一般的に接頭語や接尾語の扱いは，辞書によって，また単語によって異なっている．形態素解析のための辞書と単語辞書で接頭語，接尾語が一致しないために検索が失敗してしまうのは，形態素解析結果を使う手法全てに共通の問題である．a の失敗パターンのうち典型的な接頭語については組合せのパターンを用意することである程度対応できる．しかし，接尾語は接頭語よりも種類が多く，組合せも多様なので b の失敗パターンの対策を立てるのは難しい．

失敗パターン c： 前節で述べたように，複合要素が文中に含まれた形で解析した結果をもとに XPath を作成することで減らすことが出来る．

失敗パターン d： XPath による検索は成功しないが，辞書項目を引くという点からすれば，XPath を作成した見出し語をもとの単語辞書に残しておくことで問題は解決される．

失敗パターン e： 検索したい複合要素が対象テキストの形態素解析結果の部分文字列であるとき，XPath 検索は失敗する．これは，形態素解析を利用することの一般的な弊害で，複合要素以外の通常の単語の検索でも起こりうる．しかし，この失敗パターンは文字列検索で補うことができる．

4.2.2 XPath の書き方が検索に及ぼす影響

4.2.1 で述べた失敗例収集の段階で，形態素解析に起因するもの以外の失敗もいくつか見られたが，数が少なく，分類できる状況ではなかった．そこで，XPath の書き方による検索の問題点を示す例を挙げ，対策について述べる．

2 節で述べたように，XPath を用いることで，検索の条件を柔軟に記述できるようになり，複合要素を構成する要素が検索対象テキスト中で離れた位置に出現

表 7: 形態素解析に起因する XPath 検索の失敗例

	説明	例	
		項目単独の解析	例文中の解析
a	接頭語とくっついて形態素になってしまったもの	菓子 | 屋	お菓子 | 屋
b	接尾語とくっついて形態素になってしまったもの	収容 | 所	収容 | 所長
c	辞書項目単独の解析と例文の解析で 形態素の区切りが変わってしまったもの	友軍 | 機	友 | 軍機
d	単独の解析では形態素が分割してしまうが, 例文中では一つの形態素になるもの	遠 | 方	遠方
e	構文解析において，辞書項目以上に長い部分を 一つのまとまりとして認識してしまうもの	千 | 歳	千歳空港

しても検出できるようになった．しかし，この性質は次のような場合にはデメリットとなる．

「訳なし」(「手間がかからないこと」あるいは「たやすく簡単にできるさま」の意）という複合要素に対して//*[@orig=”訳”][following-sibling::*[@orig=”ない”]] という XPath が登録してある．ここで,「君に頼んだ訳ではないのだ」という文を GDA に変換すると図 2 のようになる．この変換結果は orig 属性が「訳」である形態素より後ろに orig 属性が「ない」である形態素が兄弟ノードとして存在するので「訳なし」の XPath がマッチしてしまう．この場合，原理的には「訳」の直後に「なし」が来るという条件の XPath にすれば問題は解決する．しかし，構成要素が必ず連続するもの，離れて出現してもよいもの，どちらにも対応するためには，全ての複合要素をあらかじめどちらかに分類しておく必要がある．

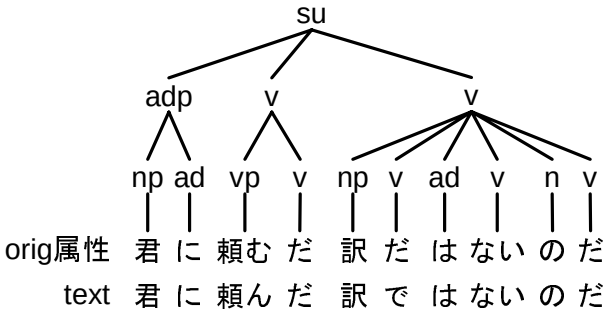

図 2: 「君に頼んだ訳ではないのだ」を GDA に変換した結果

5 まとめ

形態素解析によって複数の形態素に分かれてしまう単語辞書項目を検索する方法として，GDA に変換された対象テキストに XPath 検索をする方法を実装した．実装にあたって，辞書項目単独の解析より文脈を与えた解析から XPath を作った方がよいこと，品詞は XPath に取り入れない方がよいこと，検索時における XPath の絞りこみ方について説明した．また，サンプルデータを使って評価実験を行い，単純な形態素のマッチングだけでは検出できない辞書項目が 1 文あたり約 2.5 個あり，XPath 検索を使うと F 値 0.7 前後の精度で，検出できることを示した．また XPath 検索が失敗する例を分類し，文字列検索を補助的に用いるなどの解決策を提案した．

提案した解決策を実装し，さらなる精度の向上が得られることを検証することが今後の課題である．

参考文献

[1] 川村よし子, 初級中級日本語学習者用の辞書ツールの開発, 第 7 回ヨーロッパ日本語教育シンポジウム, 2002

[2] 橋田浩一, GDA 日本語アノテーションマニュアル, http://i-content.org/gda/tagman.html.

[3] W3C, XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath.

[4] 阿辺川武, 八木豊, 戸次徳久, 傅亮, 奥村学, 仁科喜久子, 読解支援システムのための言語非依存フレームワーク構築, 言語処理学会第 10 回年次大会発表論文集, pp.144-147, 2004.

[5] 工藤拓, 松本裕治, チャンキングの段階適用による日本語係り受け解析, 情報処理学会論文誌, Vol.43, No.6, pp.1834-1842, 2002.

[6] 日本電子化辞書研究所,EDR 電子化辞書仕様説明書第２版, Technical Report TR–045, 1995.

[7] 伊藤一茂, 斎藤博昭, マルチモーダル対話コーパス検索/再生ツールの実装, 自然言語処理 142-5, 2001.

[8] 青空文庫, http://www.aozora.gr.jp/

[9] 松本 裕治, 形態素解析システム「茶筌」, 情報処理, Vol.41, No.11, pp.1208-1214, November 2000.