日本語文型同定システムの構築

ミロシニク ロマン　加藤 恒昭
東京大学 大学院総合文化研究科
miroshnik.roma@gmail.com; kato@boz.c.u-tokyo.ac.jp

1 はじめに

日本語学習者を支援するための総合的な日本語文型同定システムを開発している。初心者の日本語学習者が、自分で日本語文を作文する時や書かれた日本語文を読み解く時に使えるような語彙辞書、漢字辞書、文型辞書を組み込んだ総合的な読解支援システムを目指している。

読解支援は高次レベル支援と低次レベル支援といった二つのタイプに分けられる [1, 4]。低次レベルの支援は単語、内容表現の訳、フリガナ、漢字の情報の提供、機能語と機能表現を含む文型の説明といった要素からなっている。高次レベルの支援は修飾関係、助詞の省略などの構文構造の情報、文の理解に必要な歴史的/文化的背景知識の提供からなっている。文献 [11] では 4 つの日本語読解支援システムの評価と分析が行われている。それらに「JL 文庫」というシステムを加えて再分析を行った。結果を表 1 に纏める。

訳とフリガナの支援はほぼ全ての読解支援システムに実装されている。漢字に関する支援は理解.com だけが実装しているが、様々な電子辞書による支援も可能で、KANJIDIC という XML 形式漢字データベースも公開されており、実装は比較的簡単である。構文に関する支援は CaboCha に基づいたものがあすなろに実装されている。歴史的/文化的背景の知識に関する支援は JL 文庫に実装されているが、人手で編集したテキストだけを提供していて、任意のテキストに対応していない。歴史的/文化的背景知識の支援はまだ大きな問題として残っている。

文型に対する支援は、文献 [5] などの文型辞典で示されているような日本語の機能表現を含む様々な文型の存在を判断して、それについての解説を表示する。文型に関する支援は既存の日本語読解支援システムでは十分に実装されていない。リーディング・チュウ太と理解.com では文型に当てはまる形態素を支援なしで表示している。WWWJDIC では、「それから」「ながら」「したがって」などの簡単な機能語と機能表現に対しては訳の支援があるが、「なければならない」「てあげる」などの比較的複雑な文型に対しては支援を行っていない。このため誤解をまねくこともありうる。例えば、WWWJDIC で「公共の交通機関がないので、自分の車で行くよりほかにはない。」という文を分析した結果は「公共」「交通機関」「自分」「行く」に対して正しく支援の情報が表示されるが、助詞「が」「で」と副助詞「ので」に対して支援情報がなく、「よりほかにはない」という文型に含まれる「ほかに」だけに対して支援情報が示される [14]。この情報だけでは特に日本語の初級学習者にとっては文の意味の理解が難しいと予想される。

これらから、総合的に文型を取り扱う読解支援システムの必要性は非常に高いと言うことができ、その核となる文型同定システムが求められている。

2 必要とされる要件

日本語学習者を支援するための日本語文型同定システムは、次のような特徴を持つこがと望ましい。

- **文型検出処理を簡単に追加・変更できる。**
  文型の定義自体様々であり、支援の目的に従って文型要素の追加を大量に行うこともありえる。プログラミングスキルを持っていない人にも簡単に文型検出処理を作れることが望ましい。
- **複雑な文型が検出できる。**
  後述するように、検出すべき文型は、連続する形態素だけでなく離れている形態素が関係したり、文型を構成する形態素の一部が省略できたり任意要素が挿入できたりなど、様々である。それらを記述し検出できることが望ましい。
- **検出処理の再利用に対応する。**
  文型を構成する形態素には重複が多いので、それを効率的に処理できることが望ましい。著者が機能表現辞書つつじ [6] に基づき、 文献 [5] に含まれている 634 文型の 16884 の異形を構成する形態素の出現分布を調査したところ、全ての異形の中の形態素出現数は 69546 であったが、出現率上位 30 種類の形態素が 43402 件出現し、すべての形態素出現数

表 1: 日本語読解支援システムの機能別の分析

	訳	フリガナ	漢字	文型（機能語/機能表現）	構文	歴史的/文化的背景の知識
リーディング・チュウ太	○	○	×	×	×	×
あすなろ	×	○	×	×	○	×
理解.com	○	○	○	×	×	×
WWWJDIC	○	○	×	△	×	×
JL 文庫	△	△	×	×	×	○

の 62.4%を占めている。出現率上位 50 種類の出現は 49581 件で、71.29%に上る。

- **文型の様々な異形を処理できる。**
  様々な異形を扱える必要がある。上述の 634 の文型が持っている異形数は 16884 件である。異形数が上位の 30 文型がすべての異形の 59%を占めている。50 文型が 74.73%に至る。「ことがある」、「さし上げる」、「ことが出来る」、「いただく」、「くださる」といった文型が 400-700 程度の異形を持つ。異形の原因は主に挿入可能な助詞、語幹の表記ゆれ、動詞の活用形とます形の活用形である。

3 アプローチ

文型同定システムを実現する方法はいくつか考えられる。

機械学習的なアプローチを利用して文型同定システムを実現するためには、既存の形態素に加えて各機能表現の接続制約や連接コストを推定するために、機能表現がラベル付けされた大規模なコーパスが必要になる。文献 [12] で指摘されているように作成コストの点から見て、そのような条件を満たす大規模コーパスを準備することは非現実的である。その上、新規文型要素を追加する時、コーパス編集と再学習が必要になる。

形態素解析結果を利用して、人手で作成した検出規則を形態素解析結果に対して適用することにより機能表現を検出するシステムの提案もある [8, 13, 7]。しかし、これらのシステムでは、検出規則を人手で作成するのに多大なコストが必要と報告されている。

形態素解析結果から機械学習によって機能表現を検出する方法も提案されている [12]。SVM を用いて、形態素解析器 ChaSen による形態素解析結果を入力として、曖昧性がある 52 表現に対して機能表現検出器を実装したと報告されている。高精度に検出できるものの、検出できる機能表現の要素を容易に追加できないという問題点がある。

本稿で提案するシステムでは、形態素解析器 MeCab の解析結果を用いて、XML 形式で記述されている規則に基づいて、4 段階の処理を行うことで文型同定を実現する。各段階の規則を容易に追加でき、後述するような非連続複合文型と繰り返し文型などの複雑な文型の検出も可能である。

4 提案システムの構成

提案するシステムは、統計的な方法とルールベース方法を組み合わせたものとなっている。形態素解析器の解析結果を利用して XML 形式で記述したパターンに従って文型要素を同定する。本システムでは、文献 [5] に含まれている文型を念頭に置いて、文型同定を実装する。

処理は次の段階に分けられている。

1. 形態素解析器 MeCab によるテキストの解析。
   分析対象となるテキストを MeCab によって解析し、結果を形態素オブジェクトの配列とする。
2. XML パターンによる形態素解析結果の修正。
   必要に応じて形態素解析結果を修正する。
3. XML パターンによる文型構成素候補の抽出。
   文中で同じような役割をなしている形態素を一つのより抽象度の高い文型構成素候補にする。
4. XML パターンによる文型要素の同定。
   文型を含んでいるかを分析し、文型要素を同定する。
5. XML パターンによる曖昧性解消。
   文型として検出される表現が実は文型としての役割を持たず、内容表現である場合があるので、それを判断して、内容表現になっているものを取り除く。

以下、それぞれの段階について説明する。

4.1 形態素解析機 MeCab によるテキストの解析

この段階では入力文を MeCab を用いて解析して、文型分析の基本となる形態素オブジェクトの配列を作成する。

4.2 XML パターンによる形態素解析結果の修正、変更

形態素解析結果を変更する。MeCab は IPA 辞書に基づいているため、独立した形態素だけではなく、「さも

なければ」「と共に」といった複数の形態素からなる機能表現が単位として用いられている。文型同定の時、そちらに含まれる形態素を別々の要素として取り扱う必要がある。加えて、ある程度の解析結果の誤り修正も必要である。例えば、漢字表記の「其れ」は誤って一般名詞「其」と助動詞「れ」として解析される傾向があるので、これを代名詞へと修正する。これらの変更と修正の内容は XML パターンとして記述される。[1]

4.3 XML パターンによる文型構成素候補の抽出

ある種の文型は、特に複数の形態素からなっている文型は、いくつかの形を持つことがある。「あまり · 否定」と「でも [Wh-word] でも」はその例である。第一の文型に当てはまるものは「この本はあまりよくない。」と「この本はあまりよくありません。」である。両方とも「あまり · 否定」の文型を含んでいるが否定を表す形態素が違う。これらを扱うために一つの文型に二つのパターンを指定することが考えられるが、パターンの条件が多くなり、コードの理解と編集が難しくなる。文型構成素候補という抽象的な要素を導入することでパターンが理解しやすくなり、パターンの改善と編集の際に作業を容易に出来ることが期待できる。この例では、「ない」「ん」が「否定」にまとめられる。

もう一つは抽象的な要素を作る処理と逆の処理である。こちらでは同じような形、読み、品詞を持っている形態素を品詞細分類の情報などを用いて更に細かい文型構成素候補に分けていく。例えば、助詞の「と」は格助詞や並立助詞などに分類される。

4.4 XML パターンによる文型要素の同定

形態素オブジェクトに付与された文型構成素候補のラベルを参照し、パターンによって文型要素の同定処理を行う。

次のような場合を含む複雑な文型の定義を扱える必要がある。

- 挿入/中略可能な形態素の存在。例えば「てあげる」という文型では、助詞「て/で」と動詞「あげる」の間に助詞「は」「も」「さえ」「すら」などが挿入可能である。
- 交換可能な形態素の存在。　例えば「てあげる」という文型では、動詞「あげる」がかな表記と漢字表記で現れることがあり、それぞれの活用形で現れる。それらの処理のことである。
- 繰り返し文型。同じ構造からなっていて、文中二回以上現れる隣り合っていない要素を含む文型を繰り返し文型と呼ぶ。例えば、「. . . やら . . . やら」、「. . . であれ . . . であれ」、「. . . と言わず . . . と言わず」などである。
- 非連続複合文型。文型の中に任意の形態素をいくつか含むことができ、文型の構造が繰り返しのパターンを含まない連続していない形態素からなる文型を被連続複合文型と呼ぶ。例えば、「というのは . . . ことだ」、「ろくに . . . ない」、「なぜなら（ば）. . . からだ」などである。

これらを記述できる XML パターンを定義する．以下では「てあげる」の文型要素同定パターンが示されている。<bunkei>は文型要素を表し、<bunkei>の name という属性は同定した文型要素に付与する。<bunkei>のタグの中に<pattern>というタグが一つ以上入っている。<pattern>のタグが<kouseiso>というタグを含む。<kouseiso>のタグは文型を構成する構成素に当てはまる。<kouseiso>のタグの中に<cand_name>のタグが入る。<cand_name>のタグが文型構成素を同定するために必要な条件を表す。形態素オブジェクトに付与された文型構成素候補ラベルと<cand_name>がマッチした場合、文型構成素（<kouseiso>）の同定を行う。<pattern>に入っているすべて構成素を同定した場合、パターンの同定を行い、文型要素の同定を行う。文型構成素の中に交換可能な<cand_name>がある場合、複数の<cand_name>のタグの記述が可能である。例えば、「てあげる」の文型パターンでは、3 番目の文型構成素が ageru_doushi_hijiritsu、ageru_doushi_hijiritsu_kanji、ageru_doushi_jiritsu、ageru_doushi_jiritsu_kanji のいずれかを含む必要がある。挿入/中略の場合、<cand_name>の条件として*を記述する。そういう場合、<kouseiso>のタグの中で相応しい文型構成素候補にマッチしなかった場合でも同定処理を終了せず、後続の<kouseiso>のタグの同定処理を行う。

```
<bunkei name="ageru2">
  <pattern>
    <kouseiso>
      <cand_name>te</cand_name>
      <cand_name>de_setsuzokujyoshi</cand_name>
    </kouseiso>
    <kouseiso>
      <cand_name>*</cand_name>
      <cand_name>ha</cand_name>
      <cand_name>mo</cand_name>
      <cand_name>sae</cand_name>
      <cand_name>sura</cand_name>
      <cand_name>nado</cand_name>
      <cand_name>nanka</cand_name>
      <cand_name>nannte</cand_name>
    </kouseiso>
    <kouseiso>
      <cand_name>ageru_doushi_hijiritsu</cand_name>
      <cand_name>ageru_doushi_hijiritsu_kanji</cand_name>
      <cand_name>ageru_doushi_jiritsu_kanji</cand_name>
      <cand_name>ageru_doushi_jiritsu</cand_name>
    </kouseiso>
    <kouseiso>
      <cand_name>*</cand_name>
      <cand_name>masu_jyodoushi</cand_name>
      <cand_name>mashi_jyodoushi</cand_name>
```

[1]以下、スペースの制約で本稿内に示せなかった XML パターンは https://github.com/MyroshnykRoman/bunkei を参照いただきたい。

```
      <cand_name>masure_jyodoushi</cand_name>
    </kouseiso>
  </pattern>
  <pattern>
    <kouseiso>
      <cand_name>tageru</cand_name>
    </kouseiso>
    <kouseiso>
      <cand_name>*</cand_name>
      <cand_name>masu_jyodoushi</cand_name>
      <cand_name>mashi_jyodoushi</cand_name>
    </kouseiso>
  </pattern>
</bunkei>
```

繰り返し文型パターンと非連続複合文型パタンは次の通りになる。

```
<bunkei name="toiwazu_toiwazu">
  <pattern type="repeat">
    <kouseiso>
      <cand_name>to_kakujyoshi_inyou</cand_name>
    </kouseiso>
    <kouseiso>
      <cand_name>iwa</cand_name>
    </kouseiso>
    <kouseiso>
      <cand_name>zu</cand_name>
    </kouseiso>
  </pattern>
</bunkei>
<bunkei name="kara_niitarumade">
  <pattern>
    <kouseiso>
    kouseisoname>kara_kakujyoshi</cand_name>
    </kouseiso>
    <kouseiso type="hirenzoku"></kouseiso>
    <kouseiso>
       <cand_name>ni_kakujyoshi</cand_name>
    </kouseiso>
    <kouseiso>
      <cand_name>itaru</candidate_name>
    </kouseiso>
    <kouseiso>
      <cand_name>made</cand_name>
    </kouseiso>
  </pattern>
</bunkei>
```

繰り返し文型の同定の際、 <pattern>タグの type 属性に repeat という値を記述する。文中すべて同定したパターンが一つの文型要素に同定される。非連続複合文型の同定の際、空の<kouseiso>のタグに hirenzoku という値を持つ type 属性を記述する。これにより、非連続タイプの<kouseiso>タグの前後<kouseiso>タグの間に一つ以上任意個の形態素オブジェクトを考慮して同定を行う。

4.5 曖昧性解消

先行研究 [12] では形態素解析結果を機械学習によって機能表現の可能性と内容表現の可能性がある曖昧性を持つ表現の検出器を作成が報告されており、表現の前後二つ形態素を素性として、F 値が 92.3 程度の検出器が実現できたとされている。機械学習の代わりに前後の二つの形態素に参照する XML パターンを利用して同じ性能を実現することを考えている。例えば、「とはいえ」に対しては後続する形態素は助動詞「ない」の場合、内容表現だと判断し、同定した文型から取り除くことができる。

5 おわりに

本論文では文型同定システムの構成を提案した。形態素解析機の解析結果を利用して 4 段階の XML パターンによって文型要素を同定する。システムの特徴としてはパターンの再利用と容易なパターンの作成と追加という利点が上げられる。2017 年 1 月現在で、文型構成素候補パターン 596 件（文献 [5] の 90%程度）と文型同定パターン 300 件（同 48%）を実装している。システムの実現の伴い、様々な日本語読解支援システム、日本語学習支援システムへの展開を予定している。

参考文献

[1] Alderson, J. C. 2000. Assessing Reading. Cambridge: Cambridge University Press.

[2] あすなろ　https://hinoki-project.org/asunaro/

[3] JL 文庫　http://jl-bunko.sakura.ne.jp/

[4] Koda, K. 1994. Second language reading research: Problems and possibilities. Applied Psycholinguistics, 15, pp.1-28.

[5] 牧野 成一, 筒井 通雄. 1989, 1995, 2008. 日本語基本文法辞典, 日本語文法辞典 [中級編], 日本語文法辞典 [上級編] The Japan Times.

[6] 松吉 俊, 佐藤 理史, 宇津呂 武仁. 2007. 日本語機能表現辞 書の編纂. 自然言語処理, 14(5), pp.123-146.

[7] 松吉 俊, 佐藤 理史, 宇津呂 武仁. 2005. 接続情報にもとづく助詞型機能表現の自動検出. 言語処理学会第 11 回年次大会発表論文集, pp.1044-1047.

[8] 中塚 裕之, 佐藤 理史, 宇津呂 武仁. 2005. 助動詞型機能表現の形態・接続情報と自動検出. 言語処理学会第 11 回年次大会発表論文集, pp.596-599.

[9] リーディング・チュウ太　http://chuta.jp/ , http://language.tiu.ac.jp/

[10] 理解.com　http://www.rikai.com/

[11] 豊田 悦子, 松下 達彦. 2015. 既存の日本語読解支援システムをどう評価するか　—日本語教育の視点から—. 言語処理学会第 21 回年次大会論文集, pp.780-783.

[12] 土屋 雅稔, 注連 隆夫, 高木 俊宏, 内元 清貴, 松吉 俊, 宇津呂 武仁, 佐藤 理史, 中川 聖一. 2007. 機械学習を用いた日本語機能表現のチャンキング. 自然言語処理, 14(1), pp.111-138.

[13] 土屋 雅稔, 宇津呂 武仁, 佐藤 理史, 中川 聖一. 2005. 形態素情報を用いた日本語機能表現の検出. 言語処理学会第 11 回年次大会発表論文集, pp.584-587.

[14] WWWJDIC　http://nihongo.monash.edu/cgi-bin/wwwjdic?1C