品詞列・係り受け部分木に基づくラベリングツールの設計と実装 –節境界ラベリングを例に–

浅原 正幸 (国立国語研究所) *
小西 光 (国立国語研究所)
田中 弥生 (神奈川大学・国立国語研究所)
加藤 祥 (国立国語研究所)

Design and Implementation of a Labeling Tool Based on Morpheme Subsequences and Dependency Subtrees – a Use Case in Clause Boundary Labeling –

Masayuki Asahara (National Institute for Japanese Language and Linguistics)
Hikari Konishi (National Institute for Japanese Language and Linguistics)
Yayoi Tanaka (Kanagawa University, National Institute for Japanese Language and Linguistics)
Sachi Kato (National Institute for Japanese Language and Linguistics)

要旨

コーパスに対する情報付与の方法として、悉皆的に人手で行うアノテーション以外に、パターンに基づくラベリング手法がある。後者のラベリング手法は、人手で行うアノテーションの前段階で用いられるほか、形態論情報や係り受け部分木に基づくパターンで記述可能な手がかり句がわかっている際に広く用いられる。しかしながら、パターンに基づくラベリングツールは様々な研究者により開発されているが、再利用性が乏しく、パターンを記述するために汎用的な記述方法が求められている。本研究では、コーパスの検索系で用いられるクエリ言語を基にした形態論情報や係り受け部分木に基づくラベリングツールを設計し、実装した。また、節境界ラベリングを例にした応用について紹介する。

1. はじめに

日本語のコーパス分析において、形態素解析器によって単語分かち書きと同時に形態論情報を付与した形態素解析結果を、人手で修正して分析するという手法が多く用いられてきた。係り受け解析器の精度が高くなるについてれ係り受け情報を用いた分析も少しずつ増えてきている。さらに、個々の研究者によってアノテーションされた統語的・意味論的情報をもとに分析する研究も行われている。

このアノテーションを行う際に、アノテーション対象が手がかり句 (cue phrase)[(1)]によって

* masayu-a@ninjal.ac.jp

(1) アノテーション対象の存在を示す語句。

ある程度表現可能な場合がある。品詞体系や係り受け構造が広く利用されているものであれば、形態論情報や係り受け部分木によって共有可能な手がかり句が表現され、手がかり句を再利用することが可能になる。(2)

ほとんどの手がかり句は各種検索系に基づく、検索クエリ相当の表現で記述することが可能である。そこで本稿では、各種検索系で可能なクエリとクエリ言語について整理し、JSON (JavaScript Object Notation) 形式での統一的なパターン記述言語を設計する。さらに、「鳥バンク」(池原 (2007)) で採用されている節境界認定基準を中心に、節境界ラベリングツールの試作を行う。

2. 形態論情報・係り受け部分木に基づく検索系

2.1 中納言

コーパス検索アプリケーション「中納言」(国立国語研究所 (2015a)) は、形態論情報に基づくコーパス検索が可能な Web アプリケーションである。現在のところ『現代日本語書き言葉均衡コーパス』(Maekawa et al. (2014)) や『日本語歴史コーパス』(国立国語研究所 (2015b)) が検索できるようになっている。

図 1 は、連体形が前置する手がかり句「かたわら、」（鳥バンク：「副詞句：二者関係：：対比：FUj001」）を中納言で検索した例である。

図 1　「中納言」による手がかり句「かたわら、」の検索フォーム

「中納言」では、クエリ言語として、SQL に似た記述言語を用いている。図 2 は上記検索フォームの内容を「中納言」のクエリ言語に変換したものである。

(2) 本稿では、規則やパターンに基づいて自動的に情報を付与する作業を「アノテーション」とは呼ばない立場をとり、ラベリングと呼ぶこととする。

```
キー: 書字形出現形 = "かたわら" AND 前方共起: 活用形 LIKE "連体形%" ON 1 WORDS % FROM キー
AND 後方共起: 書字形出現形 = "、" ON 1 WORDS FROM キー WITH OPTIONS unit="1"
AND tglBunKugiri="#" AND tglWords="20" AND limitToSelfSentence="1" AND tglKugiri="|"
AND endOfLine="CRLF" AND encoding="UTF-16LE" AND tglFixVariable="2"
```

図 2　「中納言」による手がかり句「かたわら、」の検索クエリ

2.2　ChaKi.NET Tag Search

コーパスコンコーダンサ「ChaKi.NET」(Matsumoto et al. (2006)) の Tag Search 機能でも、同様の検索が可能である。検索フォームは図 3 のようになる。

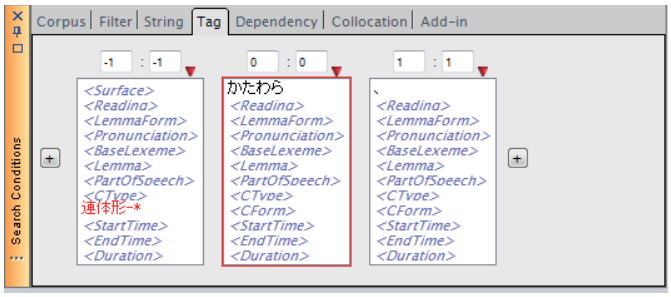

図 3　「ChaKi.NET」による手がかり句「かたわら、」の検索フォーム

この検索フォームは外部 XML ファイルに保存可能である。図 4 に Tag Search の検索条件<TagCond>の冒頭の形態素を指定している箇所を示す。

```
<TagCond><LexemeConds><LexemeCondition><PropertyPairs>
  <PropertyPair>
    <Key>CForm</Key>
      <Value xsi:type="CForm"><StrVal>連体形-*</StrVal><IsRegEx>true</IsRegEx>
      <IsCaseSensitive>true</IsCaseSensitive><ID>0</ID><Name>連体形-*</Name></Value>
  </PropertyPair>
 </PropertyPairs>
 <RelativePosition><Start>-1</Start><End>-1</End></RelativePosition>
  ...
```

図 4　「ChaKi.NET」による手がかり句「かたわら、」の検索クエリ

2.3　超大規模コーパス 検索系 (形態論情報検索)

図 5 は現在開発中の超大規模コーパス検索系のうち形態論情報を検索するための試作 UI である。「ChaKi.NET」の Tag Search を参考にした UI になっている。この UI は 3.2 節に示す JSON によるクエリ言語を発行することができる。

2.4　MREP

MREP [(3)] は MeCab の出力をベースとしたパターンマッチャーで、形態論情報に対して、品詞と表層文字列をアルファベットとする正規言語相当のパターン (図 6) にマッチすることがで

[(3)] http://www.slideshare.net/unnonouno/miura-dsirnlp6

図 5　超大規模コーパス検索系 (形態論情報検索) による手がかり句「かたわら、」の検索フォーム

きる。

- .
  任意の形態素にマッチ
- <pos=x>
  品詞が x の形態素にマッチ
- <surface=x>
  表記が x の形態素にマッチ
- X*
  X の 1 回以上 の繰り返しにマッチ
- X|Y
  X か Y にマッチ

図 6　MREP のクエリ言語仕様

2.5　ChaKi.NET Dependency Search

図 1,3,5 のフォームは主に形態素列に基づく検索フォームであった。コーパスコンコーダンサ「ChaKi.NET」の Dependency Search 機能では、係り受け部分木に基づく検索が可能である。

図 7 は、手がかり句「をいいことに」（鳥バンク：「副詞句：その他：慣用的表現：状況の悪利用：FUp202」）を ChaKi.NET Dependency Search で検索した例である。

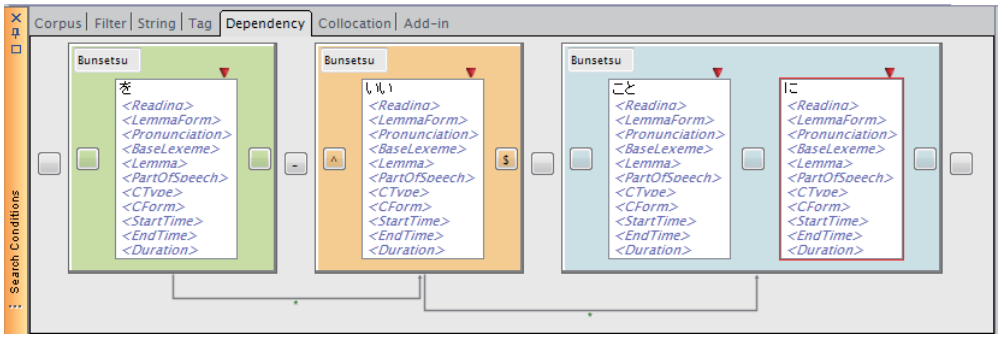

図 7　「ChaKi.NET」による手がかり句「をいいことに」の検索フォーム

この検索フォームも外部 XML ファイルに保存可能である。図 8 のように Tag Search の条件を文節内に格納するような形式で記述する。

```
<DepCond><BunsetsuConds><TagSearchCondition>
 <LexemeConds><LexemeCondition><PropertyPairs><PropertyPair>
  <Key>Surface</Key>
   <Value><StrVal>を</StrVal><IsRegEx>false</IsRegEx><IsCaseSensitive>true</IsCaseSensitive></Value>
   </PropertyPair></PropertyPairs>
   <RelativePosition><Start>0</Start><End>0</End></RelativePosition>
   <LeftConnection>32</LeftConnection><RightConnection>32</RightConnection><IsPivot>false</IsPivot>
 </LexemeCondition></LexemeConds>
 <LeftConnection>32</LeftConnection><RightConnection>45</RightConnection>
 <SegmentTag>Bunsetsu</SegmentTag>
 </TagSearchCondition>
  ...
```

図 8　「ChaKi.NET」による手がかり句「をいいことに」の検索クエリ

2.6　超大規模コーパス 検索系 (係り受け部分木検索)

図 9 は現在開発中の超大規模コーパス検索系のうち係り受け部分木を検索するための試作 UI である。「ChaKi.NET」の Dependency Search を参考にした UI になっている。この UI は 3.3 節に示す JSON によるクエリ言語を発行することができる。

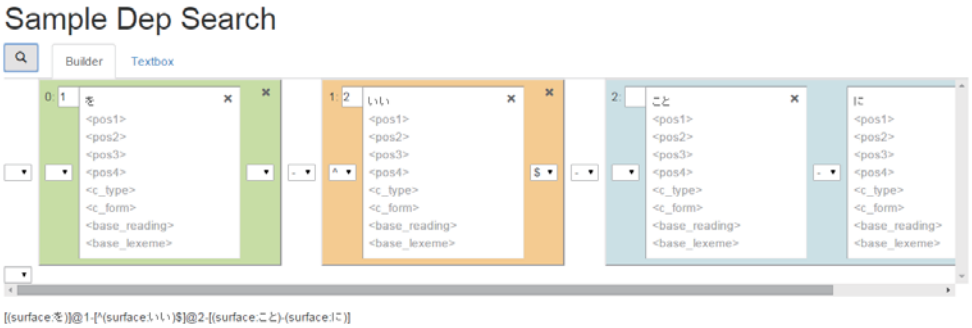

図 9　超大規模コーパス検索系 (係り受け部分木検索) による手がかり句「をいいことに」の検索フォーム

2.7　検索系のちがい

各検索系で細かな点で機能に違いがある。表 1 に各検索系機能のまとめを示す。各項目の意味は以下の通りである：

- 「形態論情報正規表現」: 形態論情報を指定する際に正規表現が指定できるか否か
- 「Ignore Case」: 形態論情報を指定する際に大文字・小文字の違いを無視できるか否か
- 「文頭・文末指定」: 形態素位置を文頭・文末からの相対位置で指定できるか否か
- 「文境界またぎ」: 検索クエリを文境界を越えて指定できるか否か
- 「語彙表・n-gram 出力」: 検索クエリに適合する結果の異なりを度数とともに出力できるか否か
- 「文節境界相対位置」は、文節境界との相対位置で形態素位置を指定できるか否か
- 「係り受け部分木」は、（文節）係り受け関係に基づく検索クエリが発行できるか否か
- 「中心位置（キー）指定」は KWIC 表示時に出力する中心位置を指定できるか否か

3.　ラベリングツールの仕様

3.1　仕様の概要

今回開発したラベリングツールは、形態素解析器 MeCab および 係り受け解析器 CaboCha の出力を入力とする。ラベリングツールの出力形式は、各形態素の右列にラベルを付与するか、

表 1　各検索系機能まとめ

検索系	中納言	ChaKi.NET Tag Search	超大規模コーパス 形態論情報	MREP	ChaKi.NET Dep. Search	超大規模コーパス 係り受け部分木
形態論情報正規表現	○	○	×	×	○	×
Ignore Case	×	○	×	×	○	×
文頭・文末指定	○	×	○	×	○	○
文境界またぎ	○	×	×	×	×	×
語彙表・n-gram 出力	×	○	×	×	○	×
文節境界相対位置	×	×	×	×	○	○
係り受け部分木	×	×	×	×	○	○
中心位置（キー）指定	○	○	○	×	○	×

拡張 CaboCha 形式の SEGMENT_S 相当のラベル (松吉ほか (2014)) を付与することによる。

パターン記述言語は、超大規模コーパス検索系で利用している JSON 形式のクエリ言語を拡張したものを用いる。図 10 にパターン記述言語の仕様について示す。

- パターン記述言語 := {"patterns": [ラベル付きパターン JSON+]}
- ラベル付きパターン JSON := {"pattern": {パターン JSON}, "label": 文字列}
- パターン JSON := 形態素列パターン JSON|係り受け部分木パターン JSON

図 10　パターン記述言語仕様（概要）

「パターン記述言語」は複数の「ラベル付きパターン JSON」からなる。ラベル付きパターンは、検索系で用いられるクエリ言語を流用した「パターン JSON」と付与するラベル (label) からなる。「パターン JSON」は「形態素列パターン JSON」と「係り受け部分木パターン JSON」からなり、それぞれ 3.2 節、3.3 節で説明する。

図 11 の例は、形態素列に基づくパターンを記述したものである。左が「補足節:名詞節:コト型:HSa100」、右が「補足節:名詞節:ノ型:HSa200」のパターンの例である。

図 12 の例は、係り受け部分木に基づくパターンを記述したものである。格要素を持つ用言がなす名詞修飾節（「名詞修飾節:補足語修飾節:限定的:MSa100」）にマッチするパターンを示す。

3.2　形態論情報系

図 13 に形態論情報の指定に利用する形態素列パターン JSON の仕様を示す。

形態素列パターン JSON は、指定する形態素の数からなる形態素 JSON 列 ("morphemes") と、各形態素 JSON の出現位置 ("positions") を指定する形態素出現位置 JSON からなる。形態素列による指定の場合には中心位置（マッチする形態素）を形態素出現位置 “0” により指定するが、係り受け部分木による指定の場合には形態素 JSON の "is_target"を True にして中心位置を指定する。

形態素 JSON は、形態素を指定するためのものである。文字列で形態論情報を完全一致で指定するため、現状では ChaKi.NET などで可能な正規表現による形態素指定ができない。今後、正規表現指定のフラグを入れることを検討する必要がある。

```
{
  "patterns": [
    {
      "pattern": {
        "morphemes": [
          {
            "base_lexeme": "事",
            "pos1": "名詞",
            "pos2": "普通名詞",
            "pos3": "一般"
          },
          {
            "pos1": "助詞"
          }
        ],
        "positions": {
          "0": {
            "min": 0,
            "max": 0
          },
          "1": {
            "min": 1,
            "max": 1
          }
        }
      },
      "label": "補足節:名詞節:コト型:HSa100"
    },
    {
      "pattern": {
        "morphemes": [
          {
            "surface": "の",
            "pos1": "助詞",
            "pos2": "準体助詞"
          }
        ],
        "positions": {
          "0": {
            "min": 0,
            "max": 0
          }
        }
      },
      "label": "補足節:名詞節:ノ型:HSa200"
    },
    ...
```

図 11　形態素列に基づくパターン例

```
{
  "patterns": [
    {
      "pattern": {
        "segments": [
          {
            "morphemes": [
              {
                "pos1": "助詞",
                "pos2": "格助詞"
              }
            ],
            "relations": {},
            "prefix_match": false,
            "suffix_match": false
          },
          {
            "morphemes": [
              {
                "c_form": "連体形-*",
"is_target": true
              }
            ],
            "relations": {},
            "prefix_match": false,
            "suffix_match": true
          },
          {
            "morphemes": [
              {
                "pos1": "名詞"
              }
            ],
            "relations": {},
            "prefix_match": false,
            "suffix_match": false
          }
        ],
        "relations": {},
        "dependencies": {
          "0": 1,
          "1": 2
        },
        "prefix_match": false,
        "suffix_match": true
      },
      "label": "名詞修飾節:補足語修飾節:限定的:MSa100"
    }
  ]
}
```

図 12　係り受け部分木に基づくパターン例

3.3　係り受け部分木系

図 14 に係り受け部分木の指定に利用する係り受け部分木パターン JSON の仕様を示す。

係り受け部分木パターン JSON は文節を指定する文節 JSON 列 ("segments") と、文節間係り受けを表す係り受け JSON 列 ("dependencies") と、文節の相対位置を表す情報 (隣接関

```
形態素列パターン JSON := {
  "morphemes": [ 形態素 JSON+],
  "positions": {
    形態素インデックス番号: 形態素出現位置 JSON+
  }
}

形態素出現位置 JSON := { "min": 出現下限位置, "max": 出現上限位置 }
```

```
形態素 JSON := {
  "surface": 文字列,
  "pos1": 文字列,
  "pos2": 文字列,
  "pos3": 文字列,
  "pos4": 文字列,
  "c_type": 文字列,
  "c_form": 文字列,
  "base_reading": 文字列,
  "base_lexeme": 文字列,
  "is_target": 真偽値
}
```

図 13　パターン記述言語仕様（形態素列パターン JSON）

```
係り受け部分木パターン JSON := {
  "segments": [ 文節 JSON+ ],
  "relations": { 隣接関係 JSON+ },
  "dependencies": { 係り受け JSON+ },
  "prefix_match": 真偽値,
  "suffix_match": 真偽値
}

隣接関係 JSON := 隣接関係ラベル
```

```
文節 JSON := {
  "morphemes": [ 形態素 JSON+ ],
  "relations": { 隣接関係 JSON+ },
  "prefix_match": 真偽値,
  "suffix_match": 真偽値
}

係り受け JSON :=
  文節インデックス番号: 係り先インデックス番号
```

図 14　パターン記述言語仕様（係り受け部分木 JSON）

係 JSON "relations", "prefix_match", "suffix_match") からなる。係り受け JSON は文節インデックス番号により指定する文節の係り先をインデックス番号で指定する。隣接関係 JSON("relations") は、隣接関係ラベルとして、文節間が隣接しているか "-"、出現順を保存するか "<"、何も規定しないか " "が指定できる。"prefix_match", "suffix_match"は文頭・文末に隣接するかを指定する。文節 JSON は形態素 JSON 列と、形態素の相対位置を表す情報 (隣接関係 JSON"relations","prefix_match", "suffix_match") からなる。文節 JSON 内の隣接関係 JSON("relations") は、隣接関係ラベルとして、文節と同様に形態素間の関係を"-"、"<"、" "により指定でき、"prefix_match", "suffix_match"は文節頭・文節末に隣接するかを指定できる。

3.4　クエリ言語をメンテナンスする UI

2.3 節・2.6 節で示した超大規模コーパス検索系の Web UI はクエリ言語を発行することが可能である。一方、JSON 形式のファイルを編集するエディタが各種開発されており、それを用いてメンテナンスすることも可能である。図 15 は XML Editor の一つである oXygen XML Editor [(4)]によりクエリ言語を編集している画面である。図 16 は Google Chrome の拡張機能で

[(4)] http://www.oxygenxml.com/

ある JSON Editor [5]によりクエリ言語を編集している画面である。

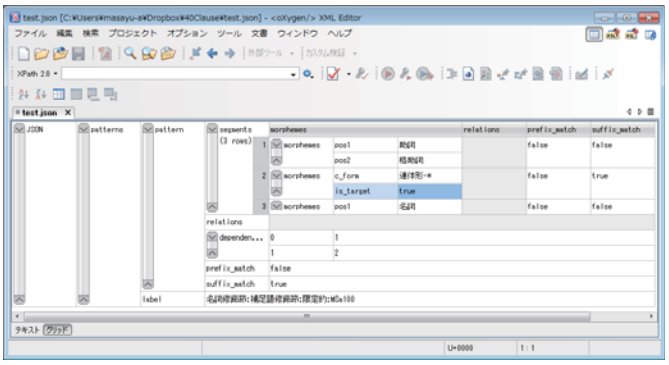

図 15　oXygen XML Editor

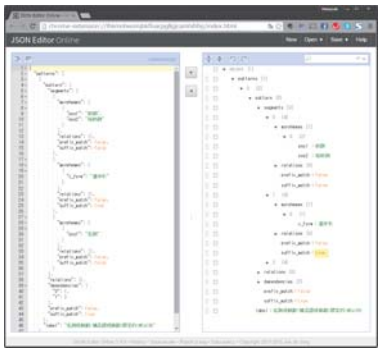

図 16　Google Chrome JSON Editor

4.　節境界ラベリングツールの試作

形態論情報基準[6]・係り受け関係基準[7]については、公開されているコーパスや解析器の出力が事実上の標準として用いられ、共有されている。

一方、節境界については、益岡・田窪品詞体系 (益岡・田窪 (1992)) に基づく丸山ほか (2004) の CBAP や「鳥バンク」(池原 (2007)) で採用されている節境界認定基準などがある。後者の節境界認定基準が公開されているパターンでもっとも細かい記述がされており[8]、解析器が公開されている一方、以下のような問題点がある。

- 品詞体系 IPADIC 品詞体系に基づいており、ChaSen 出力形式である必要がある。
- 文字コードが EUC-JP でなくてはならない。
- 意味属性コードなど、形態論情報や係り受け部分木を超える情報を参照する規則がある。

そこで、UniDic 品詞体系・CaboCha の係り受け構造に基づき、池原 (2007) に規定されている節間意味コードの第 4 段階のパターンの、UTF-8 による再記述を進めている。

5.　おわりに

本研究ではアノテーションの前段階で手がかり句がわかっている場合に用いるラベリングツールの設計と実装について説明した。手がかり句を検索系のクエリで記述可能なことから、各種検索系で発行可能なクエリを示し、各クエリ言語を整理した。整理した結果に基づき、JSON に基づくパターン記述言語を設計し、実装した。

今後の課題として、まず各種検索系との結合が考えられる。パターン記述時には、パターンが実際にコーパス中にマッチする事例を見ながら検討することが多い。検索系と結合すること

---

[5] https://chrome.google.com/webstore/detail/json-editor/lhkmoheomjbkfloacpgllgjcamhihfaj

[6] 益岡・田窪品詞体系益岡・田窪 (1992) に基づく JUMAN・IPADIC 品詞体系に基づく IPADIC/NAIST-jdic/MeCab・UniDic 小木曽・伝 (2013) など。

[7] 京都大学テキストコーパス・KNP・CaboCha など。詳細は浅原 (2013) を参照。

[8] 大分類（第 1 段階）で補足節 28093 パターン、名詞修飾節 40450 パターン、副詞節 66548 パターン、並列節 36321 パターンからなる。

で、シームレスなラベリングツールの開発が行えると考える。手始めとして、「ChaKi.NET」の一機能として実装することを検討している。

次に検索系の整理を行う。形態論情報や係り受け部分木に基づく検索系が複数開発され、検索可能なパターンが少しずつ異なっている。統一的なクエリ言語で対照分析して整理を行う。

最後に節境界ラベリングツールの展開について述べる。第一の動機として、既存のラベリング規則の通時適応がある。現代語の節境界ラベリング規則が固まり次第、近代語への拡張を行う。次に、BCCWJ に対するアノテーションを行う。新聞記事サンプルを中心に鳥バンクの節分類の第 3 段階レベルのアノテーションを進める。さらに、BCCWJ-TimeBank (Asahara et al. (2013)) に対する時制節性 (有田 (2007)) 情報付与があげられる。英語の TimeML (Pustejovsky et al. (2003)) では、SLINK として従属節-主節間の事象構造の関係を付与しているが、これに相当する情報として時制節性のアノテーションを行う。また、鳥バンク節分類と「益岡・田窪体系」(益岡・田窪 (1992)) との節ラベルの対応づけを行う。

謝辞

本研究の一部は科研費基盤 (B)「言語コーパスに対する読文時間付与とその利用」(25284083)、科研費萌芽「近代語コーパスに対する統語情報アノテーション基準策定」(15K12888)、科研費基盤 (C)「「修辞機能」と「脱文脈化程度」の観点からのテキスト分析手法確立と自動化の検討」(15K02535)、科研費若手 (B)「近代口語文翻訳小説コーパスの構築と計量的文体研究」(25770178)、国語研基幹型共同研究プロジェクト「コーパスアノテーションの基礎研究」および国語研「超大規模コーパス構築プロジェクト」によるものです。

参考文献

有田節子 (2007).『日本語条件文と時制節性』　くろしお出版.

浅原正幸 (2013).「係り受けアノテーション基準の比較」　第 3 回コーパス日本語学ワークショップ予稿集, pp. 81–90.

Asahara, M., S. Yasuda, H. Konishi, M. Imada, and K. Maekawa (2013). “BCCWJ-TimeBank: Temporal and Event Information Annotation on Japanese Text.” *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27)*.

池原悟 (2007).「意味類型パターン記述言語仕様書」　Technical report, 独立行政法人科学技術振興機構, 戦略的基礎研究事業, 高度メディア社会の生活情報技術.

国立国語研究所コーパス開発センター (2015a).『コーパス検索アプリケーション「中納言」』, https://chunagon.ninjal.ac.jp/.

国立国語研究所コーパス開発センター編 (2015b).『日本語歴史コーパス』.

Maekawa, Kikuo, Makoto Yamazaki, Toshinobu Ogiso, Takehiko Maruyama, Hideki Ogura, Wakako Kashino, Hanae Koiso, Masaya Yamaguchi, Makiro Tanaka, and Yasuharu Den (2014). “Balanced Corpus of Contemporary Written Japanese.” *Language Resources and Evaluation*, 48, pp. 345–371.

益岡隆志・田窪行則 (1992).『基礎日本語文法–改訂版–』　くろしお出版.

Matsumoto, Yuji, Masayuki Asahara, Kiyota Hashimoto, Yukio Tono, Akira Otani, and Toshio Morita (2006). “An Annotated Corpus Management Tool: ChaKi.” *Proc. of LREC-2006*, pp. 1418–1421.

小木曽智信・伝康晴 (2013).「UniDic2: 拡張性と応用可能性にとんだ電子化辞書」　言語処理学会第 19 回年次大会発表論文集, pp. 912–915.

Pustejovsky, J., J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz (2003). “TimeML: Robust Specification of Event and Temporal Expressions in Text.” *Proceedings of the 5th International Workshop on Computational Semantics (IWCS-5)*, pp. 337–353.

丸山岳彦・柏岡秀紀・熊野正・田中英輝 (2004).「日本語節境界検出プログラム CBAP の開発と評価」　自然言語処理, 11:3, pp. 39–68.

松吉俊・浅原正幸・飯田龍・森田敏生 (2014).「拡張 CaboCha フォーマットの仕様拡張」　第 5 回コーパス日本語学ワークショップ, pp. 223–232.