NPCMJ インタフェースガイドおよびユーザーズ・マニュアル１（2018-11-30）

スティーヴン・ライト・ホーン アラステア・バトラー （訳：吉本啓）

Contents

1 はじめに

本編では、NINJAL Parsed Corpus of Modern Japanese（NPCMJ）の言語データおよびその利用のためのオンライン上のインタフェースについて紹介する。 出発点となるのは、以下のページにあるインタフェースの一覧である:

http://npcmj.ninjal.ac.jp/interfaces/

このガイドの執筆はまだ進行中で不完全なものだということに注意していただきたい。インタフェースは、このマニュアルの中で論じるもの以外にも様々な機能を持っている。さらに、これらのインタフェースを使うだけでは、NPCMJ の持つ全ての側面を捉えつくすことは出来ない。このガイドは、NPCMJ の言語データを一般的なやり方で検索しようとする人のための初歩的な入門書として書かれた。 ここで書かれていない機能やテクニックについては、後に時間が許せば書くことにする。

1.1 ボタンについて

インタフェース一覧のページからインタフェース環境に入れば、元のページに戻ることなくインタフェースの様々な機能を利用していくことができる。 それぞれのページの最上部左側には、「検索インタフェース一覧」（または「コーパス概要」）、「タグ」、および個々の検索機能のための「語の依存関係」「文字列検索」「ツリー検索」のボタンが並んでいる。 また、最上部右側には、言語データの漢字かな／アルファベット表記の選択のためのボタン、日本語／英語の説明の選択のためのボタン、およびクレジット情報のボタンが並んでいる。 検索の機能のボタンのうちどれを選んでも、その下にさらに一列のボタンがあらわれる。 オレンジ色のボタンは「文字列検索について」等と書かれており、それぞれの検索機能の特徴についてのドキュメンテーションを提供する。 このボタンをクリックすると、ドキュメンテーションのテクストが表示される。 検索機能について解説したどのページからも当の検索機能が利用できるようになっている。 検索機能には、「語の依存関係」「文字列検索」「ツリー検索」がある。 これらのうちの1つを選ぶと、上から2行目にその機能の解説のためのオレンジ色のドキュメンテーション用ボタンがあらわれる。 このボタンをクリックすることで、当該の検索機能の使用法について、詳しい情報が得られる。 ここでも、すべての検索機能は直接利用することもできるし、またそれぞれのドキュメンテーションのページの中のボタンから選ぶこともできる。

1.2 謝辞

アノテーションを施された言語データには、以下のものが含まれている:

- 『河北新報』の新聞記事（newswire_KAHOKU）。河北新報社の許諾による。
- 『基礎日本語文法 — 改訂版 — 』の全例文（textbook_kisonihongo）著者の益岡隆志氏および田窪行則氏と版元のくろしお出版の許諾による。
- Basic Japanese: A Grammar and Workbook および Intermediate Japanese: A Grammar and Workbook の例文。著者の Takae Tsujioka 氏および Shoko Hamano 氏と版元の Routledge 社の許諾による。
- A dictionary of basic Japanese grammar”, “A dictionary of intermediate Japanese grammar”, および “A dictionary of advanced Japanese grammar” の例文。著者の Seiichi Makino 氏および Michio Tsutsui 氏と版元の The Japan Times 社の許諾による。
- 『てにをは辞典 — A Dictionary of Japanese Particles』の例文。著者の スー. A. 川島氏および版元の講談社インターナショナルの許諾による。

1.3 引用のためのガイドライン

個々の例文を引用したい場合は、当の例文のテクスト全体（統語木の終端ノードまたはテクスト形式として）に対し一般的な引用情報（”NPCMJ” 等）を付け、下記の形式の書誌情報に関連づけるだけで、例文の特定のためには十分である。 例文の ID 番号はつねに提供されているが、その引用が必須というわけではない。 インタフェースを通じて利用できるすべての例文について、その統語木表示画面の「コンテキスト表示」のボタンを押すと、当該のテクストに関する完全な情報を掲載したページが表示される。 これには、タイトル、日付、出典、ジャンル、所属するサブコーパス、および利用規約等が記されている。

本コーパスを利用して行った研究結果を発表するに際しては、以下の形式の引用を忘れずに行うことをお願いする。

国立国語研究所『NINJAL Parsed Corpus of Modern Japanese』（バージョン 1.0）
（http://npcmj.ninjal.ac.jp/interfaces/ [年月日]確認）.
（バージョンおよび[年月日]は適宜変更のこと）

2　インタフェース・ガイドおよびユーザーズ・マニュアル

2.1　統語解析情報付きコーパスのすすめ

統語解析情報付きコーパスを使わないと出来ないことにはどんなことがあるだろうか？いま仮に、日本語のテクストの中で馬が行っている事柄をすべて調べたいとする。すると、「馬」という語が（あるいは「馬」を含む名詞句が）主語として関係づけられている述語を探すことが課題となる。この場合、「馬が」のような表現はかなり確実に主語と見なしてよいが、それに加えて、「馬は、馬も、馬さえ、馬なら、馬に、馬から、馬の」のように、主語かも知れないがそうでないかも知れない表現も存在する。さらに、周知のように、「馬が追い手を振りきって、走って、山へ逃げた」のような文では、馬は3つの動詞の主語ではあっても、局所的に関連づけられている述語は一つしかない。文脈で既に言及されている「馬」と同一照応の代名詞もあるかも知れないので、それらを調べたいこともあるだろう。のみならず、「鳴く馬」や「しっぽが白い馬」のように、馬について描写を行う表現ではあっても、「馬」の主語としての地位が助詞や語順だけを参照して推論することのできない表現もある。統語解析情報コーパスを使うと、統語構造や文法役割の規定や意味関係のアノテーションを参照することで、「馬」の語（あるいは、「馬」を含む名詞句および「馬」を指示する代名詞）が主語として関連づけられている述語をすべて検索することができる。これについては、2.4.5 節で詳しく述べる。

もう1つ例を挙げる。「笠を手に出かけた」の文の中の「笠を手に」のような表現に興味があるとする。この表現が興味深いのは、「笠を」と「手に」との関係を、明示されていない動詞（「持って」のような）が媒介していると思われるからである。「料理を中心に」や「首都を皮切りに」のような類似の多くの表現は、「A を B として」と言い替えられるのに対し、この表現はそれが出来ないことからなお面白い。統語解析情報付きコーパスを使うと、このような、(i)従属節で、(ii)述語を欠いており、(iii)対格目的語と「に」を伴う助詞句だけからなるものをすべて検索することができる。実際のところ、このような表現は非常に稀である。以下に例をいくつか挙げる:

- 地図を手に目的地を探した。
- 共産党は独自候補の擁立を視野に準備を進めている。
- 親交のあった本願寺門主蓮如の留守中に居室に上がりこみ、蓮如の持念仏の阿弥陀如来像を枕に昼寝をした。
- 荒涼とした市街地を前に、学生らは津波被害の大きさを初めて体感した。
- 「奈良・国宝室生寺の仏たち」の開幕を前に、同展の見どころを一足早く紹介する催しが１３日、仙台市中心部で始まります。
- オリンピックを前に、選手たちは泳ぎ込んだ.

以上に挙げたのは、統語解析情報付きコーパスで出来て、ブラウザーやコンコーダンサーの検索エンジンでは不可能な事柄のうち、わずかな例にすぎない。このことは、文の意味について人が解析し、少数の抽象的な原理にもとづいてデータを構成することによって可能となるのである。日本語の文法に関して、系統立って研究されておらず、統語解析情報付きコーパスが解明の手助けになりそうな問題は数多くある。それは、統語解

析情報付きコーパスは、言語研究者が興味あるものとして取り上げたものばかりでなく、すべての事柄を解析するからである。 実際に NPCMJ を検索して、そこに何があるか見てみることを奨める。

2.2 文字列の検索

コーパスを効果的に使用するためには、それに含まれるデータおよびその構成に関する一般的な知識が必要である。 しかし、日本語に関する一般的な知識以上にコーパスに関する背景知識を必要とせずに情報にアクセスするための簡単な方法として、NPCMJ の「文字列検索」とその関連機能がある。 これは非常に強力なツールであり、検索を簡単にして（他のいくつかの検索エンジンと異なり）ワイルド・カードの使用を不要にする。例えば、現代日本語では単一の語を成さないが、単語の部分の組み合わせとしてテクストに出現する文字列を簡単に検索することができる。 日本語が分かる人なら誰でも、「にはい」という連続する文字列が単一の単語を表すものではないことを知っている。 この文字列を単一の単語として検索しようとしても（すなわち、<wb> で単語の境界を表すと、「<wb> にはい <wb>」を検索しても — これは、文字列検索に ‘Strict’ のオプションを適用することで行える）、結果として何も得られない。
他方、「にはい」は単語の部分の組み合わせとして（それらが平仮名で書かれているかぎりは）は出現する。 例えば、 ‘Character’ オプションを選んで検索することで、「<wb> に‥は‥い<wb>」（‥ は単語の境界が出現しうることをあらわす）を検索できる。 その結果として、

- 「家<wb>に<wb>は<wb>い<wb>ない」 (家には居ない).

が得られる。
　また、 ‘Liberal’ オプションを選ぶと、検索する表現の最初と最後に単語の境界を前提としない。 すなわち、「‥に‥は‥い‥」を検索することになる。 結果として得られるのは、上記のものの他に、以下のような例がある:

- 「簡単<wb>に<wb>は<wb>いか<wb>ない」(簡単には行かない)
- 「エジプト<wb>に<wb>はいろ<wb>う<wb>と<wb>して」(エジプトに入ろうとして)
- 「<wb>わけにはいかない<wb>」
- 「屋根<wb>に<wb>はい上がる」(屋根に這い上がる), etc.

‘Mine’ オプションを選ぶと、結果をかなりの程度絞り込むことができる。 1個のスペース（ここでは、アンダースコア “_” によって表すことになる）を文字列の途中で入力することにより、単語の境界を導入することができる。 例えば、 ‘Mine’ オプションの下で「に_はい」と入力すると、‥に <wb> はい‥ を検索することになり、動詞「入る」や「這い」を含む例は出力し、「居る」や「行く」の例を返すことはない。
　文字列検索に当って、 ‘Strict’ オプションを選ぶと、さらに限定された結果が得られることになる。 この場合、最初と最後に単語の境界が置かれることになる。 例えば、このオプションの下で「に_はい」と入力すると、「<wb> に <wb> はい <wb>」を検索する。 現在のところ、その結果として出力される例文は無いが、それはコーパスの中に「に」の後に独立した単語「肺」「灰」「杯」や「這い」が（少くとも平仮名では）直接後続する例文が無いからである。 また、このような結果から、「這いあがる」が単一の単位であること（すなわち、文字列全体が2つの単語でなく単一の語彙項目として分析されること）が分る。 これは有益な情報である。 動詞連用形に後続する生産的な助動詞（「始める、終わる、出す、止む、忘れる」等）が個別の単語として取り扱われる

一方で、同様に動詞連用形に後続して、「這い上がる、かき集める、言いふらす、撒き散らす、追いかける」のような語彙的複合語を形作るものもある。後者の語形については、最初の動詞と後続の動詞の間に単語の境界は存在しない。

「よくばり検索」を 'Liberal', 'Character', 'Mine' および 'Strict' のオプションと組み合わせて行うことによって、より広範囲の結果を得ることができる。この「よくばり検索」によって、単語の境界に介在することのできる文字数の上限を指定することができる。文字列「に_はい」について「よくばり検索」のオプションを 'Character' とし、介在できる文字数を1にセットすると、以下のような例を得ることができる:

- 「間には悪い感情」,
- 「基本的に鬼は怖いもの」,
- 「調べるのには向いていない」,
- 「活動には狙いが二つ」,
- 「チリにはない考え方」,
- 「おいしいにはおいしいけれど」, etc.

介在できる文字数を大きくすると、さらに多くの結果を得ることができる。
「よくばり検索」におけるオプションの機能は以下の通りである:

- 文字列 A_BC について 'Liberal' オプションを選ぶと、A と BC の間に n 個の文字が介在する例を検索する（単語の境界をどこにも設定しない）。
- 文字 A_BC について 'Character' オプションを選ぶと、A と BC の間に n 個の文字が介在する例を検索する（A の後および BC の前に単語の境界を設定する）。
- 文字列 A_BC について 'Mine' オプションを選ぶと、A と BC の間に n 個の文字が介在する例を検索する（A の前や BC の後に単語の境界を設定しない）。
- 文字列 A_BC について 'Strict' オプションを選ぶと、A と BC の間に n 個の文字が介在する例を検索する（A および BC に単語の境界を設定する）。

基本的な「文字列検索」についても「よくばり文字列検索」についても、その検索の原理については、漢字かな書きテクストについてもローマ字化されたテクストについても変わりがない。

文字列の検索に関しては制約がある。(i) 入力としてどの表記を採用したかにより結果は制約される（例えば、「うま」「ウマ」「馬」はすべて異なる結果をもたらす）。(ii) レンマにもとづく検索は出来ない（例えば、「あります」「あれば」「ある」「あった」はすべて同一のレンマ「有る」を持つ形態素を含んでいるが、当該の語が出現する例をすべて得たいのであれば、「あり、アリ、在り、有り、あれ、アレ、在れ、有れ」等の多様な語形式について個々に検索を行う必要がある。これらの制約は Tgrep-lite を使用するとかなりの程度克服できる (2.4 節を参照のこと）。

このように、文字列検索を使えば特定の文字列を含む文の検索が完全に行える。しかし、文字列検索がもっとも有用なのは、(i) 文字列がどのように分割されるか、使用者が分らない場合と (ii) 単語の連鎖に対しどのような統語構造が与えられるか、使用者が分らない場合である。例えば、たまたま文字列「ならでは」が単一の単語としてタグ付けされることを知らなかった場合、'Liberal' オプションを選んで「なら」「らで」「では」「ならで」「ならでは」「ならではの」のうちどれを検索しても、「ならでは」

を含む結果が返される。 また、もしも文字列「に向けて」が文法機能の違いに応じて2つの異なるタグ付けを与えられることを知らなかった場合、 'Liberal' オプションを選んでこの文字列を検索すれば、(P-ROLE に向けて) と (P-ROLE に) (VB 向け) (P-CONN て) の2つのパターンが得られる （パターンの意義については、下の 2.3 節で論じる）。

要するに、適格な文字列を入力してコーパスの中にそれに対応するパターンがあれば、何らかの検索結果が返される。 ディフォールトの表示によって、それぞれの例文の諸要素がどのように統語構造を構成しているかが示される。 検索結果の表示をよく見ることにより、コーパス中で形態素がどのように分割されているか、またコーパスの統語構造がどのような特徴を持っているかについて習熟することができる。

2.3 基本的構造

本コーパスは第一に、テクストに対し説明を与えたものである。 テクストは、基本的な単位（文および文の断片）に分割される。 基本的単位は、「木」の原理によって統合され、そこでは幹が枝分れしてより小さい部分となり、最終的には葉（テクストの単語）となる。「木」の代りに、大きな箱が小さな箱を内に含み、もっとも小さい箱にはブロック（テクストの単語）が入っている、と考えてもよい。 どちらのように視覚化しても役に立つ。 数学的なモデルとしてはどちらの視覚化でも同じことになるが、電子化されたテクストで木を表現するもっとも簡単な方法はラベル付きのカッコ（箱の中に箱が入っているという捉え方）である。 他方、木構造について説明する時には木の比喩を用い、祖先::子孫、兄弟姉妹::兄弟姉妹、親::子供、などの（家系樹で用いるような）親族用語を使うことが多い。 ただ1つ木の比喩には困惑させられることがある。 それは、言語学における木は逆さまに、葉（単語）を一番下にして書かれるということである。

2.3.1 葉から根へ

まず最初に、木を底辺から構築するにはどうしたらよいかを考えてみよう。 この議論は、NPCMJ において文がどのようにして分析されるかを利用者に理解してもらうために行うのである。 初めに、コーパスの中に存在することが分っている次の文を取り上げてみよう。

- 鈴木さんの言葉はすがすがしくさえあった

この文は、データとしては第一に文字を一列に並べたものである。 そこで、文全体をオンラインの文字列検索インタフェースで検索することができる。 コーパス中にこの文は1度しかあらわれないので、適切な形で検索するなら、結果としてはただ1つの項目が得られる。 間違った形で検索するなら、何の結果も得られない。 現在のところ、上記の文は単に文字を一列に並べたものにすぎないが、コーパスの中では文は語句へと分割され、それらがノードの下で構造として統合されている。 文字列検索を使ってコーパスの検索を成功させるためには、語句の区切りに文字列がマッチすることが必要である。 その1つのやり方は、”Liberal” かまたは ”Character” のオプションを選んで文字列検索を行うことである。 そのどちらを選んでも、文字と文字のすべての間を潜在的な単語の境界と見なすことになる。

- 鈴‖木‖さ‖ん‖の‖言‖葉‖は‖す‖が‖す‖が‖し‖く‖さ‖え‖あ‖っ‖た

検索の結果、上記の区切りの可能性を満たす例文が1つ得られる。 もう1つの検索のやり方は、文の区切りをコーパスにおけるのと全く同じやり方で行い、”Mine” または ”Strict” を条件として選んで検索を行うことである:

- 鈴木さん<wb>の<wb>言葉<wb>は<wb>すがすがしく<wb>さえ<wb>あっ<wb>た

これによっても、結果として唯一の文が得られる。
コーパス中のデータを単語へと分割するやり方は利用者にとって予想することが容易であろう。 というのは、単語分割は大部分学校文法にもとづいているからである。 学校文法の主要な品詞（名詞、動詞、形容詞、副詞、代名詞、助詞、助動詞等）が別個の区切り（単語）として区別され、そのそれぞれが、当該の単語の品詞名をラベル付けされたノードによって支配されている。

NPR 鈴木さん	P-ROLE の	N 言葉	P-OPTR は	ADJI すがすがしく	P-OPTR さえ	VB2 あっ	AXD た

NPR（固有名詞）は名詞の下位カテゴリーである。 単語を含むノードは、ラベル付きカッコ記法によって次のように表される:

```
(NPR 鈴木さん)
```

P-ROLE は文法役割を表す、助詞の下位カテゴリーである:

```
(P-ROLE の)
```

N は普通名詞のラベルである:

```
(N 言葉)
```

P-OPTR は助詞の下位カテゴリーであり、いわゆる「取り立て助詞」を含む:

```
(P-OPTR は)
```

ADJI はイー形容詞のカテゴリーであるが、「ーたい」「ーがたい」「ーやすい」「ーにくい」「ーづらい」のような助動詞的用法は含まない。 上記の例における ADJI 「すがすがしく」は連用形であるが、この活用情報は単語ノードのラベルには含まれない:

```
(ADJI すがすがしく)
```

P-OPTR はまた、用言の連用形の後にもあらわれることができる:

```
(P-OPTR さえ)
```

VB2 は動詞の下位カテゴリーであり、核となる述語に後続して1つまたはそれ以上の文法機能を果たす。 VB2 は通常正規の動詞から派生したものであるが、VB2 として出現する場合は語彙的意味の多くを失う。 ここで、動詞「ある」の語形は、先行する核述語の活用情報を担うための「ダミー」として出現している。 この動詞は実際には連用形（テ／タに接続する活用形）であらわれている:

```
(VB2 あっ)
```

AXD は、過去テンスを表す助動詞に与えられるカテゴリーである:

```
(AXD た)
```

それぞれの単語に品詞が与えられると、単語が出現する直接の文脈を参照して、それらがどのような構造を構成するかを推論することが可能になる。 上記の文についてこのことを、単語ごとに、左から右へと行ってみよう。 構造を構築するための基本原理は、主要語（ヘッド）が句を投射し、句の下で他の構成素と結合する、ということである。 これにより、NPR は（名詞の下位カテゴリーとして）NP （名詞句）を投射する:

```
(NP (NPR 鈴木さん))
```

この NP の直後には所有を表す助詞 P-ROLE が続き、これが主要語として PP を投射する。 言い換えれば、NP は後続の P-ROLE を補部（complement）として修飾する:

```
(PP (NP (NPR 鈴木さん))
    (P-ROLE の))
```

名詞の直前にあらわれる所有を表す助詞は、通常その名詞の補部を表示する。 これにより、PP は、それに後接する N が投射する NP の内部に含まれる:

```
(NP (PP (NP (NPR 鈴木さん))
        (P-ROLE の))
    (N 言葉))
```

ここでもまた、後続の P は NP を補部として取り、その NP を含む PP を投射する:

```
(PP (NP (PP (NP (NPR 鈴木さん))
            (P-ROLE の))
        (N 言葉))
    (P-OPTR は))
```

焦点を表す助詞が名詞に付加された場合、通常それは（何らかの文法役割によって）述語と関連づけられる。「は」の直後に続く「すがすがしく」の語はイ形容詞（ADJI）の品詞に属する。これは、動詞、形容詞、およびコピュラ表現という、述語の中核部分を構成する3種類の表現の1つである。一般に、「述語」とは節（clause）を投射できる、活用を持つカテゴリーのことである。日本語において述語は単一の語によって構成されることもあるが、様々なタイプの複数の単語の連鎖により形作られることも多い。今のところは、「すがすがしく」はそれを先行するPPを含む節を投射していると考えておく。節は IP（inflectional phrase）とラベル付けされる。

```
(IP (PP (NP (PP (NP (NPR 鈴木さん))
                (P-ROLE の))
            (N 言葉))
        (P-OPTR は))
    (ADJI すがすがしく))
```

残りの要素である「さえ」「あっ」「た」は、ADJI を中核とする述語の他の部分と見なすことができる。とりわけ、ここでは過去テンスの形容詞述語「すがすがしかった」が分割され、中核部分が P-OPTR 「さえ」により焦点を当てられている。述語を構成する残りの部分は、中核部分と同一のレベルにあらわれている。

```
(IP (PP (NP (PP (NP (NPR 鈴木さん))
                (P-ROLE の))
            (N 言葉))
        (P-OPTR は))
    (ADJI すがすがしく)
    (P-OPTR さえ)
    (VB2 あっ)
    (AXD た))
```

結果として得られるのは、文の基本的な統語解析木である。しかし、必要に応じて、文法機能に関するより多くの情報をノード・ラベルの拡張の形で付け加えることができる。例えば、PP「鈴木さんの言葉は」は ADJI「すがすがしく」の主語であるが、助詞「は」は主語の文法機能を表示するわけではない。そのため、この情報を付け加えるために、主語の構成素を PP-SBJ とラベル付けする。構成素が文法機能を果すのは他の要素と結合する時なので、機能の指定は構成素と構成素の結合が行われる句のレベルで行われる。主語を表す構成素は述語が投射する句の下で述語と結合するので、主語の文法機能のラベル付けが行われるのは節 IP のすぐ下となる。これに対して、NP 内部の助詞「の」は限定された数の解釈を伴う機能ーー所有者::被所有者（言葉の音節）、全体::部分（言葉の殆ど）、項::述語（言葉の解釈）、等ーーを伴うにすぎないので、この用法に関する限りは拡張は不要である。最後に、IP はつねに拡張タグを必要とする。上記の文は、述語で終わる単文の叙述文である。このような文には IP-MAT（inflectional phrase-matrix）のラベルを与える。

```
(IP-MAT (PP-SBJ (NP (PP (NP (NPR 鈴木さん))
                        (P-ROLE の))
                    (N 言葉))
```

(P-OPTR は))
(ADJI すがすがしく)
(P-OPTR さえ)
(VB2 あっ)
(AXD た))

以上では、比較的単純な文の構造の生成を例として述べた。しかし、同じ修飾と項・述語関係の原理を再帰的に適用することにより、非常に複雑な構造を作り出すことができる。

2.3.2 根から葉へ

次に文についてトップ・ダウン的に、根（ルート）から始めて考えてみよう。すべての文（および文の断片）はルート・ノードを持っており、それがさらにより小さなノード（通常は複雑な構成を持つ句）へと分割される。文の断片（典型的には、述語を持たない省略された発話）は、FRAG とラベル付けされたルート・ノードを持つ。これに対し、完全な文は、それが表す発話のタイプに応じてラベル付けされる。疑問文は CP-QUE と、感嘆文は CP-EXL と、また疑問意外の終助詞や外置された補足表現を伴う文は CP-FINAL とラベル付けされる。これらの各ノードは典型的には、IP-SUB ノードに加えて、後に続く要素（終助詞や補足表現）を持っている。残りのルート・ノードは発話のタイプを表示する付加的要素を伴わないもので、これには叙述文（IP-MAT）と命令文（IP-IMP）がある。

完全な文は、すべて何らかのタイプの IP 節を主文として含んでいるが、その中にさらなる節が埋め込まれることがある。従属節は IP-ADV-SCON のラベルを持っている。これらの節は通常、述部を修飾して様態または条件を規定する。並列節のうちで、最後尾の節以外は IP-ADV-CONJ のラベルを持っている。名詞句の構成要素のうち、名詞を直接修飾する節は IP-REL または IP-EMB とラベル付けされる。いくつかの例外を除いて、IP ノードは何らかの種類の述語を主要部（head）として直接含んでいる。述語とは、大雑把に定義すると、主語について何事かを述べる、活用を持つ表現のことである。もっとも多い述語のタイプは、動詞（VB とラベル付けされる）のものである。IP の直接下に（すなわち、「直接支配されて」）述語の中核部分の品詞（例えば、VB）を表すノードがあらわれ、さらにその品詞ラベルのすぐ下に単語そのもの（テクストの一部をなす文字列）があらわれることが多い。単語は終端ノードとなる。また、IP の主要部となりうる述語中核部分になることができるのは、ADJI（イ-形容詞）、ADJN（ナ-形容詞）、および [NP-PRD + AX]（名詞述語にコピュラが後続したもの）である。

IP の下でその構成要素として出現することのできる句のタイプは一定のものに限られる。言うまでもなく、上で述べたように、IP-ADV は IP の下に再帰的にあらわれることができる。他の主要な句のタイプとしては、NP（裸の名詞句）、PP（助詞句）、ADVP（副詞句）、INTJP（間投詞句）、FRAG（断片）、PU（句読点）がある。PU を除けば、これらの句のどれも、主要部に加えて他の句を含む複雑な句となることがある。

カッコを用いて与えられた構造を表す場合、「ノード A はノード B を直接含んでいる」や「ノード B はノード A に直接含まれている」のように述べてもよいし、また同じ関係を「ノード A はノード B を直接支配している」とか「ノード B はノード A に直接支配されている」のように述べてもよい。木においては、そのような関係は、ルート・ノードに近く、より高い位置にあるノード A からより低い位置にありルート・ノードに遠いノード B へと枝を描くことで示される。木は結合された構造である。すなわち、木の中のすべての要素は、他のすべての要素と枝によって結合されている。枝は2つの異なる関係を表示する。すなわち、先行関係と直接支配関係である。これらの2つの関係を組み合わせることで、木における他のより一般的な関係を定義することができる（例えば、間接的支配関係、直接後続する姉妹関係、ルートによる支配関係、等）。ノード間の関係を規定するによって、検索表現を作ることができるわけである。例えば、「食

べ」という活用形の動詞を主要部とする疑問文を探したい場合は、「食べ」のノードを直接支配する VB を直接支配する IP-SUB を直接支配する CP-QUE を検索すればよい。

2.4 Tgrep-lite を用いた検索

Tgrep-lite は、カッコで表示した木の中の様々な関係を記述するための言語 Tgrep (Pito 1994) をウエッブ上のインターフェースに使えるように改良したものである。 Tgrep は、他のより表現力のある木検索言語、特に Tgrep2 (Rohde 2005) および Tregex (Levy and Andrew 2006) の基礎となった。 私たちの検索インタフェースはまた XPath 検索構文も扱えるが、これについては後で説明することにする。

Tgrep-lite は単純でパワフルで、しかも学びやすい。 以下に、Tgrep-lite を使ってできることの例をいくつか挙げる。 品詞レベルと句レベルにおいて、ノードは四角カッコの中の文字列として指定される（例えば、[IP-MAT], [NP-SBJ], [AX] 等）。 ノードのラベルを使ってノードを指定する簡単な表現を「ノード記述」と呼ぶことにする。 これらの表現における四角カッコは、正規表現の環境を作っている。 そのため、[P] は P を含むすべての文字列、すなわち IP-MAT や PP-SBJ のような句レベルのノードおよび品詞レベルのノードである P-ROLE や NPR 等とマッチする。 何を検索したいかに応じて、ノードをきわめて正確に規定することが必要になることがある。 例えば、[^IP] は IP で始まるすべてのラベルを、また [SBJ$] は SBJ で終わるすべてのラベルとマッチする。 統語的に適格な検索を行なったからといって、つねに結果が得られるとは限らないことに注意する必要がある。 例えば、[^IP$] は IP のラベルに正確にマッチするが、NPCMJ では IP の後につねに何らかの機能情報を付け加えているため、この検索によっては何の結果も得ることができない。 また、正規表現の環境の内部では、ノードの記述を “|” を使って分けることで選言（disjunction）が利用できる。 例えば、検索表現 [IP-REL|IP-EMB] は、空所を含む関係節（IP-REL）かまたは空所の無い関係節（IP-EMB）にマッチする。 ワイルド・カードは2つのアンダースコア “__” で表し、あらゆるノードにマッチする。ノード間の関係は、直接支配（“>”）、支配（“>>”)、直接先行（“.”）、先行（“..”）等のように、シンボルを使って規定する。 終端ノード記述の否定は、単純な文字列の前、またはノードを記述する正規表現の前に “!” を付加して規定する。例えば、“[P-OPTR] < !は” とすると、「は」を除く取り立て助詞を検索する。 同様にして、関係の否定もその関係を表すシンボルの前に “!” を付加して規定する。 例えば、“[P-OPTR] !< は” とすると、「は」という助詞を支配しない取り立て助詞を検索する。 コーパスのデータの中では取り立て助詞の品詞ラベルP-OPTR は必ず取り立て助詞を一つだけ支配するため、“[P-OPTR] < !は” という検索表現は “[P-OPTR] !< は” と等価である。

検索表現の最初のノード（「焦点」となるノードを「ノード$_1$」としておく）について関係の連言（conjunction）を表現したい場合は、ノード$_1$ に対して [関係 + ノード] の対を複数付け加えればよい。 複数の関係の選言（disjunction）は Tgrep-lite では直接には表せない。 ただし、応急措置として、ド・モルガンの法則が使える（2.4.4 を参照）。

不便なことに、レンマに関する検索を Tgrep-lite で行うことはできない。 例えば、検索表現としてある動詞の辞書形（例えば、「ある」の辞書形としての「有る」）を使ったとすると、Tgrep-lite による検索の結果には、この動詞の他の活用形や他の表記（「あり、有り、ある、あっ、有っ、あろ、有ろ、あれ、有れ」等）とマッチするものは含まれない。 XPath のような他の検索言語を使えばレンマによる単語の検索が可能である。木検索のテクスト・ボックスに以下の表現を入力することで検索が行える。

```
//node[@lemma='有る']
```

2.4.1 はじめに

Tgrep-lite で利用できる関係の規定とその使用法については、後続の諸節で詳しく述べることにする。 しかしながら、Tgrep-lite の使用にあたっては、心に留めておかなければいけないことがいくつかある。 第一に、Tgrep-lite が XML エンコーディングされた木のデータベースに対し質問を行うときは、Tgrep 言語を XPath 言語に書き換えることによって行う、ということである。 NPCMJ の XML フォーマットにおける要素で、ツリーの部分に当たるものは全て node の要素名を持っている。 以降はこういった要素を「ノード」という。 XML のノードは、それが持つ属性によって互いに区別される。 ノード記述を構成する値を含む属性には、 @word と @cat の2通りがある。 XML フォーマットの木において、終端ノードは @word の属性を持ち、その他のノード（品詞ノードおよび句ノード）はすべて @cat 属性を持つ。 @word 属性と @cat 属性の両方を同時に持つノードは存在しない。 Tgrep-lite における検索表現の形式が、どちらのタイプのノードが規定されているかを決定する。 @word 属性の値は、木の葉に相当する部分（終端ノード）を構成する文字列である。 検索ボックスの中のテクストの単純な文字列（例えば、「馬」）は、書換えプログラムにより、XML フォーマットにおける @word 属性の値であると解釈される。 使用されている XPath 表現を正確に知りたければ、検索ボックスの下の “reveal” ボタンをクリックしてから検索を再びサブミットすればよい：

```
//node[self::node[@word='馬']]
```

特定の部分的文字列（例えば「りな」）を含む終端ノードを探したい場合は、正規表現の環境を使用する。 次の検索表現

```
/りな/
```

は、当該の部分文字列を含む終端ノードを探すことになる。

```
//node[self::node[matches(@word,'りな')]]
```

この検索結果には、「かぎりなく、織りなす、極まりない、怠りなく、取りなす」等が含まれる。
属性として @word を持つノードは必ず @pt という属性も持つ。 この場合、@word の値が語であれば、@pt の値は “word” となる。 しかし @word の値が痕跡（*T*）やゼロ代名詞（*pro* など）の場合、、@word の値は “zero” となる：

```
//node[self::node[matches(@pt='zero' @word='*T*')]]
```

ツリーでは以下の構造になる：

```
(ZERO *T*)
```

オンラインインタフェースではこの特殊なノードのノード記述はブレース “{}” の中に入れなければならない。 Tgrep-lite においては @pt の値と @word の値はイコール関係にあるように定義されている。 例えば {WORD} == 宇宙人 という関係が成り立ち、{WORD} < 宇宙人 をコーパスにかけても結果を得られない。 しかしイコール関係は推移関係であるため、[N] < 宇宙人 は成り立つ。

木の品詞ノードまたは句ノードのラベルは、XML フォーマットでは非終端ノードの @cat 属性の値となる。 四角カッコの中に記入した単純な文字列（例えば、“[WPRO]”）ことにより、@cat 属性の値（この場合、疑問代名詞の品詞ラベルにマッチする値）を求める検索を行う：

```
//node[self::node[matches(@cat,'wpro')]]
```

@cat 属性の値を検索することに加えて、四角カッコは正規表現の環境を作り出すことに注意が必要である。 そのため、品詞ラベルとして N を持つノードを検索する（同時に、NPR 、 NP 、 PRN のように N を含むより長いラベルを排除する）ためには、正規表現の内部の文字列に対し制限を加えることが必要となる。 記号 “^” で文字列の始まりを、また記号 “$” で文字列の終わりを表す。 そこで、

```
[^N$]
```

は以下の XPath の検索表現に相当する。

```
//node[self::node[matches(@cat,'^n$')]]
```

ワイルドカード（“__”）は、Tgrep-lite の検索においてどんなノードとでもマッチする。これを使って、終端ノードを定義することができる（他のいずれのノードも支配しないノードとして規定している）：

```
__ !< __
```

NPCMJ の現時点のリリースでは、当該のノードは 338,709 個存在する。これは、コーパス中の単語の数に他ならない。同様にして、木のルート（現在のリリースでは、20,424 個）も検索することができる。これは読者に対する課題としておく。

同一関係は、“==” を使って規定される。これは様々に役立てることができる。例えば、他の IP により直接支配される IP はつねに機能情報のタグを与えられ、しかもさらにサブタイプを伴う。並列関係にある IP-ADV-CONJ を除いた、そのような IP-ADV を探すための方法として、一般的な規定に加えて、自己同一的ノード中の部分文字列を否定することで検索を行なうことができる：

```
[IP-ADV-] > [IP] !== [CONJ]
```

ノード規定の認識条件をうまく利用していくためには、ノード・ラベルにどのようなものがあるかについての知識が必要である。巻末の付録（4）に一覧表を掲載しておく。

オンライン・インタフェースを使って終端ノードを記入する場合には、アルファベットの大文字と小文字の区別を行う必要がある。また、“*” のようないくつかの記号を Tgrep-lite に読み込ませるためには、正規表現環境の中で直前にエスケープ記号 “\” を付ける必要がある。そのため、ゼロ代名詞 *pro* を単独で検索したい時に*pro* は検索表現は十分ではあるが、正規表現の中で外な要素と並べる場合は “/*pro*/” を使わなければならない。品詞ノードおよび句ノードの記入に際しては、大文字／小文字は区別されない（すなわち、“[pro]” は “[PRO]” または “[pRo]” または “[PrO]” 等と同じ結果を生じる）。このようにすると、非疑問の代名詞に変えられた品詞タグPROを、疑問代名詞のタグWPROとともに検索する。“PRO” だけを検索したい場合には、“[^PRO$]” のように制約を付ける必要がある。このようにすると、結果として、「私、あなたがた、これ、自分」等が得られる。

ボックスの中に Tgrep-lite の構文規則に従って検索表現を記入した場合、インタフェースはそれを認識し、表現を解釈する。表現が適格であってもコーパス中にマッチするデータが無い場合、スクリーンには

```
There were no results returned.
```

（出力結果無し） と表示される。たとえ検索表現が不適格な場合でも同じメッセージが現れることに注意すること。

ここでも、検索結果のページで “reveal” の箱にチェックを入れてから検索表現を再びサブミットすると、Tgrep-lite 表現の XPath 表現への翻訳が表示される。これを見ることにより、今作成した表現が自分の意図する検索を反映しているか否かを確認することができる。

最後の注意書き：オンラインで Tgrep-lite を使う場合、ノード記述と関係記号との間に半角スペースを入れなければならない。グループ化するためのかっこ “()” （詳しくは 2.4.4 を参照） はノード記述の直前と直後に置いてもいい。

2.4.2 ノードの記述と表現の作成

Tgrep-lite 表現はノードの記述から始まる。記述の仕方は、それが終端ノードか非終端ノードかによって異なる。検索表現の最初に終端ノード（すなわち、文字列）を規定したい場合には、ノード記述は以下のいずれかの形式を取る。

- 正確にマッチする文字列（例えば、馬、国立国語研究所、等）
- 正規表現（例えば、/馬|牛|羊/、 /*pro*/、等)
- 他のノードを直接支配しないノードのワイルドカード（例えば、 __ !< __ ないし /(.*)/ !< /(.*)/）、
- 文字列または正規表現の否定（例えば、!犬、 !/^ど/、等）

単純な文字列に完全にマッチする文字列要素がコーパス中に存在する場合、終端ノードに対する検索結果が出力される。 当該の語形が機能や意味に関して曖昧であるか否かに応じて、検索結果のアノテーションは様々に異なりうる。 例えば、

```
と
```

という表現を検索した場合、接続助詞 P-CONN （例えば、「草と木と」）、 格助詞（文法役割を表す助詞） P-ROLE（例えば、「人と競い合う」）、 補文助詞 P-COMP（例えば、「『信じます』」と言って）、 およびコピュラ助動詞 AX（例えば、「観光の拠点となる」）を結果として返す。

スラッシュ “//” で囲って正規表現とすると、部分的なマッチングを規定することができる。 例えば、

```
/と/
```

という表現を検索すると、「と」を含む文字列要素すべてを結果として返す。

```
/^と/
```

を検索すると、「と」の文字で始まる全ての文字列要素を結果として返す。 また、

```
/と$/
```

を検索すると、「と」の文字で終わるすべての文字列要素を結果として返す。 さらに、

```
/^(と|ど)/
```

を検索すると、「と」または「ど」の文字で始まるすべての文字列要素を返す。

注意を要するのは、Tgrep-lite では四角括弧 “[ ]” が @cat 属性値を見る際の正規表現を示すように設計されているということである。つまり、/^[とど]/ という表現は、上の /^(と|ど)/ という表現等価ではなく、むしろ、間違った表現と見なされる。また、オンライン・インターフェースの検索表現は結局は url の一部となるので、 url を示す際に使われる特殊記号（例えば、“?” and “&”）を Java の検索表現として使うことはできない。 例えば、/か(.*?)ら$/ という表現は何の結果も返さない。

検索表現の最初に非終端ノード（品詞ノードまたは句ノード）を記述する場合には、以下のいずれかの形を取る。

- 正確な一致（例えば、[^N$]; [^NP-SBJ$]; [^IP-ADV-CONJ$]; 等)
- 部分的な一致（例えば、[N]; [VB]; [IP]; 等)
- 他のノードを直接支配するノードを表すワイルドカード（例えば、__ < __ ないし /(.*)/ < /(.*)/）。

終端ノードの場合と異なり、非終端ノードの場合は、一致の否定を表すには “!” を前接させて書くことは出来ないことに注意が必要である。その代りに、ワイルドカードと「同一」関係の否定を使わなければならない：

```
__ !== [^N$]
```

特に句ノードに関しては、ノード記述の一部として語の境界を指定したいこともあるかもしれない。 様々な種類の拡張タグを持つ句ノード（例えば、NP-SBJ, NP-OB1, 等）を検索するには、ハイフンを使用する（例えば、[NP-]）ことができる。 しかし、拡張タグを持っている NP の両方を探したい場合には、

[NP¥b]

とすると、最終部分に境界のあらわれる文字列 NP を含むすべての表現を結果として返す。 これにより、NPR のような表現は除外される。

「隣接して後続する」関係（“.”）を使って、文字列の間の境界を指定することも可能である。 例えば、検索表現として

/に$/ . /^はい/

を入力すると、文字列検索で “Mine” オプションを選んで文字列「に_はい」（すなわち、〃に<wb>はい〃）を検索したのと同じ結果が得られる。

順番を逆にした終端文字列の間に「隣接して後続する」（すなわち、“,”）という関係を指定すると同じ結果が得られる。

特定の品詞の「と」を検索するには、他のノード記述との関係を規定する必要がある。次の表現

と > [-ROLE]

を入力すると、 @word 属性の値として「と」を持ち、@cat 属性の値に “-ROLE” を含むノードにより直接支配されるものを検索する。 このように書くことによって相手を表す格助詞（P-ROLE と）が検索される。 非終端ノードについては、インタフェースの入力は大文字／小文字の区別は関係無く、また -ROLE の文字列を含むノードは品詞ノード・ラベル P-ROLE だけなので、と > [p-role]、と > [P-ROLE]、と >> [-ROLE] のいずれを検索表現と入力しても、同じ結果を得ることになる。

検索表現の最初のノードは、表現全体の「マスター」ノードとなる。 それに続く同一レベルの（すなわち、同一のグルーピングの中の）[関係 + ノード] のペアは、その「マスター」ノードに関する関係を示すことになる。「マスター」ノードは、木表示においても検索結果においても焦点を受けてハイライトされて表示される。 そのため、“[^NP$] < [^N$]” とすると、句ノード NP が焦点とされる。

検索表現の中の最初のノードは、ディフォールトによってグルーピングを開始する。さらに、[関係 + ノード] のペアはグルーピングの中の最も左端のノードに関係づけられるので、

[^N$] < 馬 > [^NP$]

という形式を検索すると、 (i) 終端ノード「馬」にマッチし、 (ii) NP に支配されるような N を探すことになる。 すなわち、一般的な形式で示すと、“A R_1 B R_2 C” は論理的に “(A R_1 B) & (A R_2 C)” に等しい。 （注意：Tgrep2 や Tregex と異なり、Tgrep-lite においては、2つの関係の連言は2つの [関係 + ノード] の対の間に “&” 記号を使用することにより表現することができない。）

複数の関係はカッコ “()” を使ってグループ化することができる。 左カッコ “(” の後の最初のノードが、左カッコと右カッコ “)” の間に挿まれる表現のマスター・ノードとなる。 そこで、次の形式

[^CP-QUE$] < ([^IP-SUB$] < ([^VB$] < 食べ))

を検索すると、「食べ」を直接支配する VB を直接支配する IP-SUB を直接支配する CP-QUE が得られる。 重要な事なので繰り返しておくが、動詞「食べる」の連用形「食べ」だけでなく、そのすべての形式（「たべる」や「タベル」を含めて）を検索するには、XPath 検索の “//node[@lemma=`食べる']” が必要になる。

2.4.3 Tgrep-lite におけるノード間の関係

ノード間の様々な関係は、支配と先行という2つの基本的関係を規定することにより定義できる。 Tgrep-lite には、あらかじめ定義された一式の関係がある。 これらの関係はすべて、1つ以上の文字からなる文字列の記号により表される。 どの関係をとってもその逆の関係が与えられており、これにより、左端の（焦点となる）位置に来るノードを自由に選べるようになっている。 例えば、「直接支配する」（“<”）関係に対しては

「直接支配される」（“>”）関係が存在する。 そこで、[^PP$] < [^NP$] は [^NP$] > [^PP$] と全く同一の構文を記述しているが、焦点だけは交替している。
Tgrep-lite では、以下の関係を利用することができる。

```
A << B      A は B の先祖である（A は B を支配する）
A >> B      A は B の子孫である（A は B に支配される）
A < B       A は B の直接の先祖（親）である（A は B を直接支配する）
A > B       A は B の直接の子孫（子）である（A は B に直接支配される）
A .. B      A は B に先行する
A ,, B      A は B に後続する
A . B       A は B の直前に置かれる
A , B       A は B の直後に置かれる
A $ B       A は B の姉妹である
A $.. B     A は B の姉妹で先行する
A $,, B     A は B の姉妹で後続する
A $. B      A は B の隣接する姉妹で先行する
A $, B      A は B の隣接する姉妹で後続する
A == B      B は A と同一のノードである
A <<, B     B は A の左端の子孫である
A <<- B     B は A の右端の子孫である
A >>, B     A は B の左端の子孫である
A >>- B     A は B の右端の子孫である
A <1 B   B は A の最初の子である
A >1 B      A は B の最初の子である
A <-1 B     B は A の最後の子である
A >-1 B     A は B の最後の子である
A <, B      B は A の最初の子である
A >, B      A は B の最初の子である
A <- B      B は A の最後の子である
A >- B      A は B の最後の子である
A <: B      B は A の唯一の子である
A >: B      A は B の唯一の子である
A <<: B     A は B を枝分かれの無い枝のみによって支配する
A >>: B     A は B に枝分かれの無い枝のみによって支配される
```

2.4.4 より複雑な関係についての注意

ノード記述や関係の否定は、当該のノード記述や関係を表す記号の直前に感嘆符（“!”）を置くことによって行う。 終端記号を否定するには、単純な文字列の直前に感嘆符（“!”）を付ければよい。
（非終端記号の否定は、このやり方では行えない。）
ノード記述同士の間の関係を否定するには、その関係を表す記号の直前に感嘆符 (“!”) を置くことによって行う。

検索表現の最初のノードが表現全体の「マスター」ノードとなる。 そして、その右側にあらわれる [関係 + ノード] の対はすべて、この「マスター」ノードに対して関係づけられる。 それゆえ、

```
A < B < C
```

は、B および C の両方を直接支配するノード A を出力する。

複数の関係を、カッコ “()” を使ってグループ化することが可能である。 左カッコ “(” およびそれと対をなす右カッコ “)” にはさまれた表現の中で、最初のノードがそのマスター・ノードとなる。 そこで、

```
A < (B < C)
```
を入力すると、ノード C を直接支配するようなノード B を直接支配するノード A を出力する。 基本的に、許される埋め込みの深さには限界が無い。 例えば、

```
A < (B < (C < D))
```
は、ノード D を直接支配するノード C を直接支配するノード B を直接支配するノード A を出力する。

検索表現の最初のノード記述は、ディフォールトによりマスター・ノードとなる。 検索表現の中で、冒頭部分以外のマスター・ノードと一緒に明示的なやり方でグループ化されていない [関係 + ノード記述] の対は、すべて最初のノード記述に関係づけられる。 そのため。検索表現中の最初のノード記述の前に左カッコを記入したとしても、それを右カッコとの間にどのようなノード記述がなされるかということに関係なく、無意味である。 すなわち、

```
A < B < C
```
のように記入しても、

```
((A < B) < C)
```
のように記入しても、全く同じことである。

Tgrep2 や Tregex と異なり、Tgrep-lite においては四角カッコ（“[ ]”）の持つ機能の一部が利用できないことに注意が必要である。 そのため、正規表現の中では、複数の完全なノード記述の間の選言は “|” を使って表示でき、

```
[PRO] > [^NP-SBJ$|^NP-OB1$|^NP-OB2$]
```
また正規表現の中に “|” を記入することで、部分的かまたは完全なノード記述を行える

```
[PRO] > [SBJ|OB1|OB2]
```
のに対し、複数の [関係 + ノード] の対の間の選言は直接表せない。 例えば、もしも名詞「人」を主要部とする主語名詞句を検索したいとする（下の 2.4.5 節の例で論じるように）と、-SBJ の拡張タグを持つ裸の主語 NP もしくは助詞を付加された主語、すなわち拡張タグ -SBJ を持つ PP の内部の主要部を探すことになるだろう。 両方の句のタイプともに -SBJ とラベルづけされているとはいえ、それらは主要部の「人」とは異なる構文上の関係を持っている。 Tgrep-lite を使用する場合、1回の検索で2つの記述を選言の形で直接含めるかわりに、2回の異なる検索を行ってその結果をまとめることが可能である。

Tgrep-lite で関係同士の選言を表現するための間接的な方法は、ド・モルガンの法則を使うことである。 -SBJ の拡張タグを持つ裸の NP と助詞を付加された主語、すなわち -SBJ を持つ PP を探すためには、次のようにすればよい：

```
人 > ([N] !== (__ !> [NP-SBJ] !> ([NP] > [PP-SBJ])))
```

2.4.5 Tgrep-lite の使用例

Tgrep-lite では一連の関係と操作が利用でき、検索のために様々な関係を規定することができる。 まず最初に、ノード A とノード B とが同一の文の中に共起しているという平凡なケースを考えてみよう。 A と B の両方を支配するルート・ノード（すなわち、他のいずれのノードによっても直接支配されないノード）を探すための検索表現を作り上げることは可能である。 A を「船」 とし、B を「海」としよう：

```
__ !> __ << 船 << 海
```
以下の検索でも同じ結果が得られるが、この場合、「船」に焦点が当てられる：

```
船 >> (__ !> __ << 海)
```

「船」を正規表現の中に入れることで、文字列の中のどこかに「船」を文字として含む任意のノードへと検索を広げることができる：

```
/船/ >> (__ !> __ << 海)
```

正規表現の内部に選言記号（“|”）を使用することで、同時に同一の条件下で複数のノード記述をいくつでも追加して検索を行うことができる：

```
/船|魚/ >> (__ !> __ << 海)
```

与えられた文脈の中での共起関係は、構文と文法役割が指定される場合には、より興味深いものとなる。主語と目的語の両方を含む（直接支配する）節の検索を例に取って考えてみよう。助詞を付加された項を導く PP も裸の項を導入する NP も、文法役割を指定する拡張タグを伴う、ということを思い出してほしい。ノード記述の中に拡張タグのみを含めることで、一般的な検索を行うことができる：

```
[IP] < [SBJ] < [OB1]
```

また、以下のように、IP が主語を直接支配し、主語が直接目的語と姉妹関係にある、と指定しても同じ検索が行える：

```
[IP] < ([SBJ] $ [OB1])
```

また、次のようにすると、コーパス中の多くの（だが、すべてとは言えない）他動詞に焦点を当てることができる。

```
[VB] > ([IP] < ([SBJ] $ [OB1]))
```

Tgrep-lite の関係を利用して、主語と目的語の語順を規定することもできる。 以下のように検索すると、目的語を姉妹としてそれに後続する主語（かき混ぜられた構文）を探すことになる：

```
[SBJ] $,, [OBJ]
```

これと同等であるが、姉妹である主語に先行する目的語は次のように規定すればよい：

```
[OBJ] $.. [SBJ]
```

さて、このユーザーズ・ガイドの冒頭で述べた最初の問題のバリエーションを、「馬」を「人」に換えて考えてみよう。普通名詞「人」が主語名詞句の主要部であるか、あるいは「人」を主要部として伴う NP が主語の位置にある空要素を指示している例を検索することを考えてみよう。2つの単純な検索–裸の主語の検索および助詞を伴う主語の検索–を行うことにより、解決の一部を見出すことができる：

```
[NP-SBJ] < ([^N$] < 人)

[PP-SBJ] < ([NP] < ([^N$] < 人))
```

また、主要部の名詞「人」を修飾する関係節の内部に主語のトレース (NP-SBJ *T*) を含む例を検索することもできる。

```
[IP-REL] < ([NP-SBJ] < /¥*T¥*/) $. ([N] < 人)
```

残りの問題の一部分は、主語の欠けている IP を少くとも1つ直接支配する IP の中の ([^N$] < 人) を主要部とする項を探すことにより（すなわち、「人」を主要部とする NP がコントロール関係における潜在的先行詞である場合を探すことにより）解決される。項の拡張タグは -SBJ2, -OB2, -LGS, OB1, および SBJ である。 ここでも、裸の項と助動詞を付加された項の検索を別に行う必要がある：

```
[SB|OB|LGS] < ([^N$] < 人) > ([IP] < [IP] !< [SBJ]))

[SB|OB|LGS] < ([NP] < ([^N$] < 人)) > ([IP] < ([IP] !< [SBJ]))
```

これらの検索の結果は求めている例文を含むが、さらに手による選択を必要とする。（TIGERSearch というオフライン・ツールを使えば、単純な表現を1つだけ使って、「人」を主要部とする NP が主語役割を持つ例をすべて特定することが可能である。これについては、3.2 節でより詳しく紹介する。）

当該の句に支配される範囲内で、左端または右端の枝から数えて、姉妹たちとの相対的な順序を指定することが可能である。 例えば、副詞で始まる文を探したければ、次のようにすればよい：

```
[IP] <, [ADVP]
```

あるいは、ADVP に焦点を当てて、他にどの姉妹ノードに後続することもない（IP の下の） ADVP の例を検索することもできる：

```
[ADVP] !$,, __ > [IP]
```

以上、Tgrep-lite で利用可能な関係のいくつかについて手短かに紹介した。 より詳しいドキュメンテーションは、以下のオンライン・ドキュメンテーションで利用可能である：

http://npcmj.ninjal.ac.jp/interfaces/cgi-bin/tree_search.sh?db=npcmj

3 研究のためのコーパスの利用

NPCMJ では一連のインタフェースを介して、けやきツリーバンク（Butler, Yoshimoto, Hiyama, Horn, Nagasaki, and Kubota 2017）のデータの一部にアクセスすることができる。 これらのデータは、高水準のアノテーションを付加された、公に利用可能なテクストを使用している。 さらに、インタフェースを通じて、研究に利用可能ないくつもの機能が存在する。 しかしながら、NPCMJ とけやきツリーバンクとでは、情報の提示や記録の仕方にシステマティックな違いがある。 NPCMJ においては、データは XML フォーマットで記録され、そこにはmecab および UniDic による形態素解析の結果と文字列の線形順序に関する情報とが含まれている。 そのため、レンマ情報の検索や、2つの文字列間の距離を指定した条件による検索（途中に介在する文字列の数を計ることによって行うもの）が比較的容易に行われる。 この種の情報は、けやきツリーバンクのフォーマットでは提供されない。

けやきツリーバンクのフォーマットは、アノテーションの容易さを考えて作られたものである。 それは、カッコで表示された木のフォーマットである。 レンマや線形順序に関する余分な情報は含まれていない。 さらに、構成素の機能情報のうちいくつかは、当の構成素とは別に、「空虚な句」（すなわち、テクスト的な内容を持たない句）として提示される。 例えば、けやきツリーバンクにおいて、「が」で表示される主語名詞句は一般的に以下の形式を取る：

```
(PP (NP (N xxx))
    (P が))
(NP-SBJ *が*)
```

NPCMJ では、この種の情報は、ラベルの拡張の形でノード・ラベルに付け加えられる。

```
(PP-SBJ (NP (N xxx))
        (P-CORE が))
```

3.1 オフライン・ツールを使った研究

特定の検索によって得られた用例の数を数える方法として、CSV（comma-separated values）フォーマットによるダウンロードがある。 結果として得られる例文のおのおのが1行に表示され、その先頭に用例数を表示する数字と文の ID が示される。 用例中の焦点に相当する語句（検索表現のマスター・ノードに相当）は XML 式の開始タグおよび終了タグによって表示される（例えば、検索の焦点が <NP-OB2> であれば、“<NP-OB2> ... </NP-OB2>”）。 そのため、同一の文 ID と例文が、タグの位置が変るだけで複数回出現することもありうる。 文中の単語はスペースによって区切られる。

得られた検索結果に対しさらに処理を加える場合の単純な例として、結果を各文の長さに応じてソートする（文中の単語数によって計った長さによる）。その一つのやり方としては、ファイルの内容に対し以下のスクリプトを使用することができる：

```
awk -F "¥",¥"" '{ print length($3), $0 }' | sort -n
```

特に、

```
$ cat /path/filename | awk -F "¥",¥"" '{ print length($3), $0 }' | sort -n
```

とすることにより、各例文の文字数を計り（$3 で同定する）、その数にもとづいて行をソートする。

CSV 形式のもう1つの特徴として、検索の焦点が XML 式のタグによりはさみ込まれる形で区別されることがある。必要に応じ、これらを書き換えて、すべての例文中の焦点が縦一列に並ぶように配置することも可能である。それには、次のようにすればよい：

```
$ cat /path/filename | sed 's@<[^/ ]¥+>@",@; s@<¥/[^/ ]¥+>@",@'
```

ダウンロードされたファイルは、様々なツールを用いた検索や処理を行うことが可能である。例えば、「CSV 形式」は処理を加えてスプレッドシート・プログラムにインポートすることができる。もしくは、Goldvarb を使って解析するために、“.Tok”ファイルを生成するのに利用することができる。

http://albuquerque.bioinformatics.uottawa.ca/GoldVarb/GoldManual.dir/GVManual.html

3.1.1 オフライン・ツールを使ったケース・スタディ

以下では、オンラインのインタフェースからデータをダウンロードし、それをオフラインのツールを使って分析するやり方について概略を述べることを目的として、簡単なケース・スタディを行う。ツリーのソースファイルに直接アクセスすることができれば、一般的な検索を利用して、特定の問題に関連するデータを収集することができる。その後で、オンラインのものよりも強力なツールを使ってデータを分析することが可能である。

ケース・スタディを行うには（そして、他の多くの同じようなタイプの操作を行うには）、tsurgeon、bash、sed、gawk のコードを使う。そのため OS としては Linux を、Windows 上でならば、Windows Subsystem for Linux (https://docs.microsoft.com/en-us/windows/wsl/install-win10) を使うことを勧める。次のステップとして、カッコ表示木のファイルを扱うための様々なツールを利用することができる。まず、GitHub のウェブサイトから次のコマンドを実行して、ディレクトリを複製しよう：

```
$ git clone https://github.com/ajb129/normalize.git
```

README ファイルの説明にあるように、munge-trees と stanford-tregex.jar もダウンロードしなければならない。そして、複製したディレクトリの中の /bin ディレクトリを PATH に加え、tregex_location の中身を、自分自身の stanford-tregex.jar の置き場所に変更しなければならない。

次に、以下のディレクトリを複製し、PATH の設定に /bin フォルダを加えよう：

```
$ git clone https://github.com/ajb129/keyaki-aid.git
```

これらのステップを踏むことによって、以下のケース・スタディを進めることが可能になる。

1つ以上の主語を含む文（すなわち、「二重主語文」およびそれらから派生可能な文）のことを考えてみよう。例えば、「すなわち駒は1手ごとに性能が変わる」のような文がある。これらの文を調べてみると、述語に最も近い位置にある「が」で表示された主語は関係名詞のことが多いことが分かる。仮に、< SBJ の主要部名詞、SBJ2 の主要部名

詞、述語> という形式の3つ組の語彙情報のリストを抽出したいとする。 上記の例文からは、< 駒、性能、変わる > という3つ組の列が得られる。 通常、これらの文はコーパスの中で、-SBJ の拡張タグを伴う句（PP または NP）を含み、さらにその後に -SBJ2 の拡張タグを持つ2番目の句 PP または NP が後続するものとしてアノテートされる。 オンライン・インターフェースでは、[SBJ2] を検索することで SBJ2 を含む文を選び出すことができる。 しかし、検索結果や抽出された情報を操作するために、統語アノテーションを利用したい場合には、カッコ形式のファイルにアクセスする必要がある。

NPCMJ の全公開済みデータはオンライン・インターフェース（以下）の概要（Overview）ページおよびNPCMJ プロジェクトのトップページからダウンロードすることができる：

http://npcmj.ninjal.ac.jp/

現状では、136のテクストがリリースされている。 以下、136のファイルについてこのステップを行い、すべてのカッコ形式のファイルが my_NPCMJ という名前のフォルダに入っていることを前提に話を進める。
まずは、専用のディレクトリ（例えば、sandbox）を作成し my_NPCMJ フォルダをその中に入れる。

これ以降は、オフライン・ツールの使用が必要になる。
そのようなツールとしては、munge-trees:

http://web.science.mq.edu.au/~mjohnson/Software.htm

Tgrep2:

https://tedlab.mit.edu/~dr/Tgrep2/

Tsurgeon および Tregex：

https://nlp.stanford.edu/software/tregex.html

がある。

次に行うべきはそれぞれのファイルに含まれるツリー（ツリーだけ）すべてを1つのファイルにまとめることである。カッコ形式でダウンロードしたファイルに含まれるメタ情報は除外しなければならない。ツリーのみを自動的に取り出すためのコマンドとして csearch_collect があるが、このコマンドを走らせるにはまず最初に、ターミナルで以下を行って、my_NPCMJ 中のファイルの拡張子を .txt から .psd に変更する必要がある：

```
$ for f in *.txt; do  mv "$f" "${f%.txt}.psd"; done
```

これですべてのツリーを集め、ひとつのファイルのまとめるためのコマンドを使うことができるようになった。 my_NPCMJ フォルダの中で、次を実行する：

```
$ csearch_collect > ../gold.psd
```

次に、sandbox フォルダーの中で、以下を走らせる；

```
$ cat gold.psd
```

これでそれぞれのツリーが一行にまとめられたことを確認できる。

ここから先は、Tsurgeon を使って、終端ノードと品詞ノードの間の WORD ノードを削除する必要がある。 このノードは NPCMJ においては情報を有するものとして使われるが、同じツールのセットを使うコーパスでは語形成あるいは正書法の情報を有する。tree_to_tnt コマンドが終端の文字列と品詞のペアを返す（これは、この後に行うパイプラインである）ようにするために、このノードは削除しておかなければならない。 上記の sandbox フォルダーの中に、以下のような内容を持つファイル change.tsurgeon を作っておこう。

```
8<8<8<8<8<8<8<8<8<8<8<

/WORD|ZERO/=x
```

```
excise x x

8<8<8<8<8<8<8<8<8<8<8<
```

このファイルを sandbox フォルダーに置き、以下を走らせる：

```
$ cat gold.psd | tsurgeon_script change.tsurgeon | sponge gold.psd
```

次に、SBJ2 というノードをもつツリーを選び出すために、次を実行する：

```
$ cat gold.psd | grep "SBJ2" > SBJ2.psd
```

これ以降は、Tgrep-lite の代わりに、少々異なった検索言語である Tgrep または Tgrep2 を使うことになることだろう。 この形式のファイルでは、Tsurgeon スクリプトを動かすこともできる。 さて、次のようにすることにより、-SBJ2 とタグ付けされた句の用例数を数えることができる：

```
$ tregex "/SBJ2/" SBJ2.psd
```

このコマンドを実行すると、次にような応答が得られる：

```
There were 699 matches in total.
```

（全部で699のマッチがあります。）

そのようなファイルはまた、語彙的な内容を含まない主語名詞句の用例 –関係節内部のトレース、様々な種類のゼロ代名詞、必須格名詞句が明示されず、構文のより高い階層にある先行詞から指示対象を継承している場合等– も含んでいる。 トレースやゼロ代名詞の形式の主語は節の中でつねに IP の下の NP としてあらわれ、それに直接支配されて、一般的な形式としては *form* の形を取る空要素が出現する。 SBJ2 の下のそのような要素は、次のようにすれば検索できる：

```
$ tregex "/SBJ2/ < /¥*/" SBJ2.psd
```

この検索の結果、全部で 24 のマッチが得られる。 SBJ2 とタグ付けされた句と共起する空の NP-SBJ についても同様にして検索することができる：

```
$ tregex "/SBJ/ < /¥*/ $ /SBJ2/ " SBJ2.psd
```

この検索により、全部で 294 のマッチが得られる。 主語の位置が空である場合の検索は、次のようにすればよい：

```
$ tregex "/SBJ2/ !$ /SBJ/ " SBJ2.psd
```

検索の結果、151 のマッチが得られる。 すなわち、SBJ2 句を伴う節 699 例のうち、実に 445 例が、その「大主語」の指示対象を局所的領域の外から得ていることになる。 このこと自体が興味深い結果であると言える。

ところで、SBJ2 句と一緒に明示的な述語があらわれない例は 13 ある。 以上の要因をすべて組み合わせて検索を行うと、二重主語構文の例文のうち、語彙項目を持つ完全な3つ組みのリストに含まれないものがいくつあるかを知ることができる。

```
$ tregex "/SBJ2/ [ < /¥*/ | !$ /ADJ|VB|PRD/ | !$ /SBJ/ | $ (/SBJ/ < /¥*/) ]" SBJ2.psd
```

上のようにすると、全部で 482 のマッチが得られる。 そこで、明示的な語彙項目を持つ3つ組みのリストには、おおよそ 699 - 482 = 217 のエントリーがあることになる。

ところで、複雑なリストを作るためには、各々の要素に対して焦点を当てることが必要である。 このことは、単純な検索だけで行うことはできない。 その代わりに、元々のデータに手を加え、抽出したい項目に「ハイライトを与える」必要がある。 NP の主要部となる、語彙的内容を持つ語は通常普通名詞（N）、固有名詞（NPR）、または代名詞（PRO）の品詞カテゴリーの下にあらわれる。 述語があらわれるのは、形容詞（ADJI,

ADJN）や動詞（VB）の品詞カテゴリーの下か、あるいは名詞述語（NP-PRD）の主要部の下である。これらのそれぞれにラベルを付けることで、それらを集めてリストを表示することができる。上記の sandbox フォルダーの中に、以下のような内容に change.tsurgeon を変えておこう。

```
8<8<8<8<8<8<8<8<8<8<8<

/SBJ¥b/ [ < N|NPR|PRO=a | < (NP < N|NPR|PRO=a) ] $ (/SBJ2¥b/ [ < N|NPR|PRO=b | < (NP < N|NPR|PRO=b) ]) [ $ /ADJ|VB/=c | $ (/PRD/ < __=c) ]

relabel a N_sbj_of_interest
relabel b N_sbj2_of_interest
relabel c predicate_of_interest

8<8<8<8<8<8<8<8<8<8<8<
```

このファイルを sandbox フォルダーに置き、以下を走らせる：

```
$ cat SBJ2.psd | tsurgeon_script change.tsurgeon | grep interest
```

次に以下を実行しよう：

```
$ cat SBJ2.psd | tsurgeon_script change.tsurgeon | grep interest > interesting_trees.psd
```

次に以下のコマンドを走らせて、リストを作成しよう。

```
$ cat interesting_trees.psd | tree_to_tnt | awk '/interest/ { print ; next } ; /EOS/ {printf("¥n") } ' > triples.txt
```

出来上がったリストは 113 項目を含む、次のようなものになるだろう。

```
童舞 N_sbj_of_interest
希少価値 N_sbj2_of_interest
高い predicate_of_interest

北部 N_sbj_of_interest
南東側 N_sbj2_of_interest
分かれる predicate_of_interest

皇位継承 N_sbj_of_interest
不安 N_sbj2_of_interest
つきまとう predicate_of_interest

etc.
```

次のコマンドは、triples.txt ファイルから SBJ2 のヘッドとなっている語を抽出する。つまり、コーパスにおいて SBJ2 として現れた語の正確なリストを提供する。

```
$ cat triples.txt | grep 'sbj2_of' | sort | uniq -c | sort -n | awk '{ print $2 }'
```

以下は「しよう・仕方がない」、「ほうがいい」、「仲が良い」等の慣用的な表現や、かなりの割合の関係名詞（relational nouns）を含んだリストである。

ステンドグラス, ファッション, あさって, スピード, ハンドル, しよう, テンポ, ほう, 下地, 不安, 予測, 事, 事故, 二極化, 仕方, 仲, 値うち, 備蓄, 入場整理券, 写本, 分析, 割合, 南東側, 参加者, 取り扱い, 問題, 国力, 売り上げ, 夜景, 大手, 大進, 実現性, 寿命, 工事, 市域, 広島, 建造物, 彩色, 復旧, 思い, 性能, 成績, 文化, 文字, 方, 旧石巻医療圏, 格差, 楽譜, 歩み, 民営化方針, 気仙, 無理, 独眼竜政宗, 現存作, 発色, 発達, 着工, 知名度, 経済規模, 者, 背, 能

力, 腹, 自宅, 色, 苦闘, 英語, 行き先, 被害, 補修, 計画, 記憶, 話, 説得力, 車, 運用数, 道路, 部分, 関係, 閲覧, 面, 面積, 類似点, 風景, 高齢化, こと, 場合, 希少価値, 歯, 気, 点, 目, 評価, 違い, 可能性, の

上記のリストは問題となるすべての要素を完全に抽出したものではない（完全な目録はいくつかの、ここで述べたものよりもずっと複雑な構造を含む）。

このケース・スタディでは一連の操作について説明したが、これは適用可能なテクニックのうちいくつかについて説明することを目的とするものにすぎない。

3.2 コーパスのさらなる利用

コーパスで使われている言語は複雑なパターンを持ち、それらが構造の点でも語彙的内容の点でも繰り返され、また変化してあらわれる。 これらのパターンに対して与えられる木構造とノード記述（アノテーション）は、一般的な文法の機能に関するいくつかの単純で基本的な原理、および特に日本語の仕組みに関する多様な経験則に従って決定される。 実際のアノテーションの背後にある諸原理は、以下のインターネット上のアドレスで利用可能なアノテーション・マニュアルに詳しく書かれている：

http://www.compling.jp/keyaki/

アノテーション・マニュアルはアノテーターを考慮して作られたものだが、日本語文法における諸々のパターンがコーパス中でどのような構造によって表されているかの豊富な情報を含んでいる。 アノテーション・マニュアルで使用されているノード記述はオンライン・インタフェースで使われているものとは少々異なっているが、わずかな例外を除けば構造は対応している、ということに注意してほしい。 オンライン・インタフェース専用に書かれたユーザー版のマニュアルが出来上がるまでは、どんな形であれコーパスを使って網羅的な研究を行いたい場合、つねにアノテーション・マニュアルの記述によって得るところがあるだろう。

例えば、アノテーション・マニュアルには、コントロールの原理がどのように定義されるかについて説明がなされ、また、それを使って「太郎は今度苦笑しながら頷いた」の文における「太郎」と「苦笑し」との間におけるような非局所的な依存関係をいかに捉えることができるかが示されている。 この文において、「太郎」は主節において「頷いた」の局所的な項となっているが、それにもかかわらず、「苦笑し」を主要部とする従属節の空虚な主語位置の指示対象の源泉として機能している。 非局所的な依存関係を形作る構文としては他にも、Across The Board 抽出における左端の主節構成素や、関係節におけるトレースの存在や、束縛情報にもとづく先行詞の代名詞への結び付きや、外置された構成素をどの場所において解釈すべきかを規定する、一般的なインデクス付けのメカニズムがある。

これらの非局所的依存関係は、コーパスのデータを TIGERSearch のフォーマットに書き換えることによって検索が可能になる。 このデータ形式については、TIGERSearch ツールとともに、近いうちに説明したい。

4 付録

この節では，使用されるすべてのタグのラベルを一覧表として掲げることにする。

品詞タグ

QUOT	引用符 (quote)
-LRB-	左カッコ (left bracket)
-RRB-	右カッコ (right bracket)
PU	句読点 (punctuation)
ADJI	イ形容詞 (い-adjective)
ADJN	ナ形容詞 (な-adjective)

ADV	副詞 (adverb)
AX	助動詞 (auxiliary verb (including copula))
AXD	テンス標識(助動詞の一部) (auxiliary verb, past tense),過去テンス
CL	助数詞 (classifier)
CONJ	並列接続詞 (coordinating conjunction)
D	限定詞 (determiner)
FN	形式名詞 (formal noun)
FW	外来語 (foreign word)
INTJ	間投詞 (interjection)
MD	モーダル要素 (modal element)
N	名詞 (noun)
NEG	否定辞 (negation)
NPR	固有名詞 (proper noun)
NUM	数詞 (numeral)
P	助詞 (particle)
P-COMP	補文助詞 (complementizer)
P-CONN	接続助詞 (conjunctional particle)
P-FINAL	終助詞 (final particle)
P-OPTR	とりたて助詞 (toritate particle)
P-ROLE	格助詞 (role particle)
PASS	受動助動詞 (passive)
PNL	連体詞 (prenominal)
PRO	代名詞 (pronoun)
Q	量化詞 (quantifier)
QN	量化名詞 (noun with quantifier)
SYM	記号 (symbol)
VB	動詞(語幹) (verb (or verb stem))
VBO	軽動詞 (light verb)
VB2	補助動詞 (secondary verb)
WADV	疑問副詞 (indeterminate adverb)
WD	疑問限定詞 (indeterminate determiner)
WNUM	疑問数詞 (indeterminate numeral)
WPRO	疑問代名詞 (indeterminate pronoun)

統語タグ

ADVP	副詞句 (adverb phrase)
CONJP	接続詞句 (conjunction phrase)
CP-EXL	感嘆節 (exclamative)
CP-FINAL	終助詞節 (projection for sentence final particle)
CP-QUE	疑問節(直接または間接) (question (direct or indirect))
CP-QUE-ADV	副詞的な疑問節 (question used adverbially)
CP-QUE-OB1	目的語として用いられた疑問節 (question used as object)
CP-QUE-PRD	述語として用いられた疑問節 (question used as a nominal predicate)
CP-THT	THAT節 (complementizer clause)
CP-THT-ADV	副詞的な THAT 節 (quote used adverbially)
CP-THT-SBJ	主語として用いられた THAT 節 (quote used as subject)
FRAG	断片 (fragment)
FRAG-IMP	断片による命令文 (imperative with fragment)
FS	言い間違え (false start)
INTJP	間投詞句 (interjection phrase)
IP-ADV	副詞節 (adverbial clause)
IP-ADV-CND	条件節 (conditional clause)
IP-ADV-CONJ	等位的な節 (coordinated clause)
IP-ADV-SCON	従属的な節 (subordinate clause)
IP-EMB	空所なし名詞修飾節 (gapless noun-modifying clause)
IP-IMP	命令節 (imperative clause)
IP-SMC	小節 (small clause)

IP-MAT	主節 (matrix clause)
IP-REL	関係節 (relative clause)
IP-SUB	準主節(clause under CP* layer)
IP-NML	名詞化節 (nominalized clause)
multi-sentence	多重文 (multiple sentence)
NML	中間名詞句 (intermediate nominal layer)
NP	名詞句 (noun phrase)
NP-ADV	副詞的名詞句 (adverbial noun phrase)
NP-LGS	論理的主語名詞句 (logical subject noun phrase)
NP-LOC	場所名詞句 (locational noun phrase)
NP-MSR	数量名詞句 (measure noun phrase)
NP-OB1	第一目的語名詞句 (noun phrase first object)
NP-OB2	第二目的語名詞句 (noun phrase second object)
NP-POS	所有名詞句 (possessive noun phrase)
NP-PRD	述語名詞句 (predicate noun phrase)
NP-SBJ	主語名詞句 (noun phrase subject)
NP-SBJ2	第二主語名詞句 (noun phrase second subject)
NP-TMP	時間名詞句 (temporal noun phrase)
NP-TPC	主題名詞句 (topic noun phrase)
NP-VOC	呼格名詞句 (vocative noun phrase)
NUMCLP	助数詞句 (numeral-classifier phrase)
PNLP	連体句 (prenominal phrase)
PP	助詞句 (particle phrase)
PP-ADV	副詞的助詞句 (adverbial noun phrase)
PP-CND	条件を表す助詞句 (conditional)
PP-CONJ	等位的助詞句 (coordination)
PP-LGS	論理的主語助詞句 (logical subject)
PP-LOC	場所助詞句 (location)
PP-MSR	数量助詞句 (measurement)
PP-OB1	第一目的語助詞句 (first object)
PP-OB2	第二目的語助詞句 (second object)
PP-PRD	述語助詞句 (predicate)
PP-PRP	目的助詞句 (purpositive particle phrase)
PP-SBJ	主語助詞句 (subject)
PP-SBJ2	第二主語助詞句 (second subject)
PP-SCON	従属節助詞句 (subordination)
PP-TMP	時間助詞句 (temporal)
PP-TPC	主題助詞句 (topic)
PP-VOC	呼格助詞句 (vocative)
PRN	カッコ挿入句 (parenthetical)

節連結を特定するためのタグ

-CND	条件 (conditional)
-SCON	従属接続 (subordinate conjunction)
-CONJ	並列接続 (coordinate conjunction)

他のタグ

LS	リスト (list item)
LST	リスト項目 (list)
META	メタ情報 (meta information)

References

Butler, Alastair, Kei Yoshimoto, Shota Hiyama, Stephen Wright Horn, Iku Nagasaki, and Ai Kubota. 2017. The Keyaki Treebank Parsed Corpus, Version 1.0. (http://www.compling.jp/keyaki/ accessed 2017/10/28).

Levy, Roger and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structure. In **5th International conference on Language Resources and Evaluation**.

Pito, Richard. 1994. tgrepdoc – documentation for tgrep. University of Pennsylvania.

Rohde, Douglas. 2005. TGrep2 User Manual version 1.15. (http://tedlab.mit.edu/~dr/Tgrep2).