TR-IT-0269

統計的言語処理による日本語形態素・構文解析

An Overview of Statistics Based Natural Language Processing: Japanese Morphological and Syntactic Analysis

河田 康裕　　柏岡 秀紀
Yasuhiro KAWATA and Kashioka HIDEKI

1998 年 8 月 31 日

概　要

本稿では、我々の統計的言語処理手法を概説する。統計的枠組みによる形態素・構文解析では、大量の分析済の学習データが必要である。コーパスに基づき文法を開発する環境、開発した日本語文法の概容、その文法に従って作成した日本語ツリーバンクについて報告する。また、文法開発とツリーバンク作成に用いたツール、作成したツリーバンクデータを、学習データとして使用して処理する統計的日本語解析の現状と、今後取り組むべき問題点についても検討した。

エイ・ティ・アール音声翻訳通信研究所
ATR Interpreting Telecommunications Research Laboratories

目次

第 1 章

背景

今日までの自然言語処理技術の研究成果として、ある程度の精度で、目的を限定された使用に耐える自然言語処理応用システム[1]が市場に出て実働している。一般に、音声認識では、確率に基づく予測モデルが、言語処理では、規則に基づく処理モデルが主流であった。近年、これらを組み合わせる手法が、提案されている。そこでは、(1) 従来の言語処理で研究されてきたような種類の知識を、統計を使って自動的に学習したり、(2) 従来の言語処理手法で、特に難しかった問題を統計手法で解決する手法などが研究されている。本稿では、ATR 統計的言語処理解析器の概容と、日本語旅行会話の解析の現状を報告する。

統計的言語処理では、全体的な手法が可能である。言語の科学では、自然言語の分析を行う際に、伝統的に、形態論、構文論、意味論、語用論、という枠組みを想定してきた。そのため、従来の自然言語処理でも、形態素解析の出力が構文解析へ、構文解析の出力が意味解析へという手法が一般的であった。しかしながら、意味的な情報が構文解析を決定したり、構文構造の情報が形態素解析を決定すると考えられることも数多くある。人間は、言語記号を時系列的に処理するが、そこには異なったレベルの処理が、同時に働いているように思われる。ここで述べる、我々の統計的自然言語処理手法では、形態素解析と構文解析を、ある程度、同時に処理するので、従来手法に比べて、それらを融合した処理を実現すると考えられる。品詞体系に、詳細な意味的情報を追加したり、学習コーパスに、体系的な語用論的注釈を付加することにより、さらに広範囲の概念を取込んだ言語処理を実現する可能性がある。

統計的言語処理は、言語学習のモデルとしても興味ある手法である。人間は、ひとりでに自然言語を習得する。言語の研究が、ある程度進んだなら、次に、言語学習の研究が可能であると言われている。自然言語を処理するモデルでは、必要な知識の学習可能性が、明確であることが望ましい。統計的自然言語処理手法のように、形態素解析や構文解析の機構が、形態素や構文構造の分析済みデータを、実現可能なアルゴリズムにより学習して内在化し、それらの経験的事実に基づく知識を拠り所に言語処理が実現する

[1] 例えば、機械翻訳や、自然言語インターフェイスなど。

ならば、それは、言語学習のモデルとなり得る。本稿で述べる統計的自然言語処理手法では、解析器は、大量の分析済ツリーバンクから確率付決定木を学習する。解析器は、学習した決定木に制御されて解析処理を行うので、同じ機構が、異なるツリーバンクを学習すると、異なる振舞いをする可能性もある。

統計的自然言語処理は、ロバスト（頑健）である。確率に基づいて処理する手法では、学習したデータから、尤度の高い解答が得られなくても、より尤度の低い解答を出すことが可能である場合が多い。ロバストさは、実用システムにとって、重要な性質である。これらの興味深い性質のため、近年、統計的言語処理への関心が高まっている。

第 2 章

ATR 統計解析器の概要

ATR 統計解析器は、文法を用いた解析器と確率付決定木を用いた解析器の折衷型として提案された。８０年代に、文脈自由文法に、統計手法を応用した確率付文脈自由文法 (PCFG) が登場した。一方で、確率付決定木を用いて、言語的知識に一切たよらない、統計的パターン認識による解析手法 [7] も提案された。ATR 統計解析器は、文法規則をはじめとする言語知識と、確率付決定木とを、両方用いて、PCFG 解析器よりも高精度で、しかも、確率付決定木のみによる解析器よりも、言語的に妥当と考えられる解析機構を目指している。

ATR 統計解析器は、文法と決定木の折衷案というだけではない。我々は、統計解析器の決定木が、詳細な言語知識を参照するならば、解析の精度は、向上すると考えているので、決定木における質問事項は、解析中の文と、解析途中の状態について、任意の質問ができるようになっている。言語知識を用いない決定木解析器では、構成要素の区切りを行うなど、単純な浅い解析では、かなり成功すると言われているが、文脈自由文法により求められるような、もう少し深い解析をする場合に、どの程度成功するのか明らかではない。確率モデルを用いて深い解析を行うためには、言語的情報を参照することが必要だと考えられる。言語的に動機付けされた質問により、可能な途中解析結果に対して、残りの部分の解析結果の相対的可能性が計算できる。

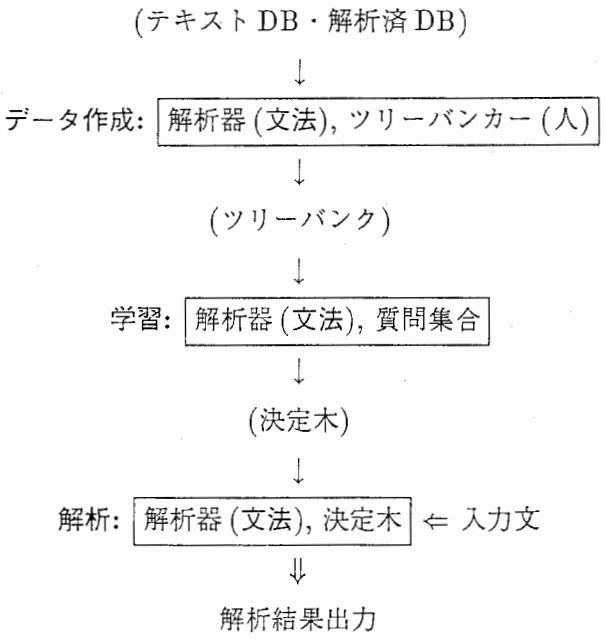
2.1 モデル

ATR 統計解析器は、確率付決定木によって動きを制御されるチャート解析器であり、同時に、決定木学習装置でもある。分析済ツリーバンクを与えると、確率付決定木を生成する。一旦、確率付決定木が生成されて組み込まれたならば、テキストを与えると、解析器として、形態素分割された品詞タグ付テキスト、または、構文木付テキストを出力する。分析済ツリーバンクは、人手を介して、半自動で作成している。[21] ツリーバンカーと呼ぶ専門の担当者が、構文解析器を道具として用い、妥当なツリーを選択して登録する。統計解析器は、こうして作成されたツリーバンクを、教師データとし

て学習し、確率付決定木を生成する。その確率付決定木により統計解析器を制御して形態素と構文の解析を行うのである。

確率付決定木は、統計的学習モデルのひとつである。問題を解決するのに、質問の集合を用いる。質問ごとに、その可能な解答の確率が、学習した事実に基づいて計算される。学習した事実を再現するために、質問の集合を、どのような順序で尋ねると最も有効か、エントロピーの減少に基づいて構成したものが確率付決定木である。この確率付決定木は、新たな問題を解決するのに用いることができる。

決定木の質問は、再帰的に構成できる述語論理型の ATR 質問言語で記述され、言語現象や、解析途中の状態に関する質問を行う。この言語を用い、使用できる述語を組み合わせて、いくつかの選択肢から、正しい答を選択するのに十分な情報を得られるような質問を用意する。どのような状況下で、どのような質問が必要な情報を得るのに最も有効であるかは、決定木学習のアルゴリズムにより決定される。この決定木生成アルゴリズムでは、質問は二値の関数でなければならないが、多値の関数は、複数の二値の関数の組み合わせに翻訳が可能なので、ATR 質問言語では多値関数を扱うことができる。



2.2　解析アルゴリズム

基礎となる解析器は、通常のチャート解析器である。チャート解析器は、文法が許容する全ての構造を、解析中の文の部分単語列によりインデックスされたチャートに格納する。[1]文法形式は、単一化文法を模した素性構造文法であるが、計算の効率化のために完全な単一化は、行わない。したがって、明示されていない素性値の解釈が、一般の

[1]実際には、全ての素性値の全ての組み合わせではなく、殆ん全てのカテゴリーに存在する 2 つの素性 (pos, barnum) の値の全ての組み合わせである。

単一化文法とは異なる。構成要素の素性値を照合しながら解析を進める文脈自由型文法の解析装置である。解析戦略は、ボトムアップ、左から右、深さ優先探索である。[2]

一般に、トップダウン解析では、再帰規則に問題があるが、チャート解析では、同じ活性弧を 2 回以上作らないので、問題はない。一般に、ボトムアップ解析では空列生成規則に問題があるが、ATR の文法では英語・日本語ともに空列生成の概念は持ち込まれていないので、問題はない。したがって、トップダウンとボトムアップ、いづれの解析戦略でも実現は可能と考えられるが、確率付決定木解析器の場合、トップダウン解析では、解析状態が、終端記号である形態素に到達するまでのエントロピーが大きくなりすぎて現実的でない。ボトムアップ解析では、決定木の質問において、はじめから終端記号に関する質問ができるので、エントロピーの減少に有利である。そのため、ボトムアップが選択された。[3]

深さ優先の探索戦略により各解析構造の派生が唯一であることを保証している。幅優先探索では、複数の途中解析状態を平行して保持しなければならない。確率モデル解析器では、各解析構造の派生は、唯一でなければならない。なぜならば、同じ解析構造が二通り以上に派生される可能性があると、一つの解析構造の確率を求めるために、全ての可能な派生の確率を計算しつくさなければならず、たとえ有限であっても、計算量は組み合わせ爆発的に増加し、確率計算が、事実上困難である。

解析器の文法は、曖昧さを孕んでいるので、多数の解析候補が生成されることがある。その場合、解析候補を生成する途中で、確率を参照し、可能性の高い解析候補だけを考慮する方法と、全ての解析候補を考慮して、確率を利用して確からしい解析結果を決定する方法が考えられる。前者の方が効率的であるかもしれないが、プログラムがきわめて複雑になるし、探索空間が網羅されない可能性があるので、研究段階の ATR 統計解析器では、後者の方針で設計された。ただし、探索空間にある全ての可能性を生成することは不可能なので、文法による構文構造のチェックに優先して、確率付決定木を参照している。これにより、確率の高い解析結果がある限り、文法に適合する解析結果がすべて生成されるわけではないが、解析結果の探索空間は網羅されていることになる。

2.3　決定木

ATR 解析器では、タグモデル、ルールモデル、方向モデルが、日本語の解析では、これらに加えてトークンモデル、の決定木モデルがそれぞれ機能する。トークンモデルは、日本語の形態素分割を予測する決定木である。タグモデルは、各形態素の品詞タグを予測する決定木、ルールモデルは、解析中、次に使用される文法規則を予測する決定

[2] 松本 BUP[13] のような左隅記号の先読みはしていない。英語版では、開始記号への到達可能行列を使用しているが、日本語版では使用していない。

[3] 文法記述者は、空列生成規則を作ることができない。

木である。それぞれのモデルは、必要に応じて、下位モデルに分けられる可能性がある。例えば、現在の我々のシステムのタグモデルでは、まず、大まかな品詞類を決定する決定木モデルと、それぞれの品詞類について、詳細な品詞タグを決定する決定木モデル群として構成している。方向モデルは、タグ付与を継続するか、ルールを適用するかの選択を予測する決定木である。この統計解析器は、構文木を左隅からボトムアップに派生するが、PCFG のように、各々の文法規則に固有の確率が付与されるのではなく、構文木派生の各段階に対して確率が付与される。確率モデルによる構文木の派生を、状態遷移と見るならば、次の形態素のタグの確率と、次に適用する文法規則の確率は、状態の確率であるのに対し、タグ付与を継続するか、ルールを適用するかの確率は、遷移の確率とみなすことができる。別の見方をするならば、タグ付与を継続するか、ルールを適用するかという予測は、構成要素の境界を予測するモデルであると考えてもよい。構成要素境界はルールモデルによっても、ある程度は予測されるが、文法規則適用は、学習データ中の文法規則適用の事実の存在によって学習されるので、「構成要素境界ではなさそうなところ」は、学習データ中に明示的には存在しない。このような、負の学習をするためには、この方向モデルがより有効であると考えられる。

ATR 統計解析器の確率モデルは、解析の各段階からの遷移の確率合計が、1 になることが保証されていない。その主な理由は、確率付決定木には、スムージング[4]をかけるため、成功しない解析段階にも 0 ではない確率が与えられる可能性があるからである。もうひとつの理由は、解析が進むと成功しないと判明するような解析段階に、一時的に入りこむことが有るので、学習データ解析時に、その確率を反映してしまうためである。各段階の遷移確率の合計を、1 に正規化することは、困難ではないが、このモデルでは、成功しない解析段階に、ごく低い確率が与えられることは、むしろ好ましい。なぜならば、正規化すれば、解析結果派生の際に、袋小路に導かれるような、ゼロに近い確率の選択肢でも、選択肢がひとつしかない場合、最大の確率が与えられ、解析の早い段階で、無駄な可能性が優先されてしまう可能性があるからである。

決定木解析器には、質問言語で書かれた質問（*.questions）の他に、質問の際に使用するボキャブラリーのリスト（*.voc）と、ボキャブラリーとビット列の対のリスト（*.bits）の三つのタイプのファイルが使用されている。これらの一部、タグ名のボキャブラリー（tag.voc）、素性名のボキャブラリー（feature.voc）、素性値名のボキャブラリー（value.voc）、ルール名のボキャブラリー（rule.voc）は、解析用の文法規則から自動生成される。質問言語中で使用されるボキャブラリーと、ボキャブラリー・ビット列対応は、このようなファイルで定義しておかなくてはならない。例えば、二値の質問には、{yes, no} だけを含む Bit.voc というファイルを用意する。質問言語では、文と文の解析途中の状態について任意の質問が可能なので、全く文法と関係のないボキャブラリーを使って質問を書くこともできる。ボキャブラリーとビット列

[4]学習した確率が、学習データをより正確に再現するように確率値を微調整する。

の対のリスト（*.bits）は、リストの中のボキャブラリーを任意のしかたで弁別することができる。

第 3 章

ATR 統計解析器文法開発環境

この章では、ATR 統計解析器の文法と、その開発環境について、概説する。

3.1 文法規則

ATR 統計解析器の文法形式は、単一化文法を模した素性構造にもとづく文法である。いわゆる単一化の操作はしないが、素性値の照合を行う拡張文脈自由文法である。一つの構成要素の左辺から一つ以上の構成要素の右辺への書き換え規則の集合である。非終端記号は、素性と値の対の集合で表現される。左辺、すなわち、親ノードは、その規則によって示された構成要素の素性値を規定する。右辺、すなわち、子ノードは、左辺の素性構造の構成要素が、いくつかの構成要素の連続に書き換えを許すとき、連続するそれぞれの構成要素の素性のそれぞれの値を規定する。

素性値の変数は、その素性値の値が一つとは限らないことを示す。一つの文法規則中で同じ変数が二つ以上あれば、代入されうる値は、互いに束縛される。その規則を適用するためには、それらの同一の変数は同一の値を持たなければならない。特定の素性の値は、代入することにより、構文構造の下から上に向かって伝播される。また、右辺の適用の制約を条件記述することができる。文法規則の一般的な形式は、次のとうりである。[1]

$$\begin{bmatrix} \text{素性}_1 = \text{値}_1 \\ \text{素性}_2 = \text{値}_2 \\ \cdots \end{bmatrix} \rightarrow \begin{bmatrix} \text{素性}_1 = \text{値}_1 \\ \text{素性}_2 = \text{値}_2 \\ \cdots \end{bmatrix} \begin{bmatrix} \text{素性}_1 = \text{値}_1 \\ \text{素性}_2 = \text{値}_2 \\ \cdots \end{bmatrix} \{\text{条件}\}$$

素性

文の構文的解釈は、文法の中で使用される素性の値により表現される。それぞれの素性値は、その素性値をもつ文、構成要素、単語などの情報の一部を表現する。素性構造

[1] 並記した素性対は、それぞれの構成要素内で連言。

は、再帰的構造体ではない。この点も、一般の単一化文法とは、形式的に異なる。構成要素は、素性名と値の対の集合であるが、値は単純記号であり、値として素性構造をとり得ない。中でも、**pos** と **barnum** の二つの素性は、解析のチャート生成や文法規則選択に特別に使用される。

現在、ATR 英語文法では 65 の素性、SLDB 体系日本語文法では 17 の素性、TDMT 体系日本語文法では 20 の素性が使用されている。例えば、現在の TDMT 体系日本語文法では、**pos** 素性の可能な値は 10 ある。それぞれの素性にはデフォルト値が設定されている。素性と素性値の対の集合は、素性の束である。素性の束は、可能な全ての素性値、を明示的あるいは暗示的に規定する。素性は、その値が明示されているときは、その値、それだけを値とする。素性の束の中で指定されていない素性値は、あらかじめ設定されたデフォルト値になる。値が変数の素性は、その素性の値の領域に含まれていれば、いづれの値でもよい。その領域は、同じ文法規則の中の別の素性対か、あるいは文法規則の条件の中で、共通の変数を与えられらた素性によってさらに制約を受ける。文法記述言語では、小文字アルファベットで始まる文字列は、定数の値と解釈され、大文字アルファベットで始まる文字列は、変数と解釈される。日本語 2 バイト文字は、定数として扱われることになっているが、動作確認が必要である。

一つの素性は、素性名、デフォルト素性値、値の領域を表す集合の三つ組により、文法ファイル (features.ascii) で定義される。

変数

素性値の変数は、文法規則の中で一般化を行うために用いる。文法規則は、文の部分記号列が、右辺すなわち子ノードの素性の束の連続に一致する場合に適用される。しかし、文脈によって、必ずしも全ての素性値が一致しなければ文法規則を適用してはならないとは限らない。ある素性だけが一致すれば、その文法規則を適用したい場合がある。そのような場合に変数を使用する。

変数の使い方は 2 種類ある。同じ一つの文法規則の中で一度だけ使用する場合と、2 ヵ所以上で使用する場合である。前者は、その素性の可能な領域内の値ならばどの値でもかまわないことを示す。したがって匿名変数である。後者は、同じ変数の使用されている値が、同一であることを示す。同じ変数の使用は、異なる別の素性の束の中にある同じ素性名の値どうしに限られているわけではない。例えば、異なる素性名の値でも、それらの領域内に共通の値があるならば、異なる素性が、同じ値を持つことを表現するために用いてもよい。この場合は、後述するように警告メッセージが出る。

変数という概念の解釈は、一般の単一化文法や、制限節文法（DCG）等とは、異なっている。一般の単一化文法では、素性値の伝播に方向性はないが、ATR 統計解析用文法では、必ずしもそうではない。ATR 統計解析用文法では、一つの規則の親ノードと別の規則の子ノードは、全ての素性値が照合したときに成功する。二つの素性値

は、次の場合に照合する。1) 両者が同じ定数項のとき、2) 一方が定数項で、もう一方がその同じ定数項が代入された変数のとき、3) 両者が、ともに同じ定数項が代入された変数のとき、4) 両者が、ともに未代入の変数であるとき、である。一つの文法規則が、ある構成要素を子ノードとして照合するとき、子ノードの中の変数の素性値は、その構成要素の定数の素性値、または既に代入された変数の素性値と照合したとき、その値を代入されるが、その逆は真ではない。文法規則が、ある構成要素を子ノードとして照合し、その規則の子ノードの中の定数の素性値と、その構成要素の未代入変数の素性値とが照合しても、その定数の値は変数に代入されない。

このように、単一化文法を模してはいるが、一般の単一化文法と異なる概念を用いているのは、処理の高速化のためである。もし、同じ文法が、通常の完全な単一化操作を伴う解釈で運用されたなら、同じ文に対して異なる解析結果の集合を返す可能性がある。これについては、後で述べる。

条件

条件は、子ノードの素性の束の中で使用された変数の値を制約する関数の形式で記述される。項として関数を取らない単純な関数は、=、==、**var** である。前二者は、左側が変数で右側が素性値の二項をとる演算子である。**var** は、変数を一項とる関数である。= は、左側の変数が、右側の値であるか、あるいは、値を持たない時、真を返す。== は、左側の変数が、右側の値である時にのみ、真を返す。したがって、変数が束縛されていない時に、両者の値が違ってくる。**var** は、その変数が、束縛されていない時に、真を返す。

項として、再帰的に条件の関数を記述して、複合的な条件記述ができるものがある。& (and)、| (or)、not である。これらは、一般のブール論理と同じ意味である。連合のスコープを表すのには、丸括弧を用いる。条件は、効率的処理のため、内部的には、有限状態オートマトンに変換される。

規則

文法規則は、左辺の一つの親ノードである素性の束から、右辺の一つ以上の子ノードである素性の束の連鎖への書き換え規則、それに子ノードの素性束の中の変数に関する条件からなる。それぞれの文法規則には、**DESCRIPTION:** と **EXASMPLE:** という注釈を書き込むことができる行区切りがある。文法をコンパイルした後は、素性のデフォルト値、文法規則中の変数、変数の数などの情報が、配列に格納されている。

品詞タグ規則は、品詞タグに素性構造の解釈を与えるための規則である。左辺の一つの親ノードである素性の束から、右辺の一つの品詞タグ記号への無条件の書き換え規則である。束縛されない変数も使用できる。

3.2 文法のコンパイル

文法は、品詞タグ規則（tags.ascii）、素性（features.ascii）、文法規則（rules.ascii）の三つのアスキーファイルに納められているが、後述する GWB ツール、または、統計解析器のプログラムが読み込む時には、バイナリ形式にコンパイルされる。そのさい、文法記述言語の形式と意味と運用上のチェックがされる。その他にも、処理の効率化の点で有用な、その文法についての情報が、解析のたびに生成する必要がないように格納される。文法がセーブされる時に、文法の 3 つの各ファイルに対応して、それぞれのボキャブラリーファイル（tag.voc, feature.voc, rule.voc）と、 全ての素性値の和集合のボキャブラリーファイル（value.voc）が、自動生成される。まず素性のボキャブラリーが読み込まれ、それから、文法規則とタグ規則が読み込まれてチェックされる。

文法の形式エラーは、チェックされ、エラーメッセージが出る。意味のチェックでは、素性の値に、その素性の値の領域にない値が代入されるとチェックされ、メッセージが出る。条件の中の変数は、有効な定数が代入されるかどうかチェックされる。運用上のチェックでは、異なる素性の値に同一の変数が割り当てられている場合 (これは有効であるが、記述者の注意を喚起するために)、警告メッセージを出す。

解析中に構成要素ができると、その構成要素の **pos** と **barnum** の 2 つの素性を鍵に、その構成要素が子ノードとして照合を試みる文法規則があるかどうかをチェックする。 **pos** と **barnum** の 2 つの素性値の全ての組み合わせについて、規則の左隅、右辺の第一番目の素性の束が、その素性値の組み合わせを許す文法規則群のテーブルを生成する。これが解析器のチャートである。

第 4 章

GWB ツール

GWB ツール (Grammarian's Work Bench Tool) は、Motif ベースの文法開発とツリーバンクのツールである。ATR の統計的自然言語処理において、GWB ツールは、重要な役割を果たす。対象言語の文法の開発には、仮定した対象言語の文法を、実際にその言語のコーパスに即して実証しなければならない。同時に、統計的言語処理のモデル化のためには、大規模な学習データを収集しなければならないが、そのツリーバンク作成のためのツールとしても GWB ツールを用いる。GWB ツールは、文法と解析器とのインターフェースである。解析中の文の木構造と素性構造を表示し、文法や解析する文の表示、検索、編集、などを行う、文法とツリーバンクを編集するための複合機能エディタである。

ここでは、GWB ツールの基本的な特徴と使い方について、簡単に述べる。GWB ツールのエディタ (Grammar editor, Rule editor, Tag editor, Featrure edior, Treebank editor) は、すべて共通して、タイトル、メニューバー、現在読み込んでいる文法名のスロット、メッセージウインドウ、それに、それぞれの状態で必要な操作ボタン、と表示用ウインドウからなる。

Treebank editor は、構文木を記述するツリーバンカーが、GWB ツールに読み込まれた文法を使用して構文解析結果を参考にし、品詞タグ付コーパスの上に文の分析結果を記述するのに用いる。Rule editor, Tag editor, Featrure edior は、文法の開発、修正を行うのに用いる。

4.1 はじめに

画面左上隅に 3 つの操作ボタンがある。右端のボタン、Window で GWB ツールの状態を選択する。これにより、ユーザは、エディタの状態を変えることができる。

Grammar Window 文法の選択をする

Rule Window 文法規則を表示、編集する (Rule editor)

Tag Window 品詞タグの定義を表示、編集する (Tag editor)

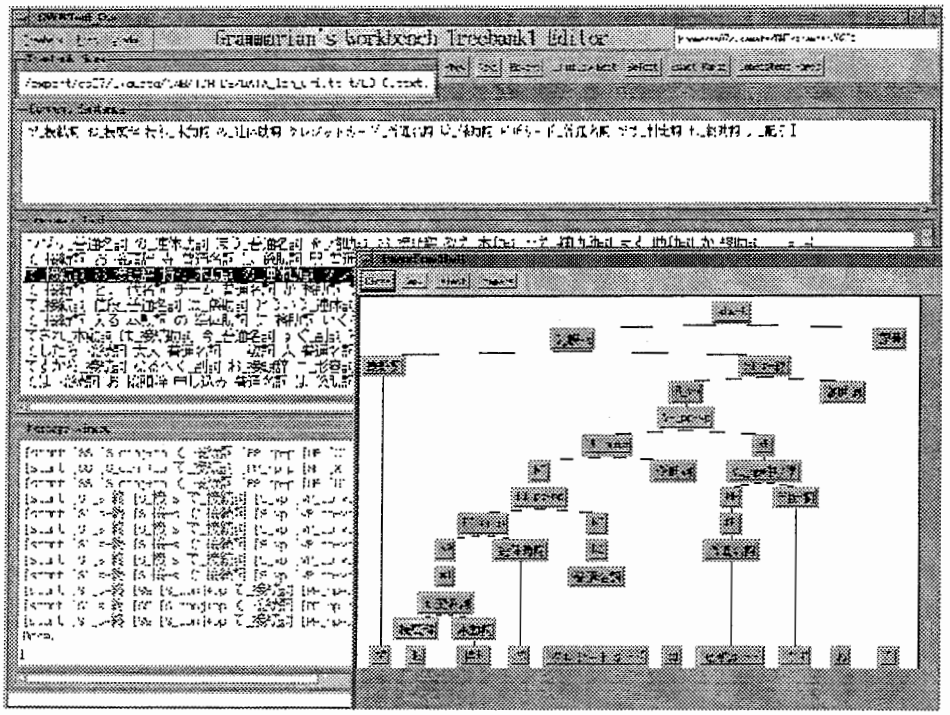


図 4.1: GWBTool の画面 (Treebank Editor)

Featrure Window 素性の定義を表示、編集する (Feature editor)

Treebank Window ツリーバンクを作成、表示、編集する (Treebank editor)

状態を選択すると、それに応じてエディタのタイトルが変化して、それぞれの画面構成が表示される。

真中のボタン、Misc は、エディタの状態にかかわらず共通である。Print Rule、Print Tag、Print Feature は、それぞれ、文法規則、品詞タグ規則、素性の定義をメッセージウインドウに出力する。Clear Message Window はメッセージウインドウの全てのテキストを消去する。

左端のボタンは、GWB ツールの状態に応じて、それぞれ Grammar, Rule, Tag, Feature, Treebank と変化する。クリックすると、それぞれの状態のときのメニューバーを表示する。

4.2 メニューバー

GWB ツールが、いづれの状態の時でも、次のようなメニューバーを表示する。状態によってそれぞれ解釈が異なるが、一般に次のような操作を行う。

New 画面をクリアして、新しいアイテムの編集を開始する。このときセーブしていない内容、変更点があれば、警告を出す。

Open 既存のアイテムをエディタ上に読み込んで開く。このときセーブしていない内容、変更点があれば、警告を出す。

Save 現在のアイテムをセーブする。このとき、そのアイテムの情報をチェックする。

Save As 現在のアイテムを別の名前でセーブする。

Print アイテムをメッセージウインドウ上に表示する。

Delete 既存のアイテムを削除する。

Quit ツールを終了する。

ここでアイテムと呼んだのは、例えば、Grammar editor の状態の時は、文法全体、Rule editor の状態の時は、文法規則、Tag editor の時は、品詞タグの定義、Feature editor の時は、素性の定義、Treebank editor の時は、ツリーバンクのファイルである。

4.3 エディタ群

Grammar editor

GWB ツールは、Grammar editor の状態で動作を開始する。全ての操作は、文法を読み込んでから可能となる。検索ウインドウとメッセージウインドウが表示される。

New 新しい文法を作る。システムは、ユーザに、これから作る新しい文法の UNIX のディレクトリ、ファイル名の入力を促す。

Open 既存の文法をエディタに読み込む。システムは、ユーザに、対話ウインドウで開くべき文法を訊いてくる。

Save 現在、読み込んでいる文法をセーブする。

Save As 現在、読み込んでいる文法を別の名前でセーブする。

Delete 指定した文法を削除する。

Quit GWB ツールを終了する。

Rule editor

文法規則の左辺と右辺の素性構造をそれぞれ表示するウインドウ、文法規則名（文法規則左辺の素性構造に対してユニークにつけられた名称）と、Description, Example の各ウインドウが表示される。これらのウインドウ上で文法の編集を行うことができる。

New 新しい文法規則を追加するため、文法規則に関する全ウインドウを空白にする。

Open 既存の文法規則の情報を開く。

Save 現在の文法規則をセーブする。編集した規則は、即座にメモリー上の文法の一部として機能する。ただし、その規則が、アスキー版の文法ファイルに書き込まれるのは、文法がセーブされる時である。

Error Check 現在の文法規則の情報をチェックする。

Print 現在の文法規則をメッセージウインドウ上に表示する。

Delete 現在の文法規則をメモリー上の文法から削除する。

Quit GWB ツールを終了する。

Tag editor

右辺を表示するウインドウが無いほかは、 Rule editor と似ている。品詞タグの素性構造を表示するウインドウ、品詞タグ名、 Description、 Example の各ウインドウが表示される。

New 新しい品詞タグを追加するため、品詞タグに関する全ウインドウを空白にする。

Open 既存の品詞タグの情報を開く。

Save 現在の品詞タグの情報をセーブする。編集した品詞タグ情報は、即座にメモリー上の文法の一部として機能する。ただし、その品詞タグ情報が、アスキー版の文法ファイルに書き込まれるのは、文法がセーブされる時である 。

Error Check 現在の品詞タグの情報をチェックする。

Print 現在の品詞タグ情報をメッセージウインドウ上に表示する。

Delete 現在の品詞タグ情報をメモリー上の文法から削除する。

Quit GWB ツールを終了する。

Feature editor

素性名、デフォルト値、値のリストを表示する各ウインドウと、メッセージウインドウが表示される。

New 新しい素性を追加するため、素性に関する全ウインドウを空白にする。

Open 既存の素性の情報を開く。

Save 現在の素性情報をセーブする。編集した素性情報は、即座にメモリー上の文法の一部として機能する。ただし、その素性の情報が、アスキー版の文法ファイルに書き込まれるのは、文法がセーブされる時である。

Print 対話ウインドウで指定した素性の情報をメッセージウインドウ上に表示する。

Delete 現在の素性をメモリー上の文法から削除する。

Quit GWB ツールを終了する。

Treebank editor

ツリーバンク名を表示するウインドウ、現在読み込んでいる文を表示する Current Sentence ウインドウ、ツリーバンクコーパス（未分析、分析済）の全文のリストを表示する Treebank Text ウインドウ、Message Window が、表示される。Current Sentence ウインドウは、Allow edit（書き込み可能）モードを選択できる。

New 新しいツリーバンクを作る。システムは、ユーザに、これから作る新しいツリーバンクの UNIX のディレクトリ、ファイル名を訊いてくる。

Open 既存のツリーバンクをエディタに読み込む。システムは、ユーザに、対話ウインドウで開くべきツリーバンクファイル名を訊いてくる。

Save 現在、読み込んでいるツリーバンクをセーブする。

Save As 現在、読み込んでいるツリーバンクを別の名前でセーブする。

Delete 指定したツリーバンクを削除する。

Quit GWB ツールを終了する。

Treebank editor の画面には、次の操作ボタンがある。

Prev 前の文を Current Sentence ウインドウに読み込む。その時、読み込まれていた文に変更があった場合は、Select される。

Next 次の文を Current Sentence ウインドウに読み込む。その時、読み込まれていた文に変更があった場合は、Select される。

Revert 解析、編集中の文を、その文を Current Sentence ウインドウに読み込んだ時の状態に戻す。

Allow Edit Current Sentence ウインドウを書き込み可能にする。トグルスイッチ。

Select 現在、Current Sentence ウインドウに表示されている状態を選択する。それ以後、Revert により戻る状態は、これで選択された状態。

Exact Parse 新たに、**Parse Tree Shell** というウインドウを開いて、Current Sentence ウインドウに読み込まれているラベル付括弧の構文木を図示する。読み込んでいる文法により、そこに記述されている木構造が受理されない場合は、エラーメッセージを出す。

Consistent Parse Current Sentence ウインドウに読み込まれている品詞タグ付テキストを、読み込んでいる文法により解析し、全ての解析結果を、**Parse Tree Shell** ウインドウ (図 4.1 右下のウインドウ) を開いて表示する。Current Sentence ウインドウに、ラベル付括弧の構文木が付与されたテキストが読み込まれている場合、読み込んでいる文法により生成される解析結果のうち、付与されている構成素構造と矛盾しない (左右の括弧が交差しない) 結果をすべて表示する。そのような構造がない場合は、エラーメッセージを出す。

Predict Parse これは、英語版 GWB ツールのみに実装されている。日本語版には、未だ実装されていない。Consistent Parse と同様のはたらきだが、統計的にいちばん尤らしい解のみを表示する。

その他

メッセージウインドウは、エディタからユーザへのメッセージを表示する。エラーメッセージは、"ERROR:"の文字列で始まり警告音を伴う。また、すべてのテキストウインドウでは、Control-s で検索用の対話ウインドウが開いて、文字列検索ができる。検索文字列を前方、または、後方に検索する機能がある。

第 5 章

解析器の日本語文法

ATR の統計的言語解析手法では、解析器に組み込まれる文法は、同時に、その解析器の統計モデルである確率付決定木の構成、成長を決定する学習データとしてのツリーバンクを作成する道具でもある。構文分析を行うツリーバンカーは、GWB ツールにこの日本語文法を読み込んで、品詞タグ付コーパスを構文解析し、文法的に可能な分析の中から、妥当な分析を選択してツリーバンクを作成する。ここでは、TDMT 品詞体系の文法について概説する。この文法は、GWB ツールを用いて、ATR 旅行会話の TDMT タグ付コーパスを構文解析するために開発した文法である。文法は 97 年 6 月から開発をはじめ、10 月からはツリーバンク作成作業と平行して、必要な変更を行ってきた。

5.1 文法規則の構成

品詞とカテゴリー

現在、解析器で使用している TDMT 体系の文法では、品詞タグは 32 種類である。複雑な言語現象を弁別する能力は、十分ではないかもしれないが、例えば、活用形情報の利用や、語の特定の (部分) 文字列の指定により、さらに細かい品詞分類を行える可能性はある。また、タスク指向の旅行会話を前提とした日本語意味分類の体系化作業が現在進行中である。これらを組み合わせることにより、ATR- ランカスター英語コーパスの約 3,000 種類に及ぶ品詞タグのように、詳細な言語情報を記述できる品詞タグ体系を構築できる可能性がある。現在の品詞体系は、TDMT 品詞体系旅行会話データベースの品詞のフィールドの記号をそのまま使用している。これは、機械翻訳システムでも使用されている品詞体系なので、基本的な日本語の文法的振舞いを弁別するには、まず妥当な数だと考えられる。[1]

品詞タグとは別に、いわゆるカテゴリーの概念が、pos 素性として想定されてい

[1] EDR コーパスの日本語品詞は 15、EDR 単語辞書の日本語品詞は 30 である。

文の類	s
動詞類	v
名詞類	n
助詞類	p
形容詞類	a
副詞類	adv
接続詞類	conj
助動詞類	aux
補文標識類	comp
その他	unspec

表 5.1: pos: 構成要素カテゴリー

る。**pos** 素性の値は、前終端記号である品詞と、全ての非終端記号を下位分類する。原則として、表 5.1のような文法カテゴリーを想定して **pos** 素性の値が決められている。

バーレベル

言語学でよく使われている、いわゆるバーレベルという概念も使用する。これは、**barnum** 素性の値として記述される。原則は、**0** が語彙レベルで、**2** が単独で生起するような句のレベル、そして、**1** がその中間、あるいは、どちらともいえない場合である。左辺と同じ素性構造が、右辺のどこかに現れるような、再帰的な規則は、原則的に 1 のレベルで記述することにしているが、必要に応じて他のレベルの再帰的規則を許している。主節にしか現れない現象、あるいは節と言えない発話の記述と、その文法的振舞いを制御するため、文の類 $(pos = s)$ については、**barnum** 素性の値は、**99** と **100** の場合がある。

素性構造

統語素性の数は、最新版（98.6.22 版）で、20 である。この統語素性の数は、必要ならば変更される可能性がある。近い将来、意味分類の体系化を品詞タグに反映すると同時に、意味素性が追加される予定である。前終端記号（品詞）をのぞく典型的な非終端記号では、デフォルト値でない値を持つ素性は、いくつかの例外を除いて、表 5.1に示す四つの素性である。それぞれの値の表す意味も、表 5.1に示す。

素性のリスト (feature.voc) と、値の領域のリスト (value.voc) は、文法をコンパイルする際に生成される。文法定義ファイルの中には、人間が読むために、**DESCRIP-**

pos	カテゴリー記号 (例、 s, n, v, p, a, adv)
barnum	語、句のバーレベル (例、 0, 1, 2)
part_of_speech	語彙レベルの場合は品詞 (例、普通名詞、本動詞)、 句のレベルの場合は文法規則名 (例、 VP_pp+vp, VP_advp+vp)
***_type**	そのカテゴリーをさらに下位分類する記号。 但し、 * の部分は、 pos 素性によって異なる。 (例、 noun_type, verb_type)

表 5.2: 標準的な素性構造

TION: と **EXAMPLE:** のフィールドがある。

以下、この章で、簡便のために、 N, V1, AP など[2]の略記を用いることがある。これは、 pos と barnum の 2 つ素性の値のみに言及する部分素性構造の略記である。前半のアルファベット文字が pos の値を表す。 barnum 素性の値が、 0 の場合、それに続く数字はない。 1 の場合は 1 が、 2 の場合は、慣例にしたがって句（フレーズ）の頭文字 P をつけた。例えば、次のような部分素性構造に言及する場合、次のような記号を使う。

$$N:\begin{bmatrix} pos=n \\ barnum=0 \\ \cdots \end{bmatrix} V1:\begin{bmatrix} pos=v \\ barnum=1 \\ \cdots \end{bmatrix} AP:\begin{bmatrix} pos=a \\ barnum=2 \\ \cdots \end{bmatrix}$$

ただし、$[pos=s, barnum=2, \ldots]$ [3]は、慣例に従って、 SP ではなく、 S と略記する。

$$S:\begin{bmatrix} pos=s \\ barnum=2 \\ \cdots \end{bmatrix}$$

また、 pos 素性にも言及しない $[\ldots, barnum=2, \ldots]$ の部分的素性構造を XP と略す場合がある。

$$XP:\begin{bmatrix} barnum=2 \\ \cdots \end{bmatrix}$$

[2]例、 S, S1, N, N1, NP, V, V1, VP, A, A1, AP, P, P1, PP, ADV, ADV1, ADVP, ...

[3]正確には、 barnum=2 以上。

文法規則の ID 名は、各規則にユニークにつけられる名称である。規則の左辺のシンボル記号として、ラベル付括弧のラベル、木構造表示の親ノードのラベルに表示される。この名前の付け方も統一している。おおむね上で説明した略記に従い、大文字でカテゴリーラベルを表わし、続けてアンダースコア、続いて構成要素の略記を小文字で表した。

(例)

VP_pp+vp : VP → PP VP

NP_pp+np : NP → PP NP

デフォルト値

単一化文法でのいわゆる "underspecify" の概念は、この文法記述法では、記法に本質的に備わってはいない。表示されている木構造の中の、あるノードに、ある素性が表示されない、あるいは、文法規則の中で、その素性が言及されていないということは、その素性が、どのような素性値も取り得るということではなく、その素性は、特別に指定されたデフォルトの定数の値を持っているということである。したがって、ある素性がないノードは、やはり、その同じ素性がないノードとしか照合できないということである。

文法を記述、あるいは読解する場合、指定されていないように見える値は、「表示しない」ことを指定された定数の値が入っていることに注意しなくてはならない。表示されていない素性の値は、たいていの場合、 "unspec" という定数の値が入っている。これはシステムが自動的にデフォルトで定める値ではなくて、素性の定義の中で、そのように定数を指定してあるだけなので、デフォルト値は、小文字で始まる文字列ならば何でもよい。ただ、他の値を代入できる変数ではないということは、他の大多数の素性ベースの単一化文法の記法と、大きく異なるので、注意が必要である。

一般の単一化記法では、$[f_1 = v_1]$ と $[f_2 = v_2]$ は、単一化し、結果は、$[f_1 = v_1, f_2 = v_2]$ になる。[4]

$$[f_1 = v_1] \sqcup [f_2 = v_2] = \begin{bmatrix} f_1 = v_1 \\ f_2 = v_2 \end{bmatrix}$$

しかし、この文法記法では、そのように表示された場合、照合できない。[5]

$$[f_1 = v_1] \sqcup\!\!/ [f_2 = v_2] = fail$$

なぜなら、$[f_1 = v_1]$ は、f_1 以外の素性値が、全てデフォルトの定数値である素性構造の簡略表現で、$[f_2 = v_2]$ は、f_2 以外の素性値が、全てデフォルトの定数値である

[4] $\sqcup$ は、単一化演算子。

[5] $\sqcup\!\!/$ は、ATR 統計解析器文法の素性照合演算子。

文の類の規則	22
動詞類の規則	25
名詞類の規則	65
副詞類の規則	11
形容詞類の規則	4
助詞句類の規則	8
語彙レベルの規則	18
その他	22

表 5.3: 文法規則数 (98.6.22)

ような素性構造の簡略表現だから、f_1 の値は、v_1 とデフォルト値$_1$、また、f_2 の値は、デフォルト値$_2$ と v_2 というように異なっているからである。

$$\begin{bmatrix} f_1 = v_1 \\ f_2 = default_2 \\ \cdots \\ f_n = default_n \end{bmatrix} \sqcup \begin{bmatrix} f_1 = default_1 \\ f_2 = v_2 \\ \cdots \\ f_n = default_n \end{bmatrix} = fail$$

5.2　文法規則

ここでは TDMT 品詞体系のツリーバンクで用いている文法規則を概観する。文法規則は、現在のところ 157 ある (表 5.2)。ここでは、規則を大まかに分類して概説する。[6]

文法規則は、「文は、述語を中心とした句である」「動詞句や形容詞句は、その主要な述語と、その左側にあるいくつかの格助詞付名詞句から成る」というような、常識的な骨組みから出発した。当然、コーパスに現れる自然言語は、そのような典型的な発話のみではありえないので、その後、ツリーバンク作成に際して、実際に日本語音声対話コーパスを解析しながら、記述的に妥当と思われる構造で受理できるように、素性構造や構成要素構造の見直しをしながら、文法を拡張していく方向で開発した。

文の類の規則

開始記号は start という名称が解析機構により定められている。音声翻訳を前提とする TDMT 品詞体系では、記号という品詞タグは、文境界に現れる文字「。」のみに付与されているので、開始記号 start は、S と記号に展開する。

[6] 個々の規則の詳細は、文法規則を定義するファイルを参照のこと

start → S　記号

S の展開規則では、動詞句、形容詞句、あるいは、感動詞のみを文とみなす規則が、典型的である。他に、名詞句のみ (「ビール。」)、副詞句のみ (「確かに。」)、その他の単語のみ (「なるほど。」「しかし。」)、等の発話を、文とみなす規則もある。これらは、会話文の文法に特徴的な規則であると考えられる。

(例)

RULE: S_vp
RULE: S_ap
RULE: S_np | pp
RULE: S_感動詞
RULE: S_感動詞 *
RULE: S_word

また、文に、感動詞、呼び掛けのための名詞句、文副詞、接続詞などの付加的な要素を付け加える規則がある。

(例)

RULE: S_感 +s
RULE: S_呼 +s
RULE: S_advp+s
RULE: S_s+adv

それに、複数の文の類をひとまとめに構成する規則がある。通常は、単独で生起しない文の類[7] が、バー 1 のレベルで記述される。

(例)

RULE: SS
RULE: S1_s+conjp

例えば、次のような、S 展開規則は、会話の文法特有の規則であると考えられる。

(例)

RULE: S_conj+pp

[7] 例えば、接続詞付の文。

談話文脈により、了解可能であるため、主語や目的語が省略されることは、よく理論研究でも指摘され議論されている。しかし、実際の会話では、述部が発話されないことも、しばしばある。

(例)

で_接続詞 ご_接頭辞 予算_普通名詞 は_係助詞 。_記号

このような場合、接続詞に助詞句が続き、文が終了してしまう。述語にあたる空所を生成することが、記述的、または、説明的に妥当かどうかは、別の問題として、この文法記法自体、空列を生成する概念が実現できないので、外心構造[8]的規則による記述を行わなうしかない。

動詞類の規則

語彙的に複合動詞とみられる動詞要素を構成するバー 0 レベルの規則がある。バー 1 のレベルでは、主に、動詞要素と助動詞要素から、動詞句を構成する規則群がある。例えば、V1* は、動詞類と助動詞類を構成要素とする再帰的規則である。バー 2 のレベルでは、動詞句の項や修飾語となる助詞句や名詞句、それに、副詞的要素と動詞句主要部とを構成要素とする規則がある。副詞類は、動詞句の類とも、文の類とも、姉妹構成要素になりうるので、文副詞や動詞句副詞といった下位分類や、前後の文脈に依存した情報があることが望ましい。

(例)

RULE: V1*
RULE: VP_pp+vp
RULE: VP_advp+vp
RULE: VP_np+vp

動詞句 (VP) と呼んでいるが、その主要な構成要素は、形容詞述語や名詞述語となっている外心構造的記述の規則がある。

(例)

RULE: VP_advp+ap
RULE: VP_np+ap
RULE: VP_pp+ap
RULE: VP_pp+np

[8] 構成要素のいづれも、全体の中心となってはいない。

名詞類の規則

名詞類の規則は、たいへん数が多い。記述文法や理論文法の文献では、人間にとって自然すぎるために見過ごされたり、理論的動機付がなくて無視されたりしがちな現象を扱う規則が、多く見られる。

名詞類には、数の表現が含まれており、それらは独特の振舞いを見せるため、名詞類の規則には、表現に特化したバーレベルの低い規則が多い。例えば、電話番号には、電話番号の文法（多くの場合、2つのグループになった7つ前後の数字の連続で、その前に市外局番や国際電話ならば国番号がついている場合がある）がある。価格の表現、日付の表現なども、多くの場合、名詞句と見られるが、文法的なきまりをもった表現である。また、使用した品詞タグ付コーパスデータでは、数字が一つずつ形態素とされている。この文法では、数字連続は、名詞類として扱われている。そして、一つの再帰規則で受理するのではなく、16 までの数字連続を受理する 16 の規則を設けた。これは、構造の無意味な階層化を避けるためと、数字がいくつ連続したかという情報を利用したいためである。16 というのは、ATR 旅行会話データベースに現れた一番長い数字連続 (クレジットカードナンバー) である。

(例)

RULE: Num4
DESCRIPTION: Num4 → 数詞 数詞 数詞 数詞

名詞句の連続は、統語素性のみからでは、構造を記述できない場合が多いにもかかわらず、解析の学習データとなるツリーバンクでは、一様ではない構造を記述したいという要請があるので、規則右辺に、さまざまな組み合わせの構造を持たせた規則がある。

(例)

RULE: NPs_np+nps
RULE: NPs_nps+np
RULE: NPs_nps+nps

規則右辺の構成要素のどれかが、構成要素全体の主要部となる内心構造[9]が原則であるが、なかには、構成要素のいづれも名詞句ではないが、全体として見ると名詞句と見なすのがよいと考えられるので、名詞句にした外心構造的記述の規則もある。

(例)

RULE: NP_pp+np EXAMPLE: 東京から大阪を下さい。

[9] いづれかの構成要素が全体の中心をなしている。

副詞類の規則

副詞類というのは、他のカテゴリーとは異なり、用法や機能的特徴のことを副詞と呼んでいる場合が多い。したがって副詞ではない品詞の語彙が、構造記述の中で副詞として分析されるようにするための書き換え規則が多い。したがって、副詞類も外心構造的規則が多い。

(例)

RULE: ADV1 形
RULE: ADV1 接続
RULE: ADV1 名

形容詞類の規則

形容詞類の規則は、副詞により修飾される形容詞句がある。連体詞は形容詞類として扱っている。

(例)

RULE: A1_advp+a1
RULE: DET

助詞句類の規則

助詞句は、助詞により文法的振舞いが決められるので、助詞句の文法上の主要部は助詞とする。典型的な助詞句は、名詞句の右側に助詞が続く構成要素であるが、左側の構成要素は、名詞句に限らず、形容詞句(「くわしく / は ...」)、副詞句(「すぐに / は ...」)、文(「どのカードで払うか / と ...」)などの場合がある。

(例)

RULE: PP_np+p
RULE: PP_ap+p
RULE: PP_advp+p
RULE: PP_s+p

その他

それ自体では、カテゴリーが不明で、文脈により動詞類、名詞類、形容詞類などが決まる形態素がある。例えば、「サ変名詞」や「サ変形容名詞」などは、右の文脈に補助動詞「する」があると、動詞類として機能するが、右の文脈が格助詞だと、名詞類として機能する。そのため、品詞タグ定義の pos 素性の値が unspec である。これらの pos 素性の値は、カテゴリーを決定する文脈と共に記述する規則において代入される。

語彙レベル、すなわちバーレベル 0 から、バーレベル 0 への書き換え規則がある。品詞タグ付コーパスでは、形態素分割されているのだが、複合名詞や複合形容詞、複合副詞などとして、一つの形態素とみることができる、あるいは、そうした方が他の文法規則との整合性がよいという場合、0 レベルの語彙規則で書換える文法規則を用いる。

(例)

RULE: 派生形容名詞 [日本 / 的]
RULE: 派生名詞 [大き / さ]
RULE: 複合形容詞 [お / 安い]
RULE: 複合代名詞 [わたくし / ども]

形容動詞という概念は、もともと品詞タグ体系には存在しないが、日本語学では標準的概念と言われるので、ツリーバンク作業の効率を維持する目的で、バーレベル 0 の語彙規則のラベルとして便宜的に取り入れた。そのカテゴリーは、連体形が形容詞類、連用形が副詞類である。

第 6 章

日本語旅行会話ツリーバンク

ATR 日本語旅行会話データベースの中から TDMT 品詞体系でタグ付されたコーパスをもとに、97 年 10 月から 98 年 7 月までの約 10ヶ月間で、約 1.8 万文のツリーバンクデータ[1]を作成した。この作業は、今後も継続し、98 年度末までに約 4 万文のツリーバンク作成を予定している。また、この他に、SLDB 品詞体系でタグ付されたコーパスをもとに、97 年 10 月までに既に作成された約 2 万文のツリーバンクもある。

カバー率

TDMT 品詞体系の文法を用いて、18,200 文のデータに対して、ツリーバンカーが妥当であると判断した構文分析を付与できたのは、17,793 文（約 97.7%）であった。分析できなかった文は、文法を改良することにより、妥当な解析ができると思われる文もあるし、著しい省略[2]や、発話途中で意図する構文構造が変化しているなど、[3]なんらかの文法のねじれが含まれており、それらの文を受理するために、無理に文法を改変しないほうがよいと判断された文もある。その他に、ツリーバンク作業時のケアレスミスなどにより、解析文法と一貫性のとれていないデータが約 0.8% みつかった。これらを総合して、現在の手法、現在の文法でも、旅行会話データベースの約 97% がツリーバンク可能と予測される。

文法の変更

ツリーバンク作業が進行する途中で、コーパスの中に、それまでの文法規則で解析できない新しい現象が現れたときは、文法を改良して対応する。途中で文法を変更すると、変更前の文法で受理されてきた構文構造が、変更後の文法では、異なる構文構造に

[1]例は、付録を参照。

[2]例、「とのことです。」、「零七五はい。」

[3]例、「日本の場合長時間の会議を禁煙にするのは難しいと聞いているんですけれども換気のことがちょっと気になってどうなんでしょうね実際のところは。」、「約歩いて 5 分です。」

なることがあり、ツリーバンクデータの一貫性に問題が生じる。そのため、決定木の学習を行うときには、統計解析器に組み込むのと同一の文法により、全てのデータが解析可能かチェックする必要がある。これは、ツリーバンク作成過程の中でも、ツリーバンクデータの蓄積が増えれば増える程、大きなコストがかかるので、文法の変更は、特に慎重に行わなければならない。

第 7 章

決定木の質問

述語論理型の ATR 質問言語により記述された質問の集合は、、決定木成長アルゴリズムにより、学習データに対してエントロピーの減少が最大になるような確率付決定木として生成される。処理対象の文について、その解析途中の状態について任意の質問ができる。ここでは、隣接する単語の共起頻度に基づいて、統計的に単語をクラス分けした単語 MI ビット [4] と、同様の手法で、隣接する文字の共起頻度に基づいて、文字を統計的に分類した文字 MI ビット [12] を採用しているので、それらに関する質問もできる。ここでは、現在、使っている主な質問事項について述べる。処理中の文字より左の要素は、形態素分割、品詞タグ付与が、時には、文法規則適用まで済んでいる場合があるので、品詞や構成要素に関する質問もできるが、処理中の文字より右の要素は、単に文字の連続であるから、文字レベルの質問しかできない。

7.1 形態素解析

形態素分割

形態素分割のための質問の集合は、トークンモデル (token.questions) である。TDMT 品詞体系の会話コーパスは、音声認識出力を想定しているので読点「、」がない。そのため、読点がある場合よりも形態素分割が難しいと考えられる。例えば、次のような質問事項を含んでいる。[1]

文字 MI ビット 処理中の文字、直前や直後の文字、2 文字前の文字などの文字 MI ビットが何であるか？

単語 MI ビット 処理中の単語、直前や直後の単語、2 単語前の単語などの単語 MI ビットが何であるか？

[1] 格助詞や接辞など、閉集合の品詞の語彙は語彙リストを用意してあるが、基本的に辞書なしで形態素解析を行う。

文字種 直前の文字と処理中の文字、処理中の文字と直後の文字などの文字種に関して、ひらがな、カタカナ、漢字のいづれであるか？

文字数 現在処理中の単語は、現在処理中の文字までで何文字か？直前の単語の文字数はいくつか？

処理中の単語 語頭に現れない文字で始まっているか？あるいは、終っているか？「日」「時」等の時間表現の単語か？単位表現の単語か？どんな機能語か？こそあど言葉か？

直前の単語 「こそあど」で始まっていたか？「感動詞」だったか？語頭に現れないない文字で終っていたか？漢字で終っていたか？

直後の文字 「日」「時」等の時間表現の文字か？単位の文字か？数字か？どんな機能語の開始文字か？語末に現れる漢字か？語頭に現れない文字か？

文字種 カタカナ、ひらがな、漢字の組み合わせはどうか？

- 直前の文字と処理中の文字について
- 処理中の文字と直後の文字について

その他の連接について 直前の単語が「う段」で終って、処理中の単語が「形式名詞」か？直前の単語が「と」または「て」で終って、処理中の文字が「思、考、言、聞」か？直前の単語がひらがな以外で終って、処理中の文字が格助詞や係助詞か？その他、「の」＋「です、ます」、国名＋「語」、「です、ます」＋終助詞、ひらがな＋「しい、しく」、カタカナ＋並立助詞、（漢字 or カタカナ or 代名詞）＋格助詞、数字＋単位、など多数。

品詞タグ付与

品詞タグモデルは、品詞タグ付与を行うための質問の集合である。現在、pos 素性値を決める決定木 (pos.questions) と、品詞タグを決める決定木 (tag.questions) の、二つを使用している。後者をさらに細かくわけて、それぞれの pos 素性値のグループに対して品詞タグを付与する決定木を、それぞれ別に作ることも可能である。トークンモデルに挙げた質問の多くは、品詞タグモデルでも使っている。

特に、品詞タグモデルで使った質問は、直前の単語の、それぞれの素性の値に関する質問である。例えば、pos 素性値は何か？ barnum 素性値は何か？ verb_type 素性値は何か？

7.2 構文解析

文法規則モデル

文法規則モデル (rule.questions) は、規則適用を決定するための質問の集合である。品詞タグ付与に比べて、構文解析の決定木モデルは、まだ、詳細な質問を用意するに至っていない。基本的な質問だけが用意されている。例えば、現在処理中の単語と、その前後の単語が、どんな機能語であるか？3語前までのそれぞれの素性の値は何か？

方向モデル

方向モデル (dir.question) は、解析器が、継続して品詞タグ付与を続けるか、あるいは、そこで文法規則を適用するかの決定を行う。用意されている質問は、例えば、3語前までのそれぞれの素性の値は何か？

第 8 章

実験、評価と考察

形態素解析（分割と品詞タグ付与）と、構文解析器の実験を行った。学習データは98.6.22 版文法により解析された、旅行会話ツリーバンクデータ約 1.8 万文である。決定木の質問集合は、試行錯誤により経験的に作られた。

8.1 形態素分割

試験文セットは、学習用ツリーバンクデータとは別の旅行会話コーパスから、会話ファイル単位で、任意に抽出した 9,773 文と 9,168 文の 2 セットを使用した。評価は、データベースの正解と比較して、一致したものを正解とした。

現在の形態素分割の精度は、表 8.1のとうりである。学習とテストに使用された TDMT 品詞体系の旅行会話コーパスは、音声認識出力を想定して、読点「、」が、全く含まれていない。そのような、べた書きの文章は、特にひらがなが連続する箇所は、人間にとっても、読みづらいもので、課せられた形態素分割の問題は、かなり難しいといえる。

トークンモデルの決定木では、深さ 5 までの 31 の質問に注目すると（表 8.2）、文字と単語の MI ビット、そして、字種の連接に関する質問が、上位に構成されている。これらの情報が、形態素分割を行う際に、かなり有効であるという事実が確認される。

統計解析器が出した誤った形態素分割の例に注目しする。完全に失敗している場合もあるが、付録の表 D.1に示すように、中には、学習データが想像され、やむを得ないと思われるような場合もある。また、正解、すなわち、作成されたもとの形態素解析データと違ってはいるが、中には、人間が見ても、システムが出した『誤答』のほうが、一見もっともらしく見える場合がある。例えば、「なければなりません」や「お待たせいたしました」などは、データベースでは一形態素となっているが、このような形態素は、全体のデータベースの形態素の体系の中でも、例外的な事実であると考えられる。このような例外を、統計的に学習するには、例外集合の規則的な共通性が、統計的に有意になるほどの膨大な学習量が必要なのではないだろうか？それは、どの程度の量なの

表 8.1: 形態素分割精度

	testset1(9773 文)		testset2(9168 文)	
学習文数	再現率	適合率	再現率	適合率
2,000	79.98	84.91	79.74	85.77
4,000	84.71	89.64	84.29	90.66
6,000	88.58	91.89	90.38	94.00
8,000	86.95	92.01	89.43	94.39
10,000	87.54	92.16	89.74	94.85
12,000	88.06	92.87	90.76	94.97
14,000	88.91	93.10	90.94	95.39
16,000	**89.18**	**93.13**	91.47	95.48
17,746	88.82	92.83	**92.34**	**95.62**

再現率 システムが出した正解形態素数 / 正解形態素数

適合率 システムが出した正解形態素数 / システム出力形態素総数

表 8.2: トークンモデル決定木

質問 \ 深さ	0	1	2	3	4	5
文字 MI	1/1	1/1	1/2		4/8	4/15
単語 MI			1/2	2/4	2/8	3/15
字種連接				2/4	2/8	4/15
その他						4/15

表 8.3: 品詞タグ付与精度

	testset1(9773 文)			testset2(9168 文)		
学習文数	再現率	適合率	タグ適合率	再現率	適合率	タグ適合率
2,000	73.88	78.44	92.37	73.39	78.93	92.03
4,000	79.86	84.51	94.27	79.38	85.38	94.18
6,000	84.22	87.37	95.08	85.89	89.33	95.03
8,000	83.23	88.07	95.72	85.79	90.55	95.93
10,000	83.63	88.04	95.53	86.34	91.26	96.22
12,000	84.39	88.99	95.83	87.45	91.50	96.35
14,000	85.59	89.63	96.27	87.90	92.21	96.67
16,000	**86.24**	**90.05**	**96.70**	**88.66**	**92.54**	**96.92**
17,746	84.51	88.33	95.15	87.82	90.93	95.10

再現率: システムの正解タグ付形態素数 / 正解形態素数

適合率: システムの正解タグ付形態素数 / システム出力形態素総数

タグ適合率: システムの正解タグ付形態素数 / システム切出し正解形態素数

か？また、その場合、過学習の効果はどうなるのか？これらについては、今後の研究課題である。

8.2 品詞タグ付与

品詞タグ付与についても、形態素分割と同様に評価した (表 8.3)。正しく形態素分割された形態素の品詞タグ付与は、かなり高い再現率で成功している。形態素分割と通して評価すると 90% 前後になる。

また、品詞タグ付与の誤りの例 (付録 D.2) に注目すると、最も多いのは、サ変名詞に普通名詞の品詞タグを付与した場合であるが、これなどは、サ変名詞とはいえ、右側に「する」の文脈で現れていなければ、用法は普通名詞となんら変わりはないのだから、普通名詞の現れる文脈に出現したサ変名詞を、データの統計的学習のみから、サ変名詞と同定するのはそうとう困難だと思われる。

第 9 章

おわりに

9.1 まとめ

ATR 統計解析器の概容と現状について述べた。統計的自然言語処理は、形態論、構文論、意味論などの同時平行処理や、言語の学習をモデル化することが可能で、ロバストで実用的可能性が大きい。ATR 統計解析器は、素性構造を使った文法を用いるチャート解析器と、確率付決定木を併せて用いる手法である。日本語処理では、形態素分割、品詞タグ付与、構文解析を行う。文法規則は、素性の束から素性の束への書き換え規則と、その条件記述からなる。素性値に変数も使用可能である。文法はコンパイル時に形式的に検査される。

文法開発と、ツリーバンク作成は、GWB ツールという多機能エディタを使用し、平行して進められた。20 の統語素性を用いた 157 の文法規則と、その文法に基づいて解析された、約 1.8 万文の TDMT 品詞体系の旅行会話ツリーバンクが作成され、約 4 万文を目標に作業が継続している。意味分類を考慮して品詞タグセットを詳細化し、意味的素性の導入を計画している。

統計解析器には、ツリーバンク作成で使用した文法の他に、形態素分割、品詞タグ付与、構文解析を正しく行うための統計情報を得るための質問の集合が備わっている。それぞれの質問集合は、学習データに則して、形態素分割、品詞タグ付与、構文解析を行う確率付決定木として構成される。学習を終えた統計解析器は、学習データの統計情報に基づいて、尤らしいやりかたで、べた書きのテキストを、形態素分割し、それぞれの形態素に品詞タグを付与し、文法に法った構文木を付与する。

ATR 旅行会話コーパスを学習して、ATR 旅行会話コーパスによりテストした場合、形態素分割の精度は、再現率 89 ～ 92%、適合率 93 ～ 96% 程度、正しい形態素分割、かつ、正しい品詞タグ付与の精度は、再現率 86 ～ 89%、適合率 90 ～ 93% 程度であった。正しく形態素分割されたものだけについて見ると、適合率は、97% 近くに及ぶ。

構文解析は、動作は確認した。決定木の質問集合が未整備でもあり、形態素分割と品

詞タグ付与における失敗も影響するので、正確な評価は未だできていない。試験入力文の 5 割強に構文解析結果が得られ、構文解析結果が得られたものについては、8 割以上が、第一候補として正解を出す。構成要素境界だけ[1]に注目するならば、9 割以上の正解率がある。

9.2 今後の課題

この手法の長所は、はじめに述べたように、異なるレベルの情報を組み合わせて使うことができ、経験的事実に基づいたロバストな処理が可能であるという点である。

短所は、決定木を用いた手法一般に言えることだが、決定木の構成は、決定木学習アルゴリズムと学習データに依存するので、処理を制御しにくい。従って、改良のための正確な指針を得ることが難しい点である。例えば、ある学習データにより、形態素分割と品詞タグ付与のための決定木を学習し、その決定木を用いた解析器により試験解析を行って、失敗した試験データを分析して改良するには、決定木の質問集合を変更するか、特定の学習データを学習することが考えられる。いづれの方法でも、決定木の構成は、変化するが、どこが、どう変化するかを、予測するのは難しい。失敗した事例を、正しく判断するのに役に立ちそうだと思われるための質問を追加しても、その質問が、決定木の中でどのような位置付で使用されるか、そもそも使用されるかどうかは、システムのアルゴリズムによる。

このような不確定さを避け、制御の自由を得るために、決定木による判断の範囲を制限して、それぞれの問題に特化した決定木モデルを作り込んでいくという可能性がある。ここで述べた品詞タグ付与のシステムでも、品詞の pos 素性を決定する決定木と、品詞タグを決定する決定木は別であったし、まだ、活用していないが、各 pos 素性の形態素ごとに、品詞タグを決定する決定木の質問集合を別々に用意することができるしくみになっている。しかし、これは、問題を分割することができるだけで、上に述べた問題の解決に向かってはいない。だからといって、決定木による問題解決の範囲を、容易に制御可能な範囲にとどめるならば、記号と規則だけによる手法と比較して、どれだけメリットがあるだろうか。文法と統計の折衷案では、その両手法の長所と短所のトレードオフを考慮して、最良の妥協点のどうやって見つけるかが、研究課題になるだろう。

現在、残っている課題として、構文解析器の評価があるが、それ以外にも、次のような課題がある。

詳細な統語分類

現在、決定木で付与している品詞タグ集合は、粗い 32 品詞である。旅行会話データベースには、活用形、活用行、などの情報もある。また、単語の (部分) 文字列から、

[1] ラベル付括弧のラベルの差は無視し、括弧の交差しなければ正解と判定する。

有効だと思われる情報を、品詞タグに持たせることもできる。それらの情報を、学習データに含めて、統語的により詳細なタグ付与が可能である。詳細な品詞タグ情報があれば、解析済の部分から得られる情報が豊富になるので、形態素分割、品詞タグ付与、構文解析、すべてにおいて、解析精度の向上が期待できる。

意味分類

現在、旅行会話データベースの語彙について、意味分類の作業が進行中である。この意味分類も、品詞タグの体系に取込んで、さらに詳細な品詞タグ付与を行う。統語的に詳細な情報と同様に、意味情報の追加も解析精度の向上に貢献すると期待できる。

決定木の構成

例えば、pos 素性のグループごとに、品詞タグ付与の決定木を作るなど、決定木の構成を検討する余地が残っている。

評価と改良の指針

構文木付文、すなわち、ラベル付括弧を施したテキストの評価手法を確率する。現在は、経験的試行錯誤しかしていない。どのような確率決定木による解析から、どのような誤りが出力されたのか、その原因が説明でき、個々の誤りを一般的に捉えてシステムの改良にフィードバックするためには、何が必要なのか、を研究する余地がある。

統計解析器の利用

解析器の文法は、独自に開発したので、TDMT 翻訳システムの構文解析器との互換性は、明らかではないが、翻訳システムの構文解析の弱点を補完するような利用方法を現在検討中である。

ツリーバンク作成に使用するエディタ、GWBTool の英語版では、統計解析器を利用し始めているが、日本語版 GWBTool でも、統計解析器を組み込むことによって、今後のツリーバンク作業効率が向上する可能性がある。

参考文献

[1] Black, E., H. Kashioka, S. Eubank, R. Garside, G. Leech and D. Magerman. (1996) "Beyond Skeleton Parsing: Producing a Comprehensive Large-Scale General-English Treebank With Full Grammatical Analysis", Proceedings of COLING-96, pp. 107-112.

[2] Black, E., S. Eubank, H. Kashioka. (1997) "The Non-Dictionay: Description and Evaluation of a Dictionaryless Semantic and Syntactic Tagger for Unrestricted English Text" 言語処理学会第3回年次大会発表論文集 pp. 309-312.

[3] Black, E., A. Finch, H. Kashioka. (1998) "The Trigger-pair Predictors in Parsing and Tagging" Coling-ACL pp. 131-137.

[4] Brown, P., V. Della Pietra, P. de Souza, J. Lai, and R. Mercer (1992) "Class-based n-gram models of natural language," *Computational Linguistics*, Vol. 18, No. 4, pp. 467–479.

[5] Breiman L., J. Friedman, R. Olshen, and C. Stone (1984) "*Classification and Regression Trees*", Wadsworth & Brooks/Cole, Monterey, CA.

[6] EDR (1996) *EDR Electronic Dictionary Version 1.5 Technical Guide.* EDR TR2-007.

[7] Magerman, D. M. (1994) "Natural Language Parsing As Statistical Pattern Recognition" *PH.D. Thesis, Stanford University*

[8] Magerman, D. M. (1995) "Statistical Decision-Tree Models for Parsing" Proceedings, 33rd Annual Meeting of ACL, pp. 276-283

[9] Marcus, M., B. Santorini, M. A. Marcinkiewicz (1993) "Building a large annotated corpus of English: the Penn Treebank." *Computational Linguistics, Vol 19.*

[10] Marcus, M., G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. (1994) "The Penn Treebank: Annotating

Predicate Argument Structure", in *Proceedings of the Human Language Technology Workshop*, Morgan Kaufmann Publishers Inc., San Francisco, CA.

[11] Miyoshi, H., K. Sugiyama, M. Kobayashi, and T. Ogino. (1996) "An Overview of the EDR Electronic Dictionary and the Current Status of Its Utilization" *Proceedings of COLING-96.*

[12] Kashioka, H., Y. Kawata, Y. Kinjo, A. Finch, and E. Black (1998) "Use of Mutual Information Based Character Clusters in Dictionary-less Morphological Analysis of Japanese", (to appear?) *Coling-ACL 1998.*

[13] Matsumoto, Y., H. Tanaka, H. Hirakawa, H. Miyoshi, and H. Yasukawa (1983) "BUP: a bottom-up parser embedded in Prolog.", New Generation Computing 1: pp. 145-158.

[14] M. Nagata (1994) "A Stochastic Japanese Morphological Analyzer Using a Forward–DP Backward–A* N–Best Search Algorithm", COLING–94, pp.201–207.

[15] D. Paul (1990) "Algorithms for an optimal a^* search and linearizing the search in the stack decoder", *Proceedings of the June 1990 DARPA Speech and Natural Language Work shop.*

[16] Quinlan, J. R. (1993) "C4.5: Programs for Machine Learning", *Morgan Kaufmann.*

[17] A. Ushioda (1996) "Heirarchical Clustering of Words and Application to NLP Tasks", Fourth Workshop on Very Large Corpora, pp.28–41,

[18] M. Yamamoto (1996) "A Re–estimation Method for Stochastic Language Modeling from Ambiguous Observations", Fourth Workshop on Very Large Corpora, pp.155–167,

[19] 柏岡秀紀, S. Eubank, E. Black. (1996) "The ATR/Lancaster General – American – English Treebank", 言語処理学会第 2 回年次大会発表論文集 1996, pp. 269-272.

[20] 柏岡秀紀, 河田康裕, 金城由美子, A. Finch、E. Black (1998) 『確率付決定木を用いた日本語構文解析』言語処理学会第 4 回年次大会発表論文集 pp. 213-216.

[21] 河田康裕, 金城由美子, 柏岡秀紀 (1998) 『日本語会話文の構文木付コーパス作成』言語処理学会第 4 回年次大会発表論文集 pp. 622-625.

[22] 黒橋禎夫, 長尾眞 (1997) 『京都大学テキストコーパス・プロジェクト』 言語処理学会第 3 回年次大会発表論文集 pp. 115-118.

[23] 中渡瀬 (1996) “正規化頻度による形態素境界の推定”, 情報処理学会, 96–NL–113, pp.13–18,

[24] 渕 武志, 松岡 浩司, 高木 伸一郎 (1997) “保守性を考慮した日本語形態素解析システム”, 情報処理学会, 97–NL–117, pp.59–66,

付録 A

ツリーバンクのデータの例

日本語ツリーバンクデータのディレクトリは、
/dept3/work22/kashioka/skel/JTREE/{SLDB,TDMT}/

[start [S_ 感+s はい _ 感動詞 [S_vp [VP [V1* [V1_np+ 判 | 準 [NP [N1 ペニンシュラニューヨーク _ 普通名詞 N1] NP] で _ 判定詞 V1_np+ 判 | 準] ございます _ 補助動詞 V1*] VP] S_vp] S_ 感+s] 。 _ 記号 start]

[start [S1_s+conjp [S_vp [VP [V1_np+ 判 | 準 [NP [N1_xp+ 準体 [S_vp [VP_pp+vp [PP_np+p [RelNP [S_vp [VP_pp+vp [PP_np+p [NP [N1 そちら _ 代名詞 N1] NP] で _ 格助詞 PP_np+p] [VP [V1* [V1 やっ _ 本動詞 V1] ている _ 助動詞 V1*] VP] VP_pp+vp] S_vp] [NP [N1_ 連体 pp+np(s) [PP_np+p [NP [N1 週末 _ 普通名詞 N1] NP] の _ 連体助詞 PP_np+p] [NP [N1 [複合名詞 2 宿泊 _ サ変名詞 割引 _ サ変名詞 複合名詞 2] N1] NP] N1_ 連体 pp+np(s)] NP] RelNP] を _ 格助詞 PP_np+p] [VP [V1* [V1* [X1 利用 _ サ変名詞 X1] し _ 補助動詞 V1*] たい _ 助動詞 V1*] VP] VP_pp+vp] S_vp] ん _ 準体助詞 N1_xp+ 準体] NP] です _ 判定詞 V1_np+ 判 | 準] VP] S_vp] が _ 接続助詞 S1_s+conjp] 。 _ 記号 start]

[start [S1_s+conjp [S_vp [VP [V1_np+ 判 | 準 [NP [N1_xp+ 準体 [S_vp [VP_pp+vp [PP_np+p [NP [N1 数 _num+n1 単位 [Num1 二 _ 数詞 Num1] [N1 単位 人 _ 普通名詞 N1 単位] N1 数 _num+n1 単位] NP] で _ 格助詞 PP_np+p] [VP_pp+vp [PP_np+p [NP_np(s)+ 副 [NP [N1 ディナー _ 普通名詞 N1] NP] 込み _ 普通名詞 NP_np(s)+ 副] で _ 格助詞 PP_np+p] [VP_pp+vp [PP_np+p [NP [N1 数 _n1 + n1 [N1 数 _num+n1 単位 [Num1 一 _ 数詞 Num1] [N1 単位 泊 _ 普通名詞
N1 単位] N1 数 _num+n1 単位] [N1 数 _num+n1 単位 [Num2 三 _ 数詞 百 _ 数詞 Num2] [N1 単位 ドル _ 普通名詞 N1 単位] N1 数 _num+n1 単位] N1 数 _n1 + n1] NP] で _ 格助詞 PP_np+p] [VP [V1* [V1* [X1 [サ変名詞 お _ 接頭辞 願い _ 本動詞 サ変名詞] X1] し _ 補助動詞 V1*] たい _ 助動詞 V1*] VP] VP_pp+vp] VP_pp+vp] VP_pp+vp] S_vp] ん _ 準体助詞 N1_xp+ 準体] NP] です _ 判定詞 V1_np+ 判 | 準] VP] S_vp] けれども _ 接続助詞 S1_s+conjp] 。 _ 記号 start]

[start [S_ 感動詞 はい _ 感動詞 S_ 感動詞] 。 _ 記号 start]

[start [S_vp [VP_pp+vp [PP_np+p [NP [N1 [複合名詞 3 週末 _ 普通名詞

特別 _ 形容名詞 割引 _ サ変名詞 複合名詞 3] N1] NP] を _ 格助詞 PP_np+p]
[VP [V1* [V1* [V1* [X1_pref+x ご _ 接頭辞 用意 _ サ変名詞 X1_pref+x]
し _ 補助動詞 V1*] ており _ 助動詞 V1*] ます _ 助動詞 V1*] VP] VP_pp+vp]
S_vp] 。 _ 記号 start]

[start [SS [S_vp [VP_pp+vp [PP_np+p [NP [X1 割引 _ サ変名詞 X1] NP]
は _ 係助詞 PP_np+p] [VP [V1_np+ 判 | 準 [NP [N1_ 連体 pp+np(s) [PP_np+p [NP
[N1 土曜日 _ 普通名詞 N1] NP] の _ 連体助詞 PP_np+p] [NP [X1 宿泊 _ サ変名詞
X1] NP] N1_ 連体 pp+np(s)] NP] で _ 判定詞 V1_np+ 判 | 準] VP] VP_pp+vp]
S_vp] [S_vp [VP_pp+vp [PP_np+p [NP_np(s)+ 副 [NP [N1_ 連体 pp+np(s)
[PP_np+p [NP [N1 ツイン _ 普通名詞 N1] NP] の _ 連体助詞 PP_np+p] [NP
[N1_prefix+n お _ 接頭辞 部屋 _ 普通名詞 N1_prefix+n] NP] N1_ 連体 pp+np(s)]
NP] のみ _ 副助詞 NP_np(s)+ 副] と _ 格助詞 PP_np+p] [VP [V1* [V1* [V1
なっ _ 本動詞 V1] ており _ 助動詞 V1*] ます _ 助動詞 V1*] VP] VP_pp+vp]
S_vp] SS] 。 _ 記号 start]

[start [S_vp [VP_pp+vp [PP_nps+p [NP_np(s)+ 副 [NPs_pp+np [PP_np(s)+ 並
[NP [N1 ディナー _ 普通名詞 N1] NP] と _ 並立助詞 PP_np(s)+ 並] [NP [N1_ 連
体 pp+np(s) [PP_np+p [NP [N1 翌朝 _ 普通名詞 N1] NP] の _ 連体助詞 PP_np+p]
[NP [N1 ブランチ _ 普通名詞 N1] NP] N1_ 連体 pp+np(s)] NP] NPs_pp+np] 込み
_ 普通名詞 NP_np(s)+ 副] で _ 格助詞 PP_nps+p] [VP [V1* [V1_np+ 判 | 準 [NP
[N1 数 _num+n1 単位 [Num4 二 _ 数詞 百 _ 数詞 九 _ 数詞 十 _ 数詞 Num4] [N1 単位
ドル _ 普通名詞 N1 単位] N1 数 _num+n1 単位] NP] で _ 判定詞 V1_np+ 判 | 準]
ございます _ 補助動詞 V1*] VP] VP_pp+vp] S_vp] 。 _ 記号 start]

[start [S_vp [VP [V1* [V1* [V1 分かり _ 本動詞 V1] まし _ 助動詞 V1*]
た _ 助動詞 V1*] VP] S_vp] 。 _ 記号 start]

[start [S_conj+s それでは _ 接続詞 [S1_s+conjp [S_vp [VP [V1_np+ 判 | 準
[NP [N1_xp+ 準体 [S_vp [VP_pp+vp [PP_np+p [NP_n1+ 曜 [N1 数 _n1 ＋ n1 [N1
数
_num+n1 単位 [Num1 十 _ 数詞 Num1] [N1 単位 月 _ 普通名詞 N1 単位]
N1 数 _num+n1 単位] [N1 数 _num+n1 単位 [Num3 二 _ 数詞 十 _ 数詞 九 _ 数詞 Num3]
[N1 単位 日 _ 普通名詞 N1 単位] N1 数 _num+n1 単位] N1 数 _n1 ＋ n1] 土曜日 _ 普
通
名詞 NP_n1+ 曜] に _ 格助詞 PP_np+p] [VP_pp+vp [PP_np+p [NP
[N1 数 _num+n1 単位 [Num1 一 _ 数詞 Num1] [N1 単位 泊 _ 普通名詞 N1 単位]
N1 数 _num+n1 単位] NP] で _ 格助詞 PP_np+p] [VP [V1* [V1* [X1 [サ変名詞
お _ 接頭辞 願い _ 本動詞 サ変名詞] X1] し _ 補助動詞 V1*] たい _ 助動詞 V1*]
VP] VP_pp+vp] VP_pp+vp] S_vp] の _ 準体助詞 N1_xp+ 準体] NP] です _ 判定詞
V1_np+ 判 | 準] VP] S_vp] が _ 接続助詞 S1_s+conjp] S_conj+s] 。 _ 記号
start]

[start [S_ 感動詞 かしこまりました _ 感動詞 S_ 感動詞] 。 _ 記号 start]

...

付録 B

日本語文法規則の例

```
RULE: start
DESCRIPTION: start → S* 記号
EXAMPLE:
PARENT:
        pos=s
        barnum=100
CHILD1:
        pos=s
        barnum=B*
        part_of_speech=POS*
        noun_type=N*
        coord_list=V*
        verb_type=V*
        mood=V*
        pp_type=P*
        p_case=V*
        adj_type=A*
        conj_type=CONJ
        compl_type=COMP
CHILD2:
        pos=unspec
        barnum=0
        part_of_speech=記号
END_OF_RULE(start)

RULE: S_vp
DESCRIPTION: S → VP  (VPは文)
EXAMPLE:
PARENT:
        pos=s
        barnum=2
        part_of_speech=s_vp
        verb_type=VT
        left_concatenation=LC
        right_concatenation=RC
CHILD1:
        pos=v
        barnum=2
        part_of_speech=POS
        verb_type=VT
        left_concatenation=LC
        right_concatenation=RC
CONDITION:
        not(((POS=vp)&(VT=感動詞))|(POS=vp_感+vp))
END_OF_RULE(S_vp)

RULE: S_word
DESCRIPTION:  S_word → 1語発話
EXAMPLE:
PARENT:
        pos=s
```

```
        barnum=99
        part_of_speech=s_word
CHILD1:
        pos=V*
        barnum=0
        part_of_speech=POS
        noun_type=V*
        verb_type=V*
        adj_type=V*
        adv_type=V*
        pp_type=V*
        compl_type=V*
        conj_type=V*
        affix_type=V*
        gyou=V*
        katuyou=V*
        katuyoukei=V*
CONDITION:
        (POS=副詞)|(POS=接続詞)|(POS=接続副詞)
END_OF_RULE(S_word)

RULE: VP_pp+vp
DESCRIPTION: VP_pp+vp → PP VP    ＮＰ{が|に|を|...}ＶＰ
EXAMPLE: [[[子供]と]は]ね]/行かないよ
         [フィリップス様の/お越し_本動詞]
PARENT:
        pos=v
        barnum=2
        part_of_speech=vp_pp+vp
        verb_type=V_TYPE
        left_concatenation=LC
        right_concatenation=RC
CHILD1:
        pos=p
        barnum=2
        part_of_speech=POS
        pp_type=P_TYPE
        p_case=V*
        noun_type=V*
coord_list=V*
        conj_type=V*
        compl_type=V*
        left_concatenation=LC
CONDITION:
        not((P_TYPE=並立)|(P_TYPE=連体))
CHILD2:
        pos=v
        barnum=2
        part_of_speech=POS*
        noun_type=V*
        verb_type=V_TYPE
        adj_type=V*
        adv_type=V*
        pp_type=P_TYPE2
        compl_type=V*
        conj_type=V*
        affix_type=V*
        gyou=V*
        katuyou=V*
        katuyoukei=V*
        right_concatenation=RC
CONDITION:
        not(P_TYPE2=連体)&(not(V_TYPE=unspec))
END_OF_RULE(VP_pp+vp)

RULE: VP_np+vp
DESCRIPTION: VP_np+vp → NP VP
EXAMPLE:
 [[np それ][vp 見れる/ん/です]
```

```
PARENT:
        pos=v
        barnum=2
        part_of_speech=vp_np+vp
        verb_type=V2
        right_concatenation=RC
CHILD1:
        pos=n
        barnum=2
        part_of_speech=V*
        noun_type=N1
        coord_list=V*
CHILD2:
        pos=v
        barnum=2
        part_of_speech=V*
        verb_type=V2
        compl_type=V*
        conj_type=V*
        affix_type=V*
        gyou=V*
        katuyou=V*
        katuyoukei=V*
        right_concatenation=RC
CONDITION:
        not((N1=人称)&(V2=感動詞))
END_OF_RULE(VP_np+vp)

RULE: NP
DESCRIPTION: NP → N1
EXAMPLE:
[np [x1 クルージング_サ変名詞]]
[np [n1 [ap あきらめる_本動詞 しか_副助詞 ない_形容詞 ] の_準体助詞 ]]
[np 二_数詞 二_数詞 三_数詞 一_数詞 二_数詞 三_数詞 四_数詞] 。_記号
[np もの_形式名詞] は考えよう
PARENT:
        pos=n
        barnum=2
        part_of_speech=np
        noun_type=N
CHILD1:
        pos=V*
        barnum=1
        part_of_speech=POS
        noun_type=N
coord_list=unspec
        verb_type=V*
        adj_type=A
        pp_type=V*
        adv_type=V*
        katuyou=V*
        katuyoukei=K
CONDITION:
     not((N=unspec)|
         (N=単位)|
         (POS=x1_advp+x1))
END_OF_RULE(NP)

RULE: N1_連体 pp+np(s)
DESCRIPTION: n1_連体 pp+np|nps → PP(連体)  NP(s)     (PP 付の NP 名詞句)
EXAMPLE:
[食事無しで一泊一万円]
[ペニンシュラという / ホテル]  [微子の / 部屋]
[エフ四十七の / 座席] [おみやげ用の / 表の / こと]
[np お客様の [nps 御名前と連絡先]]
PARENT:
        pos=n
        barnum=1
        part_of_speech=n1_連体 pp+np
```

```
        noun_type=N
        coord_list=CL
CHILD1:
        pos=p
        barnum=2
        part_of_speech=V*
        pp_type= 連体
        p_case=V*
        noun_type=V*
        coord_list=V*
CHILD2:
        pos=n
        barnum=2
        part_of_speech=POS
        noun_type=N
        coord_list=CL
        adj_type=V*
        pp_type=V*
        katuyou=V*
        katuyoukei=V*
END_OF_RULE(N1_ 連体 pp+np(s))

RULE: AP
DESCRIPTION: AP → A1_ *
EXAMPLE:
 [a1 [advp 非常に][a1 大事な]]
PARENT:
        pos=a
        barnum=2
        part_of_speech=ap
        adj_type=A
CHILD1:
        pos=V*
        barnum=1
        part_of_speech=V*
        noun_type=V*
        verb_type=V*
        adj_type=A
        katuyou=V*
        katuyoukei=V*
CONDITION:
        not(A=unspec)
END_OF_RULE(AP)

RULE: ADVP
DESCRIPTION: ADVP → (ADV1 | V1 | X1 形名)
副詞句 ADVP とみなせる ADV1、V1 および X1 形名
EXAMPLE:
[いろいろ][結構][たいへん]、、、
PARENT:
        pos=adv
        barnum=2
        part_of_speech=advp
        adv_type=ADV_TYPE
CHILD1:
        pos=CAT
        barnum=1
        part_of_speech=POS
        noun_type=N_TYPE
        verb_type=V_TYPE
        adj_type=A_TYPE
        pp_type=P_TYPE*
        adv_type=ADV_TYPE
        katuyou=V*
        katuyoukei=V*
CONDITION:
        (CAT=adv)|((CAT=v)&(not(V_TYPE= 判定詞))&(not(V_TYPE= 感動詞)))
END_OF_RULE(ADVP)
```

```
RULE: ADV1*
DESCRIPTION: ADV1* → ADVP  ADV1*  (右再帰)
EXAMPLE:
[adv1 [advp [もうすこし]][adv1 早く]]
PARENT:
        pos=adv
        barnum=1
        part_of_speech=adv1*
        adv_type=ADV
CHILD1:
        pos=adv
        barnum=2
        part_of_speech=V*
        adv_type=V*
CHILD2:
        pos=adv
        barnum=1
        part_of_speech=V*
        adv_type=ADV
CONDITION:
        not(ADV=名)
END_OF_RULE(ADV1*)

RULE: ADV1 名
DESCRIPTION: ADV1 名 → NP  (副詞とみなせる名詞句)
EXAMPLE:
[np きょう][np あす] いきます。
[np いくつ][np あとひとつ] たべますか。
[np いつか] いきましょう。
[np だれ _ 代名詞  か _ 副助詞 ]
[nps 明日  九月十七日夜七時]
PARENT:
        pos=adv
        barnum=1
        part_of_speech=adv1 名
        adv_type= 名
CHILD1:
        pos=n
        barnum=2
        part_of_speech=POS
        noun_type=N
        coord_list=V*
CONDITION:
        not(POS=np_np+ 副)&
        (N= 代名詞)|(N= 数量名詞句)|((N= 普通名詞)&((POS=np)|(POS=relnp)|(POS=np_advp+np)))
END_OF_RULE(ADV1 名)


...
```

付録 C

形態素分割質問の質問の例

```
MIWbits_of_CurrW MIBits(RelWord(PickNode(0,0),0))
MIWbits_of_PrevW MIBits(RelWord(PickNode(0,0),-1))
MIWbits_of2PrevW MIBits(RelWord(PickNode(0,0),-2))

MICWletter-1-1 Bits("MIletter.bits", WordChar(RelWord(PickNode(0,0),-1),-1,"MIlette
MICWletter_0_0 Bits("MIletter.bits", WordChar(RelWord(PickNode(0,0),0),0,"MIletter.
MICWletter_0-2 Bits("MIletter.bits", WordChar(RelWord(PickNode(0,0),0),-2,"MIletter
MICWletter_0-1 Bits("MIletter.bits", WordChar(RelWord(PickNode(0,0),0),-1,"MIletter
MICWletter+1_0 Bits("MIletter.bits", WordChar(RelWord(PickNode(0,0),1),0,"MIletter.

Connect_kata-1-1_kata-00 Bit(And(Member(WordSet("katakana.voc"),
WordChar(RelWord(PickNode(0,0),-1),-1)),
Member(WordSet("katakana.voc"),
WordChar(RelWord(PickNode(0,0),0),0))))
Connect_kata-1-1_hira-00 Bit(And(Member(WordSet("katakana.voc"),
WordChar(RelWord(PickNode(0,0),-1),-1)),
Member(WordSet("hiragana.voc"),
WordChar(RelWord(PickNode(0,0),0),0))))
Connect_kata-1-1_notkana-00 Bit(And(Member(WordSet("katakana.voc"),
WordChar(RelWord(PickNode(0,0),-1),-1)),
Not(Member(WordSet("kana.voc"),
WordChar(RelWord(PickNode(0,0),0),0)))))
Connect_hira-1-1_kata-00 Bit(And(Member(WordSet("hiragana.voc"),
WordChar(RelWord(PickNode(0,0),-1),-1)),
Member(WordSet("katakana.voc"),
WordChar(RelWord(PickNode(0,0),0),0))))

...
```

付録 D

形態素分割・品詞タグ付与にみられるデータベースと異なる出力の例

表 D.1: データベースと異なる出力例 (形態素分割)

データベース	システム出力
運転手 /	運転 / 手 /
なければなりません /	なけれ / ば / なり / ません /
コース / 料理 /	コース料理 /
精進 / 料理 /	精進料理 /
豊国神社 /	豊国 / 神社 /
ニューヨーク近代美術館 /	ニューヨーク / 近代美術館 /
乗り捨て /	乗り / 捨て /
お待たせいたしました /	お / 待た / せ / いたし / まし / た /
連絡 / 申し上げ / ます /	連絡申し上げます /
ツアーデスク /	ツアー / デスク /
自動販売機 /	自動 / 販売 / 機 /
返品 / 可能 /	返品可能 /
アジア航空 /	アジア / 航空 /
コンピューター上 /	コンピューター / 上 /
レタス / クルトン / パルメザンチーズ /	レタスクルトンパルメザンチーズ /
お / 気を付け /	お気 / を / 付け /
念のために /	念のため / に /
トヨタ車 /	トヨタ / 車 /
華厳の滝 /	華厳 / の / 滝 /
四つ角 /	四つ / 角 /
ローラースケート場 /	ローラースケート / 場 /
両国国技館 /	両国 / 国技館 /
一 / 品 / 料理 /	一品料理 /
メトロポリタン美術館 / へ /	メトロポリタン / 美術 / 館へ /
たった今 /	たった / 今 /
河原町通り /	河原町 / 通り /
マシュー / シモンズ /	マシューシモンズ /
アメリカ航空 /	アメリカ / 航空 /
予約センター /	予約 / センター /
ヒルトンホテル /	ヒルトン / ホテル /
京福電鉄 /	京福 / 電鉄 /
講演会 /	講演 / 会 /
専門店 /	専門 / 店 /
ホテル東京イン /	ホテル / 東京イン /
シーエービー /	シー / エー / ビー /
プールサイド /	プール / サイド /
...	...

表 D.2: データベースと異なる出力例 (品詞タグ)

	testset1 (9773 文)			testset2 (9168 文)		
	頻度	正解	出力	頻度	正解	出力
1	25	サ変名詞	普通名詞	45	サ変名詞	普通名詞
2	20	本動詞	普通名詞	14	形容詞	本動詞
3	14	普通名詞	本動詞	11	補助動詞	本動詞
4	13	助動詞	本動詞	11	普通名詞	本動詞
5	13	形容詞	本動詞	10	形容名詞	普通名詞
6	12	本動詞	形容詞	10	格助詞	連体助詞
7	10	普通名詞	サ変名詞	10	格助詞	接続助詞
8	9	接続詞	感動詞	9	本動詞	普通名詞
9	9	形容名詞	副詞	9	接頭辞	普通名詞
10	8	助動詞	判定詞	7	本動詞	助動詞
11	8	格助詞	接続助詞	7	普通名詞	サ変名詞
12	7	本動詞	助動詞	7	助動詞	形容詞
13	7	補助動詞	本動詞	7	格助詞	並立助詞
14	7	副詞	普通名詞	5	判定詞	格助詞
15	7	普通名詞	形容詞	5	助動詞	補助動詞
16	7	形容名詞	普通名詞	4	本動詞	形容詞
17	6	判定詞	格助詞	4	補助動詞	助動詞
18	6	接続詞	普通名詞	3	副詞	普通名詞
19	5	本動詞	判定詞	3	助動詞	本動詞
20	5	並立助詞	格助詞	3	助動詞	判定詞
21	5	準体助詞	連体助詞	3	準体助動詞	判定詞
22	5	感動詞	本動詞	3	形容名詞	本動詞
23	5	格助詞	判定詞	3	形容詞	普通名詞
24	4	普通名詞	形容名詞	2	本動詞	補助動詞
25	4	準体助動詞	判定詞	2	本動詞	判定詞
26	4	形容名詞	本動詞	2	普通名詞	連体詞
27	3	本動詞	補助動詞	2	判定詞	本動詞
28	3	接続助詞	格助詞	2	接続助詞	助動詞
29	3	接続詞	判定詞	2	接続助詞	係助詞
30	3	助動詞	補助動詞	2	接続助詞	格助詞
	...	...	...	...	...	...