日本語形態素解析システム
『茶筌』version 2.0
使用説明書

松本裕治　北内啓　山下達雄　平野善隆

平成 11 年 4 月

JUMAN	
Version 0.6	17 February 1992
Version 0.8	14 April 1992
Version 1.0	25 February 1993
Version 2.0	11 July 1994
ChaSen	
Version 1.0	19 February 1997
Version 1.5	7 July 1997
Version 1.51	29 July 1997
Version 2.0b6	20 April 1999
ChaSen for Windows95	
Version 1.0	29 March 1997

目 次

はじめに

計算機による日本語の解析において，欧米の言語の解析と比べてまず問題になるのに次の 2 点があります．一つは形態素解析の問題です．ワードプロセッサの普及などによって日本語の入力には大きな問題がなくなりましたが，計算機による日本語解析では，まず入力文内の個々の形態素を認識する必要があります．これには実用に耐えられるだけの大きな辞書も必要であり，これを如何に整備するかという問題も同時に存在します．もう一つの問題として，日本語には広く認められ同意を得られた文法，ないし，文法用語がないという現実です．学校文法の単語分類および文法用語は一般には広く知られていますが，研究者の間ではあまり評判がよくありませんし，計算機向きではありません．

日本語の解析に真っ先に必要な形態素解析システムは，多くの研究グループによって既に開発され技術的な問題が洗い出されているにも係わらず，共通のツールとして世の中に流布しているものはありません．計算機可読な日本語辞書についても同様です．

本システムは，計算機による日本語の解析の研究を目指す多くの研究者に共通に使える形態素解析ツールを提供するために開発されました．その際，上の二つ目の問題を考慮し，使用者によって文法の定義，単語間の接続関係の定義などを容易に変更できるように配慮しました．

大学で小人数で開発したシステムであり，色々な点で不完全な部分があると思います．可能な限り順次改良を重ねる予定です．皆様の寛容な利用をお願いいたします．

本茶筌システムの原形は，京都大学長尾研究室および奈良先端科学技術大学院大学松本研究室において開発された日本語形態素解析システム JUMAN(version2.0) です．JUMAN は，京都大学および奈良先端科学技術大学院大学のスタッフおよび多くの学生の協力を得て作成したものです．また，辞書に関しては，Wnn かな漢字変換システムの辞書，および，ICOT から公開された日本語辞書を利用し，独自に修正を加えました．JUMAN 2.0 をともに開発した京都大学の黒橋禎夫さん，現在キャノン勤務の妙木裕さんには特に感謝いたします．

JUMAN 開発のきっかけを作って下さった京都大学長尾真先生に感謝します．JUMAN 開発に関して様々な形で協力していただいた奈良先端大宇津呂武仁氏に感謝します．奈良先端大の知念賢一氏には，茶筌システムの開発に関して多くの助言をいただきました．奈良先端大在学時の今一修氏、今村友明氏には茶筌 1.0 の開発の際に種々の助力をいただきました。両氏および茶筌の開発に協力いただいた松本研究室のメンバーに深く感謝します。奈良先端大の鹿野清宏教授を代表とする「日本語ディクテーション基本ソフトウェアの開発」グループの方々には、IPA 品詞体系辞書の大幅な整備を行っていただきました。特に、御尽力いただいた電子技術総合研究所の伊藤氏、ASTEM の山田篤氏に感謝いたします。また，一人一人の名を挙げることはできませんが，JUMAN システムおよび茶筌システムに対して多くのコメントと質問をいただいた利用者の方々に感謝します．

平成 11 年 4 月 20 日

本システムに関するお問い合わせは以下にお願いします．

〒 630-0101
奈良県生駒市高山町 8916-5
奈良先端科学技術大学院大学
情報科学研究科 松本研究室
茶筌管理開発担当者集団
Tel: (0743)72-5240, Fax: (0743)72-5249
E-mail: chasen@cl.aist-nara.ac.jp

また以下の URL にて最新情報を提供しています．

URL: http://cl.aist-nara.ac.jp/lab/nlt/chasen.html

1 茶筌の使用法

1.1 インストール手順

1. 環境に応じて `Makefile` の `BINDIR`, `LIBDIR`, `CC`, `CFLAGS` などの項目を書き換えた後 “`make`” を実行する．これによって各プログラムがコンパイルされる．辞書を作成するためのプログラムは `mkchadic/` 以下に，茶筌本体の実行ファイルは `chasen/chasen` に作成される．
2. “`make dic`” を実行する．これによってシステム辞書が作成される．システム辞書作成に要する時間は，辞書の大きさとマシンの性能にもよるが，現在の辞書では，SPARCstation20 上で 2～3 分である．現在の標準辞書により最終的に生成されるファイルとその容量は以下の通り．

`dic/chadic.int`	約 8.6MB
`dic/chadic.pat`	約 1.3MB
`dic/chadic.ary`	約 0.9MB

3. “`make install`” を実行する．各ファイルは，デフォルトでは以下の場所にインストールされる．

`/usr/local/bin/chasen`	茶筌の実行ファイル
`/usr/local/lib/chasen/chasenrc`	chasenrc ファイル
`/usr/local/lib/chasen/mkchadic/`	辞書を作成するためのプログラム群
`/usr/local/lib/chasen/dic/`	文法・辞書ファイル

   ただし，茶筌ライブラリ `chasen/libchasen.a` および Emacs 用インタフェース `chasen/chasen.el` はインストールされない．
4. ユーザ専用の `chasenrc` ファイルを利用するには，`chasenrc` をユーザのホームディレクトリに `.chasenrc` という名前でコピーする．`.chasenrc` 内の「文法ファイル」に文法辞書が存在するディレクトリ，「PATDIC」に辞書ファイルから拡張子を取り除いたものを指定しておく．

1.2 実行方法

システムの実行ファイルは，`chasen/chasen` に作成され，`/usr/local/bin/chasen` などにインストールされる．

- 形態素解析の実行
  茶筌は，以下のように `chasen` コマンドを実行することにより起動される．

  ```
  % chasen [オプション] [ファイル名...]
  ```

  標準入力，または引数で指定されたファイルから一行ごとに文を読み込んで形態素解析処理を行なう．文字コードとしては日本語 EUC あるいは JIS(ISO-2022-JP) を受け付ける．
- 処理内容
  コスト最小 (それぞれの形態素の区切りで最小コストとの差が許容されるコスト幅以内) の解を求め，結果をオプションに従って表示する．出力時の文字コードは日本語 EUC である．各オプションの意味は次節にまとめる．

使用例

入力ファイルを引数として指定できる．以下に使用例を示す．

```
% cat temp
私は昨日学校へ行った．
% chasen temp
私      {ワタクシ/ワタシ}  私      名詞-代名詞-一般
は      ワ                 は      助詞-係助詞
昨日    キノー             昨日    名詞-副詞可能
学校    ガッコー           学校    名詞-一般
へ      エ                 へ      助詞-格助詞-一般
行っ    {イ/ユ}ッ          行く    動詞-自立        五段・カ行促音便  連用タ接続
た      タ                 た      助動詞           特殊・タ          基本形
．      ．                 ．      記号-句点
EOS
```

1.3 実行時のオプション

形態素解析の実行については，いくつかのオプションが用意されている．以下にそれをまとめる．-r など引数をともなうオプションでは，オプションと引数の間には空白があってもなくてもかまわない．

- 茶筌の起動についてのオプション

-s	茶筌サーバの起動
-P port	茶筌サーバのポート番号の指定 (-s オプションとともに使用，デフォルト値は 31000)
-D host[:port]	茶筌サーバに接続
-R	chasenrc ファイルを読み込まない (-D オプションとともに使用)
-a	環境変数 CHASENSERVER があってもスタンドアロンで実行

- 解が曖昧性を含む場合の表示方法 (曖昧性がない場合はどの方法も同じ表示となる)

-b	後方最長一致の解を一つだけ表示する (デフォルト)
-m	曖昧性のある部分だけ，複数の形態素を表示する
-p	曖昧性の組合せを展開し，すべての解を個別に表示する

- 各形態素の表示方法

-f	カラムを整えて表示 (デフォルト)
-e	完全な形態素情報を文字で表示
-c	完全な形態素情報をコードで表示
-d	各形態素を Prolog の複合項で表現し，それらをリストにしたものを出力
-v	美茶のための詳細表示
-F format	形態素を format で指定された形式で出力
-Fh	-F オプションの出力フォーマットのヘルプを表示

- その他

`-j`	句点あるいは空行を文の区切りとして解析
`-w width`	コスト幅を指定
`-C`	コマンドインタプリタを使用
`-r rc_file`	`rc_file` を `chasenrc` ファイルとして使用
`-L lang`	言語を指定
`-h`	ヘルプメッセージを出力
`-V`	茶筌のバージョンを出力

-j オプションについて

茶筌では通常，改行をもって一つの入力文字列の終了とする．そのため，文の途中で改行が挿入されているファイルを解析した場合，正しい結果が得られなくなることが多い．

そのようなときは -j オプションをつけると，句点など (デフォルトでは「 。!? 」の 4 文字) あるいは空行を文の区切りとして解析を行うようになる．

また，茶筌 1.5 以降では chasenrc ファイルの「区切り文字」の項目を指定することにより，-j オプションをつけた時の文の区切り文字を設定することができる．

1.4 茶筌サーバとクライアントの使用法

茶筌 1.5 以降では，クライアントから茶筌サーバに接続して形態素解析を行うことができる．茶筌サーバを利用するには，まず，以下のように入力してサーバを起動する．

```
% chasen -s
```

次に -D オプションをつけて茶筌を実行すると，クライアントが起動しサーバに接続して形態素解析を行う．ここで，server は茶筌サーバを立ち上げたマシンのホスト名である．

```
% chasen -Dserver filename
```

茶筌サーバでは，デフォルトでポート番号 31000 を使う．このポート番号を変更するには，-P オプションをつけて茶筌サーバを起動する．

```
% chasen -s -P31234
```

また，クライアントの起動時には -D オプションでサーバ名の後ろに ‘:’(コロン) をつけ，ポート番号を指定すればよい．

```
% chasen -Dserver:31234 filename
```

また，以下のように環境変数 CHASENSERVER を設定することにより，-D オプションをつけなくても茶筌のクライアントが起動されるようになる．

```
% setenv CHASENSERVER server:31234
```

逆に，環境変数 CHASENSERVER を設定している場合でもスタンドアロンで茶筌を実行したいときは，-a オプションをつけて起動すればよい．

茶筌サーバを起動すると，スタンドアロンの時と同じように chasenrc ファイルを読み込み，辞書や文法ファイルが読み込まれる．クライアント側ではサーバが読み込んだ辞書と文法ファイルが用いられる．つまり，クライアントの起動時には chasenrc ファイルの「文法ファイル」「PATDIC」の項目は無視され，それ以外の

項目だけが有効となる．もし，辞書や文法ファイルだけでなく他の項目についてもサーバと同じ設定でよければ，クライアントの起動時に -R オプションをつけることで chasenrc ファイルを読み込まずに解析を行わせることができる．

1.5 出力フォーマット

-F オプションや，chasenrc ファイルの「出力フォーマット」で出力フォーマットを指定することにより，解析結果の出力形式を変えることができる．

出力フォーマットの文字列の末尾に '\n' があれば，各形態素情報の表示ごとに改行を行い，文末の次に 'EOS' の 1 行を出力する．末尾に '\n' がなければ，1 文中の形態素情報を 1 行で出力し，行末に改行を表示する．

出力フォーマットの使用例をいくつかあげる．

- デフォルト (-f オプション) と同様の出力 (v-gram 版の場合)
  `"%m\t%y\t%M\t%U(%P-)\t%T␣\t%F␣\n"` または `"-f"`
- 見出し，読み，品詞をタブで区切って表示 (v-gram 版の場合)
  `"%m\t%y\t%P-\n"`
- 分かち書き
  `"%m␣"`
- 漢字かな変換
  `"%y"`
- ルビつき表示
  `"%r␣()"`

出力フォーマットの変換文字の一覧を以下に示す．

変換文字	機能
%m	見出し (出現形)
%M	見出し (基本形)
%y	読み (出現形)
%Y	読み (基本形)
%rABC	ルビつきの見出し (”A 見出し B 読み C”と表示)(※1)
%i	意味情報
%Ic	意味情報 (空文字列か”NIL”なら文字 c)(※1)
%Pc	各階層の品詞を文字 c で区切った文字列 (v-gram 版のみ)
%Pnc	1 ~ n(n:1 ~ 9) 階層目までの品詞を文字 c で区切った文字列 (v-gram 版のみ)
%h	品詞の番号
%H	品詞文字列
%Hn	n(n:1 ~ 9) 階層目の品詞 (なければ最も深い階層)(v-gram 版のみ)
%b	品詞細分類の番号 (v-gram 版の場合は 0)
%BB	品詞細分類 (なければ品詞)
%Bc	品詞細分類 (なければ文字 c)(※1)
%t	活用型の番号
%Tc	活用型 (なければ文字 c)(※1)
%f	活用形の番号
%Fc	活用形 (なければ文字 c)(※1)
%c	形態素のコスト
%S	解析文全体
%pi	パスの番号
%ps	パスの形態素の開始位置
%pe	パスの形態素の終了位置+1
%pc	パスのコスト
%ppiC	前に接続するパスの番号を文字 C で区切り列挙
%ppcC	前に接続するパスのコストを文字 C で区切り列挙
%?B/STR1/STR2/	品詞細分類があれば STR1，なければ STR2(※2)
%?I/STR1/STR2/	意味情報が NIL でも””でもなければ STR1，そうでなければ STR2(※2)
%?T/STR1/STR2/	活用があれば STR1，なければ STR2(※2)
%?F/STR1/STR2/	%?T/STR1/STR2/ と同じ
%?U/STR1/STR2/	未知語なら STR1，そうでなければ STR2(※2)
%U/STR/	未知語なら”未知語”，そうでなければ STR(%?U/未知語/STR/と同じ)(※2)
%%	% そのもの
.	フィールド幅を指定
-	フィールド幅を指定
1-9	フィールド幅を指定
\n	改行文字
\t	タブ
\\	\ そのもの
\'	' そのもの
\"	" そのもの

※1 A,B,C,c が空白文字の時は何も表示しない

※2 ‘/’ には任意の文字が使える．また，括弧 “(){}[]<>” を用いた以下のような形式が使える．

%?B(STR1)(STR2) %?B{STR1}/STR2/ %?U[STR]

1.6 コマンドインタプリタ

-C オプションにより茶筌実行時に以下のコマンドを対話的に与えることが出来る．

コマンド	
#V	-V オプションと同じ．
#F	-F オプションと同じ．ただし，フォーマット文字列をクオートする必要はない．
#w␣[数字]	コスト幅の変更．例： #w␣500
#i	様々な情報の出力．
#e␣[単語]	辞書に単語が入っているか調べる．例： #e␣茶筌
#q	茶筌の終了．
#a	辞書への単語の追加．
#f	単語を追加する辞書の指定．
#s	単語追加後のパトリシア木のセーブ．

注意

- #a, #f, #s は，茶筌の解析結果を視覚化するプログラム「美茶」用のコマンドなので，無闇に実行してはならない．
- コマンドと引数の間は必ずスペース一文字でなければならない．

2 chasenrc ファイル

chasenrc ファイルは形態素解析プログラムに必要な様々な選択肢を定義するために用いられる．これらの定義は通常，利用者のホームディレクトリにある‘.chasenrc’というファイルに記述される．起動時オプションなどによって chasenrc ファイルを指定することもできる．具体的には次のような優先順位で chasenrc ファイルが読み込まれる．

1. 起動時に -r オプションで指定されたファイル．
2. 環境変数 CHASENRC で指定されたファイル．
3. 利用者のホームディレクトリにある .chasenrc．
4. 茶筌インストール時にインストールされた chasenrc ファイル．通常は /usr/local/lib/chasen/chasenrc．

設定項目一覧を以下に示す．このうち「文法ファイル」「PATDIC」または「SUFDIC」「未知語品詞」「品詞コスト」は必ず指定されていなければならない．

1. 文法ファイルのディレクトリ
   文法ファイル (grammar.cha, ctypes.cha, cforms.cha, connect.cha.c) が存在するディレクトリを指定する．

   ```
   (文法ファイル /usr/local/lib/chasen/dic)
   ```

2. システム辞書
   システム辞書 (chadic.int) とインデックスファイル (chadic.pat または chadic.ary) を，ファイル名から末尾の拡張子を除いたものを記述することによって指定する．複数組みの辞書を指定することもでき

る．例えば以下のように指定する．

```
(PATDIC /usr/local/lib/chasen/dic/chadic
        /home/mydir/chasen/dic/mydic)
```

この例では，二組みの辞書 (chadic.int, chadic.pat および mydic.int, mydic.pat) を指定している．辞書引きに際しては，これらの辞書の両方が用いられる．[1] ．
辞書引きに SUFARY [2] を使う場合は「SUFDIC」を指定する．

```
(SUFDIC /usr/local/lib/chasen/dic/chadic)
```

SUFDIC は PATDIC に比べ，最初にインデックスファイルを読み込む時間は短いが，検索速度自体は遅いという特徴がある．解析時間を短くするには，解析文の量が少ないときは SUFDIC，多いときは PATDIC を使うとよい．
使用する辞書の最大数は，PATDIC, SUFDIC とも 5 個に設定されている．これを変更したい場合は，chasen/pat.h の MAX_DIC_NUMBER の値を変更してコンパイルしなおせばよい．

3. 未知語の品詞
未知語が発見された時に，その語をどのような品詞として連接規則を適用するかを指示する．複数の品詞を指定した時は，それぞれの品詞について連接規則が適用される．

```
(未知語品詞 (名詞 サ変接続))                ; 1 個の品詞を指定
(未知語品詞 (名詞 サ変接続) (名詞 一般))   ; 複数の品詞を指定
```

4. 品詞のコスト
形態素解析プログラムでは，解析結果の優先情報をコストとして計算している．解析に曖昧性がある場合は，コストの総計が低いものを優先することになっている「品詞コスト」では，各品詞のコストの倍率と，「未知語」についてのコストを定義する．コストは正整数の値をとる．

```
(品詞コスト
        ((*)              1)
        ((未知語)        500)
        ((名詞)           2)
        ((名詞 固有名詞)  3)
)
```

同じ品詞に対してコストの定義が複数回指定されている場合は，後のものが優先される．上の例では「名詞」の形態素のコストは 2 倍になるが，「
tt 名詞-固有名詞」以下に細分類される名詞の形態素のコストは 3 倍になる．また，先頭の ‘(*)’ の指定により，ここで明示的に定義されていない形態素のコストはすべて 1 倍 (そのままのコスト値) となる．未知語の形態素のコスト値はすべて 500 になる．

5. 連接コストと形態素コストの相対的な重みの定義
形態素解析におけるコストの計算は形態素のコストと連接のコストの総計として計算される．これら二種類のコストに異なる重みを掛けたい場合には，それを指定することができる．解析結果のコストはそれぞれのコストにここで指定された重みを乗じた値の総計として計算される．

[1] 一組の辞書には同一の形態素の登録は行なわれないが，複数の辞書に同じ形態素が登録されている場合はあり得る．このような場合は，同じ形態素が複数得られることになる．
[2] SUFARY は文字列検索パッケージである．詳しくは http://cl.aist-nara.ac.jp/lab/nlt/ss/ を参照．

```
(連接コスト重み 1)      ; デフォルト値
(形態素コスト重み 1)    ; デフォルト値
```

また，形態素解析の過程において，常にコストが最低の結果を出すのではなく，ある程度のコスト幅を許容したい場合がある．この許容幅を指定することができる．

```
(コスト幅 0)    ; デフォルト値
```

6. 未定義連接コストの定義
   連接規則ファイルに連接規則が定義されていない形態素間の連接コストを指定する．未定義連接コストを設定しないか，あるいは 0 を指定すると，連接規則が定義されていない形態素どうしは決して連接しないという意味になる．

```
(未定義連接コスト 500)
```

7. 出力フォーマット
   出力フォーマットを指定することにより，解析結果の出力形式を変えることができる．

```
(出力フォーマット "%m\t%y\t%P-\n")
```

   詳しくは 1.5 節を参照のこと．
8. **BOS** 文字列
   解析結果の文頭に表示する文字列を指定する．“%S” を使うと解析文全体を表示できる．デフォルトは空文字列 (つまり何も表示しない)．

```
(BOS 文字列 "解析文: [%S]\n")
```

9. **EOS** 文字列
   解析結果の文末に表示する文字列を指定する．“%S” を使うと解析文全体を表示できる．デフォルトは “EOS n”．

```
(EOS 文字列 "文末\n")
```

10. 連結品詞
   ある品詞の形態素が連続して出現したときに，一つの形態素として連結して出力させるときに使用する．例えば

```
(連結品詞 (名詞 数) (記号))
```

   と定義すると，連続した「名詞-数」と連続した「記号」がそれぞれ一つの形態素として出力されるようになる．ただし，「名詞-数」と「特殊」が連続している場合は連結されず，別々の形態素として出力される．
11. 区切り文字
   -j オプションをつけた時の文の区切り文字を並べ，一つの文字列にしたものを指定する (1.3 節参照)．区切り文字には全角文字，半角文字の両方を使用することができる．例えば

```
(区切り文字 "。、，!? .,!? ")
```

   と定義すると，全角文字の「。、，!?」のいずれか，または半角文字の “.,!? ” (空白文字が入っていることに注意) のいずれかの文字が文の区切りとなる．

3 茶筌ライブラリ

茶筌ライブラリ lib/libchasen.a を利用することで，茶筌のモジュールを他のプログラムに組み込むことができる．利用できるライブラリ関数には以下のものがある．

int chasen_getopt_argv(char **argv, FILE *fp)

extern int Cha_optind;

茶筌にオプションを渡す．もし茶筌の初期化が行われていなければ，初期化を行ってからオプションの設定を行う．

茶筌ライブラリではスタンドアロンによる解析のみができる．-s,-D などサーバやクライアントに関するオプションは利用できない．

argv にはコマンドラインオプションとして NULL で終わる文字列の配列を指定する．ただし argv[0] はプログラムのファイル名である．オプション指定に誤りがあった場合，ファイル・ポインタ fp にエラーメッセージを出力する．ただし fp が NULL の時は何も出力しない．

オプション指定に誤りがなければ 0 を，誤りがあれば 1 を返す．

外部変数 Cha_optind には処理したオプション (argv[0] を含む) の数が格納される．

以下に使用例を示す．chawan というプログラムにおいて，“-r /home/rikyu/chasenrc.proj -j” というオプションを茶筌に渡している．この関数の実行後 Cha_optind には 4 が代入される．

```
char *option[] = {"chawan", "-r", "/home/chamise/chasenrc.proj", "-j", NULL};
chasen_getopt_argv(option, stderr);
```

int chasen_fparse(FILE *fp_in, *fp_out)

int chasen_sparse(char *str_in, FILE *fp_out)

char *chasen_fparse_tostr(FILE *fp_in)

char *chasen_sparse_tostr(char *str_in)

スタンドアロンでの形態素解析を行う．もし茶筌の初期化が行われていなければ，初期化を行ってから形態素解析を行う．入力と出力がファイルであるか文字列であるかによって，4 つの関数がある．

chasen_fparse(), chasen_fparse_tostr() はファイル・ポインタ fp_in から読み込んだ文字列を解析する．文字コードとしては日本語 EUC あるいは JIS(ISO-2022-JP) を受け付ける．chasen_getopt_argv() で -j オプションを指定したときは，句点などを文の区切りとして解析を行う．

chasen_sparse(), chasen_sparse_tostr() は文字列 str_in を解析する．文字コードとしては日本語 EUC あるいは JIS(ISO-2022-JP) を受け付ける．

chasen_fparse(), chasen_sparse() は解析結果をファイル・ポインタ fp_out に出力する．コマンドモードで ‘#q’ を実行して茶筌を終了したときは 1 を，それ以外のときは 0 を返す．

chasen_fparse_tostr(), chasen_sparse_tostr() は解析結果を茶筌内部で確保したメモリ領域に格納し，そのポインタを返す．この領域は，次に chasen_fparse_tostr(), chasen_sparse_tostr() を呼び出すまで有効である．コマンドモードで ‘#q’ を実行して茶筌を終了したときは NULL を返す．

4 他のシステムからの利用

4.1 Prolog からの使用

- 基本的構成

茶筌システムが形態素解析を行なった結果を，SICStus Prolog がパイプを介して受けとるという構成になっている．

- 動作環境
  茶筌が仮定する Prolog 処理系は，SICStus Prolog Release 3 である．日本語を用いるためには，環境変数 `SP_CTYPE` の値を `euc` に設定する必要がある．具体的には，例えば次の一行を利用者の`.login` ファイルなどに含めればよい．

```
setenv SP_CTYPE euc
```

- 起動方法
  Prolog を起動し，'`prolog/chasen_user.pl`' を consult もしくは compile する．
  '`prolog/chasen_user.pl`' はユーザ設定用ファイルなので，ユーザが各自のディレクトリにコピーして各自の設定を行なうことを前提としている．
  解析は，以下の要領でファイル単位で行なうことができる．

```
| ?- cha.}
Input file name?  入力ファイル名.}
Output file name? 出力ファイル名.}
```

入力ファイル名，出力ファイル名を "user" とすると，端末からの入出力が可能である．

```
| ?- chatty.
```

とすれば，標準入力から入力し，標準出力へ結果を出力することができる．

- 処理内容
  Prolog 側からの一回の呼び出しによって，茶筌システムとのパイプがオープンされる．呼び出しが終了するとパイプが閉じられる．Prolog 側は，茶筌システムが形態素解析を行なった結果を受けとるだけである．
  一つの形態素について，Prolog 側が受けとるデータは，

  > morph([識別子 (ID), 開始位置 (From), 終了位置 (To), コスト (Cost), 見出し語 (Md), 読み (Ym), 基本形 (Kh), 品詞名 (Hn0), 品詞細分類 (Hn), 活用型名 (KT), 活用形名 (KF), 意味情報 (Imi), 形態素コスト (MrphCost), 前節形態素との接続コストのリスト (PreCCL), 前接形態素の識別子のリスト (PreIDL)])

  という複合項である．また，茶筌システムからデータを受けとる処理は，'`prolog/chasen.pl`' 中の `cha/3` において，`read`(+*Str*,-*MorphList*) によって上述の複合項のリスト *MorphList* を読み込むという形で行なわれている．
  Prolog 版の出力オプションは，

  > `e`： 完全な形態素情報を文字で表示 (デフォルト)
  > `f`： カラムを整えて表示
  > `s`： 見出しと品詞名を表示

  であり，述語 `cha_print_form/1` により変更することができる．

4.2 Emacs からの使用

茶筌クライアントの Emacs Lisp 版インタフェース `chasen.el` を使うことにより，Emacs 上で形態素解析を行うことができる．`chasen.el` を利用するには `chasen/chasen.el` をインストール (例えば `/usr/local/lib/mule/site-lisp/` にコピー) し，`.emacs` 中で茶筌サーバのホスト名とポート番号の設定，autoload の指定を行う．

```
(setq chasen-server-host "kyusu")
(setq chasen-server-port 31234)    ; デフォルトは 31000

(autoload 'chasen-region "chasen" "ChaSen client" t)
(autoload 'chasen-line "chasen" "ChaSen client" t)
(autoload 'chasen-highlight-class-region "chasen" "ChaSen client" t)
(autoload 'chasen-property-class-region "chasen" "ChaSen client" t)
```

それぞれの関数の詳細については chasen.el の先頭のコメント部分を参照のこと．

参考文献

[1] 益岡隆志，田窪行則：『基礎日本語文法 –改訂版–』くろしお出版，1992.

[2] 妙木裕，松本裕治，長尾眞：「汎用日本語辞書および形態素解析システム」情報処理学会第 42 回全国大会予稿集，1991.

[3] 松本裕治，黒橋禎夫，宇津呂武仁，妙木裕，長尾真：日本語形態素解析システム JUMAN 使用説明書 version 2.0，NAIST Technical Report, NAIST-IS-TR94025，1994.

[4] 北内 啓，山下 達雄，松本 裕治：「日本語形態素解析システムへの可変長連接規則の実装」, 言語処理学会第三回年次大会論文集, pp.437-440, 1997.

[5] テキストデータベース報告書 平成 8 年度, テキストグループ, データベースワークショップ, 技術研究組合 新情報処理開発機構, 1997.

付録

A 著作権および使用条件について

茶筌システムは，広く自然言語処理研究に資するため無償のソフトウェアとして開発されたものである。茶筌の著作権は，奈良先端科学技術大学院大学情報科学研究科自然言語処理学講座 (松本研究室) が保持する。本ソフトウェアの使用，改変，再配布については，特に制限を課すことはしないが，再配布については，次の事項を条件とする。

- 本ソフトウェアがその原形あるいは修正された形で再配布される場合は，再配布されるソフトウェアに茶筌 2.0 が使用されていることを明記すること。
- 再配布されるソフトウェアに，著作権に関する本節の記述と使用説明書の表紙裏のページの著作権に関する但し書きを必ず含むこと。

なお，本ソフトウェアの著作権者である奈良先端科学技術大学院大学は，原形あるいは改変された形で配布された本ソフトウェアに関連して生じる一切の損失に対して保証の責を負わないこととする。

B JUMAN 2.0 から 茶筌 2.0 への拡張点

B.1 bi-gram 版と v-gram 版の相違点

茶筌 2.0 では品詞体系や連接規則の機能などを拡張した。この機能拡張版を v-gram 版，従来のバージョンを bi-gram 版と呼ぶ。v-gram 版は bi-gram 版と文法ファイルの形式が異なっているため，辞書に互換性がない。

茶筌 2.0 ではデフォルトで v-gram 版がコンパイルされる。“`make bigram`” を実行すれば bi-gram 版の実行ファイルを作成され，bi-gram 版用の辞書を利用することができる。

v-gram 版は bi-gram 版と比べ，以下のような拡張機能や変更点がある。

1. 品詞を 2 階層から多階層に拡張した。
2. 連接規則を bi-gram の固定長から variable-gram(可変長) に拡張した。すなわち、連接する 2 つの単語 (あるいは品詞) の連接コストだけではなく、任意の長さの単語 (品詞) 列に対して単語 (品詞) の連接コストを記述することができるようになった。
3. 文節区切りの機能を持つ，長さ 0 の品詞が使える。品詞定義ファイルで品詞名の後ろに ‘/’ をつけると文節区切りとして機能する。
4. 辞書ファイル `*.dic` で単語のコスト値が省略されている場合，bi-gram 版ではコスト値が 10 となるのに対し，v-gram 版では「デフォルト品詞コスト」で指定されたコスト値 (指定されていない場合は 65535) が用いられる。
5. bi-gram 版では形態素コストと連接コストを内部で 10 倍しているが，v-gram 版ではそのままの値を使っている。
6. `*.dic` で「base」という属性を使える。見出し語の基本形などを表示する際，活用を持っていればその基本形を，活用がなく base を持っていれば base を表示する。
7. 空行に対しても “`EOS`”(正確には BOS 文字列と EOS 文字列) を表示する。
8. 解析結果の出力で，見出し語などの直後の区切りがスペースではなくタブになっている。
9. 辞書に登録されていない単語の品詞表示を「未定義語」から「未知語」に変更した。

B.2　茶筌 1.5 から 茶筌 2.0 への拡張点

ここでは v-gram 版，bi-gram 版に共通する拡張点をあげる。

1. 辞書引きに SUFARY を使えるようにすることにより，半角文字も検索できるようにした。
2. SUFARY を使って英語を解析できるようにした。
3. 文頭・文末で出力する文字列を設定できるようにした。
4. 未知語品詞とそのコストを複数指定できるようにした。
5. 出力フォーマット `"%?T/STR1/STR2/"` を使えるようにした。活用があれば `STR1`, なければ `STR2` を出力する。そのほかに `%?I`, `%?B`, `%?F`, `%?U` も使えるようにした。
6. 出力フォーマット `"%rABC"` を導入し，ルビを表示できるようにした。
7. BOS 文字列，EOS 文字列，出力フォーマットで，解析文全体を表示する`"%S"` を使えるようにした。
8. 辞書ファイルの単語コストの範囲を今までの 0 ~ 255.5 から，bi-gram 版は 0 ~ 6553.5 に，v-gram 版は 0 ~ 65535 に変更した。
9. 連接ファイルの連接コストの範囲を 0 ~ 255 から 0 ~ 65535 に変更した。

B.3　茶筌 1.0 から 茶筌 1.5 への拡張点

1. ライブラリ化を行い，茶筌のモジュールを他のプログラムに簡単に組み込めるようにした．
2. サーバ化を行い，クライアントを用いて他のマシンから解析を行うことができるようにした．また，Emacs Lisp 版のクライアントを作成した．
3. `-w` オプションでコスト幅を指定できるようにした．
4. `chasenrc` ファイルに「区切り文字」を指定することにより，jfgets() の区切り文字を設定できるようにした．半角文字を指定することも可能．また，区切り文字のデフォルトを "。!?" に変更した．
5. バッファを動的に確保することにより，文字列が長いときでも “Too many morphs” の警告が出ないようにした．
6. ViCha 用出力オプション -v を新設．
7. `-d` オプションと `-b` を同時に指定したときに`-d` の出力形式で最適解パスだけ表示できるようにした．

B.4　JUMAN 2.0 から 茶筌 1.0 への拡張点

1. 辞書検索の方法を従来の NDBM を用いて疑似的に TRIE 構造を実現する方法から，独自開発のパトリシア木を用いたものに変更した．解析に必要な辞書のサイズが約 4 分の 1 に縮小した．また，辞書のコンパイル時間が 3 ~ 40 分の 1 になった．
2. 解析システムの見直しを行ない，高速化を図った．解析速度が約 8 ~ 11 倍になった (JUMAN 2.0 との比較)．
3. 多くのプラットフォームでインストール可能になるようにコードを書き直した．また，GNU C コンパイラ (gcc) だけでなく OS 付属の C コンパイラなどでもコンパイルできるようにした．
4. 日本語 EUC だけでなく，JIS(ISO-2022-JP) の文字列も解析できるようにした．
5. 未定義連接コストの導入により，未定義語の出力を減らすことができるようになった．
6. 連結品詞を定義できるようにし，最適パスを出力する時に，その品詞の単語を一単語に連結して表示するようにした．
7. 活用語尾の読みを定義できるようにすることにより，「来る」「得る」などの読みがひらがなで表示されるようになった．
8. 入力文を改行コードで区切るのではなく，句点により区切るオプション (`-j`) を追加した．
9. `-r` オプションや環境変数 `CHASENRC` で `chasenrc` ファイルを指定できるようにした．

10. -F オプションや chasenrc ファイルの「出力フォーマット」で解析結果の出力形式を変更できるようにした．
11. 文法の見直しを行ない，品詞分類「特殊」の下の「括弧」を「括弧開」と「括弧閉」に分離した．また，同じく「特殊」の下に「空白」を定義した．「空白」は具体的には全角の空白を表す．
12. 助動詞の活用型に「助動詞べきだ型」を追加した．助動詞「べきだ」の活用を従来の「ナ形容詞」型から「助動詞べきだ型」に変更した．
13. 辞書登録語について見直し，追加削除等の修正を行なった．

C　JUMAN3.0 と 茶筌 との関係について

JUMAN 2.0 が 1994 年 7 月にリリースされて以降，京都大学長尾研究室と奈良先端大松本研究室では，それぞれ異なる方向での拡張を試みていました．京都大学では，従来の bi-gram モデルでは記述できない連接関係を記述するために連語処理や括弧の透過処理などの機能を追加し，文法ファイル，形態素辞書に大幅な修正を行なった拡張版を作成していました．奈良先端大では，今後大量の蓄積が始まると思われる日本語タグ付きコーパスから bi-gram 以上の連接規則 (単語レベルや品詞レベルの設定も含む) を自動的に学習する機能を追加するための拡張と，UNIX のハッシュデータベース NDBM に依存しない辞書の構築を考えていました．後者の拡張は UNIX 以外の OS での稼働を要求する声に対応することと辞書のコンパイル時間と検索速度の改善を目指したことによります．bi-gram 以上の連接規則に対する両者の考え方がかなり異なるため，両者の融合は見合わせることにし，いち早く完成した京都大学の拡張版が 1996 年 6 月に JUMAN3.0beta として公開されました．

奈良先端大で拡張を予定していた機能には下に示すような項目があり、茶筌 1.0 を 1997 年 2 月に公開し、以後、茶筌 1.5, 1.51 を経て、今回の茶筌 2.0b6 においてそのほとんどが実現されました。

1. (茶筌 1.0) 辞書システムの独自開発 (NDBM の棄却, パトリシア木の採用)
2. (茶筌 1.0) 解析システムの見直しと高速化
3. (茶筌 1.0) 未定義連接コスト，連接品詞，解析結果出力フォーマットの導入
4. (茶筌 1.0) JIS 文字列の解析
5. (茶筌 1.0) 活用語尾の読みの定義
6. (WinCha1.0) Windows95 への対応
7. (茶筌 1.5) ライブラリ化
8. (茶筌 1.5) サーバ化
9. (茶筌 2.0) 品詞定義の多階層化
10. (茶筌 2.0) 接続規則の可変長化
11. (茶筌 2.0) 半角文字を含む単語の辞書登録 (SUFARY を利用した辞書)
12. (茶筌 2.0) 出力フォーマットの拡充
13. 解析済みデータからの可変長接続コストの学習

D　添付の日本語辞書 (ipadic1.0) の品詞体系について

ここでは、茶筌 2.0 に添付の日本語辞書 (ipadic1.0) で採用した品詞体系について説明する。これは、情報処理振興事業協会 (IPA) で設定された IPA 品詞体系 (THiMCO97) に基づいて一部修正を加えたものである。本付録は新情報処理開発機構 (RWCP) による「テキストデータベース報告書 (平成 8 年度)」に掲載された IPA 品詞体系 (THiMCO97) の説明を許可を得て抜粋し、一部修正を施したものである。

なお、現在の IPA 品詞体系日本語辞書 (ipadic1.0) は、1998 年 5 月に公開した IPA 品詞体系日本語辞書 (ipadic1.0b2) に対して、奈良先端科学技術大学院大学情報科学研究科鹿野清宏教授を代表とする「日本語ディ

クテーション基本ソフトウェアの開発」(IPA 独創的先進的情報技術に関わる研究開発) のグループの方々に大幅な修正、改良を行っていただいたものである。

説明の書式

品詞名

以後、品詞名のことを「タグ」と呼ぶことがある。それぞれの品詞の説明の際に、以下の記号により注釈が付けられている。

#　品詞の解説

例：　単語例

*　品詞の解説についての備考

&　読み、活用形についての備考

品詞名に関する注意事項

本日本語辞書は、IPA 品詞体系 (THiMCO97) に基づいているが、茶筌の辞書として組み込む際にいくつかの変更を行なった。品詞体系の特徴と変更点について以下にまとめる。

- 品詞は、多段の階層に分類されている。例えば、「名詞 固有名詞 人名 姓」は、四段の階層よりなる品詞名である。以下では、これを「名詞-固有名詞-人名-姓」のようにハイフンで区切って表示する。茶筌 2.0 では、任意の段数の品詞階層の定義が可能になったので、これを直接文法ファイル (grammar.cha) に定義することができる。
- THiMCO97 では、「動詞 一段 連用形 自立」のように、品詞の分類と活用型、活用形が混ざり合った形で定義されていた。茶筌では、品詞の分類の定義と活用に関する定義が分離されているので、これを「動詞-自立 一段 連用形」のように 3 つの項目 (品詞名、活用型、活用形) に分けて記述することにした。
- 品詞名の定義に用いられる分類名を以下の基準に従って変更した。
  1. 「(助動詞語幹)」「(形容動詞語幹)」のように丸括弧を伴う名称の丸括弧をすべて除去した。
  2. 「動詞 接尾 (助動詞)」「形容詞 接尾 (助動詞)」として定義される「(助動詞)」の部分は冗長であるので、省略し、「動詞-接尾」「形容詞-接尾」とした。
  3. 動詞の分類には「動詞」「動詞 非自立」「動詞 接尾」に大別されるが、茶筌の品詞階層の定義では、「動詞」という記述はすべての動詞を表すので、区別のため、「動詞-自立」「動詞-非自立」「動詞-接尾」のように「自立」という細分類を追加した。同様に、活用語以外の単語のための品詞名については、「名詞」「名詞 固有名詞」「名詞 固有名詞 人名」「名詞 固有名詞 人名 姓」のような分類を、それぞれ、「名詞-一般」「名詞-固有名詞-一般」「名詞-固有名詞-人名-一般」「名詞-固有名詞-人名-姓」のように、「一般」という細分類を追加して、排他的に品詞の定義を行なった。
  4. 用言の活用形については、「未然ナイ接続」「未然レル接続」「未然ウ接続」「連用タ接続」「連用マス接続」「連用タイ接続」··· のように、後続する助動詞類に応じて細かく定義されていたが、個々の活用型については、「未然」「連用」等で異なる語尾形を持つものは少ない。よって、活用形の名称は、「未然形」「連用形」「基本形」「仮定形」「命令」を基本的な活用形とし、例外的な形のものに対してのみ、THiMCO97 の活用形名を使用した。なお、茶筌では辞書出現形に対して「基本形」という活用形を与える仕様になっているため、THiMCO97 の「見出し形」という活用形名を「基本形」という名前に変更した。

5. 「未然ウ接続」は、五段活用の動詞については「う」が接続し、その他の活用型の動詞については「よう」が接続するための活用である。しかし、「う」「よう」を単語 (助動詞) として認めず、「未然ウ接続」については、「う」あるいは「よう」がついた形を活用形として「意志形」という名称を与えた。

- ipadic1.0 に登録の単語の読み情報については、「日本語ディクテーション基本ソフトウェアの開発」グループの努力により、従来のようなふりがなではなく、より発音に近い形に修正された。例えば、係助詞の「は」の読みは「ワ」、「常識」の読みは「ジョーシキ」のように長音は「ー」によって示されている。また、綴りも品詞も等しいが読みだけが異なる単語、例えば、「私 (ワタシ/ワタクシ)」については、{ ワタシ/ワタクシ } のようにすべての可能な読みを付与し、一つの語として登録した。

D.1 名詞

名詞-一般

普通名詞、あるいは、下位分類が未定の名詞。

名詞-固有名詞-一般

一般的な固有名詞、あるいは、下位分類が未定の固有名詞。

例： 「関西国際空港」

名詞-固有名詞-人名-一般

姓と名に分けられないもの、外国人名。あるいは、姓・名の決定が未定の人名。

例： 「Ａ・Ｇ・スポルディング」

名詞-固有名詞-人名-姓

主に日本人の姓。

例： 「山田」...

名詞-固有名詞-人名-名

主に日本人の名。

例： 「太郎」...

名詞-固有名詞-組織

組織を表わす名称。

例： 「通産省」「ＮＨＫ」...

名詞-固有名詞-地域-一般

国名以外の地名を表わすもの。

例： 「アジア」「スマトラ島」

名詞-固有名詞-地域-国

\#　国の名前。

例：　「日本」「オーストラリア」...

名詞-代名詞-一般

\#　いわゆる代名詞、不定語。

例：　「それ」「ここ」「あいつ」「あなた」「あちこち」「いくつ」「どこか」「なに」「みなさん」「みんな」「わたくし」「われわれ」...

名詞-代名詞-縮約

\#　代名詞と係助詞「は」の組み合わせで、短縮した形<口語>。

例：　「ありゃ」「こりゃ」「こりゃあ」「そりゃ」「そりゃあ」

名詞-副詞可能

\#　曜日、月など時間を表す副詞的な用法を持つ名詞。量や割合などを表し副詞的に使うことのできる名詞。

例：　「金曜」「一月」「午後」「少量」...

*　元の IPA 品詞体系では、「名詞-副詞可能」のうち実際に副詞的に使われてるものを「名詞 副詞可能 副詞的」、副詞的な使用が可能であるが、副詞的に用いられていないものを「名詞 副詞可能」とラベル付けされることになっているが、ここでは、文内の用法に関係なく副詞的に働き得るものをすべて「名詞-副詞可能」と呼ぶ。

名詞-サ変接続

\#　格要素をとり、後ろに「する」「できる」「なさる」「くださる」「、」「。」が後接することができるもの。

例：　「インプット」「愛着」「悪化」「悪戦苦闘」「一安心」「下取り」「具体化」...

*　「オノマトペ（+スル）」は、[副詞-助詞類接続] とした。

名詞-形容動詞語幹

\#　いわゆる形容動詞語幹で、「な」の前に現れるもの。

例：　「健康」「安易」「駄目」「だめ」...

*　元の IPA 品詞体系では「名詞（形容動詞語幹）」となっていたが、第 2 階層の「(形容動詞語幹)」の括弧を取り除いた。

名詞-数

\#　漢数字、算用数字、および、「何(回)」「数(%)」「幾(夜)」。

例：　「0」「1」「2」「何」「数」「幾」..

名詞-非自立-一般

連体詞、「の(格助詞)」、活用語の連体形[見出し形]に接続して使われるもののうち、以下の下位分類にあてはまらないもの。いわゆる形式名詞を含む。

* 普通名詞としての用法であれば、連体修飾を受けていても[非自立]ではない。

例: 「あかつき」「暁」「かい」「甲斐」「気」「きらい」「嫌い」「くせ」「癖」「こと」「事」「ごと」「毎」「しだい」「次第」「順」「せい」「所為」「ついで」「序で」「つもり」「積もり」「点」「どころ」「の」「はず」「筈」「はずみ」「弾み」「拍子」「ふう」「ふり」「振り」「ほう」「方」「旨」「もの」「物」「者」「ゆえ」「故」「ゆえん」「所以」「わけ」「訳」「わり」「割り」「割」「ん<口語>」「もん<口語>」...

名詞-非自立-副詞可能

連体詞、「の(格助詞)」、活用語の連体形[見出し形]に接続して使われるもののうち、副詞的に働くことが可能なもの。

* 文脈上で、実際に副詞的に働いている場合には、IPA 品詞体系ではうしろに[副詞的]を付加することになっているが、それは省略した。

例: 「あいだ」「間」「あげく」「挙げ句」「あと」「後」「余り」「以外」「以降」「以後」「以上」「以前」「一方」「うえ」「上」「うち」「内」「おり」「折り」「かぎり」「限り」「きり」「っきり」「結果」「ころ」「頃」「さい」「際」「最中」「さなか」「最中」「じたい」「自体」「たび」「度」「ため」「為」「つど」「都度」「とおり」「通り」「とき」「時」「ところ」「所」「とたん」「途端」「なか」「中」「のち」「後」「ばあい」「場合」「日」「ぶん」「分」「ほか」「他」「まえ」「前」「まま」「儘」「侭」「みぎり」「矢先」...

名詞-非自立-形容動詞語幹

連体詞、「の(格助詞)」、活用語の連体形[見出し形]に接続して使われるもののうち、「な(助動詞「だ」)の体言接続」と接続可能なもの。

例: 「みたい」「ふう」「よう」

* 元の IPA 体系では、「名詞 非自立 (形容動詞語幹)」と書かれている。元の IPA 体系では、「よう」が「名詞 非自立 (助動詞語幹)」として区別されていたが、特に区別する理由がないので、「よう」も「名詞-非自立-形容動詞語幹」とした。

*

名詞-特殊-助動詞語幹

終止形[見出し形]に接続するもので、学校文法で助動詞とされている「そうだ(伝聞)」の語幹部分。

例: 「そう」

* 元の IPA 体系では、「名詞 特殊 (助動詞語幹)」と書かれている。

名詞-接尾-一般

名詞、あるいは他の品詞の語幹[ガル接続]や[連用タイ接続]に接続して複合名詞を形成する語のうち、下位の分類にあてはまらないもの。一般に「接尾語」というよりも範囲が広く、複合名詞の後ろ要素として用いられることが普通なもの。

例：　「おき」「かた」「方」「甲斐(がい)」「がかり」「ぎみ」「気味」「ぐるみ」「(～した)さ」「次第」「済(ず)み」「よう」「(でき)っこ」「感」「観」「性」「学」「類」「面」「用」...

名詞-接尾-サ変接続

#　名詞に接続して名詞を形成する接尾語のうち「スル」に前接し得るもの。

例：　「化」「視」「分け」「入り」「落ち」「買い」

名詞-接尾-助動詞語幹

#　他の品詞の連用形に接続し、学校文法で助動詞の語幹とされている「そうだ(様態)」の語幹部分。

例：　「そう」

*　元の IPA 体系では、「名詞 接尾 (助動詞語幹)」と書かれている。

名詞-接尾-形容動詞語幹

#　他の名詞や活用語の連用形に接続する接尾語で、「な」([助動詞-特殊型-体言接続]) に前接するもの。

例：　「的」「げ」「がち」

*　元の IPA 体系では、「名詞 接尾 (形容動詞語幹)」と書かれている。

名詞-接尾- 副詞可能

#　他の名詞に接続する接尾語で、副詞的に働くことが可能なもの。

*　IPA 品詞体系では、文脈上、実際に副詞的に働いているものは、うしろに [副詞的] と付加してあるが、ここでは具体的な用法にかかわらず、副詞的な使用が可能なものをすべてこの分類とした。

例：　「後(ご)」「以後」「以降」「以前」「前後」「中」「末」「上」「時(じ))」

名詞-接尾-助数詞

#　数に接続して名詞を形成する接尾。一般の「助数詞」よりも範囲が広く、数に接続する普通名詞も含まれる。

例：　「個」「つ」「本」「冊」「パーセント」「ｃｍ」「ｋｇ」「カ月」「か国」「区画」「時間」「時半」...

*　IPA 品詞体系では、これらのうち副詞的に用いられているものに「名詞 接尾 助数詞 副詞的」というタグを与えているが、これは用法に関するタグであるため、本体系では含めなかった。

名詞-接尾-特殊

#　主に用言につく特殊な接尾辞として新たに定義した。

例：　「(楽し) さ」「(考え) 方」

*　IPA 品詞体系では、「名詞 接尾」に分類されている。

名詞-接続詞的

#　単語と単語を接続する接続詞的な働きをするもの。

例：「(日本) 対 (アメリカ)」「対 (アメリカ)」「(3) 対 (5)」「(女優) 兼 (主婦)」

名詞-動詞非自立的

[助詞-接続助詞] の「て」に接続するもので、意味的には動詞的なもの。

例：「ごらん」「ご覧」「御覧」「頂戴」

注 IPA 品詞体系には、単語への分割が不可能なもの、および、ことわざ、漢詩、方言、英語などを表すタグとして「名詞 引用文字列」が用意されている。また、数式を表すためのタグ「名詞 数式」が用意されている。これらは品詞タグとは考えにくいため、本体系では含めなかった。

D.2 接頭詞

接頭詞-名詞接続

名詞 (形容動詞語幹を含む) に前接する接頭語のうち、数に接続するもの以外。

例：「お (水)」「某 (氏)」「同 (社)」「故 (~氏)」「高 (品質)」「お (見事)」「ご (立派)」

接頭詞-数接続

名詞に前接する接頭語のうち、数に接続するもの。

例：「約」「全長」「弱冠」「月 (千円)」「年 (一回)」

接頭詞-動詞接続

動詞の命令形あるいは [動詞 連用タイ接続] +「なる / なさる / くださる」に前接する接頭語。

例：「お (読みなさい)」「お (座り)」

接頭詞-形容詞接続

形容詞に前接する接頭語。

例：「お (寒いですねえ)」「バカ (でかい)」

D.3 動詞

活用形に関する注意

未然形 THiMCO97 では以下のように細かく分類されているが、語尾形に変化のない限り、「未然形」に統一した。なお、「未然ウ接続」については、「う」あるいは「よう」がついた形を活用形として「意志形」という名称を与えた。

- 未然レル接続

  # −(ラ) レル , −(サ) セルに接続するもの。

  例：「読ま」「さ」...

- 未然ナイ接続

  # −ナイに接続するもの。

例： 「読ま」「し」...

- 未然ヌ接続

  ＃ −ヌ，−（サ）シメルに接続するもの。

  例： 「読ま」「せ」「来」...

- 未然ウ接続

  ＃ −（ヨ）ウに接続するもの。

  例： 「読も」「し」...

  ＆ ipadic1.0 では、（ヨ）ウが接続した形を「意志形」と定義した。

連用形 例外的な語尾以外はすべて「連用形」という名称に統一した。

- 連用マス接続

  ＃ −マスに接続するもの。

  例： 「読み」「し」「なさい」...

- 連用タイ接続

  ＃ −タイ，−ソウ，−ヅライ，−方（かた），読点などに接続するもの。

  例： 「読み」「し」「なさり」「向かひ」「習ひ」...

- 連用タ接続

  ＃ −タ，−テに接続するもの。

  例： 「読ん」「書い」「行っ」「問う」...

基本形 THiMCO97 では、「見出し形」と呼ばれているもの。

＃ 句点，体言，−マイなどに接続するもの。

例： 「読む」「なさる」「問う」...

仮定形 THiMCO97 では、「仮定バ接続」と呼ばれている。

＃ −バ，−ドモに接続するもの。

例： 「読め」「すれ」...

命令ｉ ＃ カ変・五段ラ行特殊の命令形。およびサ変・スルの命令形「せよ」の口語形。

例： 「来い」「なさい」「せい」...

命令ｅ ＃ 五段の命令形、文語已然形、一段動詞の語幹止め命令用法（「くれ」のみ）。

例： 「読め」「（とは）いえ」「（程度の差こそ）あれ」「（やめて）くれ」...

＊ 「（やめて）くれ」は「（やめて）くれろ」の「ろ」が落ちた形。「くれる」は一段動詞の中の特殊活用型とすべきものである。なお、「（やめて）（お）くれ（なさい）」の「くれ」は [動詞-非自立 一段 連用タイ接続] であり、別のものであり、この口語形は「おくんなさい」となる。

命令ｙｏ ＃ 一段・サ変・文語（カ変）の命令形で「ｙｏ」で終わるもの。

例： 「せよ」「みよ」「来よ」...

命令ｒｏ ＃ 一段・サ変の命令形で「ｒｏ」で終わるもの。

例： 「しろ」「みろ」...

ベキ接続 ＃ 「ベキ」につづく形、サ変の場合のみ。

例： 「す」...

仮定縮約１ ＃ 仮定バ接続と「バ」とが合わさって短縮した形<口語>。

例： 「分かれりゃ」

体言接続 ＃ 文語の場合のみ。見出し形と異なる形があるもの。

例： 「助くる」（cf.「助く」）

体言接続特殊 ＃ 「る」で終る動詞が「の」などに接続する場合に音便化した形<口語>。

例： 「（何）すん（の？）」

動詞の活用型一覧 (現代語)

#【 活用形 】の位置には次のものが入る。(【 】の記号はない)

動詞-自立 カ変 【 活用形】

例： 「 くる 」「 来る 」「 やってくる 」「 やって来る 」

動詞-非自立 カ変 【 活用形】

例： 「(て) くる 」「(て) 来る 」

動詞-自立 サ変・スル 【 活用形】

「 する 」、および、[名詞 サ変接続] に接続する 「 する 」。

例： 「 する 」

動詞-非自立 サ変・スル 【 活用形】

動詞の [連用タイ接続](+ [助詞-係助詞]) に接続する 「 する 」。

例： 「 (お読み) する 」「 (~を読みも) する 」「(~を読みは) する 」...

* 「 お読みする 」は、「 お 」は [接頭詞-動詞接続]、「 読み 」は [動詞 五段・マ行 連用タイ接続]、「 する 」は [動詞 サ変・スル 見出し形 非自立]。

* 「 ~を読みもする 」「 ~を読みはする 」は、「 読み 」は [動詞 五段・マ行 連用タイ接続]、「 も 」は [助詞 係助詞]、「 する 」は [動詞 サ変・スル 見出し形 非自立]。

動詞-自立 サ変・-スル 【 活用形】

和語系のサ変動詞。

例： 「 接する 」...

* 「 -し + ない 」「 -せ + られる 」「 -せ + ぬ 」「 -し + よう 」「 -する 」「 -すれ + ば 」「 -せよ 」「 -しろ 」の形だけを [動詞 サ変・スル] とした。「 -し + 、」「 -し + た 」「 -し + たい 」などの連用形はすべて [動詞 五段・サ行] とした。

動詞-自立 サ変・-ズル 【 活用形】

和語系のザ変動詞。

例： 「 信ずる 」...

* 「 -ぜ + られる 」「 -ぜ + ぬ 」「 -ずる 」「 -ずれ + ば 」「 -ぜよ 」「 -ず + べし 」の形だけを [動詞 サ変・-ズル] とした。「 -じ + ない 」「 -じ + よう 」の未然形および 「 -じ + 、」「 -じ + た 」「 -じ + たい 」などの連用形、および 「 -じろ 」の命令形は [動詞 一段] とした。

動詞-自立 一段 【活用形】

上一段活用および下一段活用。

例： 「着る」

＊ 「病める」は、[見出し形]のみ。

動詞-非自立 一段 【活用形】

例： 「あげる」「うる」「える」「得る」「おえる」「終える」「おおせる」「かねる」「兼ねる」「かける」「きれる」「切れる」「すぎる」「過ぎる」「そこねる」「損ねる」「そびれる」「そめる」「初める」「つける」「つづける」「続ける」「(お読み)できる」「(お読み)出来る」「はじめる」「始める」「(て)いる」「(～しては)いけ(ない)」「(て)くれる」「(て)差し上げる」「(て)のける」「(て)みる」「(て)みせる」[(て)もらえる」「(て)る<口語>」

＊ 「(～しては)いけ(ない)」の終止形は「いける」。

＊ 「(勉強)できる」は[非自立]としない。

＊ 「うる」は、「うる」([見出し形])と「うれ」([仮定バ接続])のみ。「う」は[動詞 文語 見出し形]とする。

動詞-接尾 一段 【活用形】

学校文法では助動詞と呼ばれているもの。

例： 「させる」「せる」「しめる」「しむる」「られる」「れる」

動詞-自立 五段・カ行イ音便 【活用形】

五段カ行で、[助詞 接続助詞]の「て」に接続するときにイ音便になるもの。

例： 「解く」...

動詞-非自立 五段・カ行イ音便 【活用形】

例： 「つづく」「続く」「ぬく」「抜く」「(て)いただく」「(て)頂く」「(て)おく」「とく<口語>」「どく<口語>」

動詞-非自立 五段・カ行促音便 【活用形】

五段カ行で、[助詞 接続助詞]の「て」に接続するときに促音便になるもの。

例： 「いく」「行く」「ゆく」

＊ 「ゆく」には、「ゆっ(て)」の形はないが、このタグを振る。「ゆき(て)」は[動詞 文語 連用タ接続]とする。

動詞-非自立 五段・カ行促音便 【活用形】

例： 「いく」「行く」「ゆく」「く<口語>」

＊　「ゆく」には、「ゆっ(て)」の形はないが、このタグを振る。「ゆき(て)」は[動詞 文語 連用タ接続]とする。

動詞-自立 五段・ガ行【活用形】

＃　五段ガ行で、[助詞 接続助詞]の「て」に接続するときにイ音便になるもの。

例：　「継ぐ」「急ぐ」...

動詞-自立 五段・サ行【活用形】

＃　五段サ行で、[助詞 接続助詞]の「て」に接続するときに音便化しないもの。

例：　「話す」...

動詞-非自立 五段・サ行【活用形】

例：　「いたす」「致す」「だす」「出す」「つくす」「尽くす」「直す」

動詞-自立 五段・タ行【活用形】

＃　五段タ行で、[助詞 接続助詞]の「て」に接続するときに促音便になるもの。

例：　「持つ」...

動詞-自立 五段・ナ行【活用形】

＃　五段ナ行で、[助詞 接続助詞]の「て」に接続するときにハツ音便になるもの。

例：　「死ぬ」

動詞-自立 五段・バ行【活用形】

＃　五段バ行で、[助詞 接続助詞]の「て」に接続するときにハツ音便になるもの。

例：　「呼ぶ」...

動詞-自立 五段・マ行【活用形】

＃　五段マ行で、[助詞 接続助詞]の「て」に接続するときにハツ音便になるもの。

例：　「進む」...

動詞-非自立 五段・マ行【活用形】

例：　「こむ」「込む」

動詞-自立 五段・ラ行【活用形】

＃　五段ラ行で、[助詞 接続助詞]の「て」に接続するときに促音便になるもの。

例：「切る」「なる」…

動詞-非自立 五段・ラ行【活用形】

例：「おわる」「終る」「終わる」「かかる」「きる」「切る」「しぶる」「渋る」「まいる」「まわる」「回る」「やがる」「(せねば/しては)なら(ない)」「(て)ある」「(て)おる」「(て)まわる」「(て)回る」「(て)やる」「ちゃる<口語>」「じゃる<口語>」「ぢゃる<口語>」

*　「なら(ない)」の終止形は「なる」

動詞-接尾 五段・ラ行【活用形】

例：「がる」

動詞-自立 五段・ラ行特殊【活用形】

#　五段ラ行で、助動詞「ます」に接続する形および命令形が「-い」の形になるもの。

例：「いらっしゃる」「おっしゃる」「仰言る」「くださる」「下さる」「なさる」「ござる」

動詞-非自立 五段・ラ行特殊【活用形】

例：「(お読み)なさる」「(お読み)くださる」「(お読み)下さる」「(て)くださる」「(て)下さる」「(て)いらっしゃる」「(て)らっしゃる<口語>」

*　「(お読み)なさる」「(お読み)くださる」は[非自立]としたが、「(ご期待)なさる」「(ご期待)くださる」は[非自立]としない。

動詞-自立 五段・ワ行ウ音便【活用形】

#　五段ワ行で、[助詞 接続助詞]の「て」に接続するときにウ音便になるもの。

例：「問う」「乞う」「沿う(て)」「ゆう(て)」「食う(て)」「すう(て)」「負う(て)」

*　[動詞 五段・ワ行促音便]の動詞については、「て」に接続する活用語尾が「う」になっている場合のみ、このタグを振り、それ以外は、[動詞 五段・ワ行促音便]を振る。(人手修正データ中に現れたものは「ゆう」「食う」「すう」「負う」)

動詞-非自立 五段・ワ行ウ音便【活用形】

例：「たまう」「給う」

動詞-自立 五段・ワ行促音便【活用形】

#　五段ワ行で、[助詞 接続助詞]の「て」に接続するときに促音便になるもの。

例：「言う」「ゆう」「「食う」「負う」「憂う」..

*　「憂う」には「憂って」はないが、このタグを振る。(人手修正データ中に現れたものは「憂い(、)」の形のみ)

＊　[動詞 五段・ワ行促音便] の動詞については、「 て 」に接続する活用語尾が「 う 」になっている場合のみ、[動詞 五段・ワ行ウ音便] を振る。

動詞-非自立 五段・ワ行促音便 【 活用形 】

例：　「 あう 」「 合う 」「 そこなう 」「 損なう 」「(て) しまう 」「(て) もらう 」「 じゃう < 口語 > 」「 じまう < 口語 > 」「 ちまう < 口語 > 」「 ちゃう < 口語 > 」

動詞の活用型一覧 (文語)

IPA 品詞体系では、文語の活用型の細分類は行われていない。現在の辞書は、文語動詞の活用型として次のものだけを記述した。今後充実させる必要がある。活用体系が文語残存のものと、口語ではあるが歴史仮名づかいで示されているものとを含む。

動詞-自立 文語・四段 【 活用形 】

例：　「 いふ 」「 云ふ 」「 向かふ 」「 習ふ 」「 思ふ 」「 能ふ 」

＊　「 云へ (り)」は已然形だが、[動詞 文語・四段 命令ｅ] とした。

動詞-自立 文語・ラ変 【 活用形 】

例：　「 あり 」「 なり 」「 しかり 」

動詞-自立 文語・下二 【 活用形 】

例：　「 用ゆ 」

D.4　形容詞

「 見出し形 」「 仮定バ接続 」「 文語見出し形 」をそれぞれ「 基本形 」「 仮定形 」「 文語基本形 」と呼ぶ以外は、ほぼ THiMCO97 で用いられている活用形名を用いた。なお、形容詞の活用型を「 形容詞・アウオ段 」「 形容詞・イ段 」「 形容詞・文語 」に分類した。

未然ヌ接続

＃　−ヌに接続するもの。

例：　「 寒から 」...

未然ウ接続

＃　−ウに接続するもの。

例：　「 寒かろ 」...

連用タ接続

＃　−タに接続するもの。

例：「寒かっ」...

連用テ接続

#　-テ，-ナイ，-ナル，-スル，読点に接続するもの。

例：「寒く」...

連用ゴザイ接続

#　-ゴザイマスに接続するもの。

例：「寒う」「大きゅう」「のう」...

基本形

#　句点，体言などに接続するもの。

例：「寒い」「大きい」「ない」...

体言接続

#　文語活用で体言に接続するもの。

例：「寒き」「なき」...

&　基本形には「-イ」の形を入れた。

仮定形

#　-バに接続するもの。

例：「寒けれ」「なけれ」...

&　THiMCO97 では「仮定バ接続」と呼ばれていた。

命令

#　文語活用で命令形のもの。

例：「よかれ」「美しかれ」...

&　終止形には「-イ」の形を入れた。

ガル接続

#　-ガル，-ゲ，-ソウに接続するもの。

例：「寒」「悲し」...

文語基本形

#　−シで終わるもの。

例：　「良し」「遠し」「やむなし」...

文語体言接続

#　−キで終わって体言に接続するもの、「−イ」で終わる形のないもの。

例：　「悪しき」...

仮定縮約１

#　仮定バ接続と「バ」とが合わさって短縮した形１ <口語>。

例：　「欲しけりゃ」「(それが)なけりゃ(困る)」

仮定縮約２

#　仮定バ接続と「バ」とが合わさって短縮した形２ <口語>。

例：　「(それが)なきゃ(困る)」

##【活用形】の位置には次のものが入る。(【】の記号はない)

形容詞-自立 形容詞・アウオ段【活用形】

#　形容詞の活用型のうち、語幹の最後の母音がアウオのいずれかで終わるもの。

例：　「青い」「赤い」「厚い」「暑い」「熱い」...

*　IPA 品詞体系では、「ない」の文語見出し形「なし」を形容詞の文語型活用の見出し形として定義しているが、本体系では、「形容詞・アウオ段 (あるいは、形容詞・イ段)」型の「文語見出し形」という活用形として定義している。同様に、IPA 体系で形容詞の文語型活用の体言接続と定義されている「悪しき」などは、本体系では「形容詞・文語」型の「文語体言接続」形として定義した。

形容詞-自立 形容詞・イ段【活用形】

#　形容詞の活用型のうち、語幹の最後の母音がイで終わるもの。

例：　「哀しい」「楽しい」「頼もしい」...

形容詞-自立 形容詞・文語【活用形】

#　形容詞の中で「文語体言接続」の用法をもつもの。

例：　「悪し」

形容詞-非自立 形容詞・アウオ段【活用形】

#　動詞の[連用タイ接続]あるいは[連用タ接続]に後接する形容詞。

例：　「がたい」「難い」「づらい」「にくい」「やすい」「(て)よい」「(て)良い」

形容詞-非自立 形容詞・イ段【活用形】

動詞の [連用タイ接続] あるいは [連用タ接続] に後接する形容詞。

例：「らしい」「(て) いい」「(て) ほしい」「(て) 欲しい」

形容詞-接尾 形容詞・アウオ段【活用形】

学校文法では助動詞とされるもの。

例：「(食べ) たい」

D.5 副詞

副詞-一般

必ず後ろで切れるもの、連体修飾が不可能なもの。

例：「何だか」「何はともあれ」「何といっても」「何がなんだか」「何もかも」「何となく」「何とか」「何かと」「何より」「何とも (申し訳ないです)」「何しろ」「何でも (このパーティーは彼の主催だそうだ)」「なんら」「何かしら」「何やら」「何で」「何も (おこらなくったって)」「どんなに」「こんなにも」...

副詞-助詞類接続

「の」「は」「に」「な」「する」「だ」などが後続することが可能な副詞。

例：「こんなに」「そんなに」「あんなに」「なにか」「なんでも」

D.6 連体詞

連体詞

名詞を修飾する形しかもたないもの。

例：「この」「その」「あの」「どの」「いわゆる」「なんらかの」「何らかの」「いろんな」「こういう」「そういう」「ああいう」「どういう」「こんな」「そんな」「あんな」「どんな」「大きな」「小さな」「おかしな」「ほんの」「たいした」「(−も) さる (ことながら)」「微々たる」「堂々たる」「単なる」「いかなる」「我が」「同じ」「亡き」...

D.7 接続詞

接続詞

独立に現れる接続詞。

例：「が」「けれども」「そして」「じゃあ」「それどころか」...

D.8 助詞

助詞-格助詞-一般

いわゆる格助詞。

＊　「にて」「とて」も格助詞に含めた。「の」には格助詞としての用法と名詞と名詞を接続するいわゆる「AのB」用法があるが、今回は区別をせず格助詞として統一した。

例：　「から」「が」「で」「と」「に」「へ」「より」「を」「の」「にて」「とて」

助詞-格助詞-引用

#　名詞、人物の台詞「 」、会議の決定事項、理由、判決、推測表現等の直後の「と」。

例：　「(～だ)と(述べた。)」「(～である)と(して執行猶予…)」「(なん)て(魚？)<口語>」

助詞-格助詞-連語

#　格助詞と動詞との連語で、主に格助詞に相当するような働きを持つもの。

例：　「という」「といった」「とかいう」「として」「とともに」「と共に」「でもって」「にあたって」「に当たって」「に当って」「にあたり」「に当たり」「に当り」「に当たる」「にあたる」「において」「に於いて」「に於て」「における」「に於ける」「にかけ」「にかけて」「にかんし」「に関し」「にかんして」「に関して」「にかんする」「に関する」「に際し」「に際して」「にしたがい」「に従い」「に従う」「にしたがって」「に従って」「にたいし」「に対し」「にたいして」「に対して」「にたいする」「に対する」「について」「につき」「につけ」「につけて」「につれ」「につれて」「にとって」「にとり」「にまつわる」「によって」「に依って」「に因って」「により」「に依り」「に因り」「による」「に依る」「に因る」「にわたって」「にわたる」「をもって」「を以って」「を通じ」「を通じて」「を通して」「をめぐって」「をめぐり」「をめぐる」「って<口語>」「ちゅう<関西弁「という」>」「(何)ていう(人)<口語>」「っていう<口語>」「といふ」「とかいふ」

助詞-接続助詞

例：　「から」「からには」「が」「けれど」「けれども」「けど」「し」「つつ」「て」「で」「と」「ところが」「どころか」「とも」「ども」「ながら」「なり」「ので」「のに」「ば」「ものの」「や(～した)」「やいなや」「(ころん)じゃ(いけない)<口語>」「(行っ)ちゃ(いけない)<口語>」「(言っ)たって(しかたがない)<口語>」「(それがなく)ったって(平気)<口語>」

助詞-係助詞

例：　「こそ」「さえ」「しか」「すら」「は」「も」「ぞ」

助詞-副助詞

例：　「がてら」 「かも」「くらい」「位」「ぐらい」「しも」「(学校)じゃ(これが流行っている)<口語>」「(それ)じゃあ(よくない)<口語>」「ずつ」「(私)なぞ」「など」「(私)なり(に)」「(先生)なんか(大嫌い)<口語>」「(私)なんぞ」「(先生)なんて(大嫌い)<口語>」「のみ」「だけ」「(私)だって<口語>」「だに」「(彼)ったら<口語>」「(お茶)でも(いかが)」「等(とう)」「(今後)とも」「ばかり」「ばっか<口語>」「ばっかり<口語>」「ほど」「程」「まで」「迄」「(誰)も(が)([助詞-格助詞]および[助詞-係助詞]の前に位置する「も」)

助詞-並立助詞

例：「と」「たり」「だの」「だり」「とか」「なり」「や」「やら」

助詞-終助詞

例：「かい」「かしら」「さ」「ぜ」「(だ)っけ<口語>」「(とまってる)で<方言>」「な」「ナ」「なあ<口語>」「ぞ」「ね」「ネ」「ねぇ<口語>」「ねえ<口語>」「ねん<方言>」「の」「のう<口語>」「や」「よ」「ヨ」「よぉ<口語>」「わ」「わい<口語>」

＊　終助詞の「や」は「(まあいい)や」「(すごい)や」など。関西方言の「や」は、不変化活用の助動詞として扱う。

助詞-副助詞／並立助詞／終助詞

＃　「か」のうち、副助詞、並立助詞、終助詞いずれかわからないもの、たとえば、次の (a)(b)(c) のようなもの。

＃　(a)「ＡかＢか」型。例：:「(国内で運用する)か、(海外で運用する)か(。)」

＃　(b) 副詞節中。例 :「(幸いという)か(、死者はいなかった。)」「(祈りが届いたせい)か(、試験に合格した。)」

＃　(c)「かのように」。例 :「(何もなかった)か(のように振る舞った。)」

例：「か」

＊　最新の IPA 品詞体系では、これをさらに「副助詞」「並立助詞」「終助詞」に細分類しているが、本体系では区別しなかった。

助詞-連体化

＃　[名詞] に接続して体言にかかる「の」。

＊　THiMCO97 では、この用法の「の」も格助詞に分類されている。

助詞-副詞化

＃　擬音語、擬声語、擬態語及びそれに類する名詞や副詞の直後の「に」「と」。

例：「に」「と」

＊　ただし、「する」「なる」に係っているもので、状態変化を表す場合は格助詞とする。

助詞-特殊

＃　以上の分類にあてはまらないもの。短歌や俳句等に用いられる助詞などを含む。

例：「かな」「けむ」「(～しただろう)に」「(あんた)にゃ(わからん)」「(俺)ん(家)」

D.9　助動詞

助動詞 五段・ラ行アル 【 活用形 】

#　動詞型の活用の助動詞。「 である 」「 ではある 」などの「 ある 」。

例：　「 ある 」

助動詞 五段・ラ行ゴザル 【 活用形 】

#　動詞型の活用の助動詞「 ござる 」。

例：　「 ござる 」

助動詞 形容詞・イ段 【 活用形 】

#　形容詞型活用の助動詞。

例：　「 らしい 」

助動詞 特殊・ナイ 【 活用形 】

#　否定の助動詞「 ない 」の活用型。

例：　「 ない 」

助動詞 特殊・タ 【 活用形 】

#　完了を表す助動詞「 た 」の活用型。

例：　「 た 」「 だ 」

*　「 (学ん) だ 」「 (泳い) だ 」のように五段活用のガ行、ナ行、バ行、マ行に接続する場合には表層形が「 だ 」になるため、本体系では、それぞれ別の語幹をもつ形態素として定義した。

助動詞 特殊・ダ 【 活用形 】

#　断定の助動詞「 だ 」の活用型。

例：　「 だ 」

助動詞 特殊・デス 【 活用形 】

#　断定の助動詞「 です 」の活用型。

例：　「 です 」

助動詞 特殊・ジャ 【 活用形 】

#　断定の助動詞「 じゃ 」の活用型。

例：　「 じゃ 」

*　断定の「 だ 」が鈍ったもの。

助動詞 特殊・マス【活用形】

謙譲・丁寧を表わす助動詞「ます」の活用型。

例：「ます」

助動詞 特殊・ヌ【活用形】

否定の助動詞「ぬ」の活用型。

例：「ぬ」

助動詞 不変化型【活用形】

現代では活用しない助動詞。活用が想定できない口語や方言も含む。

例：「う」「よう」「まい」「(いざ行か)ん(む)」「(わから)ん<口語>」「(賜ラ)ン」「(美しい/学生)じゃ(ないか/ありません)<口語>」「~(美しい/学生)じゃん<口語>」「(~する)なかれ」「(使いたいん)や<「だ」の方言>」「やろ<方言>」「(いい)っす<口語>」「(いか)ねぇ<口語>」「(取れ)ねえ<口語>」「(負けてなら)じ」

助動詞 文語・?? 【活用形】

文語の助動詞。現在定義されている活用型は次の通り。「文語・ベシ」「文語・ゴトシ」「文語・ナリ」「文語・マジ」「文語・シム」「文語・キ」「文語・ケリ」「文語・ル」

例：「べし」「ごとし」「如し」「たり」「なり」「まじ」「き」「けり」「り」「る」

* IPA 品詞体系では、「じ」のための活用型が用意されているが、実際には不変化であるので、ここでは不変化型とした。

D.10 感動詞

感動詞

感動詞。あいさつなど。

例：「おはよう」「おはようございます」「こんにちは」「こんばんは」「ありがとう」「どうもありがとう」「ありがとうございます」「いただきます」「ごちそうさま」「さよなら」「さようなら」「はい」「いいえ」「ごめん」「ごめんなさい」...

D.11 記号

記号-一般

以下の分類以外の一般的な記号。

例：「○」「◎」「＠」「＄」「〒」「→」「＋」など。現在のところ、アルファベットもここに分類されている。

記号-句点

いわゆる句点。

例： 「。」「 .」

記号-読点

いわゆる読点。

例： 「、」「 ,」

記号-空白

全角の空白文字 (画面上には見えない)。

記号-括弧開

例： 「(」「{ 」「' 」「" 」「『 」「【 」…

記号-括弧閉

例： 「)」「 }」「 '」「 "」「 』」「 】」…

D.12 その他

その他-間投

[名詞-接尾] や、[助詞-終助詞] としにくいもの。

例： 「(だ) ァ 」