形態素解析システム
『茶筌』version 2.2.3
使用説明書

松本裕治　北内啓　山下達雄　平野善隆　松田寛　高岡一馬　浅原正幸

平成 13 年 2 月

Copyright © 2001 奈良先端科学技術大学院大学 松本研究室

Morphological Analysis System ChaSen 2.2.3 Users Manual

Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda, Kazuma Takaoka and Masayuki Asahara

Copyright (c) 2001 Nara Institute of Science and Technology All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by Nara Institute of Science and Technology.
4. The name Nara Institute of Science and Technology may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY Nara Institute of Science and Technology "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE Nara Institute of Science and Technology BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

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 2.0	15 December 1999
version 2.2.0	06 December 2000
version 2.2.1	20 December 2000
version 2.2.2	22 February 2001
version 2.2.3	24 February 2001
ChaSen for Windows	
version 1.0	29 March 1997
version 2.0	15 December 1999
NAIST Technical Report	
1st edition(NAIST-IS-TR99008)	20 April 1999
2nd edition(NAIST-IS-TR99012)	15 December 1999

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氏，現在キャノン勤務の妙木裕氏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 および茶筌 2.0 β版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克亘氏，ASTEM の山田篤氏に感謝いたします．話し言葉の解析を中心にして辞書の整備に様々な助言をいただいた奈良先端大の伝康晴氏に感謝します．奈良先端大の高林哲氏，工藤拓氏には autoconf，automake 化および RPM パッケージ作成に多くの助言をいただきました．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 12 年 12 月 06 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@is.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://chasen.aist-nara.ac.jp/

1 茶筌の使用法

1.1 インストール手順

1. 'configure' を実行する.

```
% ./configure
```

コンパイラやコンパイルオプションは自動的に設定される．これらを手動で変更したい場合は以下のように実行する.

```
% env CC=cc CFLAGS="-O2 -Wall" ./configure
```

configure の詳しい使用法については INSTALL, INSTALL-ja あるいは './configure --help' の出力を参照のこと.

2. 'make' を実行する.

```
% make
```

茶筌本体の実行ファイルは chasen/chasen に，ライブラリは lib/ に，辞書作成のプログラムは mkchadic/ 以下に作成される．OS 標準の make を使うとコンパイルに失敗することがある．その場合は GNU make を使用する.

3. 'make install' を実行する.

```
% make install
```

バージョン 2.1 からインストール先ディレクトリが変更されており，デフォルトでは以下の場所にインストールされる．PREFIX は ./configure --prefix で指定することができる (デフォルトは /usr/local).

PREFIX/bin/chasen	茶筌の実行ファイル
PREFIX/libexec/chasen/	辞書作成プログラム
PREFIX/lib/libchasen.*	茶筌ライブラリ
PREFIX/include/chasen.h	ヘッダファイル
PREFIX/share/chasen/doc/	マニュアル
PREFIX/share/chasen/prolog/	Prolog 用インタフェース

ただし，以下のものはインストールされない.

chasen/chasen.el	Emacs 用インタフェース
perl/ChaSen.pm	Perl モジュール

なお，バージョン 2.02 以前の茶筌を削除する場合は以下のコマンドを実行すればよい.

```
# rm -rf PREFIX/lib/chasen
```

Windows では，Cygwin 環境があれば同様の方法でコンパイルできる.

バージョン 2.2.2 以降，chasenrc はシステムインストール時にはインストールされなくなった．辞書インストール時に、PREFIX/etc 以下に chasenrc を置く必要がある.

1.2 実行方法

システムの実行ファイルは，'make install' によって PREFIX/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．

- 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．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
私は昨日学校へ行きました.
% chasen temp
私      ワタクシ    私      名詞-代名詞-一般
は      ハ          は      助詞-係助詞
昨日    キノウ      昨日    名詞-副詞可能
学校    ガッコウ    学校    名詞-一般
へ      へ          へ      助詞-格助詞-一般
行き    イキ        行く    動詞-自立          五段・カ行促音便    連用形
まし    マシ        ます    助動詞             特殊・マス          連用形
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
-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
-o file	解析結果出力ファイルを指定
-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
-R	クライアントモード (-D) のときは chasenrc ファイルを読み込まない，スタンドアロンのときはデフォルトの chasenrc ファイル (PREFIX/etc/chasenrc) を読み込む
-L lang	言語を指定
-lp	品詞番号と品詞名のリストを表示
-lt	活用型番号と活用型名のリストを表示
-lf	活用型番号，活用形番号と活用形名のリストを表示
-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，句読点など (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，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

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．

また，出力フォーマットに '-f', '-e', '-c' を指定すると，それぞれ -f, -e, -c と同じ出力形式になる．

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"

- 見出し語のみ

  "%m\n"

- 分かち書き (見出し語を空白で区切って表示)

  "%m␣"

- 漢字かな変換

  "%y"

- ルビつき表示．“漢字 (かな)” の形式で表示する．

  "%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, %y1	読みの第一候補 (出現形)(※ 1)
%Y, %Y1	読みの第一候補 (基本形)(※ 1)
%y0	読み全体 (出現形)
%Y0	読み全体 (基本形)
%a	発音の第一候補 (出現形)
%A	発音の第一候補 (基本形)
%a0	発音全体 (出現形)
%A0	発音全体 (基本形)
%rABC	ルビつきの見出し (“A 漢字 B かな C” と表示)(※ 2)
%i, %i1	付加情報の第一候補
%i0	付加情報全体
%Ic	付加情報 (空文字列か “NIL” なら文字c)(※ 2)
%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)(※ 2)
%t	活用型の番号
%Tc	活用型 (なければ文字c)(※ 2)
%f	活用形の番号
%Fc	活用形 (なければ文字c)(※ 2)
%c	形態素のコスト
%S	解析文全体
%pb	最適パスであれば “*”, そうでなければ “␣”
%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(※ 3)
%?I/STR1/STR2/	付加情報が “NIL” でも “”(空文字列) でもなければSTR1，そうでなければSTR2(※ 3)
%?T/STR1/STR2/	活用があればSTR1，なければSTR2(※ 3)
%?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(※ 3)
%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)(※ 3)
%%	% そのもの

変換文字	機能
.	フィールド幅の指定
-	フィールド幅の指定
1-9	フィールド幅の指定
\n	改行文字
\t	タブ
\\	\ そのもの
\’	’ そのもの
\"	" そのもの

※ **1** ipadic では,「行く (いく/ゆく)」のように形態素が複数の読みを持つ場合，その読みを「{ イ/ユ } ク」のように，半角のブレースとスラッシュを使って表している．通常の読みの出力 (出力フォーマットの %y) では，その第一候補である「イク」が出力され，%y0 を使うと読み全体である「{ イ/ユ } ク」が出力される．

※ **2** A,B,C,c が空白文字の時は何も表示しない．

※ **3** ‘/’ には任意の文字が使える．また，括弧 “(){}[]<>” を用いることもできる．以下に例をあげる．

- %?T#STR1#STR2#
- %?B(STR1)(STR2)
- %?U{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できる．

コマンド	機能
#V	-V オプションと同じ．
#F␣[文字列]	-F オプションと同じ．ただし，フォーマット文字列をクォ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

- コマンドインタプリタは，PATDIC にのみ機能する．日本語版辞書 (ipadic) 以外には作用しない．
- #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．
- ipadic に対して #e を利用する際、活用形の全展開を行なえないバグが報告されている。

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，PREFIX/etc/chasenrc に記述されるが，利用者のホームディレクトリの‘.chasenrc’というファイルに記述することもできる．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 .chasen2rc.
4. 利用者のホームディレクトリにある .chasenrc.
5. PREFIX/etc/chasenrc(デフォルトではインストールされない).

設定項目一覧を以下に示す．このうち,「PATDIC」または「SUFDIC」,「未知語品詞」,「品詞コスト」は必ず指定しなければならない.

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) が存在するディレクトリを指定する.

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

   「文法ファイル」は省略することができ，その場合 chasenrc ファイルがあるディレクトリと同じディレクトリを指定したとみなされる．茶筌に付属の辞書 ipadic1.01 以降の chasenrc ファイルでは「文法ファイル」は省略されている.

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．また，相対パス，つまり “/” で始まらないパスを記述すると，文法ファイルと同じディレクトリを指定したとみなされる．例えば以下のように指定する.

   ```
   (PATDIC chadic
           /home/rikyu/mydic/chadic)
   ```

   上の記述では，以下の二組の辞書ファイルが読み込まれる.

   (a) 文法ファイルと同じディレクトリにある chadic.int, chadic.pat

   (b) /home/rikyu/mydic/ にある chadic.int, chadic.pat

   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.

[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.

```
(SUFDIC chadic)
```

上の記述では，文法ファイルと同じディレクトリにある chadic.int, chadic.ary が読み込まれる．

chadic.ary はデフォルトでは作成されない．辞書のあるディレクトリで ‘make ary’ を実行すると作成される．

(a) /home/rikyu/mydic/ にある chadic.int, chadic.pat

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．コストは正の整数値をとる．

```
(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．上の記述では,「名詞」の形態素のコストは基本的には 2 倍になるが,「名詞-固有名詞」以下に細分類される名詞だけは形態素のコストが 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．省略した場合の重みは 1 である．

```
(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
```

6. **コスト幅**

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．コスト幅におさまるすべての解を出力するには -m オプションや-p オプションを使う．

```
(コスト幅 0)   ; デフォルト値
```

コスト幅は-w オプションでも指定することができる．その場合，-w オプションで指定したものが優先される．

7. **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．デフォルトは 0.

```
(未定義連接コスト 500)
```

8. **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")
```

出力フォーマットは-F オプションでも指定することができる．その場合，-F オプションで指定したものが優先される．詳しくは **??** 節を参照のこと．

9. **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 (何も表示しない).

```
(BOS 文字列 "解析文: [%S]\n")
```

10. **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")
```

11. **空白品詞**

茶筌は，半角の空白文字 (ASCII コード 32) とタブ (ASCII コード 9) を空白とみなし，これらを無視して解析する．通常は，解析結果に空白の情報を出力しないが,「空白品詞」を設定することにより，空白についての情報を出力するようになる．例えば，以下のように設定すると，空白を「記号-空白」として出力する．

```
(空白品詞 (記号 空白))
```

なお，出力フォーマットを “%m” に設定して，空白品詞を指定する (品詞は何でもよい) と，解析文と全く同じ出力が得られることになる．

12. **注釈**

ある文字列で始まりある文字列で終わる文字列を注釈のように扱い，その文字列の部分を無視して解析させることができる．解析結果には，その文字列が一つの形態素として出力される．

chasenrc ファイルには，開始文字列，終了文字列からなるリストと出力時の品詞名あるいはフォーマット文字列を記述する．終了文字列は省略することができ，その場合，開始文字列と一致する文字列自身を注釈として扱う．また，出力時の品詞名あるいはフォーマット文字列を省略するとその形態素についての情報を全く出力しなくなる．

```
(注釈 (("<" ">") "%m\n")
      (("「") (記号 一般))
      (("」") (記号 一般))
      (("\"" "\"") (名詞 引用文字列))
      (("［" "］"))
      )
```

例えば，上のように記述すると，以下のように解析，出力される．

- <img src="cha.gif"> のように “<” で始まり “>” で終わる文字列をそのまま出力する．
- “「” あるいは “」” を「記号-一般」として出力する．
- “"hello(again)"” のようにダブルクォーテーションで囲まれた文字列を「名詞-引用文字列」として出力する．
- “［ちゃせん］” のように “［” で始まり “］” で終わる文字列を無視して解析し，解析結果にはその文字列の情報を出力しない．

13. **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．

```
(連結品詞 ((複合名詞) (名詞) (接頭詞 名詞接続) (接頭詞 数接続))
  ((記号)))
```

例えば，上の記述では以下のように品詞を連結する．

(a) 連続した「名詞」「接頭詞-名詞接続」「接頭詞-数接続」を連結し「複合名詞」として表示する．なお,「複合名詞」は品詞定義ファイル grammar.cha に記述しておく必要がある．

(b) 連続した「記号」を連結し,「記号」として表示する．

14. **複合語出力**

形態素辞書ファイル (.dic) 内で定義した複合語について，複合語全体の形態素情報を出力する (“複合語”) か，複合語を構成する各単語の形態素情報を出力する (“構成語”) かを選択することができる．デフォルトは “複合語”．

```
(複合語出力 "複合語")
```

なお，複合語出力については -Oc, -Os オプションによっても制御することができる．

15. **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 (??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 libchasen.a, libchasen.so 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．ヘッダファイルとして chasen.h をインクルードする．利用できるライブラリ関数・変数は以下の通りである．

```
#include <chasen.h>

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．デフォルトのオプションのままでよければ，この関数を呼び出さずに以下の解析関数を呼び出してもかまわない．

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．fp が NULL のときは何も出力しない．

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/rikyu/.chasenrc.proj", "-j", NULL};
chasen_getopt_argv(option, stderr);
```

```
#include <chasen.h>

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 Perl からの使用

`perl/ChaSen.pm` を使うことにより，perl から茶筌を利用できる．インストール方法，使用方法については perl/README を参照のこと．

4.3 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/share/emacs/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] 研究開発用知的資源タグ付きテキストコーパス報告書 平成９年度, テキストサブワーキンググループ, 技術研究組合 新情報処理開発機構, 1998.

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 2.2 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．

また、上に述べた著作権は茶筌システム本体についてのものであり、ipadic をはじめとする他の辞書については、各辞書についての著作権条項があるためそちらを参照すること．

B　更新履歴

B.1　茶筌 2.0 から 茶筌 2.2 への拡張点

- 辞書とシステムの分離
  他言語の辞書整備により，辞書とシステムを分離した．chasenrc は辞書側が持ち、システムインストール時にはインストールさい。辞書インストール時に `PREFIX/etc` 以下に `chasenrc` をインストールする必要がある。
- autoconf, automake, libtool 化
  `./configure` で，自動的に環境を読み込み設定できるようにした．これに伴い，各辞書をコンパイルする際に必要になる情報を出力するプログラム `chasen-config` を導入した．

B.2　JUMAN 2.0 から 茶筌 2.0 への拡張点

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．ただし，`mkchadic/convdic` を実行することにより，bi-gram 版の辞書を v-gram 版の辞書に変換することができる．

`convdic` は bi-gram 版の辞書があるディレクトリ上で，v-gram 版の辞書を格納するディレクトリを引数として実行する．例えば以下のように実行すると，bi-gram 版の辞書がある `dic` というディレクトリと同じ階層に `dic2` というディレクトリが作成され，その中に v-gram 版の辞書が格納される．なお，`convdic` 実行後，茶筌に付属の `dic/Makefile` を v-gram 版の辞書があるディレクトリ (下の例では `dic2`) にコピーする必要がある．また，chasenrc ファイルも用意する．

```
% cd dic
% ../mkchadic/convdic ../dic2
% cp Makefile ../dic2
```

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 版の実行ファイルが作成され，bi-gram 版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 個の単語 (あるいは品詞) の連接コストだけではなく，3 個以上の任意の長さの単語 (品詞) 列に対して単語 (品詞) の連接コストを記述できる．
3. *.dic で「発音」という属性を使える．出力フォーマットの%a, %A で表示できる．また，cforms.cha で発音の語尾を定義できる．
4. *.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．英語の辞書などで使用する．
5. chasenrc ファイルの「連結品詞」の機能を拡張し，複数の種類の品詞を別々に連結できるようにした．
6. 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．つまり，“EOS” の個数が入力文の行数と一致する．
7. 解析結果のデフォルトの出力形式 (-f) 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た．
8. 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．
9. 形態素辞書ファイル *.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*.dic 中の「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．
10. 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用いる．また，bi-gram 版では形態素コストの範囲が 0〜6553.5(茶筌 1.51 以前は 0〜25.5) であるが，v-gram 版では 0〜65535 である．
11. 連接コスト 0 を「確率 1 で連接する」という意味に，-1 を「連接しない」という意味に変更した．また，連接コストの範囲を -1〜32767 に変更した．
12. 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．

B.3 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. chasenrc の「文法ファイル」を省略できるようにした.「PATDIC」「SUFDIC」が ’/’ で始まっていない場合は,「文法ファイル」のディレクトリからの相対パスとみなすようにした．
2. 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．

3. SUFARY を使って英語を解析できるようにした.
4. -D なしで -R を指定した場合は Makefile で指定した chasenrc (/usr/local/share/chasen/dic/chasenrc など) を読み込むようにした.
5. 文頭・文末で出力する文字列を設定できるようにした.
6. 未知語品詞とそのコストを複数指定できるようにした.
7. chasenrc ファイルで「空白品詞」を指定することにより，空白も解析結果に出力できるようにした.
8. chasenrc ファイルで「注釈」を指定することにより，SGML タグのような特定の文字列を空白と同様に無視して解析できるようにした.
9. -lp, -lt, -lf オプションで品詞や活用のリストを表示できるようにした.
10. -o オプションで出力ファイルを指定できるようにした.
11. 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.
12. 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.
13. chasenrc ファイルで「BOS 文字列」「EOS 文字列」を指定することにより，文頭・文末で出力する文字列を設定できるようにした.
14. 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.
15. 辞書ファイルの形態素コストの範囲を今までの 0〜25.5 から，bi-gram 版は 0〜6553.5 に，v-gram 版は 0〜65535 に変更した.
16. 連接ファイルの連接コストの範囲を 0〜255 から 0〜32767 に変更した.

B.4 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，クライアントの Emacs Lisp 版インタフェースを作成した.
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. 美茶 (ViCha) 用出力オプション -v を新設した.
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.5 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, 2.0 を経て，茶筌 2.2 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) Windows 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