Morphological Analysis System
ChaSen version 2.2.8 Manual

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

Morphological Analysis System ChaSen 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.

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 2.0   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
    version 2.2.4   18 March 2001
    version 2.2.5   24 March 2001
    version 2.2.6   17 April 2001
    version 2.2.7   22 June 2001
    version 2.2.8   23 July 2001
ChaSen for Windows
    version 1.0   29 March 1997
    version 2.0   15 December 1999
NAIST Technical Report (NAIST-IS-TR99009)
    1st edition   20 April 1999
    2nd edition   15 December 1999

Contents

1 Introduction

ChaSen is a morphological analysis system which basically has the following facilities and features.

- It segments Japanese text (sentences) string into morphemes and tags those morphemes with their parts of speech and pronunciations. It also tokenizes conjugative morphemes, i.e., it tags the conjugative morphemes with their base forms and conjugation types/forms.

- In its grammar and dictionaries, morphemes as well as connectivity of two morphemes / parts of speech are defined, where some costs are assigned to their definition.

- In its morphological analysis process, ChaSen sums up those costs of morphemes and their connectivities, then outputs results with the minimum cost.

- Basically, connectivity of two morphemes / parts of speech is defined in the form of their bi-grams. In the case of the current dictionary (ipadic1.0), connectivity of two morphemes / parts of speech and its costs are automatically extracted from a parts-of-speech tagged Japanese newspaper article corpus. In order to tune its costs, part of speech bi-gram Markov model is employed, and the probability parameters of maximum likelihood estimate (MLE) model is transformed into its connectivity costs. Similarly, costs of morphemes are also obtained from the MLE model.

2 Grammar and Dictionaries

	Morpheme Files	Grammar Files
Definition Files	Morpheme Definition Files Morpheme Dictionaries	Grammar Definition Files Parts of speech File Conjugation Types File Conjugation Forms File Connectivity Rules File
System Files	System Dictionaries Index Files	Connectivity Table Connectivity Matrix

Table 1: Grammar/Dictionary Files

As shown in Table 1, grammar and dictionary files of ChaSen system can be classified using two dimensions. According to the first dimension, they can be classified into *Definition Files* and *System Files*. Definition Files include definitions of the grammar and the morphemes of ChaSen, and are automatically compiled into System Files which are used in the morphological analysis. Using the second dimension, they can be classified into *Morpheme Files* and *Grammar Files* according to the linguistic type of the contents of the files.

The description of those grammar and dictionary files is summarized below.

1. **Definition Files**

   (a) Morpheme Definition Files

   - *Morpheme Dictionaries* (`Noun.dic`, etc.)
     define morphemes of each part of speech. A morpheme is defined as a list of its surface form (or its base form if conjugative), pronunciation, conjugation type if conjugative, and semantic information. A surface form cost (to be used in the morphological analysis) can be assigned to each morpheme definition.

   (b) Grammar Definition Files

   - *Parts of speech File* (`grammar.cha`)
     defines the set of parts of speech.
   - *Conjugation Types File* (`ctypes.cha`)
     defines the set of conjugation types for each conjugative part of speech.

- *Conjugation Forms File* (`cforms.cha`)
  defines possible conjugation forms for each conjugation type.
- *Connectivity Rules File* (`connect.cha`)
  defines connectivity of two morphemes / parts of speech in the form of their bi-grams. A connectivity cost has to be assigned to each bi-gram of morphemes / parts of speech.

2. **System Files**

  (a) Morpheme Files

   - *System Dictionaries* (`*.int`)
     is obtained by compiling morpheme dictionaries and encoding morpheme information.
   - *Index Files* (`*.pat`)
     include Patricia tree indices of system dictionaries.

  (b) Grammar Files

   - *Connectivity Table* (`table.cha`)
     defines the correspondence between the rows/columns of the connectivity matrix and the morphemes / parts of speech listed in the connectivity rules file.
   - *Connectivity Matrix* (`matrix.cha`)
     defines connectivity of two morphemes / parts of speech in the form of a matrix.

3 Morphological Analysis

3.1 Algorithm

For the string of the input Japanese sentence, ChaSen consults its morpheme dictionaries and records all the possible morphemes that are any sub-strings of the input string. Next, ChaSen calculates following two types of costs.

Morpheme Cost A cost that is assigned to each morpheme, and is calculated as the product of

- the cost of the corresponding part of speech (defined in the `chasenrc` resource file),
- relative weight of morpheme costs (defined in the `chasenrc` resource file),
- and the surface form cost (defined in the morpheme dictionaries).

Connectivity Cost A cost that is assigned to each bi-gram of morphemes, and is calculated as the product of

- the connectivity cost defined in the connectivity rules file,
- and the relative weight of connectivity costs (defined in the `chasenrc` resource file).

For the string of the input Japanese sentence, every possible segmentation into morpheme sequences and their parts of speech tagging is considered and sum of the above morpheme costs and their connectivity costs are calculated. Then, the results with the minimum cost are returned. Some cost width of beam search is defined in the `chasenrc` resource file, and at every position in the input string, morphological analysis results are pruned using this cost width of beam search.

3.2 Coping with Unknown Words

When ChaSen consults its morpheme dictionaries with some sub-string of the input string and can not find any morphemes, it assumes that the sub-string should be considered as a morpheme and behaves as if the sub-string were contained in its morpheme dictionaries, although the sub-string is assigned an extremely high cost compared with those morphemes existing in its morpheme dictionaries. Details of this facility of coping with unknown words are as follows:

- For hiragana (Japanese), kanji (Chinese), numbers, and symbols character types, ChaSen assumes each one character as a possible unknown morpheme that is not contained in its morpheme dictionaries. On the other hand, for other character types (katakana (foreign), (English) alphabet, etc.), ChaSen assumes the longest string each character of which is of the same character type as a possible unknown morpheme that is not contained in its morpheme dictionaries.

- Those morphemes that are not contained in the morpheme dictionaries are considered as having the *part of speech for unknown words*, which is defined in the `chasenrc` resource file.

- Those morphemes that are not contained in the morpheme dictionaries are assigned the *cost for unknown words*, which is defined in the `chasenrc` resource file.

3.3 Unknown Connectivity Cost

Basically, bi-grams of morphemes that does not match any rules listed in the connectivity rules file are not allowed in the morphological analysis results. However, users can allow those prohibited bi-grams in the morphological analysis by giving them an extremely high cost. This can be done by defining *unknown connectivity cost* in the `chasenrc` resource file (how to define the unknown connectivity cost are described in the next section).

4 Installation

1. Type '`./configure`' to configure the package for your system.

```
% ./configure
```

   You can give '`configure`' initial values for variables by setting them in the environment.

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

   See the file '`INSTALL`' for detail usage of configure.

2. Type '`make`'.

```
% make
```

   This produces the system. You might have to use GNU make.

3. Type '`make install`' to install programs.

```
# make install
```

   This will install the following files. `PREFIX` is defined by `./configure --prefix`. (default setting is `/usr/local`)

`PREFIX/bin/chasen`	ChaSen command
`PREFIX/libexec/chasen/`	programs for building dictionaries
`PREFIX/lib/libchasen.*`	ChaSen libraries
`PREFIX/include/chasen.h`	header files
`PREFIX/share/chasen/doc/`	manuals
`PREFIX/share/chasen/prolog/`	Prolog programs to use ChaSen

   The following files will not be installed.

`chasen/chasen.el`	Emacs lisp to use ChaSen
`perl/ChaSen.pm`	Perl modules to use ChaSen

   To remove old version of ChaSen programs, type the command below:

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

chasenrc is not installed when system is installed. You need to put chasenrc file on PREFIX/etc, when you install a dictionary package.

5 How to Use ChaSen System

5.1 Running ChaSen Program

Suppose a Japanese text file "*nihongo*", which should be encoded in Japanese EUC (Extended UNIX Code) or JIS (ISO-2022-JP). Issue the following command:

> **chasen** *nihongo*

The result of the morphological analysis is shown on the standard output. If your terminal has a direct input facility of Japanese characters, simply type

> **chasen**

then input a Japanese sentence followed by a carrige return.

5.2 Options

There are several options:

- how to run

-s	start ChaSen server
-P *port*	specify ChaSen server's port number (use with **-s**, the default is 31000)
-D *host[:port]*	connect to ChaSen server
-R	with **-D**, do not read chasenrc file,
	without **-D**, read the default chasenrc file
-a	run standalone even if environment variable **CHASENSERVER** is set

- how to print ambiguous results

-b	print one result with the least cost (default)
-m	print ambiguous parts explicitly
-p	print all possible results independently

- output format

-f	print the result in a table like format (default)
-e	print all information of each morpheme separated by a blank
-c	print all information of each morpheme in internal codes
-d	print detailed morpheme data for Prolog.
-v	print detailed morpheme data for ViCha.
-F *format*	print morpheme data with formetted output
-Fh	print help of the format of **-F** option

- miscellaneous

`-j`	Japanese sentence mode
	(assume a punctuation mark as a sentence delimiter)
`-o` *file*	write output to *file*
`-w` *width*	specify the cost width
`-C`	use command mode
`-r` *rc_file*	use *rc_file* as a chasenrc file other than the default
`-L` *lang*	specify the language of the input text
`-lp`	print the list of parts of speech
`-lt`	print the list of conjugation types
`-lf`	print the list of conjugation forms
`-h`	print the help message
`-V`	print ChaSen version number

For example, compare the default output with the results of the following.

> `chasen -m -e` *nihongo*

5.3   ChaSen Server and Client

You can use ChaSen server and its client. First, type

> `chasen -s`

to start ChaSen server.
Type

> `chasen -D`*host nihongo*

('*host*' should be the hostname of ChaSen server) to run ChaSen client.

5.4   Output Format

Notes about `-F` option.

format characters:

%m	surface form (conjugated form)	
%M	surface form (base form)	
%y	first candidate of reading (conjugated form)	
%Y	first candidate of reading (base form)	
%y0	reading (conjugated form)	
%Y0	reading (base form)	
%a	first candidate of pronunciation (conjugated form)	
%A	first candidate of pronunciation (base form)	
%a0	pronunciation (conjugated form)	
%A0	pronunciation (base form)	
%rABC	surface form with ruby	
%i	first candidate of semantic information	
%i0	semantic information	
%Ic	semantic information (if NIL, print character 'c'.)	
%Pc	parts of speech (name) of all the layers of the parts of speech hierarchy, concatenated with the character 'c'	
%Pnc	parts of speech (name) of the layers 1–n of the parts of speech hierarchy, concatenated with the character 'c'	
%h	part of speech (code)	
%H	part of speech (name)	
%Hn	the part of speech (name) at the n-th layer (if NIL, the part of speech at the most specific layer)	
%b	0 (only for the backward compatibility)	
%BB	sub-part of speech (name) (if NIL, print part of speech)	
%Bc	sub-part of speech (name) (if NIL, print character 'c')	
%t	conjugation type (code)	
%Tc	conjugation type (name) (if NIL, print character 'c')	
%f	conjugated form (code)	
%Fc	conjugated form (name) (if NIL, print character 'c')	
%c	cost value of the morpheme	
%S	the input sentence	
%pb	if the best path, "*", otherwise "␣"	
%pi	the index of the path of the output lattice	
%ps	the starting position of the morpheme at the path of the output lattice	
%pe	the ending position of the morpheme at the path of the output lattice	
%pc	the cost of the path of the output lattice	
%ppiC	the indices of the preceding paths, concatenated with the character 'C'	
%ppcC	the costs of the preceding paths, concatenated with the character 'C'	
%?B/STR1/STR2/	if sub-part of speech exists, STR1, otherwise, STR2	
%?I/STR1/STR2/	unless the semantic information is NIL and "", STR1, otherwise, STR2	
%?T/STR1/STR2/	if conjugative, STR1, otherwise, STR2	
%?F/STR1/STR2/	same as %?T/STR1/STR2/	
%?U/STR1/STR2/	if unknown word, STR1, otherwise, STR2	
%U/STR/	if unknown word, " ", otherwise, STR	
%%	'%'	
.	specify the field width	
-	specify the field width	
1-9	specify the field width	
\n	carrige return	
\t	tab	
\\	back slash	
\'	single quotation mark	
\"	double quotation mark	

example:

- same as the default output (-f option)

  "%m\t%y\t%M\t%U(%P-)\t%T␣\t%F␣\n" or "-f"

- surface forms, readings, and parts of speech separated by TAB characters

  `"%m\t%y\t%P-\n"`

- surface forms only

  `"%m\n"`

- surface forms separated by space characters

  `"%m␣"`

- kanji to kana conversion

  `"%y"`

- surface forms with ruby

  `"%r␣()"`

6   `chasenrc` Resource File

The `chasenrc` resource file is used for defining various options necessary for running ChaSen morphological analysis program. As for which `chasenrc` resource file to be used in the morphological analysis process, the following preference order holds.

1. the one given with `-r` option when running ChaSen program.

2. the one given as the environmental variable `CHASENRC`.

3. `.chasenrc` file at the user's home directory.

4. the default file given as the variable `RCPATH` in the top level `Makefile`. Usually `PREFIX/etc/chasenrc`.

The following gives options that are defined in the `chasenrc` resource file, as well as their examples.

1. Directory of grammar files (section 2).

   ```
   (GRAMMAR /usr/local/lib/chasen/dic/ipadic)
   ```

   When this option is omitted, the directory which contains `chasenrc` file is used as the directory of grammar files.

2. System dictionaries (section 2). The suffix '`.int`' has to be omitted. More than one system dictionaries can be used.

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

   In the description above, the following two directories will be read.

   (a) `chadic.int` and `chadic.pat` in the same directory as grammar files.
   (b) `chadic.int` and `chadic.pat` in `/home/rikyu/mydic/`.

   To use a package for string search, SUFARY, use `SUFDIC` option.

   ```
   (SUFDIC chadic)
   ```

   In the description above, `chadic.int` and `chadic.ary` in the same directory as grammar files will be read.

   `chadic.ary` is not created by default. To create it, type '`make ary`' in the same directory as dictionary.

The index file will be read for a shorter time with SUFDIC than PATDIC, but the time for search is longer. You had better use SUFDIC for small sentences, PATDIC for large sentences.

The maximum number of directories is 5 for both PATDIC and SUFDIC. To change this number, edit the valude `MAX_DIC_NUMBER` in `chasen/pat.h` and re-compile.

3. Part of speech for unknown words (section 3.2).

```
(UNKNOWN_POS (              ))              ; a part of speech
(UNKNOWN_POS (              ) (         ))  ; two parts of speech
```

4. Cost of each part of speech (section 3.1).

```
(POS_COST
        ((*)                1)
        ((      )          500)
        ((    )             2)
        ((               )  3)
)
```

5. Relative weights of connectivity and morpheme costs (section 3.1).

```
(CONN_WEIGHT 1)      ; defalut vaule
(MORPH_WEIGHT 1)     ; default value
```

6. Cost width of beam search (section 3.1).

```
(COST_WIDTH 0)     ; default value
```

7. Unknown connectivity cost. (section 3.3).

```
(DEF_CONN_COST 500)
```

8. Output format.

Users can specify the output format. For example, if there is the following line in `.chasenrc`, the surface form, the reading, and the part of speech will be printed.

```
(OUTPUT_FORMAT "%m\t%y\t%P-\n")
```

Note that `-f`, `-e`, `-c`, `-d` and `-F` command line options override the format defined in `.chasenrc`.

9. String for the beginning of the sentence.

```
(BOS_STRING "sentence: [%S]\n")
```

10. String for the end of the sentence.

```
(EOS_STRING "end_of_sentence\n")
```

11. Parts of speech for space and tab characters.

    ChaSen ignores space (ASCII code is 32) and tab (ASCII code is 9) characters in the analysis process, whose information is not output by default. Set 'SPACE_POS' to output those information.

    ```
    (SPACE_POS (          ))
    ```

12. Annotations.

    ChaSen can analyze sentences ignoring the certain strings like annotations, and output the information about each string as one morpheme.

    ```
    (ANNOTATION (("<" ">") "%m\n")
               (("  ") (          ))
               (("  ") (          ))
               (("\"" "\"") (              ))
               (("  " "  "))
               )
    ```

    In the description above, ChaSen will analyze and output in the following way.

    - Output the strings as it is which begin with "<" and end with ">" such as `<img src="cha.gif">`.
    - Output the strings "   " or "   " as   -
    - Output the strings surrounded by double quotations such as ""`hello(again)`"" as   -
      .
    - Analyze ignoring the strings which begin with "   " and end with "   " such as "   " and output no information about the strings.

13. Parts of speech for concatenated morphemes output.

    ChaSen concatenates morphemes of the same part of speech if the part of speech is among those specified for concatenated morphemes output.

    ```
    (COMPOSIT_POS (      ) (    ))
    ```

14. Sentence delimiter characters.

    Users can define sentence delimiter characters that are used when the ChaSen program is called with `-j` option.

    ```
    (DELIMITER␣"          .,!?␣")
    ```

7   ChaSen Library

You can use ChaSen library `$CHASEN/lib/libchasen.a` to put ChaSen's module into other programs.

8   Calling ChaSen from Other Languages

8.1   Emacs Lisp Version of ChaSen Client

Copy `$CHASEN/chasen/chasen.el` to the Emacs Lisp directory to install. Specify hostname and port number of ChaSen server, and describe autoloaded functions in your .emacs.

```
(setq chasen-server-host "kyusu")
(setq chasen-server-port 31234)    ; the default is 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)
```

9   Contact

For further information, send an email to:

chasen@is.aist-nara.ac.jp