ChaSen
Morphological Analyzer version 2.4.0
User's Manual

Yuji Matsumoto and Kazuma Takaoka and Masayuki Asahara

2007/03/19

Morphological Analysis System ChaSen 2.4.0 User's Manual
Yuji Matsumoto, Kazuma Takaoka and Masayuki Asahara
This translation of the ChaSen user's manual was made with support from the non-profit organization GSK by Eric Nichols.

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 2.3.0	16 February 2003
version 2.4.0	30 March 2007
ChaSen for Windows	
version 1.0	29 March 1997
version 2.0	15 December 1999
version 2.4.0	30 March 2007
NAIST Technical Report	
1st edition(NAIST-IS-TR99008)	20 April 1999
2nd edition(NAIST-IS-TR99012)	15 December 1999

目 次

Introduction

In the computational analysis of Japanese, unlike American and European languages, to begin with there are the following two problems. The first is the problem of morphological analysis. With the spread of word processors a big problem in the input of Japanese has gone away, but in computational analysis of Japanese, first the individual morphemes in the input sentence need to be recognized. For this, we need a dictionary as large as can be practically supported, so, at the same time, there is also the problem of how to maintain this dictionary. One more problem is the reality that in Japanese there is no widely accepted or agreed upon grammar or grammatical terminology. In grammars taught in school, in general there are word classifications and grammatical terminologies, however, amongst researchers they are not held in very high regard and are not suitable for computers.

Although morphological analyzers, a tool of foremost necessity in Japanese analysis, have already been developed by many research groups and many technological problems brought to light, there is no common tool in circulation in the world. This is true of machine-readable Japanese dictionaries as well. This system was developed to offer the many reasearchers aiming at computational analysis of Japanese a commonly usable morphological analyzer. Under these circumstances, we took into account the above two problems, and gave special consideration to making it easy for users to change the definition of the grammar and the connective relations between words. This system was developed at a university by a small number of people, and there are still areas that are not perfect, but we plan to make a series of improvements as much as possible. We hope that you will bear this in mind when using ChaSen.

The ChaSen system is based on the Japanese morphological analyzer JUMAN, version 2.0, developed at Nagao Laboratory of Kyoto Univeristy and the Graduate School of Information Science at Nara Institute of Science and Technology. JUMAN was made with the cooperation of many students and the staffs of Kyoto University and NAIST. Also, regarding the dictionary we used the dictionary from the Kana-Kanji conversion system, Wnn, and a Japanese dictionary publically released by ICOT, adding our own modifications. We are especially grateful to Sadao Kurohashi of Tokyo University, with whom we developed JUMAN 2.0, and Yutaka Myo'ki, who is currently working at Canon.

First, we would like to thank Professor Makoto Nagao for creating the opportunity to develop JUMAN. We are also grateful to Takehito Utsuro of Tsukuba University who helped us in many ways with JUMAN's development. Ken-ichi Chinen gave us many suggestions about the ChaSen system's development while he was at NAIST. We received a variety of assistance from Osamu Imaichi, Tomoaki Imamura, Akira Kitauchi while they were at NAIST during the development of ChaSen vesions 1.0 and 2.0 β, and from Tatsuo Yamashita, Yoshitaka Hirano, and Hiroshi Matsuda during the development of versions 2.0 and 2.2. We are extremely grateful to both these groups and all of the other members of Matsumoto Laboratory who helped with ChaSen's development. The "Japanese Speech Dictation Software Development Group," whose representative member is Professor Kiyohiro Shikano of NAIST carried out large-scale mantinence of the IPA POS dictionary. In particular, we would like to thank Katunobu Itou of Housei University and Kaoru Yamada from ASTEM for their assistance. We are grateful to Yasuharu Den of Chiba University for the various pieces of advice about dictionary mantinence focusing on the analysis of spoken language. We also received a lot of advice about converting ChaSen to autoconf/automake and making RPM packages from Tetsu Takabayashi and Taku Kudoh while they were at NAIST. Chooi-Ling Goh, Cheng Yuchang, and Jia Lu helped with the maintenance of the Chinese dictionary. Finally, although there are too many to name individually, we would like to thank all of ChaSen's users for the many comments and questions.

平成 19 年 3 月 30 日

Please send any inquiries regarding ChaSen to the following address.

Computational Linguistics Laboratory Graduate School of Information Science Nara Institute of Science and Technology 8916-5 Takayama, Ikoma, Nara 630-0192, Japan Tel: +81-743-72-5240, Fax: +81-0743-72-5249
E-mail: chasen@is.naist.jp
URL: http://chasen-legacy.sourceforge.jp/

1 Chasen User's Manual

1.1 Installation

1. Install the necessary tools. The following tools are necessary to compile ChaSen.
   - Darts[1] version 0.3 or later
   - libiconv (if not part of your system's standard installation)
2. Run '`configure`'.

```
% ./configure
```

   - When specifying the location of Darts' header files

```
% ./configure --with-darts=/usr/local/include
```

   - When using libiconv

```
% ./configure --with-libiconv=yes
```

   - When specifying the location of libiconv

```
% ./configure --with-libiconv=/usr/local
```

   The compiler and options will be determined automatically.

   For more information on how to use `configure` consult `INSTALL` or the output of '`./configure --help`'.

3. Run '`make`'.

```
% make
```

   ChaSen's executable is created in `chasen/chasen`; the libraries in `mkchadic/`; and the dictionary cration program in `mkchadic/`. Sometimes compilation will fail when using the OS-standard make. In that case, GNUmake should be used.

4. Run '`make install`'.

```
% make install
```

   The installation directory has changed from version 2.1 onward. Now it installs to the locations below by default. `PREFIX` can be specified with `./configure --prefix` (the default is `/usr/local`).

`PREFIX/bin/chasen`	the ChaSen executable
`PREFIX/libexec/chasen/`	dictionary construction programs
`PREFIX/lib/libchasen.*`	the ChaSen libraries
`PREFIX/include/chasen.h`	ChaSen's header file
`PREFIX/share/chasen/doc/`	documentation

[1] `http://cl.aist-nara.ac.jp/%7etaku-ku/software/darts/`

However, the following is not installed.

`perl/ChaSen.pm` Perl module

`chasenrc` is not installed with ChaSen. Instead when the dictionary (ipadic-2.6.0 or above) is installed, `chasenrc`'s path is taken from `chasen-config`, and if there is no `chasenrc` in `PREFIX/etc`, a copy is made automatically. When `PREFIX/etc` already contains a `chasenrc` file, it is not copied and must be manually updated.

1.2 Running ChaSen

The morphological analyzer's executable is installed into `PREFIX/bin/chasen` by the '`makeinstall`' command.

- Running the morphological analyzer

  ChaSen is started by running the `chasen` command in the following manner.

  ```
  % chasen [options] [filename ...]
  ```

  ChaSen reads files from standard input or specified by command line arguments one line at a time and conducts morphological analysis on each sentence.

- Processing details

  ChaSen finds the lowest cost solution (the solution where each morpheme's boundary has a variation from the minimum cost that is within the established cost threshold) and display the results following the formatting options. The meaning of each option is summarized in the next section.

- Example usage

  The input file can be given as arguments to ChaSen. For example:

  ```
  % 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 Runtime Options

ChaSen supports several runtime options. They are summariezed below. For options that take arguments such as `-r`, the argument may be optionally separated from the option with whitespace.

- Display options for ambiguous input (all display methods display in the same format for unambiguous results)

`-b`	Show only the solution with [[the rightmost longest match]] (default)
`-m`	Show multiple morphemes for only the ambiguous areas
`-p`	Expand the ambiguous combinations and show all possible solutions separately

- Display options for individual morphemes

`-f`	Display arranged in columns (default)
`-e`	Show all morphological information by category name
`-c`	Show all morphological information by category code
`-d`	Represent each morpheme as a Prolog compound term and output them as a list
`-v`	Detailed display mode for VisualMorphs
`-F format`	Output in the format specified in the `format` string
`-Fh`	`-F` Display the help for output formatting options

- Other options

`-j`	Treat full stops and empty lines as sentence boundaries
`-o file`	Specify output file
`-w width`	Manually set cost threshold
`-r rc_file`	Use `rc_file` as the `chasenrc`file
`-R`	Read the default `chasenrc` file (`PREFIX/etc/chasenrc`)
`-L lang`	Specify input language
`-lp`	Show a list of POS category codes and their names
`-lt`	Show a list of inflection category codes and their names
`-lf`	Show a list of inflection type (code), inflected form (code), inflected form (name)
`-i`	Select the input encoding (e: EUC-JP, s:Shift_JIS, w:UTF-8, u:UTF-8, a:ISO-8859-1)
`-h`	Show the help message
`-V`	Show the version number
`-s`	Restricted analysis

About the `-j` option Normally ChaSen treats the end of a line as the end as the end of an input sentence. Because of this, when analyzing a file where newlines appear in the middle of a sentence, the correct results are often not obtained.

In these cases adding the `-j` option will cause full stops and other sentence-final punctuation (by default ". 。 ! ?") or empty lines to be used for identifying sentence boundaries.

The characters used to split sentences with the `-j` option can also be specified by setting the "punctuation characters" (区切り文字) value appropriately in `chasenrc`.

1.4 Output Formats

The output format of analysis results can be changed by using the `-F` option or setting the value of "output format" (出力フォーマット) in `chasenrc`.

If there is an '`\n`' at the end of the output format string, a newline will be inserted at the end of each piece of morphological information, and '`EOS`' will be output at the end of each sentence. If there is no '`\n`' at the end of the output format string, then the morphological information for one sentence will be output on one line with a newline at the end.

Also, if the output format string contains '`-f`', '`-e`', or '`-c`,' output will match that of the corresponding option.

Here are some examples of output format string usage.

- Same as default (`-f` option)

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

- Input word, reading, POS delimited by tabs

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

- Only the input word

  `"%m\n"`

- "Wakatigaki mode" (input words divided by spaces)

  `"%m␣"`

- Kanji-kana conversion

  `"%y"`

- Ruby mode: output in the form of "Kanji (kana)"

  `"%r␣()"`

Below we give a list of all output format conversion strings and their meaning.

Conversion string	Function
%m	surface form (conjugated form)
%M	surface form (base form)
%y, %y1	first reading candidate (conjugated form)
%Y, %Y1	first reading candidate (base form)
%y0	all readings (conjugated form)
%Y0	all readings (base form)
%a	first pronunciation candidate (conjugated form)
%A	first pronunciation candidate (base form)
%a0	all pronunciation (conjugated form)
%A0	all pronunciation (base form)
%rABC	surface form with ruby (i.e. "A Kanji B kana C") (※ 1)
%i, %i1	first semantic information candidate
%i0	all semantic information
%Ic	semantic information (if "NIL" print c) (※ 1)
%Pc	part of speech (name) of all layers in the part-of-speech hierarchy, joined together by c
%Pnc	part of speech (name) of first —n— layers (n:1〜9) in the part-of-speech hierarchy, joined together
%h	part of speech (code)
%H	part of speech (name)
%Hn	part of speech (name) at the nth layer (n:1〜9) (or the deepest layer)
%b	0 (only for backwards compatibility)
%BB	sub-part of speech (name) (if "NIL" print POS)
%Bc	sub-part of speech (code) (if "NIL" print c) (※ 1)
%t	conjugated type (code)
%Tc	conjugated type (name) (if "NIL" print c) (※ 1)
%f	conjugated form (code)
%Fc	conjugated form (name) (if "NIL" print c) (※ 1)
%c	cost of morpheme
%S	the input sentence
%pb	"*" if optimal path, "* *" otherwise
%pi	the index of the path of the output lattice
%ps	Starting position of the morpheme in the
%pe	Ending position of path's morphemes +1
%pc	Cost of path
%ppiC	indices of the elements in the preceeding path, joined together by C
%ppcC	costs of the elements in the preceeding path, joined together by C
%?B/STR1/STR2/	STR1 if detailed POS category , STR2otherwise (※ 2)
%?I/STR1/STR2/	STR1 if not the empty string (even if auxiliary information is "NIL"), STR2 otherwise (※ 2)
%?T/STR1/STR2/	STR1 if conjugated, STR2 otherwise (※ 2)
%?F/STR1/STR2/	Same as %?T/STR1/STR2/
%?U/STR1/STR2/	STR1\ if unknown word, STR2 otherwise (※ 2)
%U/STR/	"未知語" if unknown word, STR otherwise (※ 2)
%%	percent sign

Conversion string	Function
.	specifies field width
-	specifies field width
1-9	specifies field width
\n	newline
\t	tab
\\	backslash
\’	single quotation mark
\"	double quotation mark

※ **1** In ipadic, when morphemes have multiple readings as in the case of ”行く (いく/ゆく),” the readings are displayed with half-width braces and back slashes readings like so: ”{ イ/ユ } ク.” In the standard output format (i.e. that of %y), the word’s first reading candidate, ”イク,” is output and with output format %y0, all of the readings, ”{ イ/ユ } ク,” are output.

※ **1** When A,B,C,c are empty strings, nothing is displayed.

※ **2** The string divider “/” can be an arbitrary string. Brackets like “(){}[]<>” are also usable. For example:

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

1.5 Constrained Analysis

”Constrained analysis” refers to a special kind of analysis that satisfies constraints used when the morphological information or boundaries for a portion of the input sentence are already known.

For example, it is possible to analyze the sentence ”にわにはにわにわとりがいる。” specifying that ”はにわ” is a noun, or that ”にわとり” should be treated as a single morpheme. このとき analysis candidates that violate the constraints, like the fourth character ”は” being treated as an independent morpheme, or ”にわとり” getting split into ”にわ” and ”とり”, will be rejected.

Input format The input for constrained analysis is the same as ChaSen’s standard output format, but reading and lemmatization information is ignored. In the following examples, tabs are represented by ⌐

```
にわ\t ニワ\t にわ\tUNSPEC
に
はにわ\t ハニワ\t はにわ\t 名詞-一般
にわとり\t ニワトリ\t にわとり\tUNSPEC
がいる。
EOS
```

Each line consists of a ”segment.” A segment can be one of the following: ”morpheme specification” ”sentence fragment” ”end of sentence” ”comment.”

- morpheme specification

  This segment represents a single morpheme, a unit that will not be split any further.
  Morpheme specification segments have part of speech information from the fourth column onward. The format is the same as ChaSen's standard output.
  If you write "UNSPEC" instead of part of speech information, ChaSen will look up the segment in its dictionary and use the corresponding entry as its results. If there is no entry, the segment will be labeled as an unknown word.

- sentence fragment

  A segment without any part of speech information represents a sentence fragment.
  The contents of this segment will be processed without any contraints. However, no candidates that cross the segment boundaries will be generated.

- end of sentence

  Lines starting with "EOS", "BOS/EOS", or "文末", and lines containing nothing but a newline mark the end of a sentence.

- annotations

  Putting "ANNO" in the part of speech information column will make that segment an annotation.
  Annotations are displayed in ChaSen's output, but they are not used in its analysis.
  The display is determined in chasenrc.

Example analysis An example of restricted analysis is given below.

Input:

```
$ chasen -s
にわ\t ニワ\t にわ\tUNSPEC
に
はにわ\t ハニワ\t はにわ\t 名詞-一般
にわとり\t ニワトリ\t にわとり\tUNSPEC
がいる。
EOS
```

Output:

```
にわ\t\t\t 未知語
に\t ニ\t に\t 助詞-格助詞-一般
はにわ\t ハニワ\はにわ\名詞-一般
にわとり\t ニワトリ\t にわとり\t 名詞-一般
が\t ガ\t が\t 助詞-格助詞-一般
いる\t イル\t いる\t 動詞-自立\t 一段\t 基本形
。\t。\t。\t 記号-句点
EOS
```

Areas of caution in restricted analysis

- During restricted analysis, even if "ANNO" is set, no output will be displayed unless comments are enabled in chasenrc.
- During restricted analysis, whitespace part of speech tagging and whitespace skipping are disabled (this is to support comments).

2 The chasenrc Resource File

The chasenrc resource file is used to define the various necessary options for running the ChaSen morphological analyzer.

These definitions are usually kept in `PREFIX/etc/chasenrc`, but they can also be stored in the file '`.chasenrc`' in the user's home directory.

The chasenrc file can also be specified by an option when chasen is initialized.

The following precendence order wil be used to determine which chasenrc file will be loaded when ChaSen is run.

1. (Unix, Windows) the file specified by the `-r` option at initialization time
2. (Unix, Windows) the file set in the `CHASENRC` environment variable
3. (Windows) The chasenrc set in the registry key `chasenrc` in `HKEY_CURRENT_USER\Software\NAIST\ChaSen`
4. (Unix) the `.chasen2rc` file in the user's home directory
5. (Unix) the file `.chasenrc` in the user's home directory
6. (Unix) `PREFIX/etc/chasenrc` (not installed by default)

A list of settings is given below.
Of these settings, "`DADIC`", "UNKNOWN_POS", and "POS_COST" absolutely must be defined.

1. **The grammar file directory setting**

   This setting specifies the directory where the grammar files (`grammar.cha, ctypes.cha, cforms.cha, connect.cha`) reside.

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

   This setting can be omitted, in which case it is assumed to be the same as the directory that the chasenrc file resides in.

   In the chasenrc file distributed with version 1.01 or later of chasen's dictionary, ipadic, "GRAMMAR" is omitted.

2. **System dictionaries**

   This setting is used to specify double array dictionaries (`chadic.{da,lex,dat}`) omitting the extensions of their file names.

   Multiple dictionary sets may also be specified.

Relative paths, i.e. paths not starting with "/", are assumed to start in the same directory as the grammar files. Here is an example.

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

In the example below, two sets of dictionaries are read in.

(a) `chadic.{da,lex,dat}` in the grammar file directory

(b) `chadic.{da,lex,dat}` in `/home/rikyu/mydic/`

When dictionary lookups are done, both of the above dictionary sets will be used.

[2] .

The setting `DADIC` is used to specify a double array dictionary for Darts.

```
(DADIC chadic)
```

In the above example, `chadic.da`, `chadic.lex`, and `chadic.dat` in the same directory as the grammar files will be read.

The maximum number of usable dictionaries is set to 32.

3. **Unknown word part of speech**

When an unknown word is detected, this setting indicates what part of speech to treat it as while applying ChaSen's connection rules. If multiple parts of speech are given, then the connection rules for each part of speech are applied.

```
(UNKNOWN_POS (名詞 サ変接続))              ; one part of speech
(UNKNOWN_POS (名詞 サ変接続) (名詞 一般))  ; multiple parts of speech
```

4. **Part of speech cost**

The morphological analyzer calculates analysis precidences as costs. When there is ambiguity while analyzing, the result with the lowest total cost is given precidence.

The part of speech cost setting is used to define the magnitude of cost associated with each part of speech as well as set the cost of unknown words. Costs must be integer values.

```
(POS_COST
        ((*)               1) ; any part of speech -- default cost 1x
        ((未知語)          500) ; unknown words -- cost 500x
        ((名詞)            2) ; nouns -- cost 2x
        ((名詞 固有名詞)   3) ; proper nouns -- cost 3x
)
```

When multiple costs are defined for a part of speech, the last cost is given precedence. In the above example, the cost of nouns (名詞) is 2, but the morpheme cost of proper nouns (名詞-固有名詞) increases

[2] The same morpheme cannot be registered in a single dictionary set multiple times, but a given morpheme may appear in multiple dictionary sets. In this case, there will be duplicates of a morpheme.

to 3. The '(*)' setting at the top indicates that the morpheme cost for parts of speech not explicitly defined should be set to 1 (i.e. no change in the total cost of the path). The cost of unknown words is set to 500.

5. **Relative weights of connectivity and morpheme costs**

   The cost in morphological analysis is calculated as the sum of morpheme cost and connectivity cost. This setting lets users assign weights to these two kinds of costs. The cost of an analysis result will be calculated as the sum of each cost multiplied by its weight. If this setting is omitted, it defaults to 1.

```
(CONN_WEIGHT 1)      ; connectivity cost of 1
(MORPH_WEIGHT 1)    ; morpheme cost of 1
```

6. **Cost threshold**

   In the process of morphological analysis, there may be situations where users want to allow all analyses within a beam search cost width. This setting is used to specify a cost width. To ouput all solutions within the cost width, use the `-m` and `-p` options.

```
(COST_WIDTH 0)   ; cost width -- default value
```

   The cost width can also be specified with the `-w` option, overriding the value set in the chasenrc file.

7. **Undefined connectivity cost**

   This setting specifies the connectivity cost for morpheme sequences not defined in the connection rule file. If an undefined connectivity cost is not given, or it is set to 0, then morpheme sequences not in the connection rule file will never be permitted. The default value is 0.

```
(DEF_CONN_COST 500) ; undefined connectivity cost of 500
```

8. **Output format**

   This settings lets users change the output format of ChaSen's results.

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

   The output format can also be specified using the `-F` flag, overriding any value set in chasenrc. For more information on formatting, see Section 1.4.

9. **BOS string**

   The setting specifies the string to display at the beginning of the results for a sentence. Using "`%S`" will display the entire input sentence. The default is the empty string.

```
(BOS_STRING "Input sentence: [%S]\n") ; BOS string is "Input sentence: [%S]"
```

10. **EOS string**

    The setting specifies the string to display at the end of the results for a sentence. Using "`%S`" will display the entire input sentence. The default is "`EOS\n`".

```
(EOS_STRING "END\n") ; EOS string is "END"
```

11. **Whitespace part of speech**

    ChaSen treats the halfspace whitespace character (ASCII code 32) and tab (ASCII 9) as whitespace and ignores them during analysis. Normally whitespace information is not included in ChaSen's output, but this can be changed by using the "SPACE_POS" setting. For example, the setting given below will output "punct-whitespace" for whitespace.

```
(SPACE_POS (punct-whitespace)) ; whitespace part of speech is "punct-whitespace"
```

    Furthermore, by setting the output format to "%m" and specifying a whitespace part of speech, uesrs can get output that is corresponds exactly to the input sentence, whitespace included.

12. **Annotations**

    This setting allows strings that begin and end with a certain sequence to be treated as an annotation and ignored during morphological analysis. In the results, the annotation string will be output as a single morpheme.

    Each annotation definition consists of a list of a start string and stop string followed by optional part of speech information or a formatting string. The stop string can also be omitted, in which case the start string itself will be treated as the annotation. If the part of speech information and format string are omitted, then absolutely no information about the annotation's morpheme will be output.

```
(ANNOTATION (("<" ">") "%m\n") ; output as is
      (("「") (記号 一般)) ; punctuation
      (("」") (記号 一般)) ; punctuation
      (("\"" "\"") (名詞 引用文字列)) ; noun quotation sting
      (("[" "]")) ; nothing will be output
      )
```

    For example, when using the above annotation definition, ChaSen will output its results in the following format.

    - text starting with "¡" and ending with "¿", such as `<img src="cha.gif">`, will be output as is
    - 記号-一般 will be output for "「" and "」"
    - 名詞 引用文字列 will be output for strings in double quotes like "hello (again)"
    - strings enclosed in square brackets like [ChaSen] will be ignored in morphological analysis and no information will be included in its output

13. **Part of speech concatenation**

    This setting is used to concatonate together morphemes of certain parts of speech that appear in succession and output them as a single morpheme.

```
(COMPOSIT_POS ((複合名詞) (名詞) (接頭詞 名詞接続) (接頭詞 数接続))
  ((記号)))
```

For example, with the above declaration of COMPOSIT_POS, parts of speech are concatonated together in the following manner.

(a) Consecutive nouns (名詞), noun prefixes (接頭詞-名詞接続), numeric prefixes (接頭詞-数接続) are concatenated together and displayed as "compound noun (複合名詞)." However, this part of speech must be defined in the part of speech definition file `grammar.cha`.

(b) Consecutive punctuation (記号) is concatenated together, and displayed as "punctuation (記号)."

14. **Compound word output**

ChaSen can be configured to treat compound words defined in the morphological dictionary file in (`.dic`) two different ways.

(a) compound (複合語): the morphological information for the entire compound word is output

(b) compositional (構成語): the compound word is decomposed into individual words, and the morphological information for eachword is output

The default setting is "compound (複合語)."

```
(OUTPUT_COMPOUND "複合語") ; output compound morphological information
```

Compound word output can also be controlled by the `-Oc` and `-Os` options.

15. **Delimiters**

This setting allows users to define the characters that are used as sentence delimiters when the `-j` option is set (see 1.3). Both half-width and full-width characters can be used as delimiters. For example, the following definition treats the full-width characters "。.、,!?" , the half-width characters "`.,!?`␣", and whitespace as sentence delimiters.

```
(DELIMITER "。.、,!?.,!? ")
```

16. **Encodings**

The character encoding that ChaSen supports can be changed by reencoding the morphological file and recompiling ChaSen. The ENCODE setting is used to indicate the encoding that ChaSen will use. For example, the following definition denotes Unicode.

```
(ENCODE "u")
```

The supported encodings are e: EUC-JP, s:Shift_JIS, w:UTF-8, u:UTF-8, a:ISO-8859-1.

3 The ChaSen Libarary

The ChaSen module can be included in other programs using the ChaSen libraries `libchasen.a` and `libchasen.so`. To do so, include the header file `chasen.h`. The following library functions and variables are accessable.

```
#include <chasen.h>
```

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

extern int Cha_optind;
```

Pass ChaSen options. If ChaSen has not been initialized, initialize it before setting the options. If ChaSen's defaults options are acceptable, calling this function can be omitted.

`argv` is an array of NULL-terminated strings containing the command line options for ChaSen. `argv[0]` always contains the program name. When there is an error in the options, an error message is output to the file at file pointer `fp`. No output is produced when `fp` is set to NULL.

When there are no errors in the option settings, 0 is returned. When there is an error, 1 is returned.

The number processed options (including `argv[0]`) is stored in the external variable `Cha_optind`.

The following is a usage example.

In the program `chawan`, the options '`-r /home/rikyu/chasenrc.proj -j`' are passed to ChaSen. After `chasen_getopt_argv()` is called, `Cha_optind` is assigned 4.

```
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);
```

These functions perform morphological analysis on the input. If ChaSen has not been initialized, it is initalized before proceeding. There are 4 functions, differing on whether the input and output are strings or file pointers.

`chasen_fparse()` and `chasen_fparse_tostr()` performs morphological analysis on strings read from a file pointer. When the `-j` option is set in `chasen_getopt_argv()`, ChaSen tokenizes the input sentences with delimiters before parsing.

`chasen_sparse()` and `chasen_sparse_tostr()` perform morphological analysis on the string `str_in`.

`chasen_fparse()` and `chasen_sparse()` output the results of morphological analysis to the file pointer `fp_out`. The return value of these functions is 0.

`chasen_fparse_tostr()` and `chasen_sparse_tostr()` store the results of morphological analysis in ChaSen's internal memory, and return a pointer to the region of memory. This region of memory can be accessed until `chasen_fparse_tostr()` or `chasen_sparse_tostr()` is called again.

4 Using ChaSen from Other Systems

4.1 Using ChaSen from Perl

ChaSen can be called in Perl by using the `perl/ChaSen.pm` Perl module. Consult the perl/README file for information on installation and usage.

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] 山下達雄, 松本裕治: 「形態素解析視覚化システム ViJUMAN version 1.0 使用説明書」, NAIST Technical Report, NAIST-IS-TR96005, 1996.

[5] 山下達雄, 松本裕治: 「形態素解析結果の視覚化システム ViJUMAN とその学習機能」, 情報処理学会研究報告 96-NL-115, pp.29-34, September 1996.

[6] 平野 善隆: 「用言の活用を考慮した韓国語品詞体系の提案とそれを用いた韓国語形態素解析」, 奈良先端科学技術大学院大学修士論文, NAIST-IS-MT9551092, March 1997.

[7] 山下達雄: 「規則と確率モデルの統合による形態素解析」, 奈良先端科学技術大学院大学修士論文, NAIST-IS-MT9551119, March 1997.

[8] 山下達雄, 松本裕治: 「コスト最小法と確率モデルの統合による形態素解析」, 情報処理学会研究報告 96-NL-119, May 1997.

[9] 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.

[10] 「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.

[11] 松田 寛: 「品詞タグ付きコーパス作成支援環境の構築」, 奈良先端科学技術大学院大学修士論文, NAIST-IS-MT9851103, March 1999.

[12] 北内 啓, 宇津呂 武仁, 松本 裕治: 「誤り駆動型の素性選択による日本語形態素解析の確率モデル学習」, 情報処理学会論文誌 Vol. 40, No. 5, p.p.2325-2337, May 1999.

[13] 松田 寛, 桐山 和久, 山田 悟史, 吉野 圭一, 松本裕治: 「部分形態素解析を用いたコーパスの品詞体系変換」, 情報処理学会研究報告 99-NL-134, p.p.23-30, Nov. 1999.

[14] Masayuki Asahara: Extended Statistical Model for Morphological Analysis, 奈良先端科学技術大学院大学修士論文, NAIST-IS-MT9851001, March 2000.

[15] 松田 寛, 松本 裕治: 「品詞タグ付きコーパス作成支援 GUI ツール VisualMorphs」, 情報処理学会研究報告 2000-NL-137, p.98, June, 2000.

[16] 浅原 正幸, 松本 裕治: 「統計的日本語形態素解析に対する拡張 HMM モデル」, 情報処理??会研究報告 2000-NL-137, p.p.39-46, June, 2000.

[17] Masayuki Asahara, Yuji Matsumoto: Extended Models and Tools for High-performance Part-of-Speech Tagger, Proceedings of COLING 2000, July, 2000.

[18] 浅原 正幸, 松本 裕治: 「誤り駆動による統計的品詞タグづけモデルの拡張」, 情報処理学会研究報告 2000-NL-139, p.p.25-32, Sep. 2000.

[19] 松本 裕治: 「形態素解析システム『茶筌』」, 情報処理 Vol.41 No.11, p.p.1208-1214, Nov. 2000.

[20] 伝 康晴, 浅原 正幸: 「リレーショナル・データベースによる統合的言語資源管理環境」, 第 1 回「話し言葉の科学と工学」ワークショップ, Feb. 2001.

[21] 伝 康晴, 宇津呂 武仁, 山田 篤, 浅原 正幸, 松本 裕治: 「話し言葉研究に適した電子化辞書の設計」, 第 2 回「話し言葉の科学と工学」ワークショップ, pp. 39-46, Feb. 2002.

[22] 浅原 正幸, 松本 裕治: 「形態素解析とチャンキングの組み合わせによる日本語テキスト中の未知語出現箇所同定」, 情報処理学会研究報告, 自然言語処理研究会, SIGNL-154, pp.47-54, 2003

[23] 中川 哲治, 工藤 拓, 松本 裕治 : 「Support Vector Machine を用いた形態素解析と修正学習法の提案」, 情報処理学会論文誌, Vol.44, No.5, pp.1354-1367, May 2003.

[24] Taku Kudo, Kaoru Yamamoto, Yuji Matsumoto: "Appliying Conditional Random Fields to Japanese Morphological Analysis", EMNLP-2004, 2004.

[25] 松本裕治, 高岡一馬, 浅原正幸, 工藤拓 : 「茶筌と南瓜による日本語解析–構文情報を用いた文の役割分類」人工知能学会誌, Vol.19, No.3, pp.334-339, 2004.

[26] Chooi-Ling Goh, Masayuki Asahara, and Yuji Matsumoto: "Chinese Word Segmentation by Classification of Characters", International Journal of Computational Linguistics and Chinese Language Processing , Vol.10, No.3, pp.381-396, September, 2005.

[27] Chooi-Ling Goh, Masayuki Asahara, and Yuji Matsumoto: "Training Multi-Classifiers for Chinese Unknown Word Detection" , Journal of Chinese Language and Computing, Vol.15, No.1, pp.1-12, 2005.

[28] ゴーチュイリン, 鄭育昌, 浅原正幸, 松本裕治: 「中国版茶筌の開発」, 言語処理学会第 11 回年次大会発表論文集, pp.245-248, 2005.

[29] 浅原正幸, 高橋由梨加, 松本裕治: 「異表記同語情報を付与した辞書の整備」, 言語処理学会第 11 回年次大会発表論文集, pp.604-607, 2005.

[30] 工藤 拓: 「形態素周辺確率を用いた分かち書きの一般化とその応用」, 言語処理学会第 11 回年次大会発表論文集, 2005.

[31] Chooi-Ling Goh, Masayuki Asahara, and Yuji Matsumoto: "Machine Learning-based Methods to Chinese Unknown Word Detection and POS Tag Guessing", Journal of Chinese Language and Computing, Vol.16, No.4, pp.185-206, 2006.

[32] 東藍, 浅原正幸, 松本裕治: 「条件付確率場による日本語未知語処理」 情報処理学会研究報告, 自然言語処理研究会, SIGNL-173, pp.67-74, 2006.

[33] 東藍, 工藤拓, 浅原正幸, 松本裕治: 「日本語未知語処理のための大規模未解析データの利用法」情報処理学会研究報告, 自然言語処理研究会, SIGNL-179, 2007.

Appendix

A Regarding Copyright and Usage Restrictions

The ChaSen morphological analyzer was developed as free software to widely aid research on natural language processing. ChaSen's copyright is held by Computational Linguistics Laboratory, Graduate School of Information Science, Nara Institute of Science and Technology. There are not any particular restrictions imposed on use and modification of this software, however, the following conditions apply to its redistribution.

B The Connection between JUMAN 3.0 and ChaSen

Since JUMAN 2.0 was relased in July of 1994, Nagao Laboratory at Kyoto University and Matsumoto Laboratory at Nara Institute of Science and Technology have been trying different approaches to its expansion. At Kyoto University, researchers have been working on adding functionality for processing multi-word expressions and parsing bracketed expressions in order to describe connective relations that cannot be represented by existing bi-gram models, and they have produced expanded versions of the grammar files and morphological dictionaries with large-scale updates. At NAIST, anticipating the accumulation of a large amount of tagged Japanese data, we focused on adding functionality for automatically learning connection rules that go beyond bi-grams (including word and part of speech label tagging) and the development of dictionaries that do not depend on the NDMB Unix hash database. The latter improvement aimed at addressing the requests to use the software on operating systems other than UNIX and improve the compilation time and search speed of the dicitonaries.

Because the two approaches to connetivity rules going beyond bi-grams are fairly different, we decided to fork into separate projects, and Kyoto University's expanded version was soon released as JUMAN 3.0 beta in June of 1996.

NAIST's fork was renamed ChaSen and version 1.0 was released in February of 1992. The planed improvements to JUMAN were made through the release of versions 1.5 through 2.3, and with the release of ChaSen 2.4, almost all of the planned features had been added. Development progressed on the following schedule.

1. (ChaSen 1.0) development of system-independent dictionaries (replacement of NDBM with binary trees)
2. (ChaSen 1.0) Refactored and improved performance of system
3. (ChaSen 1.0) Support for undefined connectivity costs ,compound parts of speech , and user definition of output formatting
4. (ChaSen 1.0) Support for JIS encoding
5. (ChaSen 1.0) Definitions for readings of inflectional endings
6. (WinCha 1.0) Support for Windows
7. (ChaSen 1.5) Converted to library
8. (ChaSen 1.5) Converted to server
9. (ChaSen 2.0) Stratification of POS definitions

10. (ChaSen 2.0) Variable length connection rules
11. (ChaSen 2.0) Created a dictionary for words with half-width characters (dictionary using SUFARY)
12. (ChaSen 2.0) Expansion of output formats
13. (ChaSen 2.0) Training of models using variable-length connection costs
14. (ChaSen 2.4) Restricted analysis

C The Future of Morphological Analyzers

A morphological analyser called "MeCab" has been released by Taku Kudoh[3] . MeCab uses a discriminitive training model known as Conditional Random Field , as opposed to the generative Hidden Markov Model used by ChaSen. In [24], the MaCab's model is shown to have better accuracy than ChaSen's. MeCab's other characteristic is it can output "Soft Wakatigaki" [30][4] . In ChaSen's current framework, it is not possible to support new models of analysis and freely design training features like in MeCab.

Recently there have also been various improvements relating to dictionaries. For the new JUMAN dictionary[5] together with the selection of a fundamental lexicon of Japanese, information about orthigraphical variations forms is also being prepared. UniDic, a dictionary developed by Professor Den's group at Chiba University that was recently released, is said to be easy to use not just for natural language processing researchers, but also for researchers in Arts and Humanities and speech processing. At NAIST, we plan to screen the entries in IPADIC and release a Japanese dictionary annotated with information about orthographical variations and compound words. We plan to rename the new dictionary and remove the ICOT entries that were a pending problem for IPADIC. We are also planning to release a dictionary for Chinese morphological analysis with Penn Chinese Treebank part of speech information once issues regarding usage rights have been settled. We have discussed the release of the Chinese morphological analysis dictionary with MeCab's author, Taku Kudoh, and we will simultaneously release versions for use with both ChaSen and MeCab.

One problem that neither JUMAN, ChaSen, or MeCab has addressed is unknown word processing (i.e. the handling of words not in the dictionary). Machine learning models to solve this problem are currently under development at NAIST [32, 33]. Sometime in the future we would like to release a morphological analyzer with a different framework than ChaSen that can support unknown word processing.

[3] http://mecab.sourceforge.net/
[4] http://mecab.sourceforge.net/soft.html
[5] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman.html