長・中単位解析ツール Comainu ver. 0.6
ユーザーズマニュアル

内元 清貴　　小澤 俊介　　伝 康晴

2011 年 6 月 14 日

目 次

1 Comainu とは

Comainu は，音声研究に適した中単位，及び，構文・意味研究に適した長単位を自動構成するツールです．本ツールは以下の機能を持ちます．

長単位解析 平文または短単位列を入力すると，長単位を付与した短単位列を出力することができる．

中単位境界解析 平文または短単位列もしくは長単位情報を付与された短単位列を入力すると，中・長単位を付与した短単位列を出力することができる．

文節境界解析 平文または短単位列を入力すると，文節境界を付与した短単位列を出力することができる．

本文書では，中・長単位解析ツール Comainu について説明します．

2 インストール

2.1 動作環境

Comainu は以下の環境を必要とします．

- MS-Windows:
  OS: MS-Windows NT5.0 以上 (Windows XP, Windows 7 で動作確認)
- UNIX:
  OS: Linux
  Perl: 5.8.8 以上
  Perl/Tk: 804.028 以上

長単位解析には以下が必要です．このうち，CRF++と MIRA については必要に応じてインストールしてください．

- YamCha: Yamcha-0.33 以上
- CRF++: CRF++-0.54 以上
- MIRA: MIRA-0.10 以上

形態素解析を利用する場合は以下も必要となります．

- ChaSen または MeCab: chasen-2.4.1 以上 または mecab-0.98 以上
- UniDic: unidic-1.3.12 以上
- Unidic: unidic-2.1.0 以上

- Unidic-tool: unidic-tool-1.0 以上

また，中単位解析をする場合は以下も必要となります.

- Java runtime: Java 1.6.0 以上
- MSTParser: MSTParser 0.5.0 以上
  MSTParser は Comainu のパッケージに同梱されています.

2.2 インストール手順（MS-Windows）

Comainu-X_XX-win32.exe 及び Comainu-X_XX-model-win32.exe をそれぞれダブルクリックしてインストールプログラムを起動し，指示に従ってプログラムとモデルのインストールを行います.

図 1: セットアップ画面 (MS-Windows 版)

2.3 インストール手順（Unix）

まず，Comainu-X_XX-src.tgz，Comainu-X_XX-model.tgz を展開する.

```
tar zxf Comainu-X_XX-src.tgz
tar zxf Comainu-X_XX-model.tgz
```

次に，トップディレクトリに移動し、configure を実行し，実行環境を設定する。

```
./configure
```

configure では表 1 の項目を設定してください.

表 1: 設定項目（Unix）.

項目	概要	デフォルト
perl	Perl のパス	/usr/bin/perl
java	Java のパス	/uar/bin/java
yamcha-dir	Yamcha のパス	/usr/local/bin
chasen-dir	Chasen のパス	/usr/local/bin
mecab-dir	Mecab のパス	/usr/local/bin
unidic-dir	Unidic のパス	/usr/local/unidic
unidic2-dir	Unidic2 のパス	/usr/local/unidic2
unidic-db	Unidic2 のデータベースのパス	/usr/local/unidic2/share/unidic.db
svm-tool-dir	TinySVM のパス	/usr/local/bin
crf-dir	CRF++のパス	/usr/local/bin
mira-dir	MIRA のパス	/usr/local/bin
mstparser-dir	MST Parser のパス	mstparser

Unix/Linux 環境の場合，以下のように設定します.

```
./configure –perl ”/usr/bin/perl” –java ”/usr/bin/java” \
    –yamcha-dir ”/usr/local/bin” –chasen-dir ”/usr/local/bin” \
    –mecab-dir ”/usr/local/bin” –unidic-dir ”/usr/local/unidic” \
    –unidic2-dir ”/usr/local/unidic2” –svm-tool-dir ”/usr/local/bin” \
    –crf-dir ”/usr/local/bin” –mira-dir ”/usr/local/bin”
```

Cygwin もしくは MSYS/MinGW 環境の場合は以下のように設定します.

```
./configure –perl ”c:/Perl/bin/perl” –java ”c:/usr/bin/java” \
    –yamcha-dir ”c:/yamcha-0.33/bin” \
    –chasen-dir ”c:/Program Files/ChaSen” \
    –mecab-dir ”c:/Program Files/MeCab/bin” \
    –unidic-dir ”c:/Program Files/unidic” \
    –unidic-dir ”c:/Program Files/unidic2” \
    –svm-tool-dir ”c:/TinySVM-0.09/bin” \
    –crf-dir ”c:/CRF++-0.54” –mira-dir ”c:/MIRA-0.10/bin”
```

3 解析 (GUI 版)

3.1 メニュー

ファイルメニュー:

- (F) ファイル

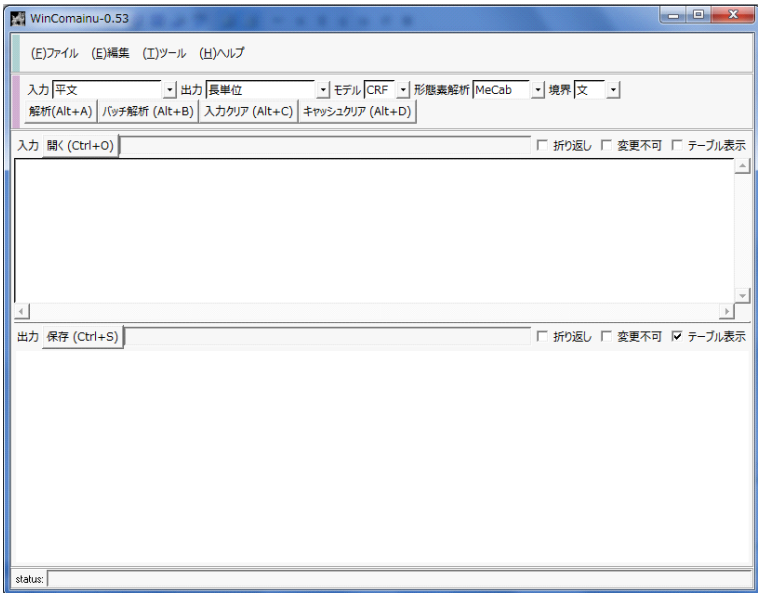

図 2: Comainu(GUI 版)

  - (N) 新しいウィンドウ [Ctrl+N]
    新しいウィンドウを開きます.
  - (O) 開く [Ctrl+O]
    入力ファイルを開いて入力テキストに設定します.
  - (S) 名前を付けて保存 ... [Ctrl+S]
    出力テキストを指定した出力ファイルに保存します.
  - (C) 閉じる [Ctrl+W]
    ウィンドウを閉じます.
  - (X) 終了 [Ctrl+Q]
    終了します.

- (F) 編集
  - (U) 元に戻す [Ctrl+Z]
    テキストエリアの変更を元に戻します.
  - (R) やり直す [Ctrl+Y]
    テキストエリアの変更を元に戻します.
  - (X) 切り取り [Ctrl+X]
    テキストエリアの選択範囲を切り取ります.
  - (C) コピー [Ctrl+C]
    テキストエリアの選択範囲をコピーします.

- (V) 貼り付け [Ctrl+V]
  テキストエリアにコピーを貼り付けます.
- (A) すべて選択 [Ctrl+A]
  テキストエリアの内容をすべて選択します.

- (T) ツール
  - (I) 入力
    入力種別を選択します.
  - (O) 出力
    出力種別を選択します.
  - (M) モデル
    モデル種別を選択します.
  - (T) 形態素解析
    形態素解析種別を選択します.
  - (K) 境界
    境界を選択します.（文または単語）
  - (A) 解析 [Alt+A]
    入力テキストを解析し，出力テキストに結果を表示します.
  - (B) バッチ解析 [Alt+B]
    バッチ解析ダイアログを開きます.
  - (C) 入力クリア [Alt+C]
    入力テキスト，出力テキストをクリアします.
  - (D) キャッシュクリア [Alt+D]
    キャッシュをクリアします.
  - (O) 設定 [Alt+O]
    設定ダイアログを開きます.

- (H) ヘルプ
  - (H) ヘルプ [F1]
    ヘルプを表示します.
  - (A) WinComainu について
    WinComainu の情報を表示します.

ツールバー：

- [入力] コンボボックス
  入力種別を選択します.

- [出力] コンボボックス
  出力種別を選択します.
- [モデル] コンボボックス
  モデル種別を選択します.
- [形態素解析] コンボボックス
  形態素解析種別を選択します.
- [境界] コンボボックス
  文境界または単語境界を選択します.
- [解析] ボタン
  入力テキストを解析し，出力テキストに結果を表示します.
- [バッチ解析] ボタンバッチ解析ダイアログを開きます.
- [入力クリア] ボタン
  入力テキスト，出力テキストをクリアします.
- [キャッシュクリア] ボタン
  キャッシュをクリアします.

入力テキストペイン：入力ファイルと入力テキストを表示します.

- [開く] ボタン
  入力ファイルを開いて入力テキストに設定します.
- [折り返し] チェックボタン
  入力テキストエリアを折り返します.
- [変更不可] チェックボタン
  入力テキストエリアを変更不可にします.

出力テキストペイン：出力ファイルと出力テキストを表示します.

- [保存] ボタン
  出力テキストを指定したファイルに保存します.
- [折り返し] チェックボタン
  出力テキストエリアを折り返します.
- [変更不可] チェックボタン
  出力テキストエリアを変更不可にします.

3.2 解析手順

解析手順は以下の通りである.

1. 下記の中から入力形式を選択する. ([入力] コンボボックス)
   平文, **BCCWJ**, **BCCWJ (長単位情報付き)**, **KC**, **KC (長単位情報付き)**
   ファイルの形式については 7 章を参照.

2. 下記の中からを解析タイプを選択する. ([出力] コンボボックス)

   文節 : 文節境界解析を行う.

   長単位 (境界のみ) : 長単位解析 (境界のみ) を行う.

   長単位 : 長単位解析を行う.

   長単位・文節 : 長単位解析を行い, その解析結果に基づき文節解析を行う[1].

   中単位 : 中単位解析を行う. 入力形式が「平文」「BCCWJ」の場合は長単位解析を行った後, 長単位解析をする.

   長単位・中単位・文節 : 中・長単位解析, 及び, 文節境界解析を行う[1].

3. 長単位解析モデル (SVM, CRF, MIRA) を選択する. ([モデル] コンボボックス)

4. 入力形式で「平文」を選択した場合, 形態素解析器 (Chasen, Mecab) を選択する. ([形態素解析] コンボボックス)

5. 解析タイプに「長単位」を選択した場合, 境界 (文, 単語) を選択する. ([境界] コンボボックス)
   「単語」を選択した場合, 入力中の長単位境界情報を利用して, 長単位解析を行う.「単語」は長単位境界情報が既知である場合に利用できる.

6. 解析するファイルを入力する, もしくは, 入力テキストペインに直接入力する.

7. [解析] ボタンを押して, 解析を開始する.
   [バッチ解析] ボタンを押した場合, 指定したディレクトリ以下に含まれる入力ファイルを全て解析する[1].

8. 解析終了後, 図 3 のように, 出力テキストペインに出力結果が出力される.
   [保存] ボタンを押すことにより, 出力をファイルに保存できる.「バッチ解析」の場合は自動的にファイルに出力されます.

---

[1] 長単位の自動解析結果に基づいて文節境界解析を行うため,「**文節**」を選択した場合の文節境界解析とは結果が異なります.

[1] 入力形式が「KC」もしくは「KC(長単位情報付き)」の場合は拡張子が ".KC" のファイル, それ以外の場合は拡張子が ".txt" のファイルを解析する.

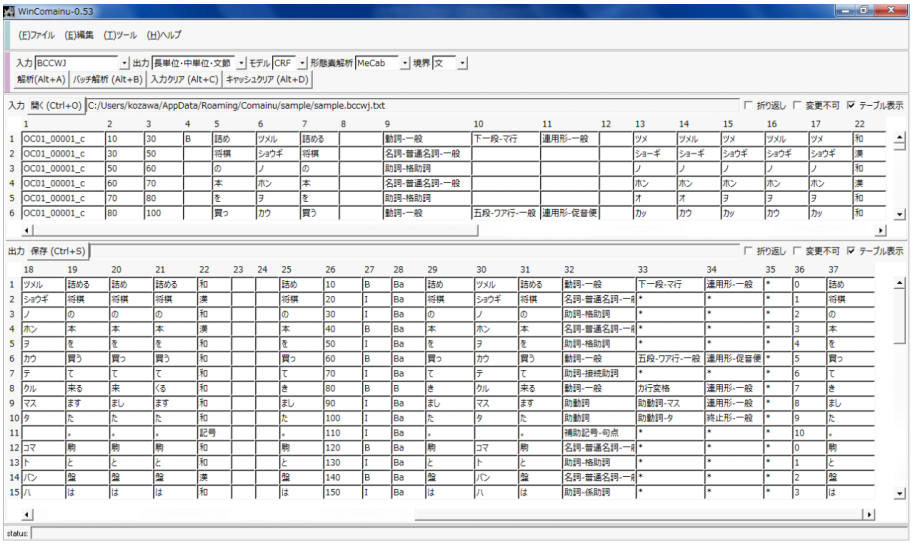

図 3: 出力例 (GUI 版)

4 解析 (CUI 版)

4.1 長単位解析

長単位解析モデル ⟨long-model-file⟩ と長単位学習ファイル ⟨long-train-kc⟩ を用いて解析ファイル ⟨test-bccwj⟩ (⟨test-kc⟩, ⟨test-file⟩) を長単位解析し，その結果をディレクトリ ⟨out-dir⟩ に出力します．長単位学習ファイルはファイル自体が存在する必要はありません．解析ファイルの形式には BCCWJ，KC，平文の 3 種類があります．ファイル形式については 7 章を参照してください．

入力が BCCWJ 形式の場合

```
./script/comainu.pl bccwj2longout ⟨long-train-kc⟩ ⟨test-bccwj⟩ \
    ⟨long-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl bccwj2longout train.KC sample/sample.bccwj.txt \
        train/SVM/train.KC.model out
```

入力が KC 形式の場合

```
./script/comainu.pl kc2longout ⟨long-train-kc⟩ ⟨test-kc⟩ \
    ⟨long-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kc2longout train.KC sample/sample.KC \
        train/CRF/train.KC.model out
```

入力が平文の場合

./script/comainu.pl plain2longout ⟨long-train-kc⟩ ⟨test-file⟩ \
⟨long-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl plain2longout train.KC sample/plain/sample.txt \
train/CRF/train.KC.model out

長単位解析モデルはデフォルトでは CRF を利用します．SVM や MIRA を利用する場合は以下のように --luwmodel で指定してください．

./script/comainu.pl bccwj2longout **--luwmodel SVM** ⟨long-train-kc⟩ \
⟨test-bccwj⟩ ⟨long-model-file⟩ ⟨out-dir⟩

また，形態素解析はデフォルトでは MeCab を利用します．Chasen を利用する場合は以下のように --suwmodel で指定してください．

./script/comainu.pl plain2longout **--suwmodel chasen** ⟨long-train-kc⟩ \
⟨test-file⟩ ⟨long-model-file⟩ ⟨out-dir⟩

長単位境界が既知で，長単位品詞情報のみを解析したい場合は，以下のように --boundary で word を指定します．

./script/comainu.pl bccwj2longout **--boundary word** ⟨long-train-kc⟩ \
⟨test-bccwj⟩ ⟨long-model-file⟩ ⟨out-dir⟩

また，長単位の境界情報のみを出力したい場合は，以下のように --luwmrph で without を指定します．

./script/comainu.pl bccwj2longout **--luwmrph without** ⟨long-train-kc⟩ \
⟨test-bccwj⟩ ⟨long-model-file⟩ ⟨out-dir⟩

4.2 文節解析

文節境界解析モデル ⟨bnst-model-file⟩ を用いて解析ファイル ⟨test-bccwj⟩ (⟨test-kc⟩, ⟨test-file⟩) を文節境界解析し，その結果をディレクトリ ⟨out-dir⟩ に出力します．解析ファイルの形式には BCCWJ，KC，平文の 3 種類があります．

入力が BCCWJ 形式の場合

./script/comainu.pl bccwj2bnstout ⟨test-bccwj⟩ ⟨bnst-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl bccwj2bnstout sample/sample.bccwj.txt \
train/bnst.model out

入力が KC 形式の場合

./script/comainu.pl kc2bnstout ⟨test-kc⟩ ⟨bnst-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kc2bnstout sample/sample.KC train/bnst.model out

入力が平文の場合

./script/comainu.pl plain2longout ⟨test-file⟩ ⟨bnst-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl plain2longout sample/plain/sample.txt \
train/bnst.model out

4.3 中単位解析

中単位解析モデル ⟨mid-model-file⟩ を用いて解析ファイル ⟨test-bccwj⟩ (⟨test-kc⟩) を中単位解析し，その結果をディレクトリ ⟨out-dir⟩ に出力します．解析ファイルの形式は BCCWJ(長単位情報付き) もしくは KC(長単位情報付き) の 2 種類です．

入力が BCCWJ(長単位情報付き) 形式の場合

./script/comainu.pl bccwjlong2midout ⟨test-bccwj⟩ ⟨mid-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl bccwjlong2midout sample/sample.bccwj.txt \
train/MST/train.KC.model out

入力が KC(長単位情報付き) 形式の場合

./script/comainu.pl kclong2midout ⟨test-kc⟩ ⟨mid-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kclong2midout sample/sample.KC \
train/MST/train.KC.model out

4.4 長単位・文節境界解析

長単位解析モデル ⟨long-model-file⟩ と長単位学習ファイル ⟨long-train-kc⟩，文節境界解析モデル ⟨bnst-model-file⟩ を用いて解析ファイル ⟨test-bccwj⟩ (⟨test-file⟩) を長単位・文節境界解析し，その結果をディレクトリ ⟨out-dir⟩ に出力します[2]．長単位学習ファイルはファイル自体が存在する必要はありません．解析ファイルの形式は BCCWJ もしくは平文の 2 種類です．

[2] 文節境界は長単位の自動解析結果に基づいて解析されるため，4.2 節の文節境界解析の結果とは異なる場合があります．

入力が BCCWJ 形式の場合

```
./script/comainu.pl bccwj2longbnstout ⟨long-train-kc⟩ ⟨test-bccwj⟩ \
    ⟨long-model-file⟩ ⟨bnst-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl bccwj2longbnstout train.KC \
        sample/sample.bccwj.txt train/CRF/train.KC.model \
        train/bnst.model out
```

入力が平文の場合

```
./script/comainu.pl plain2longbnstout ⟨long-train-kc⟩ ⟨test-file⟩ \
    ⟨long-model-file⟩ ⟨bnst-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl plain2longbnstout train.KC \
        sample/plain/sample.txt train/CRF/train.KC.model \
        train/bnst.model out
```

4.5 中・長単位解析

長単位解析モデル ⟨long-model-file⟩ と長単位学習ファイル ⟨long-train-kc⟩，中単位解析モデル ⟨mid-model-file⟩ を用いて解析ファイル ⟨test-bccwj⟩ (⟨test-kc⟩，⟨test-file⟩) を中・長単位解析し，その結果をディレクトリ ⟨out-dir⟩ に出力します．長単位学習ファイルはファイル自体が存在する必要はありません．解析ファイルの形式には BCCWJ，KC，平文の 3 種類があります．

入力が BCCWJ 形式の場合

```
./script/comainu.pl bccwj2midout ⟨long-train-kc⟩ ⟨test-bccwj⟩ \
    ⟨long-model-file⟩ ⟨mid-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl bccwj2midout train.KC \
        sample/sample.bccwj.txt train/CRF/train.KC.model \
        train/MST/train.KC.model out
```

入力が KC 形式の場合

```
./script/comainu.pl kc2midout ⟨long-train-kc⟩ ⟨test-kc⟩ \
    ⟨long-model-file⟩ ⟨mid-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kc2midout train.KC sample/sample.KC \
        train/CRF/train.KC.model train/MST/train.KC.model out
```

入力が平文の場合

./script/comainu.pl plain2midout ⟨long-train-kc⟩ ⟨test-file⟩ \
⟨long-model-file⟩ ⟨mid-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl plain2midout train.KC sample/plain/sample.txt \
train/CRF/train.KC.model train/MST/train.KC.model out

4.6 中・長単位・文節境界解析

長単位解析モデル ⟨long-model-file⟩ と長単位学習ファイル ⟨long-train-kc⟩，中単位解析モデル ⟨mid-model-file⟩，文節境界解析モデル ⟨bnst-model-file⟩ を用いて解析ファイル ⟨test-bccwj⟩ (⟨test-kc⟩，⟨test-file⟩) を中・長単位・文節境界解析し，その結果をディレクトリ ⟨out-dir⟩ に出力します[3]．長単位学習ファイルはファイル自体が存在する必要はありません．解析ファイルの形式は BCCWJ もしくは平文の 2 種類があります.

入力が BCCWJ 形式の場合

./script/comainu.pl bccwj2midbnstout ⟨long-train-kc⟩ ⟨test-bccwj⟩ \
⟨long-model-file⟩ ⟨mid-model-file⟩ ⟨bnst-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl bccwj2midbnstout train.KC \
sample/sample.bccwj.txt train/CRF/train.KC.model \
train/MST/train.KC.model train/bnst.model out

入力が平文の場合

./script/comainu.pl plain2midbnstout ⟨long-train-kc⟩ ⟨test-file⟩ \
⟨long-model-file⟩ ⟨mid-model-file⟩ ⟨bnst-model-file⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl plain2midbnstout train.KC \
sample/plain/sample.txt train/CRF/train.KC.model \
train/MST/train.KC.model train/bnst.model out

[3] 文節境界は長単位の自動解析結果に基づいて解析されるため，4.2 節の文節境界解析の結果とは異なる場合があります.

5　モデルの学習

解析に用いるモデルを学習データから学習します．学習は CUI 版のみで利用できます．また，学習ファイルの形式は KC もしくは KC(長単位情報付き) のみとなります．

5.1　長単位解析モデルの学習

長単位学習ファイル ⟨long-train-kc⟩ を学習し，モデルをディレクトリ ⟨out-dir⟩ に出力します．

```
./script/comainu.pl kc2longmodel ⟨long-train-kc⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kc2longmodel sample/sample.KC trainCRF
```

長単位解析モデルとして SVM または MIRA を利用する場合は以下のように --luwmodel にて指定します．

```
./script/comainu.pl kc2longmodel --luwmodel MIRA \
    ⟨long-train-kc⟩ ⟨out-dir⟩
```

5.2　文節解析モデルの学習

文節境界学習ファイル ⟨bnst-train-kc⟩ を学習し，モデルをディレクトリ ⟨out-dir⟩ に出力します．

```
./script/comainu.pl kc2bnstmodel ⟨bnst-train-kc⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kc2bnstmodel sample/sample.KC trainBnst
```

5.3　中単位解析モデルの学習

中単位学習ファイル ⟨mid-train-kc⟩ を学習し，モデルをディレクトリ ⟨out-dir⟩ に出力します．

```
./script/comainu.pl kclong2midmodel ⟨mid-train-kc⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kc2midmodel sample/sample.KC trainMST
```

6　評価

解析結果を参照データと比較することにより評価します．評価は CUI 版のみで利用できます．ファイル形式は KC もしくは KC(長単位情報付き) のみとなります．

6.1　長単位解析結果の評価

正解データ ⟨ref-kc⟩ と長単位解析結果 ⟨kc-lout⟩ を比較し，その結果をディレクトリ ⟨out-dir⟩ に出力する

```
./script/comainu.pl kc2longeval ⟨ref-kc⟩ ⟨kc-lout⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kc2longeval sample/sample.KC \
        out/sample.KC.lout out
```

6.2　文節解析結果の評価

正解データ ⟨ref-kc⟩ と文節境界解析結果 ⟨kc-bout⟩ を比較し，その結果をディレクトリ ⟨out-dir⟩ に出力する

```
./script/comainu.pl kc2bnsteval ⟨ref-kc⟩ ⟨kc-bout⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kc2bnsteval sample/sample.KC \
        out/sample.KC.bout out
```

6.3　中単位解析結果の評価

正解データ ⟨ref-kc⟩ と中単位解析結果 ⟨kc-mout⟩ を比較し，その結果をディレクトリ ⟨out-dir⟩ に出力する

```
./script/comainu.pl kclong2mideval ⟨ref-kc⟩ ⟨kc-mout⟩ ⟨out-dir⟩
ex.) ./script/comainu.pl kc2mideval sample/sample.KC \
        out/sample.KC.mout out
```

7　ファイル形式

Comainu の入出力に用いる項目の一覧を表 2 に示す．以降では，この項目番号を利用して入出力形式を説明する．

表 2: 入出力項目一覧.

項目番号	項目名	概要
1	file	ファイル名
2	start	短単位 start
3	end	短単位 end
4	BOS	文境界
5	orthToken	短単位書字形
6	reading	短単位語彙素読み
7	lemma	短単位語彙素
8	meaning	短単位語義
9	pos	短単位品詞
10	cType	短単位活用型
11	cForm	短単位活用形
12	usage	短単位用法
13	pronToken	短単位発音形
14	pronBase	短単位発音形基本形
15	kana	短単位仮名形
16	kanaBase	短単位仮名形基本形
17	form	語形
18	formBase	語形基本形
19	formOrthBase	語形代表表記
20	formOrth	語形代表表記出現形
21	orthBase	出現形終止形
22	wType	短単位語種
23	charEncloserOpen	丸付き数字 1
24	charEncloserClose	丸付き数字 2
25	originalText	短単位オリジナルテキスト
26	order	長単位 order
27	BOB	文節境界
28	LUW	長単位境界
29	l_orthToken	長単位書字形
30	l_reading	長単位語彙素読み
31	l_lemma	長単位語彙素
32	l_pos	長単位品詞
33	l_cType	長単位活用型
34	l_cForm	長単位活用形
35	depend	短単位間の係り受け情報
36	MID	中単位 ID
37	m_orthToken	中単位書字形

7.1 BCCWJ

BCCWJ 形式では，表 2 の 1〜25 までの項目を入力とし，1〜34 までの項目を出力する．以下の項目を**タブ**区切りしたものが入出力形式となる (入力例ではスペース区切りになっています).

入力 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

出力 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34

入力例 (BCCWJ)

```
OC01_00001_c 10 30 B 詰め ツメル 詰める * 動詞-一般 下一段-マ行 連用形-一般 * \
ツメ ツメル ツメ ツメル ツメ ツメル 詰める 詰め 詰める 和 * * 詰め
OC01_00001_c 30 50 * 将棋 ショウギ 将棋 * 名詞-普通名詞-一般 * * * \
ショーギ ショーギ ショウギ ショウギ ショウギ ショウギ 将棋 将棋 将棋 漢 * * 将棋
```

ただし，長単位境界情報を利用して長単位解析をする場合は 1〜28 までの項目を入力とする．

7.2 BCCWJ(長単位情報付き)

BCCWJ(長単位情報付き) 形式では，表 2 の 1〜34 までの項目を入力とし，1〜37 までの項目を出力する．以下の項目を**タブ**区切りしたものが入出力形式となる．

入力 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34

出力 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37

7.3 KC

KC 形式では，以下の項目を**スペース**区切りしたものが入出力となる．ただし，評価を行う場合は，入力形式を出力形式と同じものにする．また，文境界は「EOS」，文節境界は「*B」で表現する．

入力 5 6 7 9 10 11 17 18 19 20 21 23 24 22

出力 5 6 7 9 10 11 17 18 19 20 21 23 24 22 32 33 34 30 31 29

入力例 (KC)

```
*B
詰め ツメル 詰める 動詞-一般 下一段-マ行 連用形-一般 ツメ ツメル 詰める 詰め * * 和
将棋 ショウギ 将棋 名詞-普通名詞-一般 * * ショウギ ショウギ 将棋 将棋 * * 漢
の ノ の 助詞-格助詞 * * ノ ノ の の * * 和
*B
本 ホン 本 名詞-普通名詞-一般 * * ホン ホン 本 本 * * 漢
を ヲ を 助詞-格助詞 * * ヲ ヲ を を * * 和
*B
買っ カウ 買う 動詞-一般 五段-ワア行-一般 連用形-促音便 カッ カウ 買う 買っ * * 和
て テ て 助詞-接続助詞 * * テ テ て て * * 和
*B
き クル 来る 動詞-非自立可能 カ行変格 連用形-一般 キ クル 来る 来 * * 和
まし マス ます 助動詞 助動詞-マス 連用形-一般 マシ マス ます まし * * 和
た タ た 助動詞 助動詞-タ 終止形-一般 タ タ た た * * 和
。 * 。 補助記号-句点 * * * * 。 。 * * 記号
EOS
```

7.4 KC(長単位情報付き)

KC(長単位情報付き) 形式では，以下の項目を**スペース**区切りしたものが入出力となる．ただし，評価を行う場合は，入力形式を出力形式と同じものにする．また，文境界は「EOS」，文節境界は「*B」で表現する．

入力 5 6 7 9 10 11 17 18 19 20 21 23 24 22 32 33 34 30 31 29

出力 5 6 7 9 10 11 17 18 19 20 21 23 24 22 32 33 34 30 31 29 35 36 37

7.5 平文

平文を入力して解析を行うと，以下の形式 (**タブ**区切り) で出力される．文節境界解析の場合，文節境界に「*B」が付与される．

出力 (長単位解析) 4 5 13 6 7 9 10 11 22 32 33 34 30 31 29

出力 (文節境界解析) 4 5 13 6 7 9 10 11

出力 (中単位解析) 4 5 13 6 7 9 10 11 22 32 33 34 30 31 29 35 36 37

7.6 設定ファイル

設定ファイル (インストールディレクトリ/etc/data_format.conf) を編集することにより，入力形式を変更することができます[4].

[4]出力形式の変更はできません

A　コマンドラインの関数・引数一覧

表 3: 関数一覧.

関数名	入力	出力
plain2bnstout	平文	文節
plain2longout	平文	長単位/長単位境界
plain2longbnstout	平文	長単位/長単位境界，文節
plain2midout	平文	長単位/長単位境界，中単位
plain2midbnstout	平文	長単位/長単位境界，中単位，文節
bccwj2bnstout	BCCWJ	文節
bccwj2longout	BCCWJ	長単位/長単位境界
bccwj2longbnstout	BCCWJ	長単位/長単位境界，文節
bccwj2midout	BCCWJ	長単位/長単位境界，中単位
bccwj2midbnstout	BCCWJ	長単位/長単位境界，中単位，文節
bccwjlong2midout	BCCWJ(長単位情報付き)	中単位
kc2bnstmodel	KC	文節解析モデル
kc2bnstout	KC	文節
kc2bnsteval	KC	文節解析の評価結果
kc2longmodel	KC	長単位解析モデル
kc2longout	KC	長単位/長単位境界
kc2longeval	KC	長単位解析の評価結果
kclong2midmodel	KC(長単位情報付き)	中単位解析モデル
kclong2midout	KC(長単位情報付き)	中単位
kclong2mideval	KC(長単位情報付き)	中単位解析の評価結果

表 4: 引数一覧.

引数名	概要
help	ヘルプを表示します
debug	デバッグモードで実行します
version	Comainu のバージョン情報を表示します
help-method	Comainu の関数のヘルプを表示します
list-method	Comainu の関数リストを表示します
force	ツールのパスをチェックせずに実行します
boundary	word を指定すると，長単位境界情報を用いて解析します.
luwmrph	長単位解析時に長単位品詞情報を出力するかを指定します. 出力する場合は with(default)，しない場合は without
suwmodel	形態素解析器 (mecab, chasen) を指定します.
luwmodel	長単位解析モデル (CRF, SVM, MIRA) を指定します.
perl	perl のパスを指定します.
java	java のパスを指定します.
comainu-home	Comainu のディレクトリのパスを指定します.
yamcha-dir	Yamcha のパスを指定します.
chasen-dir	Chasen のパスを指定します.
mecab-dir	MeCab のパスを指定します.
unidic-dir	Unidic のパスを指定します.
unidic2-dir	Unidic2 のパスを指定します.
unidic-db	Unidic2 のデータベースファイルのパスを指定します.
svm-tool-dir	TinySVM のパスを指定します.
crf-dir	CRF++のパスを指定します.
mira-dir	MIRA のパスを指定します.
mstparser-dir	MSTParser のパスを指定します.
comainu-svm-bip-model	長単位品詞解析モデルのパスを指定します.
comainu-output	出力ディレクトリのパスを指定します.
comainu-temp	一時ファイルの保存先ディレクトリのパスを指定します.