Fumi Package Documentation

対応 NYSOL バージョン: Ver. 1.2, 2.0

revise history:
October 6, 2014 : first release

2014 年 10 月 6 日

目次

第 1 章

はじめに

1.1　概要

本「Fumi（文）」パッケージは、日本語のテキストマイニングに関する複数のコマンドから構成される。

データマイニングはこれまで、数値情報を中心とした定量的・定型的なデータを主に扱ってきた。しかし近年、コンピュータの性能向上やデータ分析技術の発展などにより、非定型的な情報を扱うことも可能となってきた。非定型情報の代表格が、「人が書いた文章」（自然言語）である。

本パッケージを用いると、日本語の形態素解析・構文解析を容易に実行することができる。その結果は CSV として出力されるので、M コマンドによる各種の処理ができるため、さまざまな分析モデルに投入することが可能となる。

なお本パッケージでは、京都大学情報学研究科 黒橋・河原研究室が開発する形態素解析システム JUMAN、構文解析システム KNP を用いている。JUMAN および KNP についての詳細は、以下の公式ページを参照のこと。

- 形態素解析システム JUMAN http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN
- 構文解析システム KNP http://nlp.ist.i.kyoto-u.ac.jp/index.php?KNP

1.2 インストール

本パッケージは全て nysol パッケージに含まれている。詳しくは nysol パッケージのインストールの説明 (http://www.nysol.jp/install) を参照のこと。

ただし、本パッケージに含まれるコマンドの実行には、形態素解析システム JUMAN、構文解析システム KNP を別途インストールしておく必要がある。JUMAN と KNP それぞれの公式ページからアーカイブをダウンロードし、それに含まれる `README` ファイルもしくは `INSTALL` ファイルを参考にインストールしておくこと。

本パッケージが前提としている JUMAN のバージョンは 7.0 以上、KNP のバージョンは 4.0 以上である。

1.3　JUMAN および KNP について

形態素解析システム JUMAN および構文解析システム KNP は、いずれも京都大学 大学院情報学研究科 知能情報学専攻知能メディア講座 言語メディア分野 黒橋・河原研究室 (http://nlp.ist.i.kyoto-u.ac.jp/) がその著作権を有している。取り扱いについては、以下を参照のこと。

1.3.1　JUMAN

1.3.2　KNP

第 2 章

テキストマイニングコマンド

2.1　mjuman.rb JUMAN による形態素解析

テキストファイルで与えられた文書ファイルを JUMAN で形態素解析し、その結果を CSV で出力する。複数の文書ファイルをまとめて指定できるほか、OS のマルチタスク機能を用いた並列処理も可能である。

2.1.1　書式

```
mjuman.rb I= O= [P=] [mp=] [log=] [-mcmdenv] [--help]
```

```
I=          : 文書ファイルが格納されたパス名
O=          : 解析結果の CSV ファイルを格納するパス名
P=          : JUMAN が直接出力する結果を格納するパス名 (省略時は出力しない)
mp=         : 並列処理の数。デフォルト値は 2
log=        : JUMAN が出力するエラーログを格納するファイル名
-mcmdenv    : 内部で利用している MCMD のメッセージ出力レベルを環境変数に任せる。
            : 省略時は警告とエラーメッセージのみ出力 (KG_VerboseLevel=2)。
--help      : ヘルプメッセージの表示
```

入力ファイル例

テキストファイルは、1 行 1 文にしておくことが望ましい。文字コードは UTF-8 である必要がある。

```
子どもはリンゴがすきです。
望遠鏡で泳ぐ少女を見た。
```

出力ファイル例

形態素解析の結果は、CSV として出力される。

```
aid,sid,tid,word,orgWord,daiWord,yomi,class1,class2,class3,class4,annotation
test.txt,0,0, 子ども, 子ども, 子供, こども, 名詞, 普通名詞, , , 代表表記:子供/こども カテゴリ:人
test.txt,0,1, は, は, , は, 助詞, 副助詞, , ,
test.txt,0,2, リンゴ, リンゴ, 林檎, りんご, 名詞, 普通名詞, , , 代表表記:林檎/りんご カテゴリ:植物
:
```

CSV の内容は JUMAN の出力に依拠するが、各項目の意味は以下の通りである。

```
aid         : 入力ファイル名
sid         : 行番号（センテンス ID）
tid         : 形態素番号（トークン ID）
word        : 語（原形）
orgWord     : 文中での表記
daiWord     : 代表表記
yomi        : 読み
class1      : 品詞（レベル 1）
class2      : 品詞（レベル 2）
class3      : 品詞（レベル 3）
class4      : 品詞（レベル 4）
annotation  : 意味情報
```

2.1.2 利用例

例 1: **基本例**

text ディレクトリに文書ファイル test.txt を置き、形態素解析を実行する。結果は csv ディレクトリに出力する。

```
$ more text/test.txt
子どもはリンゴがすきです。
望遠鏡で泳ぐ少女を見た。
$ mjuman.rb I=text O=csv
#MSG# KNP: reading text/test.txt
#MSG# JUMAN: MP-2 aid=test.txt sid=0 (sentences:1/2, articles:1/1)
#MSG# JUMAN: MP-2 aid=test.txt sid=1 (sentences:2/2, articles:1/1)
#MSG# JUM2CSV 1/1
#MSG# Elapse: 0.048sec, # of sentences=2, # of articles=1
#MSG#   0.024sec/sentence, 0.048sec/article
#MSG#   mpCount=2, poolSize=1000
#MSG#   maxLen=512Byte, maxSec=30sec, sizeLimit=2000MB
#END# /Users/maegawa/.rvm/rubies/ruby-2.0.0-p247/bin/mjuman.rb I=text O=csv
$ more csv/test.txt
aid,sid,tid,word,orgWord,daiWord,yomi,class1,class2,class3,class4,annotation
test.txt,0,0,子ども,子ども,子供,こども,名詞,普通名詞,,,代表表記:子供/こども カテゴリ:人
test.txt,0,1,は,は,,は,助詞,副助詞,,,
test.txt,0,2,リンゴ,リンゴ,林檎,りんご,名詞,普通名詞,,,代表表記:林檎/りんご カテゴリ:植物
test.txt,0,3,が,が,,が,助詞,格助詞,,,
test.txt,0,4,すきだ,すきです,好きだ,すきです,形容詞,,ナ形容詞,デス列基本形,代表表記:好きだ/
test.txt,0,5,。,。,,。,特殊,句点,,,
test.txt,1,0,望遠,望遠,望遠,ぼうえん,名詞,普通名詞,,,代表表記:望遠/ぼうえん カテゴリ:抽象物
test.txt,1,1,鏡,鏡,鏡,かがみ,名詞,普通名詞,,,代表表記:鏡/かがみ 漢字読み:訓 カテゴリ:人工物-
test.txt,1,2,で,で,,で,助詞,格助詞,,,
test.txt,1,3,泳ぐ,泳ぐ,泳ぐ,およぐ,動詞,,子音動詞ガ行,基本形,代表表記:泳ぐ/およぐ
test.txt,1,4,少女,少女,少女,しょうじょ,名詞,普通名詞,,,代表表記:少女/しょうじょ カテゴリ:人
test.txt,1,5,を,を,,を,助詞,格助詞,,,
test.txt,1,6,見る,見た,見る,みた,動詞,,母音動詞,タ形,代表表記:見る/みる 補文ト 自他動詞:自:
test.txt,1,7,。,。,,。,特殊,句点,,,
```

例 2: JUMAN **の結果（オリジナル）も出力する例**

JUMAN の結果（オリジナル）も juman ディレクトリに出力しておく。

```
$ more text/test.txt
子どもはリンゴがすきです。
望遠鏡で泳ぐ少女を見た。
$ mjuman.rb I=text O=csv P=juman
#MSG# KNP: reading text/test.txt
#MSG# JUMAN: MP-2 aid=test.txt sid=0 (sentences:1/2, articles:1/1)
#MSG# JUMAN: MP-2 aid=test.txt sid=1 (sentences:2/2, articles:1/1)
#MSG# JUM2CSV 1/1
#MSG# Elapse: 0.054sec, # of sentences=2, # of articles=1
#MSG#   0.027sec/sentence, 0.054sec/article
#MSG#   mpCount=2, poolSize=1000
#MSG#   maxLen=512Byte, maxSec=30sec, sizeLimit=2000MB
#END# /Users/maegawa/.rvm/rubies/ruby-2.0.0-p247/bin/mjuman.rb I=text O=csv P=juman
$ more juman/test.txt
子ども こども 子ども 名詞 6 普通名詞 1 * 0 * 0 "代表表記:子供/こども カテゴリ:人"
は は は 助詞 9 副助詞 2 * 0 * 0 NIL
リンゴ りんご リンゴ 名詞 6 普通名詞 1 * 0 * 0 "代表表記:林檎/りんご カテゴリ:植物
が が が 助詞 9 格助詞 1 * 0 * 0 NIL
すきです すきです すきだ 形容詞 3 * 0 ナ形容詞 21 デス列基本形 29 "代表表記:好きだ/すきだ 反
。 。 。 特殊 1 句点 1 * 0 * 0 NIL
EOS
望遠 ぼうえん 望遠 名詞 6 普通名詞 1 * 0 * 0 "代表表記:望遠/ぼうえん カテゴリ:抽象物"
```

```
鏡 かがみ 鏡 名詞 6 普通名詞 1 * 0 * 0 "代表表記:鏡/かがみ 漢字読み:訓 カテゴリ:人工物-その
で で で 助詞 9 格助詞 1 * 0 * 0 NIL
泳ぐ およぐ 泳ぐ 動詞 2 * 0 子音動詞ガ行 4 基本形 2 "代表表記:泳ぐ/およぐ"
少女 しょうじょ 少女 名詞 6 普通名詞 1 * 0 * 0 "代表表記:少女/しょうじょ カテゴリ:人"
を を を 助詞 9 格助詞 1 * 0 * 0 NIL
見た みた 見る 動詞 2 * 0 母音動詞 1 タ形 10 "代表表記:見る/みる 補文ト 自他動詞:自:見える/
。 。 。 特殊 1 句点 1 * 0 * 0 NIL
EOS
```

2.2 mknp.rb KNP による構文解析

テキストファイルで与えられた文書ファイルを KNP で構文解析し、その結果を XML 構造で出力する。複数の文書ファイルをまとめて指定できるほか、OS のマルチタスク機能を用いた並列処理も可能である。

2.2.1 書式

```
mknp.rb I= O= [P=] [mp=] [log=] [-mcmdenv] [--help]
```

I=	: 文書ファイルが格納されたパス名
O=	: 解析結果の XML ファイルを格納するパス名
P=	: KNP が直接出力する結果を格納するパス名 (省略時は出力しない)
mp=	: 並列処理の数。デフォルト値は 2
log=	: KNP が出力するエラーログを格納するファイル名
-mcmdenv	: 内部で利用している MCMD のメッセージ出力レベルを環境変数に任せる。
	: 省略時は警告とエラーメッセージのみ出力 (KG_VerboseLevel=2)。

入力ファイル例

テキストファイルは、1 行 1 文にしておくことが望ましい。文字コードは UTF-8 である必要がある。

```
子どもはリンゴがすきです。
望遠鏡で泳ぐ少女を見た。
```

出力ファイル例

構文解析の結果は、XML として出力される。

```
<?xml version='1.0' encoding='UTF-8'?>
<article id='test.txt'>
 <sentence id='0' text='子どもはリンゴがすきです。'>
  <chunk id='0' link='2' phraseType='格助詞句' caseType='ガ２格' phrase='子供' phraseTok='子
   <token id='0' class1='名詞' class2='普通名詞' word='子ども' orgWord='子ども' daiWord='子供
   <token id='1' class1='助詞' class2='副助詞' word='は' orgWord='は'/>
  </chunk>
  <chunk id='1' link='2' phraseType='格助詞句' caseType='ガ格' phrase='林檎' phraseTok='リン
   <token id='2' class1='名詞' class2='普通名詞' word='リンゴ' orgWord='リンゴ' daiWord='林檎
   <token id='3' class1='助詞' class2='格助詞' word='が' orgWord='が'/>
  </chunk>
  <chunk id='2' link='-1' phraseType='用言句' phraseTok='すきだ' rawPhrase='すきです。' phrase
   <token id='4' class1='形容詞' class3='ナ形容詞' class4='デス列基本形' word='すきだ' orgWord
   <token id='5' class1='特殊' class2='句点' word='。' orgWord='。'/>
  </chunk>
 </sentence>
:
```

この XML から係り受け関係を抽出するには、mcaseframe.rb コマンドを用いる。

2.2.2 利用例

例 1: 基本例

text ディレクトリに文書ファイル test.txt を置き、構文解析を実行する。結果は xml ディレクトリに出力する。

```
$ more text/test.txt
子どもはリンゴがすきです。
```

```
望遠鏡で泳ぐ少女を見た。
$ mknp.rb I=text O=xml
#MSG# KNP: reading text/test.txt
#MSG# KNP: MP-2 aid=test.txt sid=0 (sentences:1/2, articles:1/1)
#MSG# KNP: MP-2 aid=test.txt sid=1 (sentences:2/2, articles:1/1)
#MSG# KNP2XML 1/1
#MSG# Elapse: 0.149sec, # of sentences=2, # of articles=1
#MSG#   0.075sec/sentence, 0.149sec/article
#MSG#   mpCount=2, poolSize=1000
#MSG#   maxLen=512Byte, maxSec=30sec, sizeLimit=2000MB
#END# /Users/maegawa/.rvm/rubies/ruby-2.0.0-p247/bin/mknp.rb I=text O=xml
$ head -n20 xml/test.txt
<?xml version='1.0' encoding='UTF-8'?>
<article id='test.txt'>
  <sentence id='0' text='子どもはリンゴがすきです。'>
    <chunk id='0' link='2' phraseType='格助詞句' caseType='ガ２格' phrase='子供' phraseTok='
      <token id='0' class1='名詞' class2='普通名詞' word='子ども' orgWord='子ども' daiWord='
      <token id='1' class1='助詞' class2='副助詞' word='は' orgWord='は'/>
    </chunk>
    <chunk id='1' link='2' phraseType='格助詞句' caseType='ガ格' phrase='林檎' phraseTok='リ
      <token id='2' class1='名詞' class2='普通名詞' word='リンゴ' orgWord='リンゴ' daiWord='
      <token id='3' class1='助詞' class2='格助詞' word='が' orgWord='が'/>
    </chunk>
    <chunk id='2' link='-1' phraseType='用言句' phraseTok='すきだ' rawPhrase='すきです。' ph
      <token id='4' class1='形容詞' class3='ナ形容詞' class4='デス列基本形' word='すきだ' or
      <token id='5' class1='特殊' class2='句点' word='。' orgWord='。'/>
    </chunk>
  </sentence>
  <sentence id='1' text='望遠鏡で泳ぐ少女を見た。'>
    <chunk id='0' link='3' phraseType='格助詞句' caseType='デ格' phrase='望遠鏡' phraseTok='
      <token id='0' class1='名詞' class2='普通名詞' word='望遠' orgWord='望遠' daiWord='望遠
      <token id='1' class1='名詞' class2='普通名詞' word='鏡' orgWord='鏡' daiWord='鏡' cate
```

例 2: KNP **の結果（オリジナル）も出力する例**

KNP の結果（オリジナル）も knp ディレクトリに出力しておく。

```
$ more text/test.txt
子どもはリンゴがすきです。
望遠鏡で泳ぐ少女を見た。
$ mknp.rb I=text O=xml P=knp
#MSG# KNP: reading text/test.txt
#MSG# KNP: MP-2 aid=test.txt sid=0 (sentences:1/2, articles:1/1)
#MSG# KNP: MP-2 aid=test.txt sid=1 (sentences:2/2, articles:1/1)
#MSG# KNP2XML 1/1
#MSG# Elapse: 0.147sec, # of sentences=2, # of articles=1
#MSG#   0.074sec/sentence, 0.147sec/article
#MSG#   mpCount=2, poolSize=1000
#MSG#   maxLen=512Byte, maxSec=30sec, sizeLimit=2000MB
#END# /Users/maegawa/.rvm/rubies/ruby-2.0.0-p247/bin/mknp.rb I=text O=xml P=knp
$ head knp/test.txt
# S-ID:1 KNP:4.11-CF1.1 DATE:2014/07/28 SCORE:-21.86138
* 2D <文頭><SM-主体><SM-人><ハ><助詞><体言><係:未格><提題><区切:3-5><主題表現><格要素><連用
+ 2D <文頭><SM-主体><SM-人><ハ><助詞><体言><係:未格><提題><区切:3-5><主題表現><格要素><連用
子ども こども 子ども 名詞 6 普通名詞 1 * 0 * 0 "代表表記:子供/こども カテゴリ:人" <代表表記:
は は は 助詞 9 副助詞 2 * 0 * 0 NIL <かな漢字><ひらがな><付属>
* 2D <ガ><助詞><体言><係:ガ格><区切:0-0><格要素><連用要素><正規化代表表記:林檎/りんご><主辞
+ 2D <ガ><助詞><体言><係:ガ格><区切:0-0><格要素><連用要素><名詞項候補><先行詞候補><正規化代
リンゴ りんご リンゴ 名詞 6 普通名詞 1 * 0 * 0 "代表表記:林檎/りんご カテゴリ:植物
が が が 助詞 9 格助詞 1 * 0 * 0 NIL <かな漢字><ひらがな><付属>
* -1D <文末><句点><用言:形><レベル:C><区切:5-5><ID:（文末）><係:文末><提題受:30><主節><格要
```

2.3 mcaseframe.rb 格フレームの抽出

KNP の解析結果から、格フレームを抽出し出力する。

格フレームとは、用言とそれに係る格助詞句からなる組をいい、「リンゴ（が）」＋「好き」、「望遠鏡（で）」＋「見る」のように表現される。本コマンドは、mknp.rb コマンドが出力した構文解析結果（XML）を読み込み、格フレームを抽出して CSV に出力する。

2.3.1 書式

```
mcaseframe.rb I= o= [-key] [-mcmdenv] [--help]
```

I=	: mknp.rb で parsing した結果 xml ファイルが格納されたパス名
o=	: 出力する格フレームファイル名
-key	: key 型フォーマットで出力する。
-mcmdenv	: 内部で利用している MCMD のメッセージ出力レベルを環境変数に任せる。 省略時は警告とエラーメッセージのみ出力 (KG_VerboseLevel=2)。
--help	: ヘルプメッセージの表示

格フレームの抽出

mknp.rb コマンドの出力する XML は次のようになっている（抜粋）。

```
<sentence id='0' text='子どもはリンゴがすきです。'>
 <chunk id='0' link='2' phraseType='格助詞句' caseType='ガ２格' phrase='子供' phraseTok='子
  <token id='0' class1='名詞' class2='普通名詞' word='子ども' orgWord='子ども' daiWord='子供
  <token id='1' class1='助詞' class2='副助詞' word='は' orgWord='は'/>
 </chunk>
 <chunk id='1' link='2' phraseType='格助詞句' caseType='ガ格' phrase='林檎' phraseTok='リン
  <token id='2' class1='名詞' class2='普通名詞' word='リンゴ' orgWord='リンゴ' daiWord='林檎
  <token id='3' class1='助詞' class2='格助詞' word='が' orgWord='が'/>
 </chunk>
 <chunk id='2' link='-1' phraseType='用言句' phraseTok='すきだ' rawPhrase='すきです。' phrase
  <token id='4' class1='形容詞' class3='ナ形容詞' class4='デス列基本形' word='すきだ' orgWord
  <token id='5' class1='特殊' class2='句点' word='。' orgWord='。'/>
 </chunk>
</sentence>
```

上の例だと、chunk id='0'「子どもは」は link='2' に、chunk id='1'「リンゴが」も link='2' になっており、いずれも chunk id='2'「すきです」に係っていることがわかる。図にすると次のような係り受け関係である。

```
子どもは──┐
リンゴが──┤
      すきです。
```

本コマンドを利用すると、係り受け関係は次のような CSV として抽出される。

```
aid,sid,cid,contrastConj,denial,declinableWord,lid,caseWord,case
test.txt,0,2,,,すきだ,0,子ども,ガ２
test.txt,0,2,,,すきだ,1,リンゴ,ガ
```

CSV の各項目の意味を以下に示す。

aid : 入力ファイル名
sid : 行番号（センテンス ID）
cid : チャンク ID
contrastConj : 逆接接続詞
denial : 否定語を伴うチャンクのとき 1
declinableWord : 用言句
lid : 格助詞句のチャンク ID
caseWord : 格助詞句
case : 格助詞句の種類

2.3.2 利用例

例 1: 基本例

前節の解説で用いてる例。1 行が 1 つの格フレームとなっている。

```
$ more xml/test.txt
<?xml version='1.0' encoding='UTF-8'?>
<article id='test.txt'>
  <sentence id='0' text='子どもはリンゴがすきです。'>
    <chunk id='0' link='2' phraseType='格助詞句' caseType='ガ２格' phrase='子供' phraseTok='
      <token id='0' class1='名詞' class2='普通名詞' word='子ども' orgWord='子ども' daiWord='
      <token id='1' class1='助詞' class2='副助詞' word='は' orgWord='は'/>
    </chunk>
    <chunk id='1' link='2' phraseType='格助詞句' caseType='ガ格' phrase='林檎' phraseTok='リ
      <token id='2' class1='名詞' class2='普通名詞' word='リンゴ' orgWord='リンゴ' daiWord='
      <token id='3' class1='助詞' class2='格助詞' word='が' orgWord='が'/>
    </chunk>
    <chunk id='2' link='-1' phraseType='用言句' phraseTok='すきだ' rawPhrase='すきです。' ph
      <token id='4' class1='形容詞' class3='ナ形容詞' class4='デス列基本形' word='すきだ' or
      <token id='5' class1='特殊' class2='句点' word='。' orgWord='。'/>
    </chunk>
  </sentence>
  <sentence id='1' text='望遠鏡で泳ぐ少女を見た。'>
    <chunk id='0' link='3' phraseType='格助詞句' caseType='デ格' phrase='望遠鏡' phraseTok='
      <token id='0' class1='名詞' class2='普通名詞' word='望遠' orgWord='望遠' daiWord='望遠
      <token id='1' class1='名詞' class2='普通名詞' word='鏡' orgWord='鏡' daiWord='鏡' cate
      <token id='2' class1='助詞' class2='格助詞' word='で' orgWord='で'/>
    </chunk>
    <chunk id='1' link='2' phraseType='用言句' phrase='泳ぐ' phraseTok='泳ぐ' rawPhrase='泳
      <token id='3' class1='動詞' class3='子音動詞ガ行' class4='基本形' word='泳ぐ' orgWord=
    </chunk>
    <chunk id='2' link='3' phraseType='格助詞句' caseType='ヲ格' phrase='少女' phraseTok='少
      <token id='4' class1='名詞' class2='普通名詞' word='少女' orgWord='少女' daiWord='少女
      <token id='5' class1='助詞' class2='格助詞' word='を' orgWord='を'/>
    </chunk>
    <chunk id='3' link='-1' phraseType='用言句' phraseTok='見る' rawPhrase='見た。' phrase='
      <token id='6' class1='動詞' class3='母音動詞' class4='タ形' word='見る' orgWord='見た'
      <token id='7' class1='特殊' class2='句点' word='。' orgWord='。'/>
    </chunk>
  </sentence>
</article>mcaseframe.rb I=xml o=caseframe.csv
#END# /Users/maegawa/.rvm/rubies/ruby-2.0.0-p247/bin/mcaseframe.rb I=xml o=caseframe.csv
more caseframe.csv
aid,sid,cid,contrastConj,denial,declinableWord,lid,caseWord,case
test.txt,0,2,,,すきだ,0,子ども,ガ２
test.txt,0,2,,,すきだ,1,リンゴ,ガ
test.txt,1,3,,,見る,0,望遠鏡,デ
test.txt,1,3,,,見る,2,少女,ヲ
```

例 2: key **型フォーマットによる出力**

-key オプションを付加して実行すると、用言と、その用言に係る格助詞句が行に展開されて出力される。

```
$ mcaseframe.rb -key I=xml o=caseframe2.csv
#END# /Users/maegawa/.rvm/rubies/ruby-2.0.0-p247/bin/mcaseframe.rb -key I=xml o=caseframe2.c
$ more caseframe2.csv
aid,sid,cid,contrastConj,denial,lid,word,type
test.txt,0,2,,,2, すきだ, 用言
test.txt,0,2,,,0, 子ども, ガ２
test.txt,0,2,,,1, リンゴ, ガ
test.txt,1,1,,,1, 泳ぐ, 用言
test.txt,1,3,,,3, 見る, 用言
test.txt,1,3,,,0, 望遠鏡, デ
test.txt,1,3,,,2, 少女, ヲ
```

2.4　mnewdic.rb コーパスからの隣接単語ペア候補出力

コーパス（大量の文例からなるテキストファイル）から、辞書に登録すべき隣接単語ペアの候補情報を出力する。

JUMAN が標準で搭載する辞書は一般的なもののため、特定分野のテキストをマイニングするには、1 つの語として解釈（形態素解析）して欲しい語が複数の語に分割されてしまうことがある。mnewdic.rb コマンドは与えられたコーパスを分析し、並んで出現する頻度が高く 1 つの語として解釈すべき可能性が高い語のペアをリストアップして CSV に出力する。

この CSV を参考に登録の要否を人目で判断し、mjumandic.rb コマンドに与えることで JUMAN への辞書登録が容易になる。

2.4.1　書式

```
mnewdic.rb [i=] [O=] [S=] [n=] [seed=] [-dai] [-mcmdenv] [--help]
```

i=	: コーパスファイル名
O=	: 出力ディレクトリ名
S=	: 単語ペア出現件数最小値
n=	: 単語ペアごとに出力する文例数
seed=	: 乱数の種
-dai	: 見出し語として代表表記を使う
-mcmdenv	: 内部で利用している MCMD のメッセージ出力レベルを環境変数に任せる。 省略時は警告とエラーメッセージのみ出力 (KG_VerboseLevel=2)。
--help	: ヘルプメッセージの表示

入力ファイル例

i=パラメータで指定するコーパスファイルには、1 行 1 文にしたテキストファイルを与える。

```
3 年ぶりにウォークマンを買ったけど、育休中はあまり活躍の余地がないですね。
待機児童解消の方がいい気がするけど。
:
```

出力ファイル例

O=パラメータで指定したディレクトリには、words.csv ファイルと corpus.csv ファイルが出力される（nkf コマンドがインストールされている場合、両ファイルの文字コードを Shift JIS に変換したファイルも同時に出力される）。

```
見出し語, 品詞, 読み, カテゴリ, ドメイン,pid,word1,word2,freq
職場復帰,,,,,0, 職場, 復帰,31
授業参観,,,,,1, 授業, 参観,28
会議参加,,,,,2, 会議, 参加,26
:
```

words.csv ファイルの各項目の意味を以下に示す。

見出し語 : 見出し語
品詞 : 品詞
読み : 読み
カテゴリ : カテゴリ
ドメイン : ドメイン
pid : pid
word1 : 語 1
word2 : 語 2
freq : 出現頻度

corpus.csv は、登録候補の語がどのようなテキストに出現したのかを確認するために参照すればよい。

```
pid,id,text
0,52,"神戸で始めての、育休後職場復帰セミナーを開催しました。"
0,317,"僕の知り合いは 2 人子どもを産んで、立て続けに産休＋育休を取って、職場復帰した。"
:
```

2.4.2 利用例

例 1: **基本例**

```
$ head tweets.txt
3 年ぶりにウォークマンを買ったけど、育休中はあまり活躍の余地がないですね。
待機児童解消の方がいい気がするけど。
読売テレビ（日本テレビ系列）ウェークアップ！ぷらすに蓮舫ネクスト規制改革担当大臣が生出演！
この学生さんは、仕事に不利じゃなかったら、3 年育休取れるのも良いな、って思ってるよね。
今の人事制度のまま育休 3 年とか、前以上に女性が締め出されるだけでは。
女子大生でも分かる、3 年間の育児休暇が最悪な結果をもたらす理由。（中嶋よしふみ）
保育園を中心に期間とか決めるの、おかしいよな〜。
女性が必ず子育てしなきゃならない社会なら結婚絶対したくない…
育休とかの前に、母親に育児に専念させるなら女性の雇用よりもまず、男性の雇用、給料なんだよね。
安倍総理きた！ 育児休暇三年は…女としては嬉しいけど、会社に申し訳ないよねえ
$ mnewdic.rb i=tweets.txt O=newdic
#MSG# start to parse each line...
#MSG# working at line 0
#MSG# working at line 100
#MSG# working at line 200
#MSG# working at line 300
#MSG# working at line 400
#MSG# working at line 500
#MSG# working at line 600
#MSG# working at line 700
#MSG# working at line 800
#MSG# working at line 900
#MSG# working at line 1000
#END# /Users/maegawa/.rvm/rubies/ruby-2.0.0-p247/bin/mnewdic.rb i=tweets.txt O=newdic
$ ls newdic
corpus.csv
corpus_sjis.csv
words.csv
words_sjis.csv
$ head newdic/words.csv
見出し語,品詞,読み,カテゴリ,ドメイン,pid,word1,word2,freq
職場復帰,,,,,0,職場,復帰,31
授業参観,,,,,1,授業,参観,28
会議参加,,,,,2,会議,参加,26
育休延長,,,,,3,育休,延長,19
子育て支援,,,,,4,子育て,支援,18
育児休暇３,,,,,5,育児休暇,３,18
```

待機児童ゼロ,,,,,6, **待機児童**, **ゼロ**,17
規制緩和,,,,,7, **規制**, **緩和**,16
給付金,,,,,8, **給付**, **金**,15

2.5 mjumandic.rb CSV から JUMAN 辞書への変換

CSV で与えられた辞書データを、JUMAN の辞書に変換する。

CSV はテキストエディタを用いて記述しても構わないが、`mnewdic.rb` コマンドが出力する CSV を利用することもできる。

2.5.1 書式

```
mjumandic.rb [i=] [O=] [exe=] [-mcmdenv] [--help]
```

`i=`	: CSV の辞書ファイル名
`O=`	: JUMAN の辞書を格納するディレクトリ名
`exe=`	: makeint 等のコマンドパス (デフォルトは/usr/local/bin) JUMAN を通常の方法でインストールすれば指定する必要はないはず。
`-mcmdenv`	: 内部で利用している MCMD のメッセージ出力レベルを環境変数に任せる。 省略時は警告とエラーメッセージのみ出力 (KG_VerboseLevel=2)。
`--help`	: ヘルプメッセージの表示

入力ファイル例

`i=`パラメータで与える辞書ファイルの例を示す。**見出し語**、**読み**、**品詞**、**カテゴリ**、**ドメイン** の 5 項目があればよい。

```
id，見出し語，読み，品詞，カテゴリ，ドメイン
1，連結営業利益，れんけつえいぎょうりえき，普通名詞，抽象物，ビジネス
2，米国債，べいこくさい，，抽象物，ビジネス
3，上方修正，じょうほうしゅうせい，サ変名詞，抽象物，ビジネス
4，日本航空，にほんこうくう，組織名，，
5，夏目漱石，なつめそうせき，人名，日本，姓
6，安倍首相 安倍晋太郎 安倍晋太郎首相，あべしゅしょう，人名，日本，姓名
```

各項目の意味は以下の通りである。

見出し語 見出し語には、表記ゆれなどの複数の見出し語を半角空白で区切って列挙できる。見出し語がないとエラーとなる。

品詞 品詞は名詞のみ対応しており、以下に示す名詞の下位の品詞を「品詞」項目に登録する。
普通名詞, サ変名詞, 時相名詞, 数詞, 副詞的名詞, 固有名詞, 人名, 組織名, 地名
品詞が省略されると、「普通名詞」が指定されたものとする。品詞の体系は以下の URL を参照のこと。
http://www.unixuser.org/~euske/doc/postag/

読み 読みがないとエラーとなる。

カテゴリ カテゴリは以下の 22 種（省略可能）
人, 組織・団体, 動物, 植物, 動物-部位, 植物-部位, 人工物-食べ物, 人工物-衣類, 人工物-乗り物
人工物-金銭, 人工物-その他, 自然物, 場所-施設, 場所-施設部位, 場所-自然, 場所-機能
場所-その他, 抽象物, 形・模様, 色, 数量, 時間

ドメイン ドメインは以下の 12 種（省略可能）
文化・芸術, 交通, レクリエーション, 教育・学習, スポーツ, 科学・技術, 健康・医学
ビジネス, 家庭・暮らし, メディア, 料理・食事, 政治

カテゴリとドメインは、普通名詞とサ変名詞にのみ有効な項目である。

カテゴリとドメインは、以下の URL を参考に登録する。わからなければ省略してもよい。

http://nlp.ist.i.kyoto-u.ac.jp/DLcounter/lime.cgi?down=http://nlp.ist.i.kyoto-u.ac.jp/nl-resource/knp/20090930-juman-knp.ppt&name=20090930-juman-knp.ppt

2.5.2 利用例

例 1: 基本例

```
$ more dic1.csv
id, 見出し語, 読み, 品詞, カテゴリ, ドメイン
1, 連結営業利益, れんけつえいぎょうりえき, 普通名詞, 抽象物, ビジネス
2, 米国債, べいこくさい, , 抽象物, ビジネス
3, 上方修正, じょうほうしゅうせい, サ変名詞, 抽象物, ビジネス
4, 日本航空, にほんこうくう, 組織名, ,
5, 夏目漱石, なつめそうせき, 人名, 日本, 姓
6, 安倍首相 安倍晋太郎 安倍晋太郎首相, あべしゅしょう, 人名, 日本, 姓名
7, ２ちゃんねる にちゃんねる, にちゃんねる, , ,
$ mjumandic.rb i=dic1.csv O=jumandic
#END# kgcut f=品詞, 見出し語, 読み i=dic1.csv
#END# kgdelnull f=見出し語, 読み
#END# kgsortf f=見出し語
#END# kguniq k=見出し語 o=/tmp/__MTEMP_68157_70357348549040_0
#END# mcsvin i=/tmp/__MTEMP_68157_70357348549040_0
Mon Jul 28 01:38:37 2014
/usr/local/share/juman/dic/JUMAN.grammar parsing... done.

Mon Jul 28 01:38:37 2014
/usr/local/share/juman/dic/JUMAN.katuyou parsing... done.

Mon Jul 28 01:38:37 2014
/usr/local/share/juman/dic/jumandic.tab parsing... done.

jumandic.dic parsing... done.

execution time:    0.000s
processor time:    0.000s
File Name "/Users/maegawa/git/nysol/nysol/doc/fumi/jp/examples/jumandic/jumandic.dat"

## 10 entry  814 th char
Saving pat-tree "/Users/maegawa/git/nysol/nysol/doc/fumi/jp/examples/jumandic/jumandic.pat"
QUIT
#MSG# jumandic 内の jumandic.dat,jumandic.pat の２つのファイルがユーザ辞書として必要となる。
#MSG# ~/.jumanrc ファイルを編集し、これらのファイルが格納されたパス名を以下のように追加登録す
#MSG# (辞書ファイル
#MSG#         /usr/local/share/juman/dic
#MSG#         /usr/local/share/juman/autodic
#MSG#         /usr/local/share/juman/wikipediadic
#MSG#         /Users/maegawa/git/nysol/nysol/doc/fumi/jp/examples/jumandic
#MSG# )
#END# /Users/maegawa/.rvm/rubies/ruby-2.0.0-p247/bin/mjumandic.rb i=dic1.csv O=jumandic
$ ls jumandic
jumandic.dat
jumandic.dic
jumandic.int
jumandic.pat
$ more jumandic/jumandic.dic
(名詞 (サ変名詞 ((読み じょうほうしゅうせい) (見出し語 上方修正) (意味情報 "代表表記:上方修
(名詞 (人名 ((読み なつめそうせき) (見出し語 夏目漱石) (意味情報 "")))))
(名詞 (人名 ((読み あべしゅしょう) (見出し語 安倍首相 安倍晋太郎 安倍晋太郎首相) (意味情報 "
(名詞 (組織名 ((読み にほんこうくう) (見出し語 日本航空) (意味情報 "代表表記:日本航空/にほん
(名詞 (普通名詞 ((読み べいこくさい) (見出し語 米国債) (意味情報 "代表表記:米国債/べいこくさ
(名詞 (普通名詞 ((読み れんけつえいぎょうりえき) (見出し語 連結営業利益) (意味情報 "代表表記
(名詞 (普通名詞 ((読み にちゃんねる) (見出し語 ２ちゃんねる にちゃんねる) (意味情報 "代表表
```