テキスト簡易要約器 Posum
version 1.50.2 マニュアル

望月 源

2002 年 1 月 10 日
IS-TM-2002-002

北陸先端科学技術大学院大学情報科学研究科

Copyright © 2001,2002
Hajime Mochizuki

目 次

はじめに 1

1 Posum のインストール 2
1.1 配布プログラムと動作環境 . . . 2
1.2 インストール . . . 2

2 Posum の使用方法 4
2.1 Posum プログラム群について . . . 4
2.2 簡単な使用方法 . . . 4
2.3 詳細な説明 . . . 6
2.3.1 文の重要度を計算するプログラムの説明 . . . 6
2.3.2 文や文字を抽出するプログラムの説明 . . . 15

3 df データファイル作成方法 17
3.1 tfidfsw 用 df データ . . . 18
3.2 tfidfsw_cha 用 df データ . . . 18
3.3 cfidfsw 用 df データ . . . 18
3.4 cfidfsw_cha 用 df データ . . . 19
3.5 df インデックスの作成 . . . 19

4 Query Biased な要約への対応 20
4.1 tfidfsw での使用方法 . . . 20
4.2 tfidfsw_cha での使用方法 . . . 21
4.3 cfidfsw での使用方法 . . . 21
4.4 cfidfsw_cha での使用方法 . . . 22

5 助詞による重みの記述 23
5.1 助詞重み記述ファイルとは . . . 23
5.2 助詞重み記述ファイルの説明 . . . 23

おわりに 24

はじめに

本稿は，テキスト簡易要約器 Posum のインストール方法および使用方法を説明するマニュアルです．

Posum は，テキスト自動要約における基本的な手法である「重要文抽出」の手法の中で，テキスト中の単語の重要度や，単語間のつながりを利用した単語の重要度を元にする手法によって，要約を作成するための比較的簡単なプログラム群で構成されています．

Posum を構成するプログラム群の仕組は単純ですが，オプションが多く存在するため，組み合わせて用いることにより，多様な重要度計算を試みることが可能になっています．本マニュアルでは，全てのオプションについて説明をしてあります．細かく利用したい方は説明をお読みになり，いろいろ試行してみて下さい．

また特に難しいオプションを利用しなくても，基本的な重要文抽出型の要約が作成できますので，簡単なテキスト要約システムとして，あるいは，比較実験などのベースラインシステムとして，気軽に Posum を利用頂ければ幸いです．

なお，本稿では要約についての細かい説明や記述は行っていません．テキスト自動要約の動向などに関しては [2]，[3]，Posum で用いている要約手法に関しては，[1] や [4] などを参考にして下さい．

また，Posum を利用するためには，語彙的連鎖計算プログラム Lexical Chainers version 1.50.2 が必要になります．こちらの文献 [7] も合わせてお読み下さい．

本システムに関する問い合わせ先
〒 923-1292 石川県能美郡辰口町旭台 1-1
北陸先端科学技術大学院大学 情報科学研究科
望月 源
E-mail: motizuki@jaist.ac.jp
WWW: http://galaga.jaist.ac.jp:8000/~motizuki/software/posumcl/

1 Posum のインストール

ここでは，Posum バージョン 1.50.2 のインストール方法を説明する．

1.1 配布プログラムと動作環境

- 動作に必要な環境，アプリケーション等
  - Posum は，UNIX 系 OS*上で動作する．
  - 語彙的連鎖計算プログラム，Lexical Chainers version 1.50.2 がインストールされている必要がある．(必須)
    (配布元) http://galaga.jaist.ac.jp:8000/~motizuki/software/chainers/
  - jperl5 がインストールされ，利用できる必要がある．(必須)
  - ChaSen(『茶筌』[8]) のライブラリを組み込んで使用する場合は，事前に ChaSen(2.2.x) をインストールする必要がある．ChaSen は奈良先端科学技術大学院大学，松本研究室で作成，配布されている．(http://chasen.aist-nara.ac.jp/index.html.ja)
  - ChaSen 以外の形態素解析器として， juman3.61 を利用することもできる．(ただし，ライブラリは利用できない)．JUMAN[6] は，京都大学言語メディア研究室で作成，配布されている．(http://www-nagao.kuee.kyoto-u.ac.jp/nl-resource/juman.html)
- 配布プログラムの取得
  - 以下の配布元より Posum version 1.50.2 をダウンロードする．
    http://galaga.jaist.ac.jp:8000/~motizuki/software/posumcl/

1.2 インストール

1. 準備
   - Lexical Chainers 1.50.2 がインストールされていない場合は，インストールする.
   - Lexical Chainers のインストール先を$MOTSROOT とする．(posum も同じディレクトリにインストールされる)
   - Posum のソースを展開したディレクトリを$POSUMSRC とする．
2. $POSUMSRC/conf/conf を編集する．
   - MOTSROOT
     Lexical Chainers のインストール先のディレクトリをフルパスで指定する．POSUM も，ここで指定した場所にインストールすることになる．指定しなけれ

*動作確認済は，FreeBSD, Solaris 5.6, Solaris7, 色々な linux．

ば，/usr/local がデフォルトとして選択されるが，その場合は Lexical Chainers のインストール先が一致していないとインストールできないので注意．

- CHAINC
  chasen.h のある場所を指定する．ChaSen のライブラリを使う場合のみ指定すること．
- CHALIB
  chasen のライブラリ libchasen.a のある場所を指定する．ChaSen のライブラリを使う場合のみ指定すること．

3. $POSUMSRC で，configure を--enable-conffile 付きで実行する．

   % ./configure --enable-conffile

   なお，conf file を使用しないで，直接 --prefix を指定することも可能である．その場合，conf 内の各変数を必要に応じて，環境変数としてセットしてから，以下を実行すること．

   % ./configure --prefix=$MOTSROOT

4. make を実行する
   % make

5. make install を実行する
   % make install

   これにより，$MOTSROOT/bin にプログラムがインストールされる．

以上で，インストール終了．

2 Posum の使用方法

ここでは Posum の使用方法を説明する．

2.1 Posum プログラム群について

Posum のプログラム群は「文の重要度を計算するプログラム」と「(元のテキストから要約となる) 文や文字を抽出するプログラム」の大きく 2 つにわかれる．

1. 文の重要度を計算するプログラム：

   (1-1) tfidfsw
   tf または tfidf ベースで文の重要度を計算する．

   (1-2) tfidfsw_cha
   tf または tfidf ベースで文の重要度を計算する．ChaSen のライブラリを取り込んで処理する．

   (1-3) cfidfsw
   Lexical Chainers によって作成された語彙的連鎖ベースで文の重要度を計算する．

   (1-4) cfidfsw_cha
   Lexical Chainers によって作成された語彙的連鎖ベースで文の重要度を計算する．ChaSen のライブラリを取り込んで処理する．

2. 文や文字を抽出するプログラム：

   (2-1) extsents
   1 のプログラムで計算した重要度を使って，文を抽出する．

   (2-2) leadsum
   1 とは関係なく先頭から指定文数だけ抽出する (lead 手法)．

   (2-3) extsent_c.perl
   1 のプログラムで計算した重要度を使って，文字単位で抽出する．

   (2-4) lextsent_c.perl
   1 とは関係なく先頭から指定文字数だけ抽出する (lead 手法)．

2.2 簡単な使用方法

ここでは，Posum の基本となる使用方法について非常に簡単に説明する．Posum では，「文の重要度を計算するプログラム」と，「2．文や文字を抽出するプログラム」を組み合せて使うのが基本となる．なお，ここでは例として，目的別にそれぞれ 1 つの組み合せを示すが，それ以外の組み合わせや，各プログラムのオプションを変えて要約を作成することもできる．詳細な説明は，2.3節を参照のこと．

なお，ここであげるすべての場合において要約対象となる元のテキストファイル (TEXTFILE) として，1 行 1 文となっているものを用意する．(プログラムは改行コードを 1 行の区切り

だとみなすので，文の途中で改行されていたり，複数の文が改行なしに連続しているファイルは正しく扱うことができないので注意)．

その 1 重要度の高い文から順番に指定した割合の文数を出力

ここでは「1．文の重要度を計算するプログラム」として，tfidfsw_cha
「2．文や文字を抽出するプログラム」として，extsents を使用する．

% tfidfsw_cha < TEXTFILE | extsents -sum_val 0.2 TEXTFILE

これで，元のテキスト (TEXTFILE) の総文数の「20%の文」を出力する．つまり，TEXTFILE が 20 文であれば，重要度の高い 4 文が出てくることになる．

その 2 重要度の高い文から順番に，指定文字数だけ出力

ここでは「1．文の重要度を計算するプログラム」として，tfidfsw_cha
「2．文や文字を抽出するプログラム」として，extsent_c.perl を使用する．

% tfidfsw_cha < TEXTFILE | extsent_c.perl 30 TEXTFILE

これで，元のテキスト (TEXTFILE) の重要度の高い文から順番に「30 文字」に達するまで文字を出力する．つまり，文の途中でも 30 文字までいったらそこで終了する．

その 3 先頭から順番に，指定した割合の文数を出力

ここでは「1．文の重要度を計算するプログラム」を使わずに「2．文や文字を抽出するプログラム」として，leadsum だけを使用する．

% leadsum -sum_val 0.2 < TEXTFILE

これで，元のテキスト (TEXTFILE) の先頭の文から順番に，総文数の「20%の文」を出力する．つまり，TEXTFILE が 20 文であれば，先頭から 4 文が出てくることになる．

その 4 先頭から順番に，指定文字数だけ出力

ここでは「1．文の重要度を計算するプログラム」を使わずに「2．文や文字を抽出するプログラム」として，lextsent_c.perl だけを使用する．

% lextsent_c.perl 30 < TEXTFILE

これで，元のテキスト (TEXTFILE) の先頭から 30 文字を出力する．

2.3 詳細な説明

ここでは，Posum の各プログラムを詳細に説明する．

2.1節の分類に沿って「1．文の重要度を計算するプログラム」4 つと「2．文や文字を抽出するプログラム」4 つを順に説明する．

2.3.1 文の重要度を計算するプログラムの説明

(1-1) tfidfsw の使い方

tfidfsw は，元のテキスト (TEXTFILE) を ChaSen や juman で形態素解析した結果から，名詞，動詞，形容詞と未定義語[†]の各単語を情報として用いて[‡]，各文の重要度を計算する． 各文の重要度は，その文に出現する各語の重みを，その出現頻度 (term frequency:TF) や出現文書頻度の逆数 (inverse document frequency:IDF) を用いた tf や tf.idf によって計算し，基本的に足し合わせることで計算する．

各語の重みは次のようになる．

- tf で計算する場合

$$w_i = tf_i \tag{1}$$

ここで tf_i は単語 i の文書内の出現頻度である．

- tf.idf で計算する場合

$$w_i = tf_i \times log\frac{N}{df_i} \tag{2}$$

ここで tf_i は単語 i の文書内の出現頻度，N は文書集合内の総文書数，df_i は単語 i の出現する文書数である．

- 入力
  形態素解析結果ファイル
  ChaSen 2.2.x か JUMAN の -e オプション付きの結果
  あるいは KNP[5] の-tab オプション付きの結果
- 出力
  各文の重要度

プログラムは以下のように使用する．
% tfidfsw [options] < 形態素解析結果ファイル

(例: chasen -e TEXTFILE | ./tfidfsw [options])
(例: juman -e TEXTFILE | ./tfidfsw -jum [options])

オプションの説明：

[†]デフォルトでは名詞と同等に扱われる．
[‡]後述する [-F] オプションにより品詞選択が行なえる．

- 入力タイプ
  使用する形態素解析器の出力形式に対応するタイプを指定する．使用できる形式は，-knp, -cha, -jum のどれかであり，デフォルトは -cha である．

  -knp ： KNP の -tab 形式出力を使用

  -cha ： ChaSen 2.2.x の -e 形式出力を使用 (default)

  -jum ： Juman3.61 の -e 形式出力を使用

- フィルタルール記述ファイルの指定

  -rule <filename> ：
  使用する形態素解析器の結果に対応した除外単語リストなどを記述したファイルをフルパスで指定する．

  指定がない時は，上の「入力タイプ」と連動して，
  デフォルト (-cha) では$MOTSROOT/rule/mrule0 を読み込む．
  -knp, -jum の時は， $MOTSROOT/rule/mrule1 を読み込む．

  「フィルタルール記述ファイル」の詳細は，文献 [7] の 4 節を参照せよ．

- 重みを計算する語に関するオプション

  -F <num> ： デフォルト値=0

  tf の計算で考慮する語の品詞を指定する．組み合わせは以下のとおり
  0．名詞 動詞 形容詞
  1．名詞 —— 形容詞
  2．名詞 —— ——
  3．名詞 動詞 ——
  4．—— 動詞 ——
  5．—— 動詞 形容詞
  6．—— —— 形容詞

- 単語の重み付けのためのオプション

  -twm <0,1,2> ： 単語の重み付け方法の指定
  0: 重み付けしない (1 単語 1 点)
  1: tf を重みにする (default)
  2: tfidf を重みにする．
  df データファイルが別途必要なので，3節「df データファイル作成方法」を参照すること．

  -df <filename> ：
  -twm が 2 の場合に参照する df データファイルを指定する．

- Query-Biased な要約を作成するためのオプション

  -qf <filename> ：
  query biased を行なう場合の query ファイルを指定する．Query Biased への対応については，4節を参照すること．

-qwgt <num> : デフォルト値=3
query biased を行なう場合の重みを指定する．

- 文の長さによる正規化のためのオプション

-nm <0-5> : デフォルト値=0
文の重要度を文の長さによって正規化をする場合の方法の指定
0：正規化なし (default)．
1：文中の単語 (内容語) 数で割る．
2：各文の偏差値で割る．
3：文のスコアの 2 乗を単語 (内容語) 数で割る．
4：単語数 ×(-nm_val で指定した値) で計算する．
5：(単語数で割った値/2)+(各文の偏差値で割った値) で計算する．

-nm_val <num> : デフォルト値=1
-nm 4 の時の重み

- その他

-h : help を表示する．

(1-2) tfidfsw_cha の使い方

tfidfsw_cha は，元のテキスト (TEXTFILE) をプログラム内部で ChaSen により形態素解析し， 名詞，動詞，形容詞と未定義語[§]を情報として用いて[¶]，各文の重要度を計算する．各文の重要度は，その文に出現する各語の重みを，その出現頻度 (term frequency:TF) や出現文書頻度の逆数 (inverse document frequency:IDF) を用いた tf や tf.idf によって計算し，基本的に足し合わせることで計算する．

各語の重みは次のようになる．

- tf で計算する場合：式 (1) で計算
- tf.idf で計算する場合：式 (2) で計算

- 入力
1 行 1 文となっている，テキストファイル (TEXTFILE).
- 出力
各文の重要度

プログラムは以下のように使用する．
% tfidfsw_cha [options] < TEXTFILE

オプションの説明：

- フィルタルール記述ファイルの指定

[§]デフォルトでは名詞と同等に扱う．
[¶]後述する [-F] オプションによって品詞選択ができる．

-rule <filename> :
使用する形態素解析器の結果に対応した除外単語リストなどを記述したファイルをフルパスで指定する．

指定がない時[‖]は，上の入力タイプと連動して，
デフォルト (-cha) では$MOTSROOT/rule/mrule0 を読み込む．
-knp, -jum の時は，$MOTSROOT/rule/mrule1 を読み込む．

「フィルタルール記述ファイル」の詳細は，文献 [7] の 4 節を参照せよ．

- 重みを計算する語に関するオプション

  -F <0-6> ： デフォルト値=0

  tf の計算で考慮する語の品詞を指定する．組み合わせは以下のとおり
  0．名詞 動詞 形容詞
  1．名詞 —— 形容詞
  2．名詞 —— ——
  3．名詞 動詞 ——
  4．—— 動詞 ——
  5．—— 動詞 形容詞
  6．—— —— 形容詞

- 単語の重み付けのためのオプション

  -twm <0,1,2> : 単語の重み付け方法の指定
  0: 重み付けしない (1 単語 1 点)
  1: tf を重みにする (default)
  2: tfidf を重みにする．
  df データファイルが別途必要なので，3節「df データファイル作成方法」を参照すること．

  -df <filename> :
  -twm が 2 の場合に参照する df データファイルを指定する．

- Query-Biased な要約を作成するためのオプション

  -qf <filename> :
  query biased を行なう場合の query ファイルを指定する．Query Biased への対応については，4節を参照すること．

  -qwgt <num> : デフォルト値=3
  query biased を行なう場合の重みを指定する．

- 文の長さによる正規化のためのオプション

  -nm <0-5> : デフォルト値=0
  文の重要度を文の長さによって正規化をする場合の方法の指定

[‖]tfidfsw_cha では ChaSen のライブラリを用いるので，通常-knp や-jum に対応したフィルタルールは必要ないが，ユーザが定義したルールファイルを使えるようにするため，このオプションは存在する．

0：正規化なし (default)．
1：文中の単語 (内容語) 数で割る．
2：各文の偏差値で割る．
3：文のスコアの 2 乗を単語 (内容語) 数で割る．
4：単語数 ×(-nm_val で指定した値) で計算する．
5：(単語数で割った値/2)+ (各文の偏差値で割った値) で計算する．

-nm_val <num> ：デフォルト値=1
-nm 4 の時の重み

- その他

-h ：help を表示する．

(1-3) cfidfsw の使い方

cfidfsw は，元のテキスト (TEXTFILE) から Lexical Chainers[7] のどれかによって計算した語彙的連鎖を情報として用いて，各文の重要度を計算する．各文の重要度は，TF.IDF の考え方を語彙的連鎖にあてはめた CF.IDF を用いる．すなわち，その文に出現する各語の重みを，その語が属している語彙的連鎖を構成する語の出現頻度 (chain frequency:CF) や語彙的連鎖の出現文書頻度の逆数 (inverse document frequency:IDF) を用いた cf や cf.idf によって計算し，基本的に足し合わせることで計算する．

各語の重みは次のようになる．

- cf で計算する場合

$$w_i = \mid i \mid \tag{3}$$

ここで，$\mid i \mid$ は文書内の連鎖 i の構成単語数である．

- cf.idf で計算する場合

$$w_i = \mid i \mid \times log\frac{N}{df_i} \tag{4}$$

ここで，$\mid i \mid$ は文書内の連鎖 i の構成単語数，N は文書集合内の総文書数，df_i は連鎖 i の出現する文書数である．

- 入力
Lexical Chainers の出力 (chain_file)
オプションはコマンドにより異なるが，シソーラスを用いる場合は多義性が解消されない場合でも 1 つだけを出力する '-1' オプションをつける必要がある．
('chainer_th -1', 'chainer_thcha -1', 'chainer_co', 'chainer_cocha')

- 出力
各文の重要度

プログラムは以下のように使用する．
% cfidfsw [options] < chain_file

オプションの説明：

- 語彙的連鎖作成時に使用した形態素の形式 (入力タイプ)
  chain_file の作成時に使用した形態素解析器の出力形式に対応するタイプを指定する．使用できる形式は，-knp, -cha, -jum のどれかであり，デフォルトは -cha である．

  -knp : KNP の -tab 形式出力を使用

  -cha : ChaSen 2.2.x の -e 形式出力を使用 (default)

  -jum : Juman3.61 の -e 形式出力を使用

- 語彙的連鎖作成時に使用した連鎖の情報源
  語彙的連鎖を構成するための基準として使用したものを 1 つ指定する．

  -the : シソーラスを利用 (default)

  -co : 語の共起関係を利用

  -rep : 同語の反復を利用

- 語彙的連鎖の重み付けのためのオプション

  -cwm <0,1,2> : 語彙的連鎖の重み付けモード指定
  0: 重み付けしない (1 単語 1 点)．
  1: cf を重みにする (default)．
  「cf」は語彙的連鎖を構成する語の数
  2: cf.idf を重みにする．
  df データファイルが別途必要なので，3節「df データファイル作成方法」を参照すること．

  -df <filename> :
  cwm が 2 の場合に参照する df データファイルを指定する．

- Query-Biased な要約を作成するためのオプション

  -qf <filename> :
  query biased を行なう場合の query ファイルを指定する．Query Biased への対応については，4節を参照すること．

  -qwgt <num> : デフォルト値=3
  query biased を行なう場合の重みを指定する．

- 文の長さによる正規化のためのオプション

  -nm <0-5> : デフォルト値=0
  文の重要度を文の長さによって正規化をする場合の方法の指定
  0：正規化なし (default)．
  1：文中の単語 (内容語) 数で割る．
  2：各文の偏差値で割る．
  3：文のスコアの 2 乗を単語 (内容語) 数で割る．
  4：単語数 ×(-nm_val で指定した値) で計算する．
  5：(単語数で割った値/2)+(各文の偏差値で割った値) で計算する．

  -nm_val <num> : デフォルト値=1
  -nm 4 の時の重み

- 語彙的連鎖のデータから，使用する連鎖を選択するためのオプション

  -min_chntf <INT> : デフォルト値=1
  　何語以上で構成される場合に連鎖であると認めるかを指定する．

  -thre_mode <0,1,2> : デフォルト値=0
  　連鎖のスコア (重要度) によって足切りする (無効にする) 連鎖を選ぶモードを指定する．
  　0 : 足切りしない (default)．
  　1 : 連鎖スコアの平均 × -thre_val で指定した値 以上の連鎖だけを対象にする．
  　2 : -thre_val で指定した値以上の連鎖だけを対象にする．

  -thre_val <num> : デフォルト値=0.5
  　-thre_mode 0 以外の時の重みを指定する．

  -chain <INT> : デフォルト値=全単語数
  　chain length threshold の分母を指定する．同一の連鎖内での最初の単語と最後の単語の出現位置の差 (距離) が，全単語数/(-chain で指定した値) 以上であれば有効な連鎖とする．

  -gap <INT> : デフォルト値=1
  　gap lengh threshold の分母を指定する．同一の連鎖内での単語と単語の出現位置の差 (距離) が，全単語数/(-gap で指定した値) 以上離れたら別の連鎖として分割する．

- 付属する助詞による単語の重み付けのためのオプション

  -wrule <filename> :
  　付属する助詞による重みを記述したファイルを指定する．省略時は，入力タイプに応じて自動的にファイルが選択される．
  　入力タイプが，-cha の時は$MOTSROOT/rule/wrule0 が選択される．-jum,-knp の時は$MOTSROOT/rule/wrule1 が選択される．

  　詳しくは，5節「助詞による重みの記述」を参照すること．

  -no_pw : 助詞による重み付けをしない．
  　デフォルトでは，重み付けがされる．

- その他

  -h : help を表示する．

  -hh : help(2) を表示する．

(1-4) cfidfsw_cha の使い方

cfidfsw_cha は，元のテキスト (TEXTFILE) から Lexical Chainers のどれかによって計算した 語彙的連鎖を情報として用いて，各文の重要度を計算する．cfidfsw との違いは，Query biased な要約を作成する際に，バイアスをかける語を指定するファイルをテキストとして，内部で形態素解析を行なう点のみである．cfidfsw の場合は Query biased を行なう場合のファイルを事前に作成しておく必要があるが，

cfidfsw_cha では-qf で指定したテキストを ChaSen により形態素解析し，バイアスをかける語を獲得することができる[**] ．

なお，各単語の重要度は cfidfsw と同様に計算する (式 (3) および式 (4)) ．

- 入力
  Lexical Chainers の出力 (chain_file)
  オプションはコマンドにより異なるが，シソーラスを用いる場合は多義性が解消されない場合でも 1 つだけを出力する '-1' オプションをつける必要がある．
  ('chainer_th -1', 'chainer_thcha -1', 'chainer_co', 'chainer_cocha')
- 出力
  各文の重要度

プログラムは以下のように使用する．
% cfidfsw_cha [options] < chain_file

オプションの説明：

- 語彙的連鎖作成時に使用した連鎖の情報源
  語彙的連鎖を構成するための基準として使用したものを 1 つ指定する．

  -the : シソーラスを利用 (default)

  -co : 語の共起関係を利用

  -rep : 同語の反復を利用

- 語彙的連鎖の重み付けのためのオプション

  -cwm <0,1,2> : 語彙的連鎖の重み付けモード指定
  0: 重み付けしない (1 単語 1 点)．
  1: cf を重みにする (default)．
  「cf」は語彙的連鎖を構成する語の数
  2: cf.idf を重みにする．
  df データファイルが別途必要なので，3節「df データファイル作成方法」を参照すること．

  -df <filename> :
  cwm が 2 の場合に参照する df データファイルを指定する．

- Query-Biased な要約を作成するためのオプション

  -qf <filename> :
  query biased を行なう場合の query ファイルを指定する．Query Biased への対応については，4節を参照すること．

[**]cf.idf 値を計算する語は語彙的連鎖を元にしているのに対し，バイアスをかける語としてここで獲得する語は単語そのものとなっているので，厳密にはこの Query Biased の方法は正確でない．しかし，Query Biased 用ファイルを作る繁雑さを避ける利点があるため，このプログラムを用意した．また，語彙的連鎖を構成する基準がシソーラスに基づく場合，語彙的連鎖は概念で計算されているのに対し，バイアスの対象が単語そのものになってしまうので，この組み合わせによる使用はできない．cfidfsw_cha は，こうした点を理解した上で使用して欲しい．

-qwgt <num> : デフォルト値=3

query biased を行なう場合の重みを指定する．

-F <0-6> : デフォルト値=0

query ファイル内で，バイアスをかけるために考慮する語の品詞を指定する．組み合わせは以下のとおり

0．名詞 動詞 形容詞

1．名詞 —– 形容詞

2．名詞 —– ——

3．名詞 動詞 ——

4．—– 動詞 ——

5．—– 動詞 形容詞

6．—– —– 形容詞

-rule <filename> : デフォルトでは参照しない．

除外単語リストなどを記述したファイルを指定する．これは，query ファイル内の単語から特定の語を除外するために用いる．デフォルトでは参照しないが，指定する場合は，ChaSen 用のフィルタルール記述ファイルを指定するのが普通である．

- 文の長さによる正規化のためのオプション

-nm <0-5> : デフォルト値=0

文の重要度を文の長さによって正規化をする場合の方法の指定

0：正規化なし (default)．

1：文中の単語 (内容語) 数で割る．

2：各文の偏差値で割る．

3：文のスコアの 2 乗を単語 (内容語) 数で割る．

4：単語数 ×(-nm_val で指定した値) で計算する．

5：(単語数で割った値/2)+(各文の偏差値で割った値) で計算する．

-nm_val <num> : デフォルト値=1

-nm 4 の時の重み

- 語彙的連鎖のデータから，使用する連鎖を選択するためのオプション

-min_chntf <INT> : デフォルト値=1

何語以上で構成される場合に連鎖であると認めるかを指定する．

-thre_mode <0,1,2> : デフォルト値=0

連鎖のスコア (重要度) によって足切りする (無効にする) 連鎖を選ぶモードを指定する．

0：足切りしない (default)．

1：連鎖スコアの平均 × -thre_val で指定した値 以上の連鎖だけを対象にする．

2：-thre_val で指定した値以上の連鎖だけを対象にする．

-thre_val <num> : デフォルト値=0.5

-thre_mode 0 以外の時の重みを指定する．

-chain <INT> : デフォルト値=全単語数
chain length threshold の分母を指定する．同一の連鎖内での最初の単語と最後の単語の出現位置の差 (距離) が，全単語数/(-chain で指定した値) 以上であれば有効な連鎖とする．

-gap <INT> : デフォルト値=1
gap lengh threshold の分母を指定する．同一の連鎖内での単語と単語の出現位置の差 (距離) が，全単語数/(-gap で指定した値) 以上離れたら別の連鎖として分割する．

- 付属する助詞による単語の重み付けのためのオプション

-wrule <filename> :
付属する助詞による重みを記述したファイルを指定する．省略時は，入力タイプに応じて自動的にファイルが選択される．
入力タイプが，-cha の時は$MOTSROOT/rule/wrule0 が選択される．
-jum,-knp の時は$MOTSROOT/rule/wrule1 が選択される．

詳しくは，5節「助詞による重みの記述」を参照すること．

-no_pw : 助詞による重み付けをしない．
デフォルトでは，重み付けがされる．

- その他

-h : help を表示する．

-hh : help(2) を表示する．

2.3.2 文や文字を抽出するプログラムの説明

(2-1) extsents の使い方

extsents は，tfidfsw, tfidfsw_cha, cfidfsw, cfidfsw_cha のどれかによって計算された各文の重要度を元に，指定された要約率分の文を要約として出力する．

- 入力
(1-1),(1-2),(1-3),(1-4) いずれかの結果 (sscr_file) と，元のテキスト (TEXTFILE)
- 出力
重要文

プログラムは以下のように使用する．
% extsents [options] TEXTFILE < sscr_file

オプションの説明：

-ttl <INT> : デフォルト値=0
先頭 INT 文を無条件に出力する．

-sum_mode <0,1> : デフォルト値=1
出力モードを指定する．
0 : 全文から (-sum_val で指定した値) 文だけ出力
1 : 全文 × (-sum_val で指定した値) 文だけ出力 (default)

-sum_val <num> : デフォルト値=0.3
-sum_mode が 0 の場合は，出力文数を整数 (INT) で指定する．
-sum_mode が 1 の場合は，$0 < num <= 1.0$ の範囲で指定する．

-form_mode <0-5> : 出力形式の指定，デフォルト値=0
0 : 文章 (default)
1 : スコア 文章
2 : 文番号 文章
3 : 文番号 スコア
4 : スコア
5 : 文番号

-D : debug mode

(2-2) leadsum の使い方

leadsum は，指定された要約率分の文を先頭から要約として出力する．

- 入力
  元のテキスト (TEXTFILE)
- 出力
  重要文

プログラムは以下のように使用する．
% leadsum [options] < TEXTFILE

オプションの説明：

-sum_mode <0,1> : デフォルト値=1
出力モードを指定する．
0 : 全文から (-sum_val で指定した値) 文だけ出力
1 : 全文 × (-sum_val で指定した値) 文だけ出力 (default)

-sum_val <num> : デフォルト値=0.3
-sum_mode が 0 の場合は，出力文数を整数 (INT) で指定する．
-sum_mode が 1 の場合は，$0 < num <= 1.0$ の範囲で指定する．

-form_mode <0-5> : 出力形式の指定，デフォルト値=0
0 : 文章 (default)
1 : スコア 文章
2 : 文番号 文章
3 : 文番号 スコア
4 : スコア
5 : 文番号

-D : debug mode

-h : help を表示

(2-3) extsent_c.perl の使い方

extsent_c.perl は，tfidfsw, tfidfsw_cha, cfidfsw, cfidfsw_cha のどれかによって計算された各文の 重要度を元に，指定された要約率分の文字列を要約として出力する．

- 入力
  要約の文字数 (SUMLENGTH_C) と，(1-1),(1-2),(1-3),(1-4) のいずれかの結果 (sscr_file) と，元のテキスト (TEXTFILE)．
- 出力
  重要文 (指定した文字数だけ)

プログラムは以下のように使用する．

% extsent_c.perl [options] SUMLENGTH_C TEXTFILE < sscr_file

オプションの説明：

-L <INT> : デフォルト値は 0
先頭 INT 文をスコアに関係なく無条件に出力する．

-d : debug mode

-h : help を表示

(2-4) lextsent_c.perl の使い方

lextsent_c.perl は，指定された要約率分の文字列を先頭から要約として出力する．

- 入力
  要約の文字数 (SUMLENGTH_C) と，元のテキスト (TEXTFILE)
- 出力
  重要文 (指定した文字数だけ)

プログラムは以下のように使用する．

% lextsent_c.perl [options] SUMLENGTH_C < TEXTFILE

オプションの説明：

-d : debug mode

-h : help を表示

3 df データファイル作成方法

ここでは，idf の計算に使用する df データファイルの作成方法を説明する．

3.1 tfidfsw 用 df データ

1. df のデータを取りたいテキストから以下の形式で dffile を作成する．

   1 行目 ：N[SPACE] テキスト数
   2 行目以降：単語 i[SPACE] 出現テキスト数

   N number_of_documents
   Word1 df1
   Word2 df2
   ...

2. dffile をソートする．

3. 3.5節「df インデックスの作成」を見て変換する．

3.2 tfidfsw_cha 用 df データ

1. df のデータを取りたいテキストの一覧表 (フルパス)，TEXTLIST を作成する．

2. 1 の TEXTLIST を使って，以下を実行する．
   % $MOTSROOT/bin/gettfdf < TEXTLIST > tfdata
   これにより，dffile というファイルに TEXTLIST 内の df のデータが入る[††]．

3. 3.5節「df インデックスの作成」を見て変換する．

3.3 cfidfsw 用 df データ

非常に手間のかかる作業である．
まず，chainer_th および chainer_thcha 用の df データ作成方法について述べる．

1. df のデータを取りたいテキスト全ての語彙的連鎖を作成する．(-1 オプション付きで実行)

2. 1 のテキスト群から以下の形式で，dffile を作成する．

   1 行目 ：N[SPACE] テキスト数
   2 行目以降：語義 i[SPACE] 出現テキスト数

   N number_of_documents
   Sense1 df1

[††]注意：TEXTLIST で与えるテキストは，基本的に 1 行 1 文で改行されているものを想定している．そのため，1 行に複数文があり，改行がない場合，正常に動作しない．

Sense2 df2
...

ここで，語義とは，chainer_th, chainer_thcha の出力の 4 番目のフィールドのことである．

3. dffile をソートする．
4. 3.5節「df インデックスの作成」を見て変換する．

手順 2 の dffile 作成では今のところスクリプト等を用意していないので，各自集計用のプログラムなどを自作する必要がある．

次に，chainer_co および chainer_cocha 用の df データ作成方法について述べる．

1. df のデータを取りたいテキスト全ての語彙的連鎖を作成する．(-1 オプション付きで実行)
2. 1 のテキスト群から以下の形式で，dffile を作成する．

1 行目 ：N[SPACE] テキスト数
2 行目以降：単語 i[SPACE] 出現テキスト数

N number_of_documents
Word1 df1
Word2 df2
...

ここで，語義とは，chainer_co,chainer_cocha の出力の 4 番目のフィールドのことである．

3. dffile をソートする．
4. 3.5節「df インデックスの作成」を見て変換する．

3.4 cfidfsw_cha 用 df データ

cfidfsw 用 df データと同じである．

3.5 df インデックスの作成

dffile から df データのインデックスは以下の手順で作成する．

1. ソートした dffile を用意する．(dfs とする)

2. インデックスを作成する．
   % $MOTSROOT/sbin/patrash_int -WF df < dfs
3. 以下のファイルが作成されたこと確認する．
   df.wan, df.idx

これにより，df データが利用できるようになる．なお，Posum で使用する際は，オプションとして，-weight_mode 2 -df df などととすれば結果に反映される．

4 Query Biased な要約への対応

例えば，情報検索の結果として選ばれた文書を要約する場合，ユーザの興味は検索要求 (Query) として既に与えられている．この時，通常の要約を行なうよりも，Query に出現する語の重みを考慮して要約を作成した方が, ユーザにとって好ましい要約であると考えることができる．このような考えから Query Biased な要約では，指定した特定の語について，通常とは違う重みをつけて重要度の計算を行なう．Posum では，この query biased な要約に対応している．ただし，各プログラムによって，指定方法や準備するファイルが異なるため，順に説明する．

4.1 tfidfsw での使用方法

1. 以下の形式で，重みを付けたい単語の qfile を作成する．
   単語 i[SPACE] 適当な tf 値 (だいたいは 1)

   Word1 1
   Word2 1
   ...
2. -qf qfile を付けて実行する．
3. 必要に応じて，-qwgt の指定により，query biased の重みを調整する．デフォルト値は 3．

用意した qfile 内の単語が元のテキストの中にあれば，指定した重みが付くことになる．なお，各単語の重要度は次のようになる．以下で α は-qwgt で与える値である．

- tf の場合

$$w_i = \begin{cases} tf_i & \text{バイアス対象でない} \\ \alpha \times tf_i & \text{バイアス対象} \end{cases} \tag{5}$$

- tf.idf の場合

$$w_i = \begin{cases} tf_i \times log\frac{N}{df_i} & \text{バイアス対象でない} \\ \alpha \times tf_i \times log\frac{N}{df_i} & \text{バイアス対象} \end{cases} \tag{6}$$

4.2 tfidfsw_cha での使用方法

1. 重みを付けたい単語を書いたファイル (qfile) を用意する．
   ここでは qfile は普通のテキストでもよい．実行時に，tfidfsw_cha の内部で ChaSen により形態素解析され，-F オプションで指定された品詞のみが考慮される．

2. -qf qfile を付けて実行する．

3. 必要に応じて，-qwgt の指定により，query biased の重みを調整する．デフォルト値は 3．

用意した qfile 内の単語がオリジナルテキストの中にあれば，指定した重みが付くことになる．

なお，各単語の重要度は tfidfsw の場合と同様である (式 (5) および式 (6))．

4.3 cfidfsw での使用方法

まず，chainer_th および chainer_thcha での query biased の方法について述べる．

1. 以下の形式で，重みを付けたい語彙的連鎖の qfile を作成する．
   語義 i[SPACE] 適当な cf 値 (だいたいは 1)

   Sense1 1
   Sense2 1
   ...

   ここで語義とは，chainer_th, chainer_thcha の出力の 4 番目のフィールドを指す．このフィールドと，ここで用意した Sense を exact マッチングするので，正確に用意する必要がある．

2. -qf qfile を付けて実行する．

3. 必要に応じて，-qwgt の指定により，query biased の重みを調整する．デフォルト値は 3．

次に，chainer_co および chainer_cocha での query biased の方法について述べる．

1. 以下の形式で，重みを付けたい語彙的連鎖の qfile を作成する．
   単語 i[SPACE] 適当な tf 値 (だいたいは 1)

   Word1 1
   Word2 1

...

ここで，単語とは，chainer_co, chainer_cocha の出力の 4 番目のフィールドを指す．

2. -qf qfile を付けて実行する．
3. 必要に応じて，-qwgt の指定により，query biased の重みを調整する．デフォルトは 3．

なお，各単語の重要度は次のようになる．以下で α は-qwgt で与えた値である．

- cf の場合

$$w_i = \begin{cases} \mid i \mid & \text{バイアス対象でない} \\ \alpha \times \mid i \mid & \text{バイアス対象} \end{cases} \tag{7}$$

- cf.idf の場合

$$w_i = \begin{cases} \mid i \mid \times log\frac{N}{df_i} & \text{バイアス対象でない} \\ \alpha \times \mid i \mid \times log\frac{N}{df_i} & \text{バイアス対象} \end{cases} \tag{8}$$

4.4 cfidfsw_cha での使用方法

1. 重みを付けたい単語を書いたファイル (qfile) を用意する．
2. -qf qfile を付けて実行する．
3. 必要に応じて，-qwgt の指定により，query biased の重みを調整する．デフォルト値は 3．

ここで用いた qfile 内の単語が元のテキストの中にあれば，重みが付くことになる．
また，次の方法も使用できる．

1. 重み付けに用いたいテキストファイル (TEXTFILE) を用意する．
2. -qf TEXTFILE を付けて実行する．
3. 必要に応じて，-qwgt の指定により，query biased の重みを調整する．デフォルト値は 3．また，TEXTFILE 内の単語で重み付けに使用する品詞を選択したい場合は-F オプション，除外リストなどを用いたい場合は，-rule オプションを使用する．

各単語の重要度は cfidfsw の場合と同様である (式 (7) および式 (8))．

なお，cfidfsw_cha の場合，query biased による重みをかける対象を，指定した qfile(または TEXTFILE) からプログラムにより自動的に選択する．この時，対象は概念番号でなく，単語となるので，chainer_th および，chainer_thcha を語彙的連鎖の計算用に用いている場合で，シソーラスベースの連鎖計算をしたファイルに対しては，cfidfsw_cha による query-biased はできない．

5　助詞による重みの記述

ここでは付属する助詞による重みを記述するファイルについて説明する．

5.1　助詞重み記述ファイルとは

「助詞重み記述ファイル」とは，$MOTSROOT/rules/wrule*ファイルのことであり，次の働きをする．

- 語彙的連鎖を構成する語に付属する「助詞」(や語の品詞そのもの) の種類に応じて，個別の語の重みを調整するための値を定義する．

デフォルトで用意されている助詞重み記述ファイルは以下の 2 つである．

- ChaSen 2.2.x に対応：$MOTSROOT/rules/wrule0
- juman3.61 と knp に対応：$MOTSROOT/rules/wrule1

Posum の cfidfsw と cfidfsw_cha では，ユーザ独自の重みを定義した助詞重み記述ファイルを用いることもできる．その場合は -wrule オプションを利用する．

5.2　助詞重み記述ファイルの説明

例として，ChaSen 2.2.x を用いた語彙的連鎖に対応した$MOTSROOT/rules/wrule0 をとりあげ説明する (図 1)．

- 先頭が「＃」で始まる行は無視される．
- 有効なレコードは以下の形式を持つ．
  {weight\t 分類名 \t 記述 }

  ここで「weight」は重みを意味し，「分類名」，「記述」は Lexical Chainers の出力形式における「付属する助詞」と「助詞」をそれぞれ意味する[‡‡]．また，「記述」が「*」であるものは「分類名」だけが一致すればよいことを意味する．

  例えば，図 1において，最初の 3 つの「kaku」は以下のように解釈される．
  「1.4　kaku　が」：　分類名が kaku で記述が「が」であれば重みは 1.4．
  「1.2　kaku　を」：　分類名が kaku で記述が「を」であれば重みは 1.2．
  「1.0　kaku　*」　：　分類名が kaku で記述であれば重みは 1.0．
  なお，wrule 内の表記順に解釈されるので，前に記述されているものが優先されるので順番は重要である．

[‡‡] これは，形態素フィルタルール記述ファイルにより定義されている．詳しくは，文献 [7] の 4 節を参照せよ

```
# weight\t 分類名 \t 記述
#
1.4   kaku        が
1.2   kaku        を
1.0   kaku        *
1.4   fuku        は
1.2   fuku        も
1.0   fuku        *
1.0   shu         *
1.0   setsuzoku   *
1.0   kakari      *
1.0   tokushu     *
1.0   fukushika   *
1.0   heiritsu    *
1.0   rentaika    *
0.8   yougen      *
1.0   no          *
```

図 1: 助詞重み記述ファイルの例

おわりに

Posum は，基本的なテキスト自動要約の手法を手軽に扱えるプログラム群です．あまり難しいことを考えずに，プログラムデフォルトの方法で重要文抽出型の要約を作ることができます．また，色々なオプションが用意されており，凝った使い方も可能ですので，興味のある方は色々試してみて下さい．

参考文献

[1] 奥村学, 望月源. テキストを自動的に要約する技術. bit.2000.2 月号, 共立出版, pp. 37–42, 2000.

[2] 奥村学, 難波英嗣. テキスト自動要約技術の現状と課題. Technical Report IR-RR-98-0010I, 北陸先端科学技術大学院大学, 1998.

[3] 奥村学, 難波英嗣. テキスト自動要約に関する研究動向. 自然言語処理, Vol. 6, No. 6, pp. 1–26, 1999.

[4] 望月源, 奥村学. 語彙的連鎖に基づく要約の情報検索タスクを用いた評価. 自然言語処理, Vol. 7, No. 4, pp. 63–77, 2000.

[5] 黒橋禎夫. 日本語構文解析システム KNP version 2.0 b6 使用説明書, 1998.

[6] 黒橋禎夫, 長尾真. 日本語形態素解析システム JUMAN version 3.61, 1999.

[7] 望月源. 語彙的連鎖計算プログラム Lexical Chainers version 1.50.2 マニュアル. JAIST Technical Memorandum, IS-TM-2002-001, 2002.

[8] 松本裕治, 北内啓, 山下達雄, 平野善隆, 松田寛, 高岡一馬, 湯原正幸. 形態素解析システム『茶筌』version 2.2.7 使用説明書, 2001.