高速単語共起解析に基づく語彙的連鎖作成システム Lexical Chainers の改良

望月 源　北陸先端科学技術大学院大学情報科学研究科

目 次

はじめに **1**

1 Lexical Chainers 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 3

2 Lexical Chainers の使用方法 **5**
2.1 chainer_th の使い方 5
2.2 chainer_thcha の使い方 7
2.3 chainer_co の使い方 8
2.4 chainer_cocha の使い方 10

3 シソーラスデータ作成方法 **11**
3.1 分類語彙表を使用する場合 11
3.2 他のシソーラスを使用する場合 11

4 形態素フィルタルール記述ファイル **12**
4.1 形態素フィルタルール記述ファイルとは 12
4.2 形態素フィルタルールの説明 12

5 Lexical Chaiers の応用システム **15**

おわりに **15**

図 目 次

1 フィルタルール記述ファイルの例 13

表 目 次

1 使用条件とプログラム名の対応表 2

はじめに

本稿では，汎用連想計算エンジンＧＥＴＡにより高速に計算される単語の出現頻度情報を用いて単語間の語彙的結束性を計算し，テキスト内の語彙的連鎖を作成するシステム Lexical Chainers の最新プログラム,version 1.50.2 について説明する．

語彙的連鎖 (lexical chain) とは，文書中で語彙的結束性 (lexical cohesion) を持つ語の連続のことを言う．語彙的結束性とは，文と文の表層的な関係による結びつきである結束性 (cohesion)[1][2] の 1 つであり，文書中に出現する語と語の間にある結束性を指す．他の結束性である代名詞，省略などと異なり，関連する 2 つの語が両方とも文書中に存在することから，計算機による計算が比較的容易であるという特徴がある．

Lexical Chainers は，この語彙的結束性を持つ語の連続である語彙的連鎖を自動的に計算する．ただし，ある語とある語の間に語彙的結束性が存在するかどうかを決定するためには，ある基準が必要であり，その基準は 1 つとは限らない．そのため Lexical Chainers は，代表的な基準として,「同一の語の繰り返し」,「シソーラス」,「単語の共起関係」の 3 つを選んで用いることができる複数のプログラムのパッケージとなっている*.

この内,「単語の共起関係」は，シソーラス上には記述されていないが，意味的に関連のある語を計算する．そのため，特に未知語や固有名詞などが重要となるテキスト内での語彙的連鎖を計算する上で有効な基準であると考えられる．しかし，共起関係の解析には，一般に大規模なコーパスを用いた語の共起関係に関する統計情報を使用する必要がある．そのため，3 つの基準の内で最も計算が複雑であると同時に，実用上は，高速に共起計算を行なう必要がある．Lexical Chainers では，この共起計算に汎用連想計算エンジンＧＥＴＡを用いることで，高速な単語共起解析を行ない語彙的連鎖の計算を実現している．

文書中には複数の語彙的連鎖が存在し，それぞれの連鎖はある 1 つの話題に関連する語の連続であると捉えることができる．そのため，計算された語彙的連鎖は，その文書の文脈情報として用いることが可能である．

なお，本稿では語彙的結束性に基づく語彙的連鎖の説明とその計算方法については扱わないので，他の文献 [7] を参照されたし．

*パッケージ名が Lexical Chainers と複数形になっているのは，この理由による．

1 Lexical Chainers のインストール

ここでは，Lexical Chainers 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

- 動作環境
  Lexical Chainers は，UNIX 系 OS 上で動作する．

  Lexical Chainers には，4 種類の語彙的連鎖計算プログラムが含まれており，語彙的連鎖を計算する基準の違いで，大きく次の 2 つに分かれる．

  – 同一の語の繰り返し，またはシソーラスに基づくプログラム：
    chainer_th, chainer_thcha
  – 語の共起関係に基づくプログラム：
    chainer_co, chainer_cocha

  また，それぞれの基準で，プログラムに，ChaSen のライブラリを組込むものと，組込まないものがある． chainer_thcha, chainer_cocha が，ライブラリを組込むものである．

  プログラム名との対応を表 1. にまとめて示すので各自の目的に合わせてプログラムを選択して欲しい．

表 1: 使用条件とプログラム名の対応表

語彙的連鎖構成基準	形態素解析器	プログラム名	必要なもの
同語の繰り返し シソーラス	juman3.61(*1) ChaSen2.2.x(*2) その他 (*4)	chainer_th	シソーラス (*3)
語の共起	上と同じ	chainer_co	GETA(*5)
同語の繰り返し シソーラス	ChaSen2.2.x	chainer_thcha	ChaSen2.2x シソーラス
語の共起	上と同じ	chainer_cocha	ChaSen2.2x GETA

上記各プログラムの動作に必要な他のアプリケーション等は以下のようになる．

(*1) juman3.61
JUMAN[5] 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)

(*2) ChaSen 2.2.x
ChaSen「茶筌」[8] 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)

(*3) シソーラス
現在のところ，国立国語研究所の分類語彙表の索引を使うのが一番簡単な方法である．なお，後述する指定形式にあわせてファイルを用意することで，別のシソーラスも利用可能である．(詳しくは 3節を参照せよ)

(*4) その他
形態素解析結果の形式を ChaSen や JUMAN と同様にし，フィルタを用意することによって，JUMAN,ChaSen 以外の形態素解析器の処理結果も利用可能である．

(*5) GETA
GETA は，(株) 日立製作所中央研究所において開発されている，汎用連想検索エンジンである．そのため別途取得，インストールする必要がある．GETA の入手等については，(株) 日立製作所中央研究所の丹羽さん (yniwa@harl.hitachi.co.jp) に直接コンタクトされたし (2002 年 1 月 10 日現在)．

- 配布プログラムの取得
  - 以下の配布元より Lexical Chainers version 1.50.2 をダウンロードする．
    http://galaga.jaist.ac.jp:8000/~motizuki/software/chainers/

1.2 インストール

1. 準備
   - 語の共起に基づく語彙的連鎖を作成したい場合 (chainer_co または chainer_cocha) は，GETA をインストールする．
   - ChaSen ライブラリを取り込んだ語彙的連鎖プログラムを使う場合は ChaSen をインストールする．
2. インストール先の決定
   Lexical Chainers をインストールするディレクトリを決める．以下，$MOTSROOT と表記する．また，プログラムソースがあるディレクトリを$MOTSSRC とする．
3. $MOTSSRC/conf/conf を希望に合わせ編集する．
   - 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．指定しなければ，/usr/local がデフォルトとして選択される．(全プログラム共通で使用)

- GETAROOT
  GETA の存在するディレクトリを指定する．GETA を使う場合のみ指定すること．(chainer_co, chainer_cocha で使用)
- 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．(chainer_thcha, chainer_cocha で使用)

4. $MOTSSRC 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

5. make を実行する
   % make

   chainer_th, chainer_thcha, chainer_co, chainer_cocha の内，chainer_th は必ず作成される．後のものは conf/conf で希望したものだけが作成される．

6. 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 Lexical Chainers の使用方法

ここでは，Lexical Chainers の使用方法をプログラム別に説明する．

2.1 chainer_th の使い方

1. シソーラスの準備
   このプログラムは，シソーラスデータがないと動かないので作成する．分類語彙表などのシソーラスがない場合は，とりあえず dummy を使ってデータを作成する．

   % cd $MOTSROOT/data
   % $MOTSROOT/sbin/mkidx.sh dummy dummy

   正式なシソーラスデータの作成方法は，3節「シソーラスデータ作成方法」を参照せよ．

2. 使用方法
   プログラムは以下のように使用する．

   % ./chainer_th [options] < 形態素解析結果ファイル
   (例: chasen -e TEXTFILE | ./chainer_th [options])
   (例: juman -e TEXTFILE | ./chainer_th -jum [options])

   オプションの説明：

   - 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[4] 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出力形式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．

   - 連鎖を作る候補語に関するオプション

     -F <0-6> ： デフォルト値=0

連鎖を作る候補語となる品詞を指定する．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

- 概念階層のレベル設定
  階層は使用するシソーラスに依存し，最大 5 階層まで扱うことができる†.

  -i <0-5> ：デフォルト値=2
  シソーラスの概念階層の 何レベルまで 使用するかを指定する．
  0 の場合は，シソーラスを使用しないで，同じ語の繰り返しのみになる．

  -j <0-5> ：デフォルト値=0
  シソーラスの概念階層の 何レベルを 使用するか指定する．
  (-i とは異なり，指定したレベルのみを使う)．
  0 の場合は，このオプションを無効にする．

- シソーラスの設定に関するオプション

  -D <index> ：
  使用するシソーラスを指定する．
  ただし事前に$MOTSROOT/data/の下に，シソーラスを加工したファイルを作成しておく必要がある．詳しくは，3節「シソーラスデータ作成方法」を参照せよ．

- 出力に関するオプション

  -t <INT> ：デフォルト値=1
  何語以上連なったら有効な語彙的連鎖として出力するかを指定する．

  -1 ：多義性が完全に解消できない場合に，最初の 1 語義だけを表示する．

  -V ：処理中にメッセージを出す.

  -W ：連鎖の候補となる単語以外も単語数にカウントする．

  -h ：help を表示する．

3. 出力形式

   出力の 1 行目は，{ 総文数 総候補語数 総形態素数 } を表わす.

   2 行目以降は，1 レコードが 1 単語の情報を表し，複数レコードによって 1 種類の語彙的連鎖を表わしている．1 つの語彙的連鎖は空行の次のレコードから空行までである．1 レコードの形式は以下のようになる．

†プログラムは 5 階層に対応しているが，シソーラスデータ作成プログラムが 4 階層までしか対応していないため，現在は実質 4 階層までしか扱えない

- 同じ語の繰り返しに基づく場合 (-i 0 で chainer_th を使用)
  { 文番号 語番号 通し番号 見出し語 見出し語 付属する助詞 助詞 }
  例：
  5 9 63 要求 要求 kaku を
  19 11 249 要求 要求 kaku が
  (空行)
- シソーラスに基づく場合
  { 文番号 語番号 通し番号 概念番号 見出し語 付属する助詞 助詞 }
  例：
  3 7 45 165 難しい yougen no
  19 13 251 165 困難 yougen no
  (空行)

「通し番号」は -W オプションの有無に依存する．
-W なし：文書の先頭から現在までに出現した候補語の通し番号を表す．
-W 付き：文書の先頭から現在までに出現した形態素の通し番号を表す．

「付属する助詞の分類」は，その単語が名詞である場合は直後の助詞の種類を示し，動詞か形容詞である場合には yougen(用言) と記述されている．

助詞の分類と意味の対応は，$MOTSROOT/rule/mrule0 や mrule1 あるいは自分で指定したフィルタルール記述ファイルの助詞の項目に記述されているとおりに出力される‡．

2.2 chainer_thcha の使い方

1. シソーラスの準備
   このプログラムは，シソーラスデータがないと動かないので作成する．分類語彙表などのシソーラスがない場合は，とりあえず dummy を使ってデータを作成する．

   ```
   % cd $MOTSROOT/data
   % $MOTSROOT/sbin/mkidx.sh dummy dummy
   ```

   正式なシソーラスデータの作成方法は，3節「シソーラスデータ作成方法」を参照せよ．

2. 使用方法
   プログラムは以下のように使用する．

‡形態素解析器により助詞の分類が異なるので，統一することはせず，フィルタルール記述ファイルを設けて対応することにした．

% ./chainer_thcha [options] < TEXTFILE

オプションの説明：

- 入力タイプ
  ChaSen ライブラリを使用するので，入力タイプは -cha のみになる．
- それ以外のオプションは，chainer_th と同じ．

3. 出力フォーマット
   出力フォーマットは，chainer_th と同じ．

2.3 chainer_co の使い方

1. 使用方法
   プログラムは以下のように使用する．

% ./chainer_co handle [options] < 形態素解析結果のファイル

(例: chasen -e TEXTFILE | ./chainer_co mai94)
(例: juman -e TEXTFILE | ./chainer_co mai94 -jum)

ここで handle とは，GETA の handle を意味する．

オプションの説明：

- 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出力形式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．
- 連鎖を作る候補語に関するオプション

-F <0-6> ： デフォルト値=0

連鎖を作る候補語となる品詞を指定する．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

- 共起計算に関するオプション

-handle <str> ：
GETA で使用するコーパスの handle name を指定する．

-calc_mode <num> ：
共起スコアの計算式を指定する．
1 . TF.IDF： $df(x,y) \times log(N/(df(x) \times df(y)))$
2 . 相互情報量： $df(x,y) \times N/(df(x) \times df(y))$
10 . コサイン距離：$\sum(xi \times yi)/\sqrt{x_i{}^2} \times \sqrt{y_i{}^2}$

-threshold <num> ：
クラスタ (連鎖) 作成時の共起スコアの閾値を指定する．

-N <INT> ：
共起スコア計算に使用するコーパスの総文書数．

- 出力に関するオプション

-compth ：chainer_th と compatible な出力にする．

-W ：連鎖の候補となる単語以外も単語数にカウントする．

-h ：help を表示する．

2. 出力形式
-compath オプション付きで実行した場合は，chainer_th と同じ．

それ以外の場合は次のようになる．

出力の 1 行目はテキストの，{ 総文数 総候補語数 総形態素数 } を表わす．

2 行目以降は，1 レコードが 1 単語の情報を表し，複数レコードによって 1 種類の語彙的連鎖を表わしている．1 つの語彙的連鎖は空行の次のレコードから空行までである．
1 レコードの形式は以下のようになる．

{ 文番号 語番号 通し番号 見出し語 付属する助詞の分類 助詞 }
例：
1 10 10 携帯 no no

1 12 12 電話 no no
(空行)

「通し番号」は -W オプションの有無に依存する.
-W なし : 文書の先頭から現在までに出現した候補語の通し番号を表す.
-W 付き : 文書の先頭から現在までに出現した形態素の通し番号を表す.

「付属する助詞の分類」は，その単語が名詞である場合は直後の助詞の種類を示し，動詞か形容詞である場合には yougen(用言) と記述されている.

助詞の分類と意味の対応は，$MOTSROOT/rule/mrule0 や mrule1 あるいは自分で指定したフィルタルール記述ファイルの助詞の項目に記述されているとおりに出力される[§].

2.4 chainer_cocha の使い方

1. 使用方法
   プログラムは以下のように使用する.

   % ./chainer_cocha handle [options] < TEXTFILE

   ここで handle とは，GETA の handle を意味する.

   オプションの説明：

   ChaSen ライブラリを使用するので，入力タイプは-cha のみになる.

   それ以外のオプションは，chainer_co と同じ.

2. 出力形式

   -compath オプション付きで実行した場合は，chainer_th と同じ.

   そうでなければ，chainer_co と同じ.

[§]形態素解析器により助詞の分類が異なるので，統一することはせず，フィルタルール記述ファイルを設けて対応することにした.

3　シソーラスデータ作成方法

ここでは，chainer_th と chainer_thcha で参照するシソーラスデータの作り方を説明する．なお，$MOTSROOT は，Lexical Chainers をインストールしたルートディレクトリを意味する．また，シソーラスのための作業ディレクトリは $MOTSROOT/data である．

3.1　分類語彙表を使用する場合

分類語彙表の場合は，索引データである 'sakuin' ファイルを用いる．なお，sakuin が $BGH にあるものとして説明する．

1. 作業ディレクトリに移動する．

   % cd $MOTSROOT/data

2. nkf などを用いて分類語彙表の索引 (sakuin) を EUC に変換する．

   % nkf -s -e < $BGH/sakuin > sakuin.euc

3. シソーラスデータを作成する．
   シソーラスデータは，スクリプト$MOTSROOT/sbin/mkidx.sh を使って作成する．

   % $MOTSROOT/sbin/mkidx.sh bgh sakuin.euc

4. データの確認をする

   ここまでの作業により，$MOTSROOT/data には，bgh.dat, bgh.wan, bgh.idx の 3 つのファイルが作成される．

これにより，chainer_th, chainer_thcha にオプションとして -D bgh を付けることで，分類語彙表がシソーラスデータとして使用できるようになる．

3.2　他のシソーラスを使用する場合

基本的に，分類語彙表の索引データと同様のフォーマットになるように使用するシソーラスを加工することで，利用できるようになる．

1. ファイルの準備
   以下のフォーマットでファイルを作成する．文字コードは EUC で作成すること．

   見出し語, よみ,ID1,ID2,ID3,ID4

   例： dummy ファイルの中身
   だ, だ,0.0000,0,0,0
   み, み,0.1000,0,0,0
   い, い,0.2000,0,0,0

ファ, ファ,0.1000,0,0,0
イ, イ,0.0000,0,0,0
ル, ル,0.0000,0,0,0

2. 上記のファイルからシソーラスデータを作成する.

   % $MOTSROOT/sbin/mkidx.sh sample dummy

3. データの確認をする.

   ここまでの作業により，$MOTSROOT/data には，sample.dat sample.wan sample.idx が作成される.

これにより，chainer_th, chainer_thcha にオプションとして -D sample を付けることでこのファイル (シソーラス) が利用できるようになる.

4 形態素フィルタルール記述ファイル

ここでは，フィルタルール記述ファイルとその記述方法について説明する.

4.1 形態素フィルタルール記述ファイルとは

「形態素フィルタルール記述ファイル」とは，$MOTSROOT/rules/mrule* ファイルのことであり，以下の働きをする.

- 語彙的連鎖を作成する際に，形態素解析結果の中で有効な単語を決定する.(除外する単語の指定)
- 助詞の分類とその意味の対応を決める.

Lexical Chainers のデフォルトとして用意してあるフィルタルール記述ファイルは，以下の 2 つである.

- ChaSen 2.2.x 用 : $MOTSROOT/rules/mrule0
- juman3.61, knp 用 : $MOTSROOT/rules/mrule1

Lexical Chainers では，-rule オプションにより，ユーザ独自の形態素フィルタルール記述ファイルをプログラムに渡すことができるので，フィルタを自作することも可能である.

4.2 形態素フィルタルールの説明

例として，ChaSen 2.2.x 用のフィルタである$MOTSROOT/rules/mrule0 をあげ説明する (図 1).

```
# action\t filed1\t 表記 1(\t filed2\t 表記 2...)\n
#
#
; 名詞
6               3   名詞
; 形容詞
3               3   形容詞
; 動詞
2               3   動詞
-1              2   する
-1              2   ある
-1              2   なる
-1              2   できる
-1              2   よる
-1              2   とする
-1              0   よって
-1              0   より
-1              0   つれて
-1              0   ついて
-1              0   という
-1              0   した
; 助詞
9               3   助詞
kaku            4   格助詞
shu             4   終助詞
setsuzoku       4   接続助詞
kakari          4   係助詞
tokushu         4   特殊
fukushika       4   副詞化
fuku            4   副助詞
heiritsu        4   並立助詞
rentaika        4   連体化
; 未定義語
6               1   未知語
#================================
```

図 1: フィルタルール記述ファイルの例

- 先頭が「#」で始まる行は無視される.
- 先頭が「;」で始まる行はプログラム内部で区別する品詞についての情報の記述開始を意味する.
  この例では,「名詞」,「形容詞」,「動詞」,「助詞」,「未定義語」という 5 つの品詞を内部的な品詞として用いる.

- 「; 品詞」の直後の行は形態素解析における表記との対応を示す.
  この行の形式は { コード \t フィールド番号 \t 表記 } である.

  ここで,「コード」はプログラム内部で扱う品詞 (直前の行で指定しているもの) に割りあてるコードを意味する.「フィールド番号」と「表記」は形態素解析の結果における何番目の「フィールド」がどのような「表記」になっている場合が対応するかを示している.

  例：
  ; 名詞
  6　3　名詞

  この 2 行の解釈は，形態素解析の結果の 3 フィールド目が「名詞」と表記された場合，内部的に「名詞」であり「6」というコードを割りあてる，と解釈される．また,「形容詞」=3,「動詞」=2,「助詞」=9 である．なお,「未定義語」は 6 であり，これは名詞と同じように扱うことを意味する.

- 3 行目以降 (が存在する場合) は，以下の形式で記述される.

  {action\t フィールド番号 1\t 表記 1(\t フィールド番号 2\t 表記 2\t...)}

  ここで「フィールド番号」と「表記」は，形態素解析結果の何番目のフィールドがどのような表記である場合に「action」で表わされる性質と一致するかを意味する.

  「action」の意味は内部で扱う品詞の種類により次のように異なる.

  - 名詞，形容詞，動詞，未定義語では除外する語を指定する．action=-1
  - 助詞の場合は，助詞の分類とその意味の対応を決める．action=分類名

  例：
  ; 動詞
  2　3　動詞
  -1　2　する
  -1　2　ある
  -1　2　なる

  これは，動詞の 3 行目以降，形態素解析の結果の 第 2 フィールドが「する」「ある」「なる」... の場合は action -1 と解釈する (この場合除外を意味する) ことを表わしている.

  また，タブで区切って 「フィールド番号 \t 表記」の組を続ければ，条件を厳しくすることができる.

例：条件を厳しくした場合
; 名詞
6　3　名詞
-1　4　接尾　2　的

このようにすると，名詞-接尾の中で「的」という語だけを削除することになる．

助詞の場合は次のようになる．

例：助詞の場合
; 助詞
9　　3　助詞
kaku　4　格助詞
shu　　4　終助詞
　　…

ここで kaku や shu という表記が最終的な語彙的連鎖の出力にも表われることになる．また，タブで区切って「フィールド番号 \t 表記」の組を続ければ，分類を細かくすることができる．

例：助詞の条件を細分化した場合
; 助詞
9　　3　助詞
gakaku　4　格助詞　2　が
wokaku　4　格助詞　2　を

5　Lexical Chaiers の応用システム

Lexical Chainers によって計算され出力される語彙的連鎖データそのものは，語彙的結束性を持つ単語のグループであるので何かに直接利用できるというものではない．しかし，談話構造解析などの自然言語処理の応用分野における基礎的なデータとして広く利用可能である．これまでに Lexical Chainers を用いた研究として，パッセージ検索 [3]，や自動要約 [6]，テキストセグメンテーションなどがある．この内，自動要約については，テキスト簡易要約器 Posum としてプログラムが公開されており，次の URL より入手可能である．

http://galaga.jaist.ac.jp:8000/~motizuki/software/posumcl/

おわりに

本稿では，汎用連想計算エンジンＧＥＴＡにより高速に計算される単語の出現頻度情報を用いて単語間の語彙的結束性を計算し，テキスト内の語彙的連鎖を作成するシステム Lexical Chainers の最新プログラム,version 1.50.2 について説明した．

参考文献

[1] H.A.K. Halliday and R. Hasan. *Cohesion in English*. Longman, 1976.

[2] M. Okumura and T. Honda. Word sense disambiguation and text segmentation based on lexical cohesion. In *Proc. of the 15th International Conference on Computational Linguistics*, pp. 755–761, 1994.

[3] 望月源, 岩山真, 奥村学. 語彙的連鎖に基づくパッセージ検索. 自然言語処理, Vol. 6, No. 3, pp. 101–126, 1999.

[4] 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.

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

[6] 望月源. テキスト簡易要約器 Posum version 1.50.2 マニュアル. JAIST Technical Memorandum, IS-TM-2002-002, 2002.

[7] 望月源, 岩山真, 奥村学. 語彙的結束性に基づく語彙的連鎖の計算. JAIST Technical Memorandum, IS-TM-2000-002, 2000.

[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.