テキスト情報処理に関する演習
(研究室での演習編)

森 辰則

mori@forest.eis.ynu.ac.jp

注意:　座学の資料と合わせて読んでください。

演習の内容 (全体)

- １～数万記事の新聞記事を対象とする。
  - XML形式で記述された実際の2紙を予定。
- テキストに対する前処理として、EUC-JPエンコーディングの一行一文形式に変換。
- 既存の形態素解析器、係り受け解析器を用いて、基本的な解析を行う。
- その結果から、
  - 既存のフィルタプログラムを用いて、形態素や品詞の統計量を調べてみる。
  - 文節や係り受け関係の情報を取り出すためのフィルタプログラムを自作。文節の頻度等の統計量を調べてみる。
- 情報抽出に関する演習を行う。
  - 用語の定義・説明を抽出する。
    - 形態素結果を用いて、「…とは、～。」(「と」と「は」の品詞は、それぞれ、「助詞-格助詞-一般」、「助詞-係助詞」)という形式を持つ文をすべて取り出す。

計算機環境の利用方法

1. ユーザ名とパスワードが記された用紙を受け取る。後ほど返却のこと。
2. 指定されたPCの前に着席。
3. 上記ユーザ名とパスワードにより、Windowsにログオン。
   - これで、Windows上で行うことが一通りできます。Webブラウザも利用可能。
4. TeraTermを起動し、Unixシステム(Solarisマシン)にログイン。
   - 「スタート」⇒「すべてのプログラム」⇒「…TeraTerm Pro…」⇒「…TeraTerm Pro…」を選択
   - 起動したTeraTermにおいて、上記ユーザ名とパスワードを入力すると、Unixシステムに接続して利用可能となる。
   - エディタはemacs等が利用可能。emacsの中でかな漢字変換を行うためには、Ctrl+¥ (コントロールキーを押しながらバックスラッシュ)。もう一度押すと英数字入力に戻ります。

※ 演習はUnixシステムの上で行います。

第1日目

- 演習1: テキストに対する前処理
- 演習2: ChaSen用いた形態素解析
- 演習3: CaboChaを用いた係り受け解析
- 演習4: 既存のフィルタプログラムを用いた簡単な統計処理
- 第1日目のレポート執筆のための、レポート内容の整理と、データ整理等

- 時間が余ったら、第2日目の演習に進む

演習1:テキストに対する前処理

- 目的
  - テキストデータに対する前処理の役割を理解し、既存のフィルタプログラムの組み合わせで簡単な前処理が実現できることを理解する。
- 演習
  - 対象となるテキストデータの格納されているファイル。
    - /home/foster3/CompSciSEx/10/Corpus/mai99_0201-0210.xml
    - /home/foster3/CompSciSEx/10/Corpus/yomi99_0201-0210.xml
  - 上記テキストデータの入ったファイルを入力として、EUC-JPエンコーディングの一行一文形式に変換し、ファイルに保存する。
    - 末尾の資料を参考にして、いくつかのフィルタプログラムを組み合わせて実現する。
    - 変換結果の確認には、ファイルの内容を画面表示するlessなどが使える。
- 結果の記録
  - 実際に行ったフィルタプログラムの組み合わせ方を記録。
  - 演習結果の整理。レポート作成のための材料収集。
- 考察
  - 失敗事例に対する分析
    - 失敗箇所について、本来の正しい解析結果がどうなるべきかを提示
    - 失敗箇所について、処理の問題点を指摘
    - 失敗箇所について、どのような処理が追加できれば問題点が解決できるのかを指摘し、可能であれば、改善。

<DOC>
<DOCNO>JA-990201001</DOCNO>
<LANG>JA</LANG>
<SECTION>１面</SECTION>
<AE>無</AE>
<WORDS>772</WORDS>
<HEADLINE>［余録］京都の東寺は、いうまでもなく…</HEADLINE>
<DATE>1999-02-01</DATE>
<TEXT>
　京都の東寺は、いうまでもなく、真言宗を開いた弘法大師ゆかりの大寺で、密教文化の宝庫として名高い。不動明王をはじめ恐ろしいお顔の密教仏が七堂伽藍（がらん）にずらりと並ぶ▲そればかりか、東寺は１６世紀まで宗教的権威と大荘園領主の二面をもっていたから、聖・俗双方にわたる膨大な古文書が伝わった。その一部が、…
(中略)
…という。中世の社寺では不倫など珍しくもないし名誉な話でもない。それを、なぜ東寺は後生大事に伝えようとしたのか。ところで、今世紀最大の不倫弾劾裁判は、どうなるのだろう。
</TEXT>
</DOC>

京都の東寺は、いうまでもなく、真言宗を開いた弘法大師ゆかりの大寺で、密教文化の宝庫として名高い。
不動明王をはじめ恐ろしい…
(中略)
…という。
中世の社寺では不倫など珍しくもないし名誉な話でもない。
それを、なぜ東寺は後生大事に伝えようとしたのか。
ところで、今世紀最大の不倫弾劾裁判は、どうなるのだろう。

演習2: ChaSen用いた形態素解析

- 目的
  - 形態素解析器ChaSenを実例として、形態素解析器が行う処理について理解する。
- 演習
  - 演習1で作成した一行一文形式のテキストデータファイルを入力とし、ChaSenを動作させ、形態素解析結果をファイルに保存する。
  - ChaSenの動かし方
    - chasen -o outputfilename inputfilename
- 結果の記録
  - 出力ファイルの中身を調べ、特徴的な解析例をいくつか採取する。
  - ChaSenの出力については、末尾の資料を参照。
- 考察
  - 失敗事例に対する分析
    - 失敗箇所について、本来の正しい解析結果がどうなるべきかを提示
    - 失敗箇所について、処理の問題点を指摘

演習3: CaboCha用いた係り受け解析

- 目的
  - 係り受け解析器CaboChaを実例として、係り受け解析器が行う処理について理解する。
- 演習
  - 演習1で作成した一行一文形式のテキストデータファイルを入力とし、CaboChaを動作させ、係り受け解析結果をファイルに保存する。
  - CaboChaの動かし方
    - 木形式の場合: cabocha -o outputfilename inputfilename
    - 表形式の場合: cabocha -f 1 -o outputfilename inputfilename
- 結果の記録
  - 出力ファイルの中身を調べ、特徴的な解析例をいくつか採取する。
  - CaboChaの出力については、末尾の資料を参照。
- 考察
  - 失敗事例に対する分析
    - 失敗箇所について、本来の正しい解析結果がどうなるべきかを提示
    - 失敗箇所について、処理の問題点を指摘

演習4: 既存のフィルタプログラムを用いた簡単な統計処理(1)

- 目的
  - 形態素解析結果を既存のフィルタプログラムで解析・整理することにより、与えられたテキストに関する簡単な統計量を求めることができることを理解する。
- 演習
  - 与えられた新聞記事すべてを対象とする。
  - ChaSenの出力について、既存のフィルタプログラムのみを使って次の統計量を求める。新聞2紙を結合して行った場合と、それぞれ単独で行った場合の3種類を行う。
  - 形態素の出現頻度
    - 頻度上位100件の形態素を求める。
    - 頻度5以上の形態素を求める。
  - 品詞の出現頻度
    - 頻度上位100件の品詞を求める。
    - 頻度5以上の品詞を求める。

ヒント

各行がタブ等で区切られた項目からなる場合、各行のn番目の項目だけを抜き出すにはどうするか？

座学のレポート課題になっていた調査対象のコマンドにありましたよ！

演習4: 既存のフィルタプログラムを用いた簡単な統計処理(2)

- 結果の記録
  - 結果をファイルに記録し、レポートに報告するための情報を集める。一部の出力は大きくなるので、そのまま付録にするのには適さない。
- 考察
  - 新聞2紙を結合して行った場合と、それぞれ単独で行った場合の3種類について、頻度分布の類似点、相違点を考察せよ。

第2日目

- 演習5: 簡単なフィルタプログラムの作成
  - フィルタ1: 各文節の文字列と係り先情報を取り出す
  - フィルタ2: 文節の係り元、係り先情報を取り出す
  - フィルタ3: 固有表現を取り出す
- 演習6: 作成したフィルタプログラムを用いた簡単な統計処理
- 演習7: 簡単な情報抽出処理
- 第2日目のレポート執筆のための、レポート内容の整理と、データ整理等

※ フィルタ3の作成、ならびに、これを用いた演習については、余裕があれば、で結構です。

※ 演習7についても、余裕があれば、で結構です。

演習5: 簡単なフィルタプログラムの作成(1)

- 目的
  - 各種既存プログラムが出力する結果を独自の視点で解析するために必要となるフィルタプログラムを作成し、テキスト処理の適用範囲を拡大する手法を理解する。
- 演習
  - 次に示す動作をするフィルタプログラムの処理アルゴリズムを考える。
  - 作成するプログラムの仕様(入力の仕様、出力の仕様のおおよそは決まっているが更に詳細について)を定義する。
  - C言語、Java等、みなさんが使えるプログラミング言語によりフィルタプログラムを作成する。森研究室で利用可能なプログラミング言語には、以下のものを含む複数のものがある(括弧内がコマンド名)。エディタには、emacs, vi等が使える。
    - C(gcc), C++(g++), Java, Perl 5 (perl5.8.9), Ruby 1.6.8 (ruby), Python 2.5 (python2.5) 他
- 結果の記録
  - プログラムリストを付録とするとともに、代表的な例について動作結果を示す。
- 考察
  - 採用した処理アルゴリズムについて説明するとともに、作成したプログラムが最初に決めた仕様を満足するものであったかどうかを調べ、考察せよ。
  - 改善すべき点があれば、指摘をせよ。

演習5: 簡単なフィルタプログラムの作成(2)

- フィルタ1: CaboChaの表形式出力から、文節の出現順に、以下の情報を取り出す。ただし、文末を表すEOSはそのまま出力する。ただし、■は空白を表す。

  文節番号■係先番号■文節文字列

- フィルタ2: フィルタ1の出力を受け、係り受け関係にある文節の組のリストを以下の形式で出力する。ただし、文末を表すEOSはそのまま出力する。

  係り元文節■係り先文節

- フィルタ3: CaboChaの表形式出力から、固有表現の出現順に、以下の情報を取り出す。ただし、文末を表すEOSはそのまま出力する。

  開始形態素番号■固有表現■固有表現の種類

演習5: 簡単なフィルタプログラムの作成(3) : フィルタ1の動作例

```
* 0 3D 1/2 3.90668393
鳩山        ハトヤマ        鳩山    名詞-固有名詞-人名-姓          B-PERSON
首相        シュショウ      首相    名詞-一般                      O
は          ハ              は      助詞-係助詞                    O
* 1 3D 1/2 5.32724304
首相        シュショウ      首相    名詞-一般                      B-LOCATION
官邸        カンテイ        官邸    名詞-一般                      I-LOCATION
で          デ              で      助詞-格助詞-一般               O
* 2 3D 1/2 0.00000000
オバマ                              未知語                         B-PERSON
大統領      ダイトウリョウ  大統領  名詞-一般                      O
と          ト              と      助詞-格助詞-一般               O
* 3 -10 1/2 0.00000000
会見        カイケン        会見      名詞-サ変接続                O
し          シ              する    動詞-自立     サ変・スル連用形 O
た          タ              た      助動詞        特殊・タ基本形   O
。          。              。      記号-句点                      O
EOS
```

※一つのファイルには複数の文がはいっており、それゆえ、EOSが複数回現れるのが普通であることに注意。この例はたまたま、一文しかファイルに入っていなかった例である。複数文ある場合にはその出現順に結果をEOSで区切って出力すること。

```
0 3 鳩山首相は
1 3 首相官邸で
2 3 オバマ大統領と
3 -1 会見した。
EOS
```

演習5: 簡単なフィルタプログラムの作成(4) : フィルタ2の動作例

```
0 3 鳩山首相は
1 3 首相官邸で
2 3 オバマ大統領と
3 -1 会見した。
EOS
```

```
鳩山首相は 会見した。
首相官邸で 会見した。
オバマ大統領と 会見した。
EOS
```

※一つのファイルには複数の文がはいっており、それゆえ、EOSが複数回現れるのが普通であることに注意。この例はたまたま、一文しかファイルに入っていなかった例である。複数文ある場合にはその出現順に結果をEOSで区切って出力すること。

演習5: 簡単なフィルタプログラムの作成(5) : フィルタ3の動作例

```
* 0 3D 1/2 3.90668393
鳩山        ハトヤマ        鳩山    名詞-固有名詞-人名-姓          B-PERSON
首相        シュショウ      首相    名詞-一般                      O
は          ハ              は      助詞-係助詞                    O
* 1 3D 1/2 5.32724304
首相        シュショウ      首相    名詞-一般                      B-LOCATION
官邸        カンテイ        官邸    名詞-一般                      I-LOCATION
で          デ              で      助詞-格助詞-一般               O
* 2 3D 1/2 0.00000000
オバマ                              未知語                         B-PERSON
大統領      ダイトウリョウ  大統領  名詞-一般                      O
と          ト              と      助詞-格助詞-一般               O
* 3 -10 1/2 0.00000000
会見        カイケン        会見      名詞-サ変接続                O
し          シ              する    動詞-自立    サ変・スル連用形  O
た          タ              た      助動詞       特殊・タ基本形    O
。          。              。      記号-句点                      O
EOS
```

※一つのファイルには複数の文がはいっており、それゆえ、EOSが複数回現れるのが普通であることに注意。この例はたまたま、一文しかファイルに入っていなかった例である。複数文ある場合にはその出現順に結果をEOSで区切って出力すること。

⇩

```
0 鳩山 PERSON
3 首相官邸 LOCATION
6 オバマ PERSON
EOS
```

演習5: 簡単なフィルタプログラムの作成(6): C言語のプログラムを書くときのヒント

- char *fgets(char *s, int n, FILE *f)
  - オープンしたファイルfから文字配列sに一行読み込む。ただし、最大でもn-1文字を超えないように読み込むので、長い行は複数に分けて読み込まれる。
- sscanf(char *s, char *format, …)
  - scanf()と同じであるが、標準入力ではなくて、文字配列sから書式に従って読み込む。
- char *strcat(char *s1, char *s2)
  - 文字配列s1にある文字列の末尾にs2にある文字列の内容をコピーしてつなげる。
- size_t strlen(char *s)
  - 文字配列sの長さ(バイト数)を返す。末尾の'¥0'は含まない。
- int strcmp(char *s1, char *s2)
  - 二つの文字列s1, s2を比較して、s1のほうが辞書順で後ならば正の整数、辞書順が同じならば0、辞書順で前ならば、負の整数を返す。

※いずれも、unix上で
man　関数名
とすると使い方がわかります。Webを調べても結構です。

演習6: 作成したフィルタプログラムを用いた簡単な統計処理(1)

- 目的
  - 独自に作成したフィルタプログラムを既存のフィルタプログラムと組み合わせることにより、独自の統計量を求める手法を理解する。
- 演習
  - 与えられた新聞記事すべてを対象とし、次の各統計量を求める。
  - 新聞2紙を結合して行った場合と、それぞれ単独で行った場合の3種類を行う。
- 結果の記録
  - 結果をファイルに記録し、レポートに報告するための情報を集める。一部の出力は大きくなるので、そのまま付録にするのには適さない。
- 考察
  - 新聞2紙を結合して行った場合と、それぞれ単独で行った場合の3種類について、頻度分布の類似点、相違点を考察せよ。

演習6: 作成したフィルタプログラムを用いた簡単な統計処理(2)

- フィルタ1と既存のフィルタを用いて以下のものを求める。
  - 文節の出現頻度
    - 頻度上位100件の文節を求める。
    - 頻度5以上の文節を求める。
- フィルタ2と既存のフィルタを用いて以下のものを求める。
  - 係り受けの出現頻度
    - 頻度上位100件の係り受け関係(係り元文節と係り先文節の組)を求める。
    - 頻度5以上の係り受け関係を求める。

演習6: 作成したフィルタプログラムを用いた簡単な統計処理(3)

- ある特定の表現を決め、それに係っている係り元文節の出現頻度
  - 頻度上位100件の係り元文節を求める。
  - 頻度2以上の係り元文節を求める。
- ある特定の表現を決め、それが係っている係り先文節の出現頻度
  - 頻度上位100件の係り先文節を求める。
  - 頻度2以上の係り先文節を求める。

- フィルタ3と既存のフィルタを用いて以下のものを求める。
  - 固有表現の出現頻度
    - 頻度上位100件の固有表現を求める。
    - 頻度3以上の固有表現を求める。

演習7:簡単な情報抽出処理

- 目的
  - 各種解析器の出力を用いて、特定の情報を抽出する手法を理解する。
- 演習
  - 新聞記事より、用語の定義・説明をしている文を抽出することを例とする。
  - 形態素結果を入力として、「…とは、～。」(「と」と「は」の品詞は、それぞれ、「助詞-格助詞-一般」、「助詞-係助詞」)という形式を持つ文のみを出力するプログラムを作成する。
  - 与えられた新聞記事すべてを対象として、プログラムを動作させる。
- 結果の記録
  - 結果をファイルに記録し、レポートに報告するための情報を集める。一部の出力は大きくなるので、そのまま付録にするのには適さない。
- 考察
  - 失敗事例に対する分析
    - 上記の手法により、「用語の定義・説明をしている文」を抽出するという当初の目的が達せられたかを調べ、失敗事例を分析する(誤って採用したもの、誤って取りこぼしたもの)。
    - 改善案を考察し、余裕があれば、その効果について検討する。

資料

日本語テキストに対する一般的な処理手順

1. テキストに対する前処理---文字列処理
   i. 文字エンコーディングの統一
   ii. 不要な記号類の除去
   iii. 「一行一文」化
2. 基本的な自然言語処理
   i. 形態素解析
   ii. 係り受け解析
   iii. 固有表現抽出
3. 目的に応じて実際に行いたい処理
   - 演習では、
     - 確率的モデルを導出するための基本として、各種言語表現の登場回数(頻度)の計算
     - 情報抽出の基本として、ある言語パタンに照合する文の抽出。

※以下では、OSとして、Unix系列のものを仮定する。

テキストに対する前処理

入力となるテキストの例

- 新聞記事の場合

XMLに基づくタグにより、情報が付与されているテキストを扱うことが多い。

<DOC>
<DOCNO>980713345</DOCNO>
<SECTION>総合</SECTION>
<AE>無</AE>
<WORDS>184</WORDS>
<HEADLINE>[選挙]98参院選　両国関係は継続－－参院選の結果に関連し、米国連大使</HEADLINE>
<TEXT>
【ワシントン12日岸本正人】リチャードソン米国連大使は12日、CNNテレビに出
（中略）
に関係なく両国関係は変わらないとの見方を強調した。参院選が自民党から野党への政権交代を伴うものでないことを踏まえて発言したものとみられる。
</TEXT>
</DOC>

- このような情報がたくさん収録されたファイル

文字エンコーディング

- ある言語の文字を符号化する方法
  - ある一文字をどのようなデータ列(バイト列)で表現するか。
- 日本語に関連するエンコーディング
  - JIS (ISO-2022-JP)
    - 日本語
  - Shift_JIS
    - 日本語
  - EUC-JP
    - 日本語
  - UTF-8 (Unicode)
    - 一つのエンコーディングに複数の言語の文字セットが入っている。

16進数で文字列データの中身を表示すると

JIS (55 bytes)

```
1B 24 42 46 7C 4B 5C 38 6C 1B 28 42 20 20 1B 24
42 25 28 25 73 25 33 21 3C 25 47 25 23 25 73 25
30 1B 28 42 20 32 30 30 39 0A 09 1B 24 42 24 47
24 39 21 23 1B 28 42
```

Shift_JIS (37 bytes)

```
93 FA 96 7B 8C EA 20 20 83 47 83 93 83 52 81 5B
83 66 83 42 83 93 83 4F 20 32 30 30 39 0A 09 82
C5 82 B7 81 42
```

適切に表示された文字列

日本語　エンコーディング　2009
です。

EUC-JP (37 bytes)

```
C6 FC CB DC B8 EC 20 20 A5 A8 A5 F3 A5 B3 A1 BC
A5 C7 A5 A3 A5 F3 A5 B0 20 32 30 30 39 0A 09 A4
C7 A4 B9 A1 A3
```

日本語文字　14文字
日本語文字以外　9文字
（　数字　4文字
　空白　3文字
　TA　1文字
　改行　1文字　）

UTF-8 (51 bytes)

```
E6 97 A5 E6 9C AC E8 AA 9E 20 20 E3 82 A8 E3 83
B3 E3 82 B3 E3 83 BC E3 83 87 E3 82 A3 E3 83 B3
E3 82 B0 20 32 30 30 39 0A 09 E3 81 A7 E3 81 99
E3 80 82
```

文字エンコーディングの統一

- 方法1: テキストの文字エンコーディングを変換するソフトウェアを使い、テキスト側で統一する。
  - Unix環境では、iconvやnkfというソフトウェアなど。
  - あるいは、エディタで変換。
- 方法2: 作成するプログラム側でテキストファイルの読み込み時に、文字エンコーディングを変換して統一する。

不要な記号類の削除

- 各種「正規表現」を扱えるフィルタプログラムがUnix環境下では使える。
- フィルタプログラム(補足資料へ)
  - 入力から与えられたファイルに対して一定の作業を順次行って出力を得る
- ストリームエディタsedを使ってみよう。
  - フィルタプログラムの一種
  - 例: HTMLのタグをすべて外す。
    - `sed -e 's/<[^>]*>//g' <input.txt >output.txt`

s/*regexp*/*str*/g
正規表現*regexp*に照合するすべて(g)の部分文字列を*str*におきかえる。

注意 sedの処理結果において日本語表示が乱れる場合には、sedの代わりに
/usr/xpg4/bin/sed
を使ってみる。**以下同様**

「一行一文」形式に変換

- 句点類(。,？,！,. 等)の後に改行を入れる。
- 例1：sedを使って「。」の直後に改行を入れるシェルスクリプト
  - シェルスクリプト: シェルに与えるコマンド列をファイルにして、再利用しやすくしたもの

sent_break.sh

```
#!/bin/sh

sed 's/。/。¥
/g'
```

この行の末尾は、
¥(バックスラッシュ)の後に改行
であることに注意。改行自身が置き換え先の文字列の一部なのでこれを普通の改行として認識しないようにエスケープしている(¥を直前につけている)。
置き換え先の文字列に改行がはいっているので、sedに与えるs/regexp/str/gが二行に亙っているように見える。

このシェルスクリプトをコマンドとして実行できるようにするには、
chmod a+x sent_break.sh

- 例2: シェル上で直接sedを起動する場合は、¥自身を解釈されないように、エスケープするので、以下の通り。

```
sed 's/。/。¥¥
/g'
```

この行の末尾は、上のコメントと同じ理由で、
¥¥(バックスラッシュ二つ)の後に改行
であることに注意。

ここまでをつなげると

- 部品を「パイプライン」でつなげて実行
  - エンコーディング統一➔不要な記号類の削除➔「一行一文」形式への変換➔必要な形式の文を抽出

```
nkf -e < input.xml | sed 's/<[^>]*>//g' | ./sent_break.sh |
   grep -v '^$' | grep '。$' | sed 's/^ //' > output.txt
```

※ 紙面の都合で折り返しているが、一行にする。↩

```
nkf -e < input.xml |
sed 's/<[^>]*>//g' |
./sent_break.sh |
grep -v '^$' |
grep '。$' |
sed 's/^ //' > output.txt
```

文字エンコーディングをEUC-JPに

<…>という文字列(つまり、タグ)をすべて削除

句点「。」の次に改行を挿入

正規表現^$照合する行(すなわち空行)を削除し、残りの行を取り出す

句点「。」でおわっている行のみを取り出す

文頭にある全角空白を削除する

基本的な自然言語処理

基本的な自然言語処理

- 続く処理においてどの情報が必要かによって、ここで行う処理を選ぶ。
- 例えば、単語レベルの処理ならば、「形態素解析」
  - キーワード抽出
  - 情報検索のための索引作成
  - 情報抽出
- 単語と単語の間の関係や、文節、文単位の情報を扱うのであれば、「係り受け解析」
  - 機械翻訳
  - 質問応答
- テキストに現れる固有表現が必要であれば、「固有表現抽出」

形態素解析器ChaSen

- 形態素解析器の役割
  - 文を形態素の列に分解
    - 形態素: 意味を持つ最小の言語単位。語を構成する。
  - 各形態素の文法的な役割(品詞)を付与
- 使い方
  - その1: ファイル内の各文(「一行一文」形式)を解析

    chasen *filename*
  - その２: 標準入力からの各文(「一行一文」形式)を解析

    chasen

出力例

入力は「一行一文」形式

項目の区切りはTAB

鳩山首相は首相官邸でオバマ大統領と会見した。

出力は「一行一形態素」形式

```
鳩山	ハトヤマ	鳩山	名詞-固有名詞-人名-姓
首相	シュショウ	首相	名詞-一般
は	ハ	は	助詞-係助詞
首相	シュショウ	首相	名詞-一般
官邸	カンテイ	官邸	名詞-一般
で	デ	で	助詞-格助詞-一般
オバマ		未知語
大統領	ダイトウリョウ	大統領	名詞-一般
と	ト	と	助詞-格助詞-一般
会見	カイケン	会見	名詞-サ変接続
し	シ	する	動詞-自立	サ変・スル	連用形
た	タ	た	助動詞	特殊・タ	基本形
。	。	。	記号-句点
EOS
```

文末はEOS

見出し 出現形　ヨミ　見出し 基本形　品詞　活用型　活用形

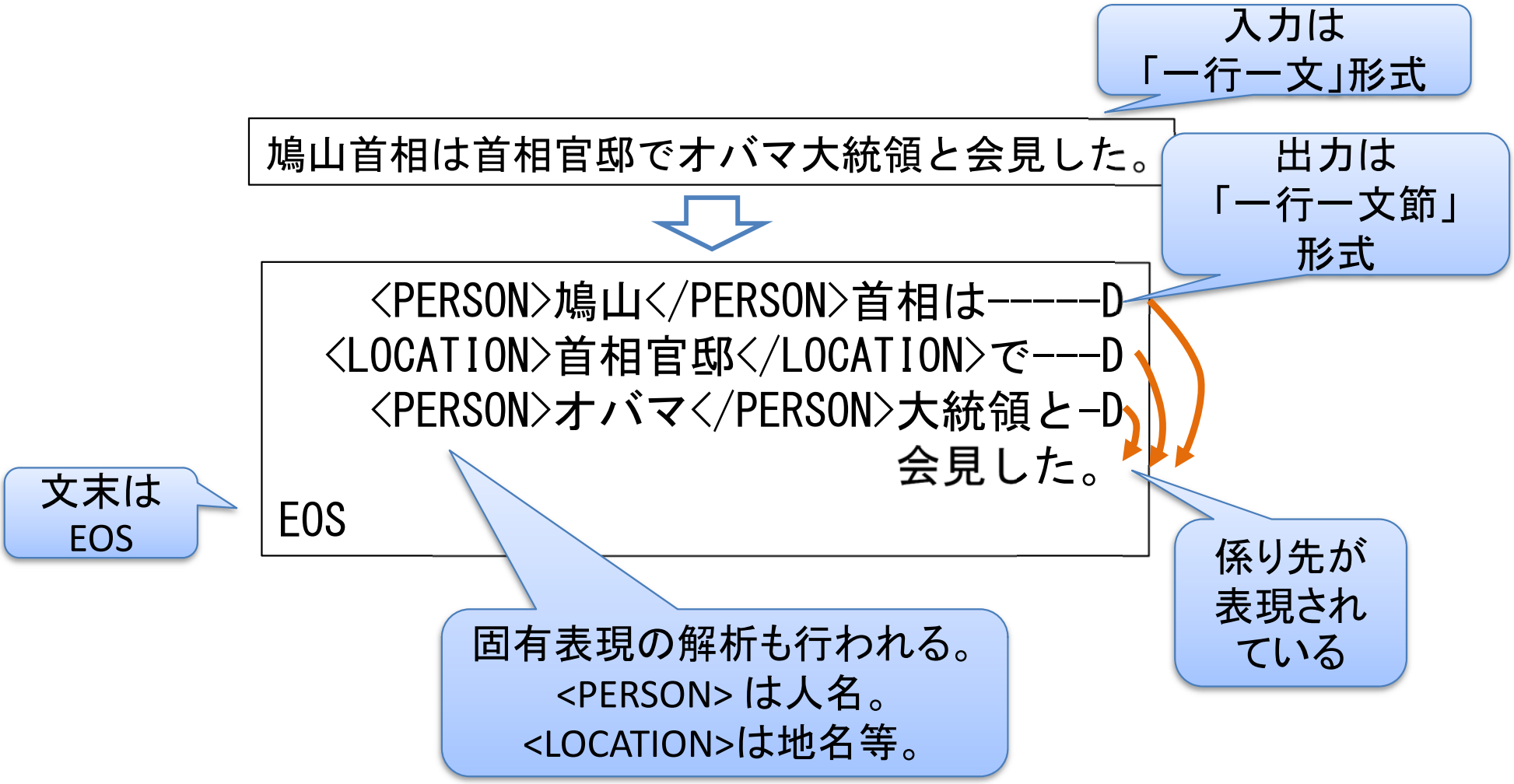
係り受け解析器CaboCha

- 係り受け解析器の役割
  - 文節にまとめ上げる
    - 文節: 一つの自立語+任意個の付属語
  - 文節間の係り受け関係を見つける
    - 各文節は、必ず、自分より後に現れる一つの文節に係る。
  - さらに、CaboChaでは固有表現の抽出も
- 使い方
  - ChaSenと同じように、ファイルから入力することも、標準入力から入力することもできる。いずれも「一行一文」形式。
  - 出力形式1: 木形式 (人間が理解する時に便利)

    `cabocha` *filename*
  - 出力形式2: 表形式 (解析結果を処理する時に便利)

    `cabocha –f1` *filename*

出力例1: 木形式


入力は
「一行一文」形式
鳩山首相は首相官邸でオバマ大統領と会見した。
出力は
「一行一文節」
形式
<PERSON>鳩山</PERSON>首相は-----D
<LOCATION>首相官邸</LOCATION>で---D
<PERSON>オバマ</PERSON>大統領と-D
会見した。
EOS
文末は
EOS
固有表現の解析も行われる。
<PERSON> は人名。
<LOCATION>は地名等。
係り先が
表現され
ている

出力例1: 表形式

鳩山首相は首相官邸でオバマ大統領と会見した。

＊　文節番号　係り先文節番号　種類　主辞/機能語位置　係りスコア

文節の開始

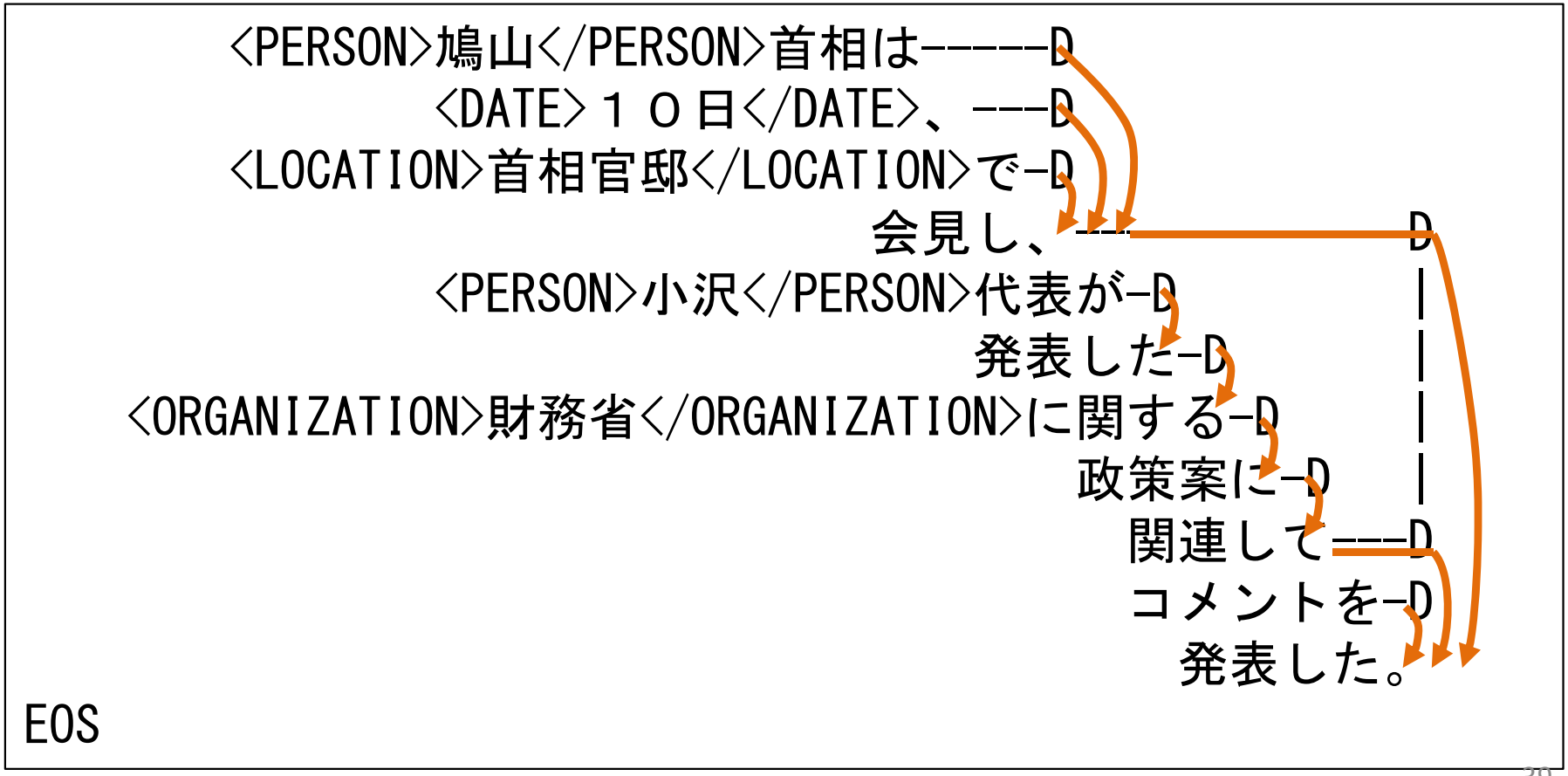
固有表現の出現位置をIOB2法で表現したもの。B-... が開始、I-... が継続

文節内の形態素の情報を一行一形態素形式で表示

```
＊ 0 3D 1/2 3.90668393
鳩山	ハトヤマ	鳩山	名詞-固有名詞-人名-姓	B-PERSON
首相	シュショウ	首相	名詞-一般	O
は	ハ	は	助詞-係助詞	O
＊ 1 3D 1/2 5.32724304
首相	シュショウ	首相	名詞-一般	B-LOCATION
官邸	カンテイ	官邸	名詞-一般	I-LOCATION
で	デ	で	助詞-格助詞-一般	O
＊ 2 3D 1/2 0.00000000
オバマ		未知語		B-PERSON
大統領	ダイトウリョウ	大統領	名詞-一般	O
と	ト	と	助詞-格助詞-一般	O
＊ 3 -10 1/2 0.00000000
会見	カイケン	会見	名詞-サ変接続	O
し	シ	する	動詞-自立	サ変・スル連用形	O
た	タ	た	助動詞	特殊・タ基本形	O
。	。	。	記号-句点	O
EOS
```

出力例2: 木形式

鳩山首相は10日、首相官邸で会見し、小沢代表が発表した財務省に関する政策案に関連してコメントを発表した。



出力例2: 表形式

鳩山首相は10日、首相官邸で会見し、小沢代表が発表した財務省に関する政策案に関連してコメントを発表した。

```
＊ 0 3D 1/2 1.00783064
鳩山	ハトヤマ	鳩山	名詞-固有名詞-人名-姓		B-PERSON
首相	シュショウ	首相	名詞-一般		O
は	ハ	は	助詞-係助詞		O
＊ 1 3D 2/2 0.34240832
１	イチ	１	名詞-数		B-DATE
０	ゼロ	０	名詞-数		I-DATE
日	ニチ	日	名詞-接尾-助数詞		I-DATE
、	、	、	記号-読点		O
＊ 2 3D 1/2 2.59904509
首相	シュショウ	首相	名詞-一般		B-LOCATION
官邸	カンテイ	官邸	名詞-一般		I-LOCATION
で	デ	で	助詞-格助詞-一般		O
＊ 3 10D 1/1 3.77343430
会見	カイケン	会見	名詞-サ変接続		O
し	シ	する	動詞-自立　サ変・スル　連用形		O
、	、	、	記号-読点		O
＊ 4 5D 1/2 1.54907214
小沢	オザワ	小沢	名詞-固有名詞-人名-姓		B-PERSON
代表	ダイヒョウ	代表	名詞-サ変接続		O
が	ガ	が	助詞-格助詞-一般		O
```

```
* 5 6D 1/2 1.00299848
発表　　ハッピョウ　　発表　　名詞-サ変接続　　0
し　　シ　　する　　動詞-自立　　サ変・スル　　連用形　　0
た　　タ　　た　　助動詞　　特殊・タ　　基本形　　0
* 6 7D 0/1 0.80009980
財務省　　ザイムショウ　　財務省　　名詞-固有名詞-組織　　B-ORGANIZATION
に関する　　ニカンスル　　に関する　　助詞-格助詞-連語　　0
* 7 8D 1/2 1.74374475
政策　　セイサク　　政策　　名詞-一般　　0
案　　アン　　案　　名詞-接尾-一般　　0
に　　ニ　　に　　助詞-格助詞-一般　　0
* 8 10D 1/2 5.71648647
関連　　カンレン　　関連　　名詞-サ変接続　　0
し　　シ　　する　　動詞-自立　　サ変・スル　　連用形　　0
て　　テ　　て　　助詞-接続助詞　　0
* 9 10D 0/1 0.00000000
コメント　　コメント　　コメント　　名詞-サ変接続　　0
を　　ヲ　　を　　助詞-格助詞-一般　　0
* 10 -10 1/2 0.00000000
発表　　ハッピョウ　　発表　　名詞-サ変接続　　0
し　　シ　　する　　動詞-自立　　サ変・スル　　連用形　　0
た　　タ　　た　　助動詞　　特殊・タ　　基本形　　0
。　　。　　。　　記号-句点　　0
EOS
```

本日の講義に対応するレポート

以下の項目に関する調査結果を報告せよ。

- 次のコマンドはUnix系OSで良くつかわれるフィルタプログラムである。それぞれの動作について、簡単な例示により解説せよ。
  - cut
  - sort,　sort –n,　sort –nr　　(ただし、n, r はオプションである。)
  - uniq,　uniq –c　(ただし、cはオプションである。)
  - wc
  - head, tail
  - cat, paste, join
  - egrep
  - sed
- 正規表現について調査をせよ。特に、以下の項目を含むこと。
  - リテラル文字とメタ文字。特にメタ文字のうち、|, *, +, (,), . , [,], ^, $
- 補足資料にある「単語頻度を求める」例において、パイプラインを構成する各コマンドの動作を説明し、これらをパイプラインで接続することにより、どのように単語頻度を求め、表示しているかを説明せよ。
- ChaSenやJuman等、形態素解析器が使っている日本語の品詞体系は、みなさんが学校で学んだいわゆる「学校文法」(橋本文法)によるものと若干異なる。形態素解析器Jumanが以下のURLで公開されているので、Jumanが採用している品詞体系と学校文法の品詞体系で異なる点をいくつか指摘し、その違いを解説せよ。

http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN

注意：
参考資料です。
これは、座学のレポート課題の記述です。
ここに挙げられているコマンドを使うとよいことが…

補足資料

復習: 標準入力・標準出力

- 標準入力・標準出力
  - Unix系OSではファイルを明示的にopenしなくても、入出力が行える。
  - 標準入力: 入力用に準備されている仮想ファイル
    - シェルの環境では、指定がなければキーボード入力。
    - C言語では、scanf(), getchar() 等の関数が標準入力からの入力。
  - 標準出力: 出力用に準備されている仮想ファイル
    - シェルの環境では、指定がなければ画面(コンソール)出力。
    - C言語では、printf(),putchar()等の関数が標準出力への出力。
- フィルタプログラム
  - 入力を標準入力、出力を標準出力としたプログラム
  - 「リダイレクション」、「パイプライン」機能により、ツールとして柔軟に使える

復習: リダイレクション

- リダイレクション
  - 標準入力や標準出力を特定のファイルに切り替えること。
  - コマンド*command*の標準出力をファイル*out_filename*への出力に切り替える

    command > *out_filename*
  - コマンド*command*の標準入力をファイル*in_filename*からの入力に切り替える

    command < *in_filename*
  - コマンド*command*の標準入力をファイル*in_filename*からの入力に、標準出力をファイル*out_filename*への出力に切り替える

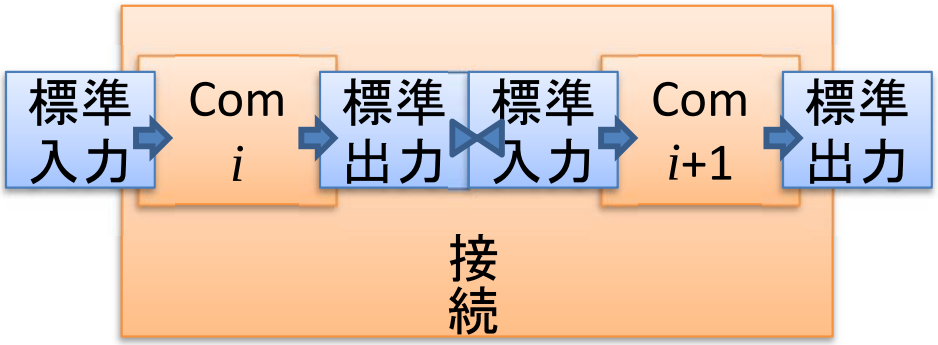
    command < *in_filename* > *out_filename*

※ 通常の状況

復習: パイプライン(1)

- 基本
  - 二つ以上のコマンドをつなげて一つのプログラムとして動作させる
  - $command_i$の標準出力を、$command_{i+1}$の標準入力に接続する。シェルでは、縦棒 '|' を使ってコマンド入力。

$$command_i \mid command_{i+1}$$

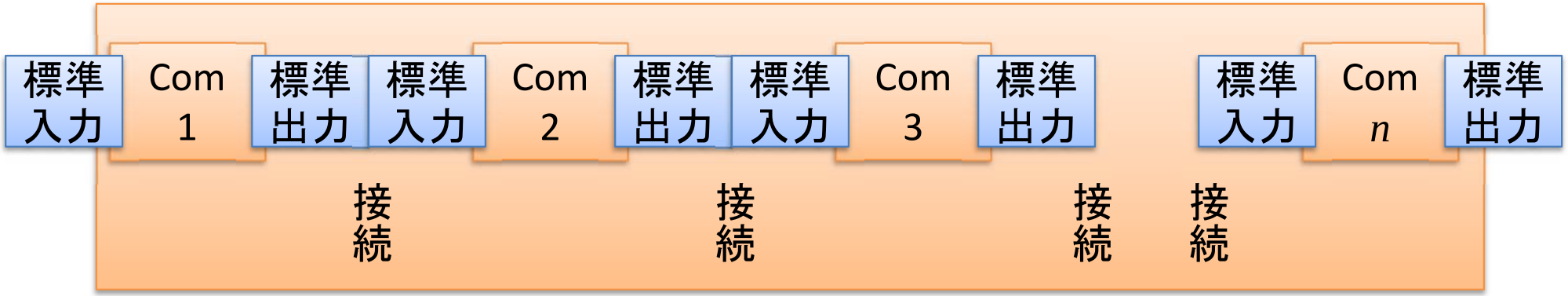


一つのコマンドのようにふるまう

復習: パイプライン(2)

- 複数のコマンドをつなげて使う
  - $command_1, command_2, \ldots, command_n$
    をつなげて一つのプログラムとして動作させる

    $command_1 \mid command_2 \mid \ldots \mid command_n$



復習:パイプライン(3) 例

- あるディレクトリにあるファイルやディレクトリの数を求める。
  - コマンドls(ファイル名の表示)の出力をコマンドws(ファイル中の単語数、行数、ファイルサイズを求める)の入力に接続する。

```
ls  | ws -w
```

- あるファイルf.txt(内容は英語)に現れる単語の頻度を求める。

```
sed 's/ /¥¥
/g' < f.txt | egrep -v '^$' | sort | uniq -c | sort -nr
```

ここは¥¥の後に改行が入っていることに注意。
¥¥+改行で改行文字それ自身を表している。