テキスト情報処理に関する演習

森 辰則

mori@forest.eis.ynu.ac.jp

注意: 演習の時間のときにも本資料を使いますので、忘れずに持参してください。

テキスト情報処理

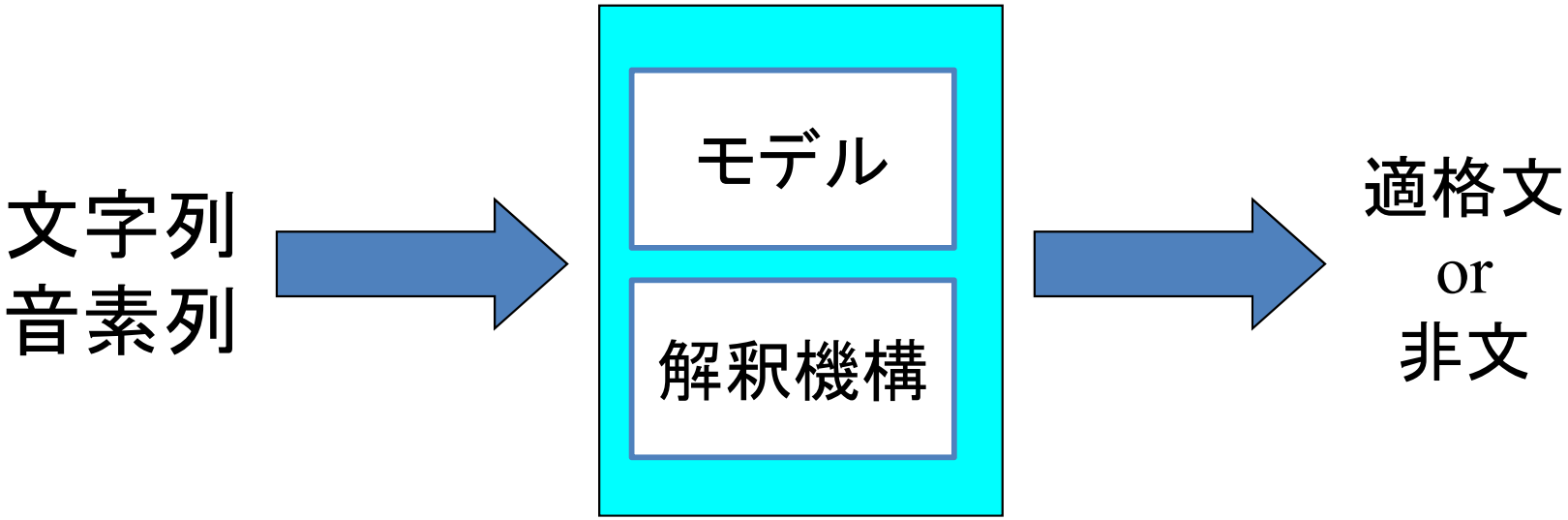
- テキスト=文字により記述される情報
  - 本、新聞、板書、看板、etc.
  - Web、電子メール、電子図書、etc.
- テキスト情報処理
  - コンピュータにより、テキストを扱う処理全般
  - 簡単なものから、
    - 文字列としての処理
    - 文字列検索、正規表現による検索、etc.
  - 高度なものまで、
    - 言語としての処理:自然言語処理（後述）
    - 情報抽出、機械翻訳、etc.
  - テキストをどのようにモデリングし、処理するかに依存

「言語」の「モデリング」

- 言語
  - 人間の使う言語、すなわち、自然言語
    - cf. 人工言語（コンピュータ用の言語など）
- モデリング
  - 「言語の持つ性質」を形式的な枠組みで記述。記述されたものが**言語の形式モデル**
  - その枠組みを用いて計算機による処理を可能とする（**自然言語処理**といいます）
  - **言語のモデルに期待される事柄は…**

言語のモデルに期待されること(1/2)

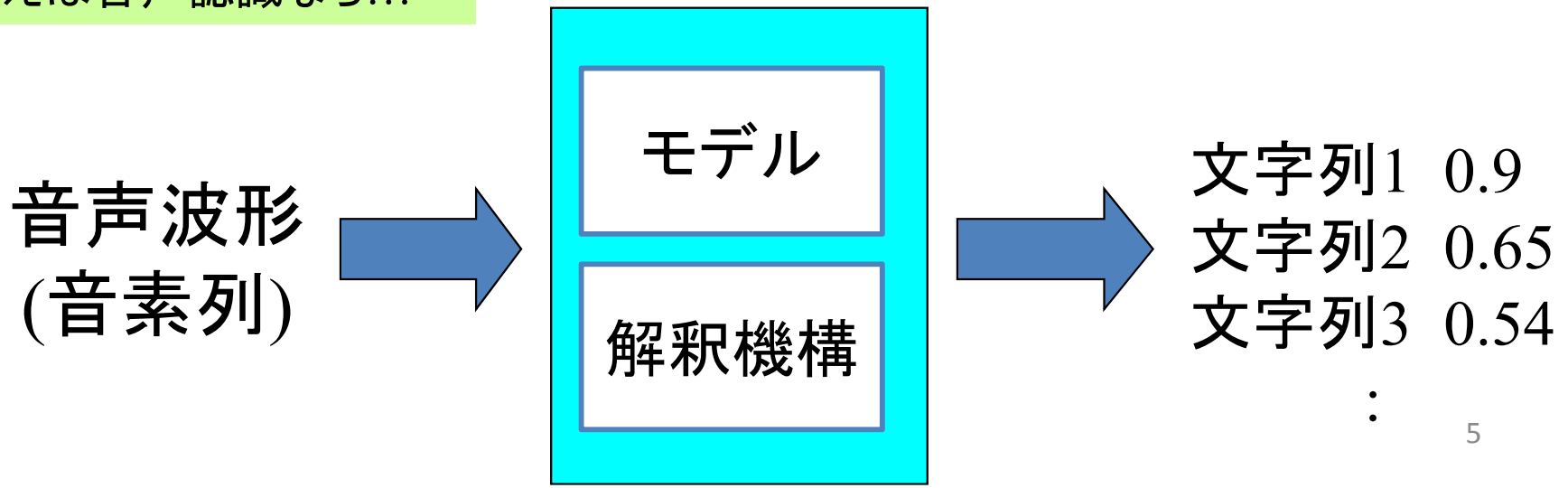
- 言語学的観点からすると…
  - ある言語において，当該の文が適格性を持つか，**非文であるかがわかること**
    - 適格文と非文
      - 今日は天気がよい。
      - *良くが今日天気は。



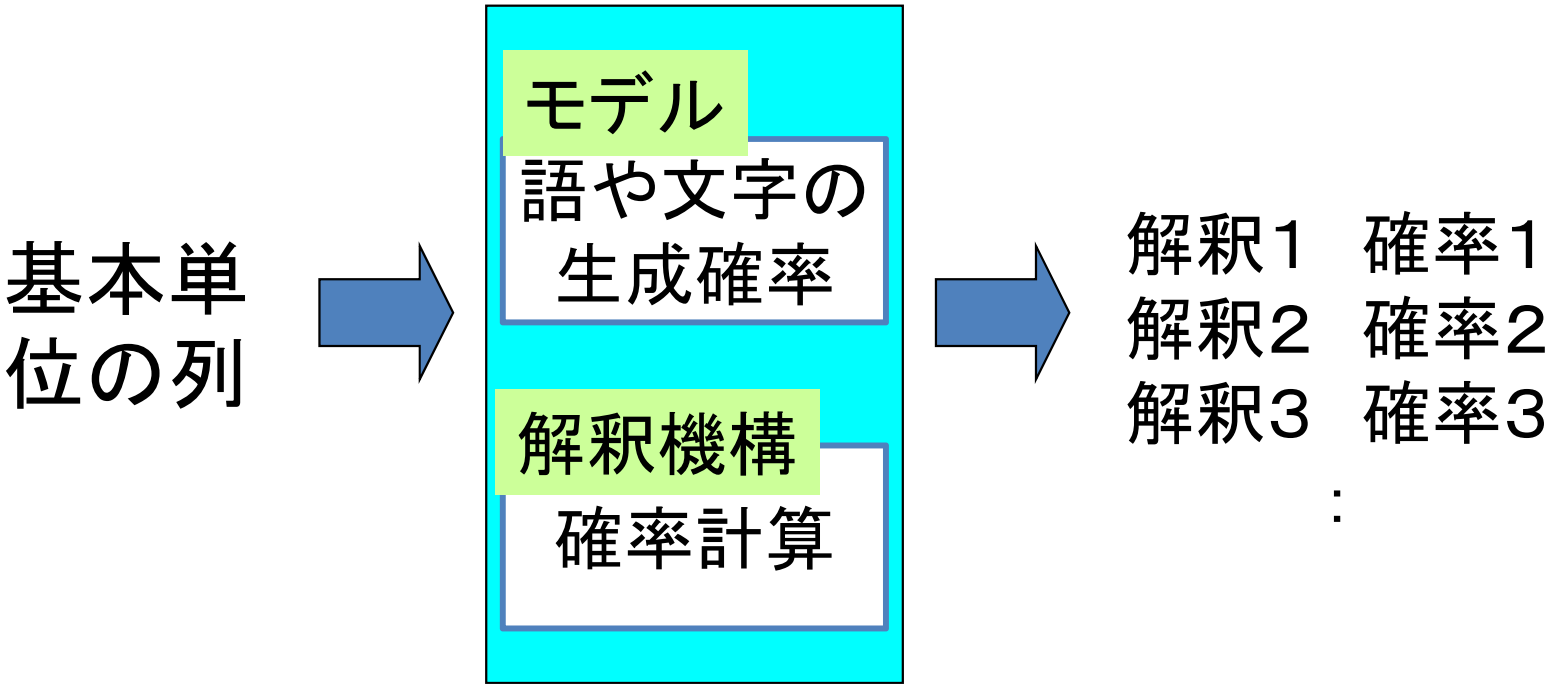
言語のモデルに期待されること(2/2)

- 言語処理の観点からすると…
  - 入力(音声，文字)に対して適切な応答をしてくれること
  - 入力に対して，**「解」が見つける**ことができて，なおかつ，それらの間に**「優先順位」をつけられる**こと

例えば音声認識なら…



確率に基づくモデル化



1. 確率による言語のモデル
2. n-gram
3. 音声認識の例

確率による言語モデル(1/2)

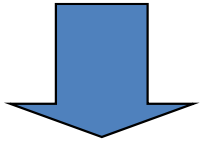
- 文書を「基本単位」の列として捉え，基本単位の出現に関する確率を考える.
  - ある時点において，次に生じる「基本単位」の出現確率によって，言語をモデル化
- 「基本単位」
  - 文字(alphabet)
  - 単語(word)

確率による言語モデル(1/3)

例:「昨日、私…」と読んだときに、次に来る文字は？

- 「が」,「は」,「も」,「に」 ←助詞類、高い確率
- 「で」 ←同じ助詞でも、確率が低い
- 「立」 （の学校の入学試験が…）←単語の途中
- 「物」 （を持って帰った）←単語の途中

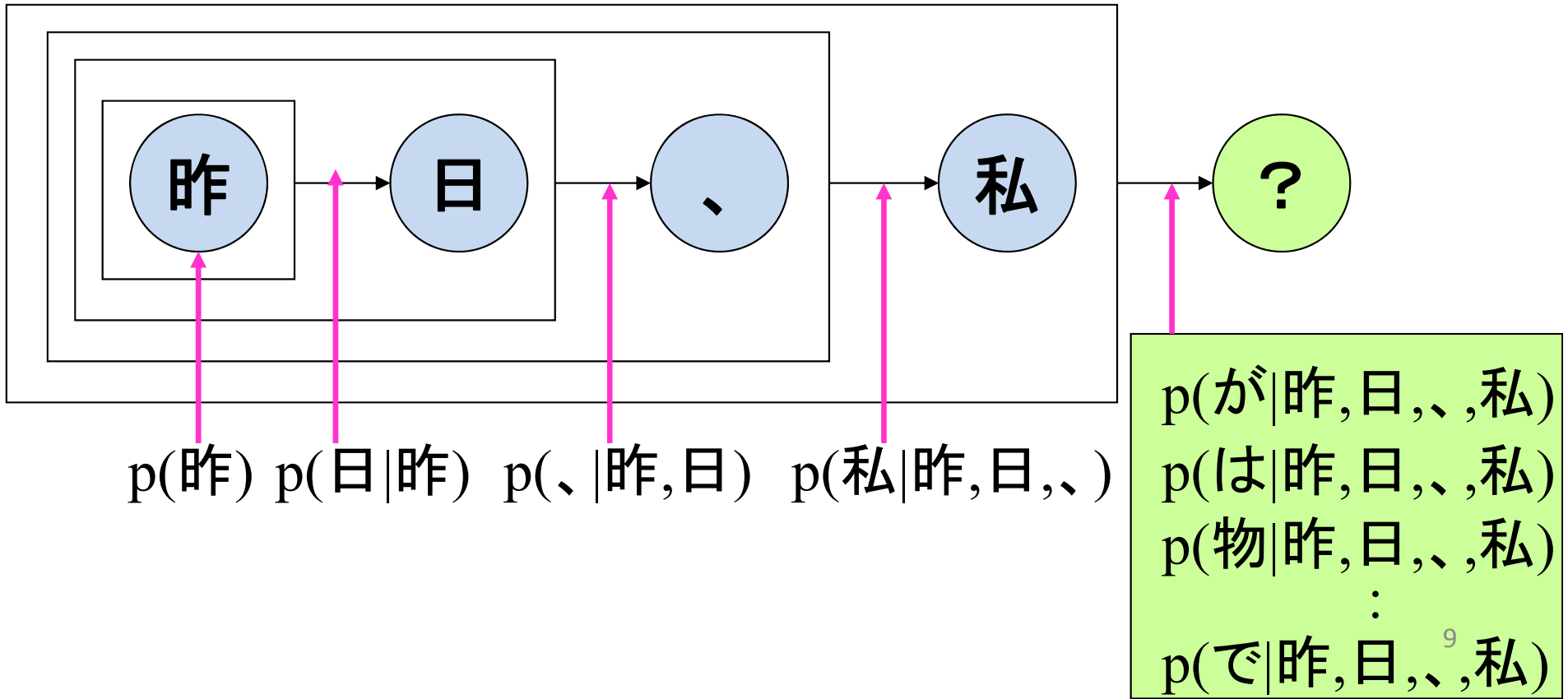
確率による優先順位： で ＜ 立、物 ＜ が、は、に、を



言語に内在する制約を反映して、生起確率が得られるはず

確率による言語モデル(2/3)

- ある「基本単位」に注目したときに、その直前までの文脈でその基本単位の生起確率が決定される。



確率による言語モデル(3/3)

- 一般化すると、列$w_1,w_2,w_3,...w_n$の生起確率は...

$$p(w_1, w_2, w_3, ..., w_n)$$
$$= p(w_1) \cdot p(w_2 \mid w_1) \cdot p(w_3 \mid w_1, w_2) \cdot ... \cdot p(w_i \mid w_1, ..., w_{i-1}) \cdot ... \cdot p(w_n \mid w_1, ..., w_{n-1})$$
$$= \prod_{i=1}^{n} p(w_i \mid w_1, ..., w_{i-1})$$

- 条件付確率(conditional probability):
  - ある事象w_jが「すでに生起している」という仮定の下での，事象w_iの生起確率．事後確率(posterior probability)とも.
  - cf. $p(w_j)$のように，ある事象自身の確率は事前確率(prior probability)と呼ばれる.
  - 条件を入れ替えるには...(推定しやすいものから求める)

$$p(w_j \mid w_i) = \frac{p(w_i, w_j)}{p(w_i)}$$
$$= \frac{p(w_i \mid w_j) \cdot p(w_j)}{p(w_i)} = \frac{p(w_i \mid w_j) \cdot p(w_j)}{\sum_{w_k} p(w_i \mid w_k) \cdot p(w_k)}$$

先の確率モデル式の問題点と解決策

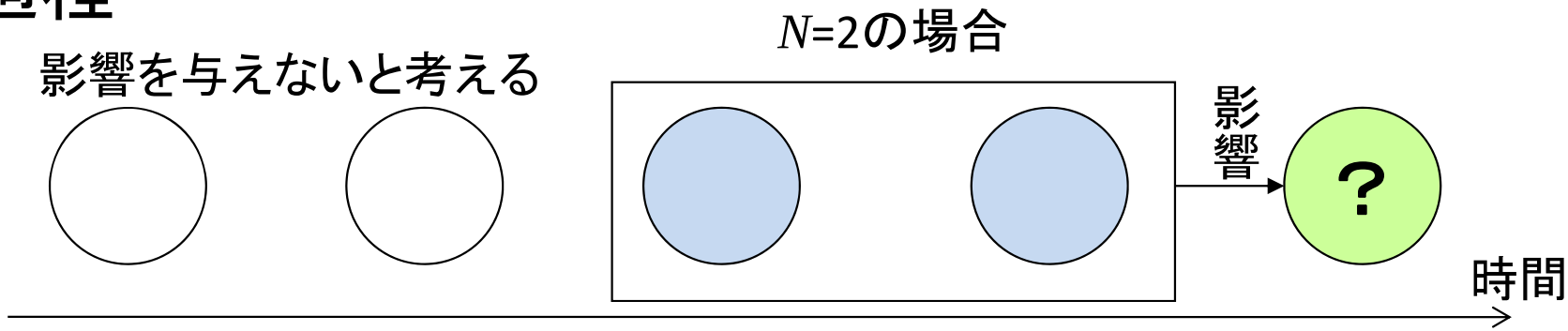
- 様々な単語/文字の組み合わせについて，条件付確率を求めるのが困難
  - 基本要素の種類数をLとすると、Lのi乗個もの異なる文字列に対する確率値が必要

$$p(w_i \mid w_1, \ldots, w_{i-1})$$

- 現実的なモデルにするには…
  - 組み合わせ数を削減する必要あり
  - 単語/文字の履歴$w_1, \ldots, w_{i-1}$を「同値類」(類似する状況)に纏め上げる
  - N重マルコフモデルによる近似: 直前のN個だけを考慮し、それ以前を無視する。

N重マルコフ過程とN−gramモデル

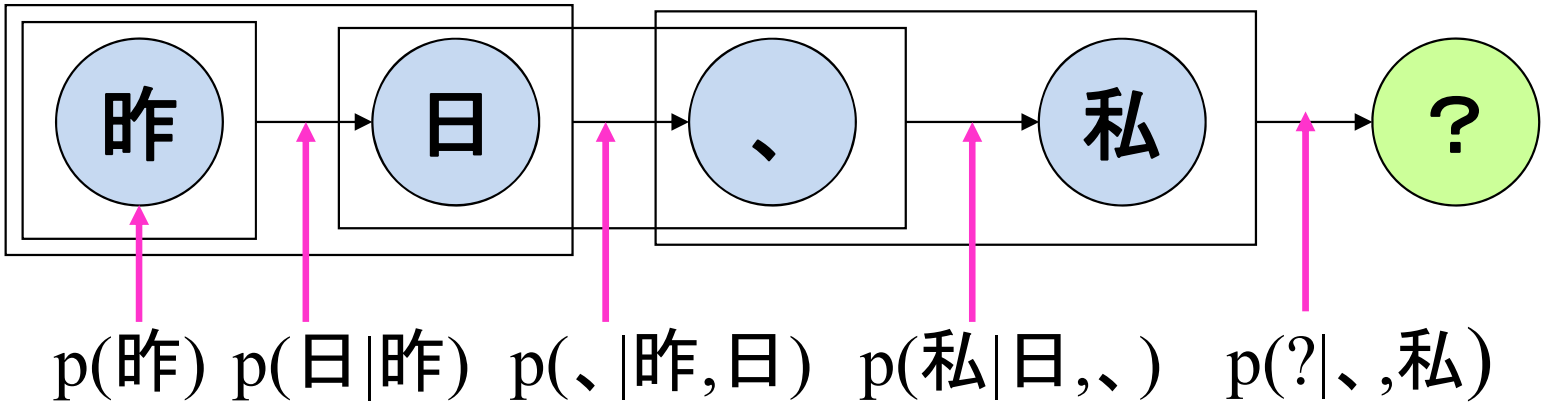
- N重マルコフ過程
  - ある事象の生起に関するモデル
  - ある時点で生起する事象の確率が，その直前のN個の時点で生起した事象だけの影響を受ける過程



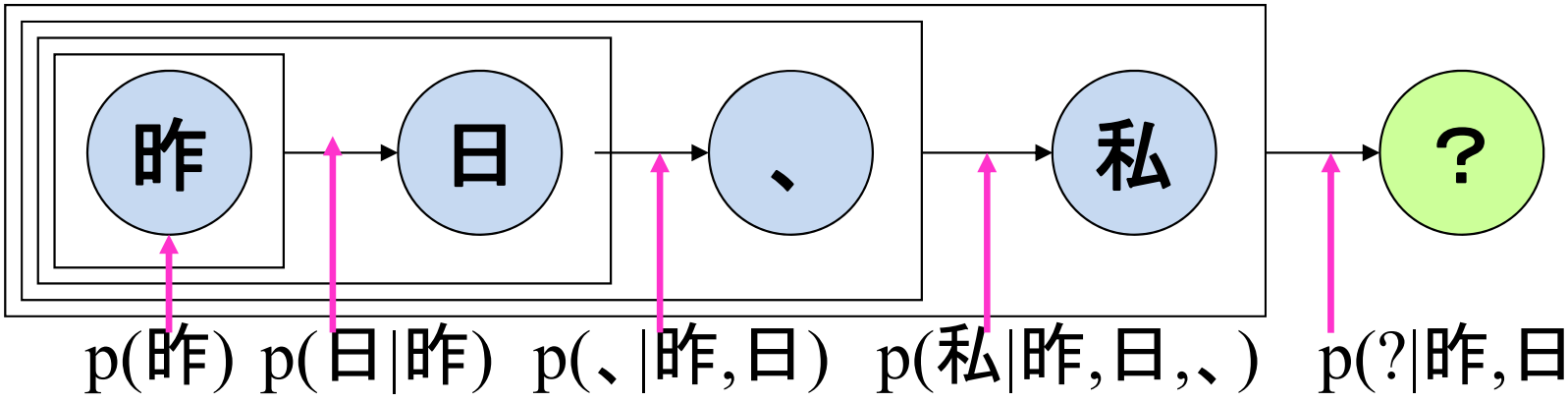
- N-gramモデル
  - N-gramによる基本単位の生起確率モデル
  - 基本単位の生起を(N-1)重マルコフ過程で近似したモデル

例: 2重マルコフ過程

- 直前の2つの事象までが次の事象に影響を与えると仮定



- 本来は…



N-gram

- N個の(隣接した)基本単位の列
- 1-gram(「ゆにぐらむ」), 2-gram(「ばいぐらむ」), 3-gram(「とらいぐらむ」)
- 例:「にわにはにわにわとりがいる．」における，文字N-gram
  - 1-gram の頻度
    - 「に」4,「わ」3,「は」1,「と」1,「り」1,「が」1,「い」1,「る」1,「．」1
  - 2-gram の頻度
    - 「にわ」3,「わに」2,「には」1,「はに」1,「わと」1,「とり」1,「りが」1,「がい」1,「いる」1,「る．」1
  - 3-gram の頻度
    - 「にわに」2,「わには」1,「にはに」1,「はにわ」1,「わにわ」1,「にわと」1,「わとり」1,「とりが」1,「りがい」1,「がいる」1,「いる．」1
- m重マルコフ過程は(m+1)-gramの統計量から推定される。

(N-1)重マルコフ過程による確率モデルの近似

- 確率モデルは次のように近似される

$$p(w_1, w_2, w_3, ..., w_n)$$
$$= p(w_1) \cdot p(w_2 \mid w_1) \cdot p(w_3 \mid w_1, w_2) \cdot ... \cdot p(w_i \mid w_1, ..., w_{i-1}) \cdot ... \cdot p(w_n \mid w_1, ..., w_{n-1})$$
$$= \prod_{i=1}^{n} p(w_i \mid \underbrace{w_1, ..., w_{i-1}}_{\text{i-1個}})$$

近似

$$p(w_1) \cdot p(w_2 \mid w_1) \cdot p(w_3 \mid w_1, w_2) \cdot ... \cdot p(w_i \mid w_{i+N+1}, ..., w_{i-1}) \cdot ... \cdot p(w_n \mid w_{n+N+1}, ..., w_{n-1})$$
$$= \prod_{i=1}^{n} p(w_i \mid \underbrace{w_{i+N+1}, ..., w_{i-1}}_{N\text{-1個}})$$

(N-1)重マルコフ過程の確率推定

- 実際の文書を用いる。
  - コーパス: 例文集
  - 扱いたい文書に「近い」、大量の文書
- N-gramの頻度と(N-1)-gramの頻度から求める。

$$p(w_i \mid w_{i-N+1}, \ldots, w_{i-1})$$
$$= \frac{freq(w_{i-N+1}, \ldots, w_i)}{freq(w_{i-N+1}, \ldots, w_{i-1})}$$
$$= \frac{p(w_{i-N+1}, \ldots, w_i)}{p(w_{i-N+1}, \ldots, w_{i-1})}$$

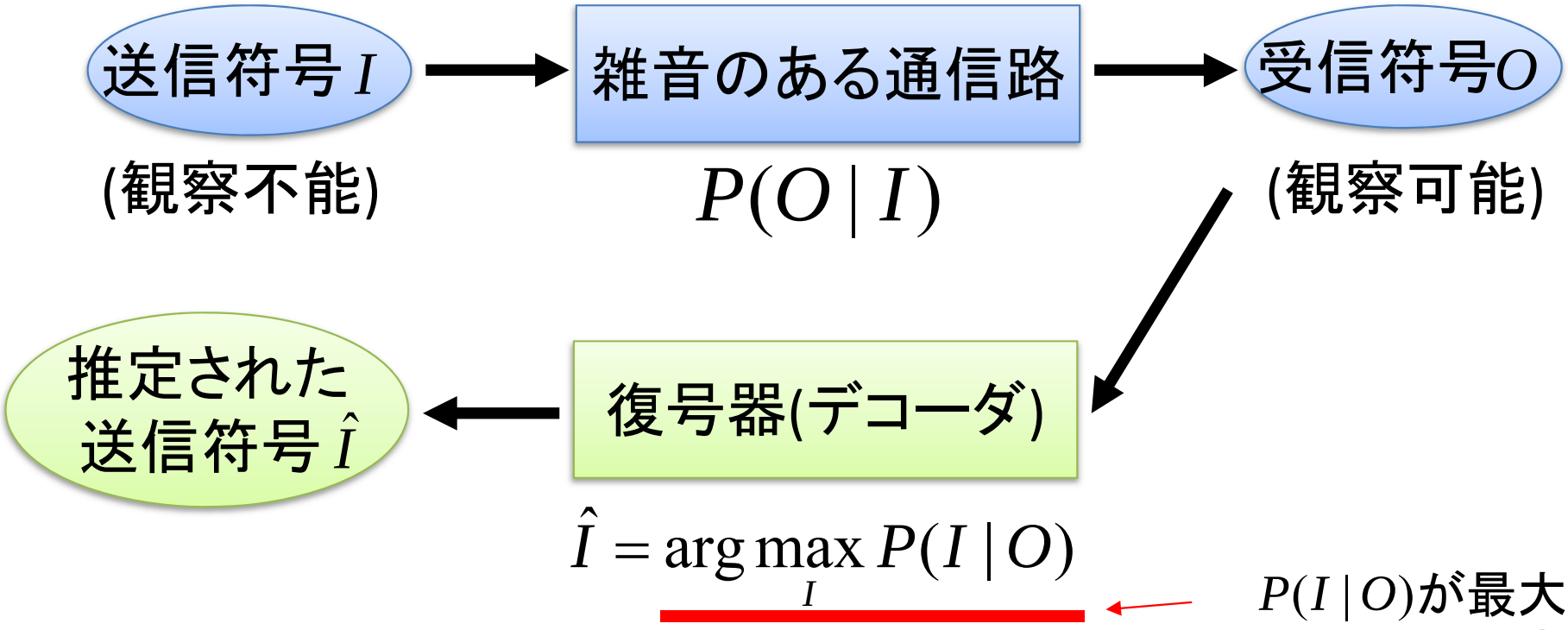
freq() : コーパス中の頻度

Noisy Channel Model(1)

- Noisy Channel Model = 雑音のある通信路のモデル
- ある言語処理を確率的な過程として考える
  - 「送信信号」が雑音のある通信路を通過することによって、「受信信号」が得られているととらえる。
  - つまり、通信路に重畳する「雑音の入り方」が「何らかの確率的な処理」であると考える。
  - 「受信信号」が実際に見えている「モノ」で、「送信信号」が実際に知りたい「モノ」
  - 「受信信号」から「入力信号」を推定する「復号器(デコーダ)」に実際の言語処理を対応させる。

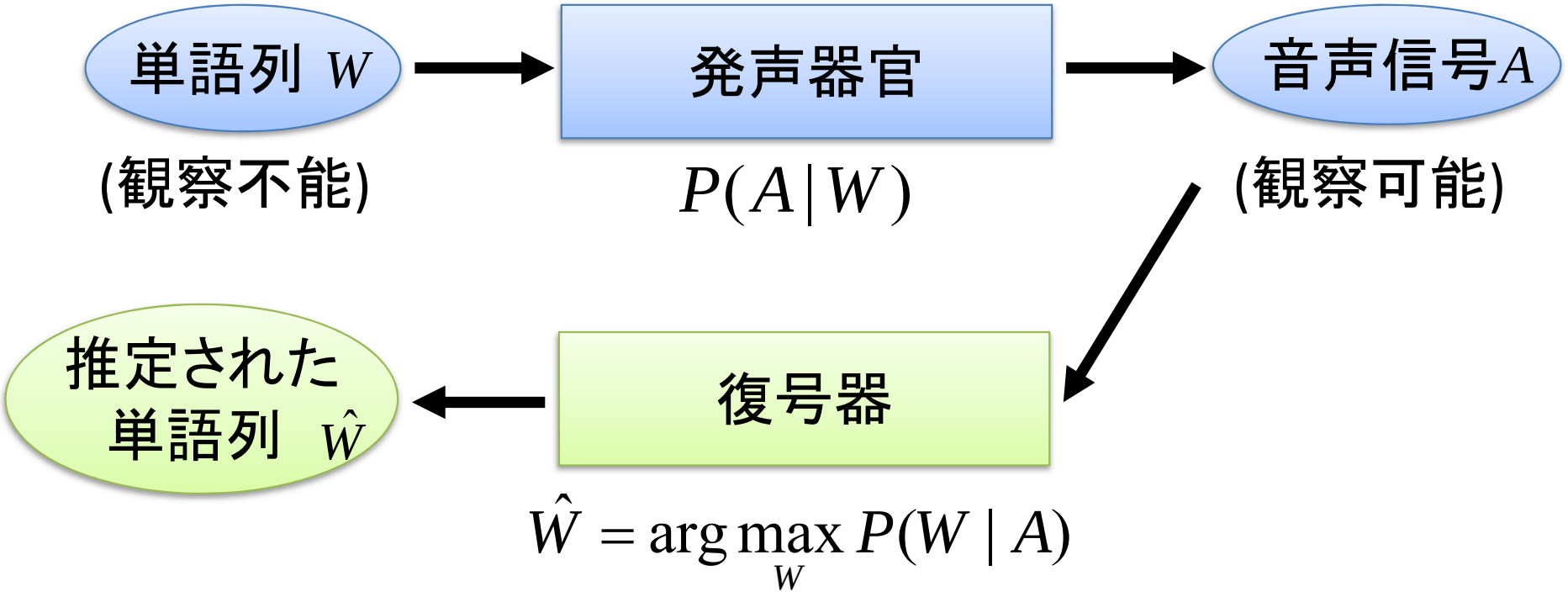
Noisy Channel Model (2)

- 通信の基本：受信符号から送信符号を復元(=復号)する。



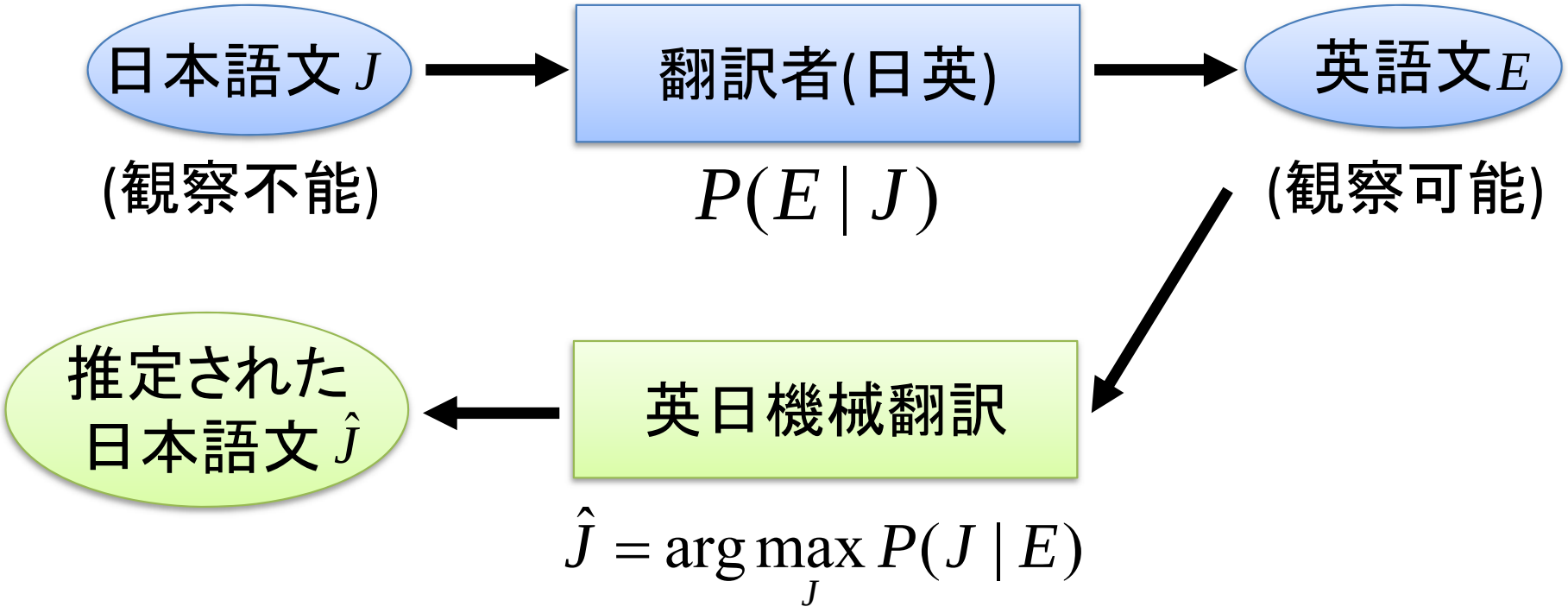
Noisy Channel Model (3)

- 音声認識の場合



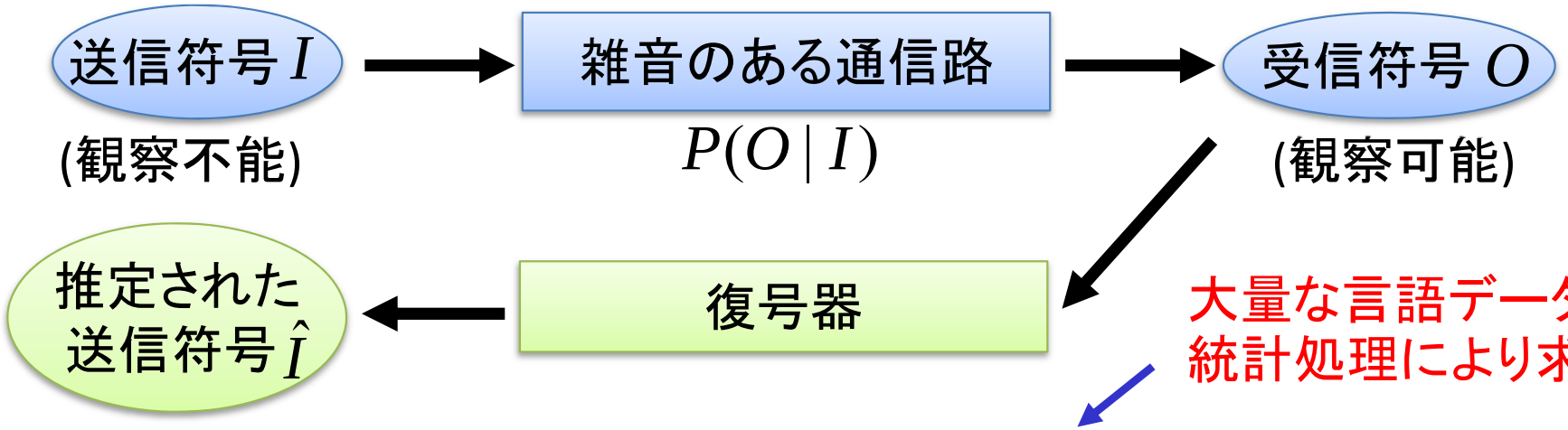
Noisy Channel Model (4)

- 英日機械翻訳の場合



最尤復号

- 受信符号 O が与えられたときの尤もらしい送信符号
- 誤る確率が最小のもの



大量な言語データから
統計処理により求める。

$$\hat{I} = \arg\max_I P(I|O)$$

$$= \arg\max_I \frac{P(I)p(O|I)}{P(O)}$$

$$= \arg\max_I P(I)P(O|I)$$

$P(I)$: 送信信号の事前確率

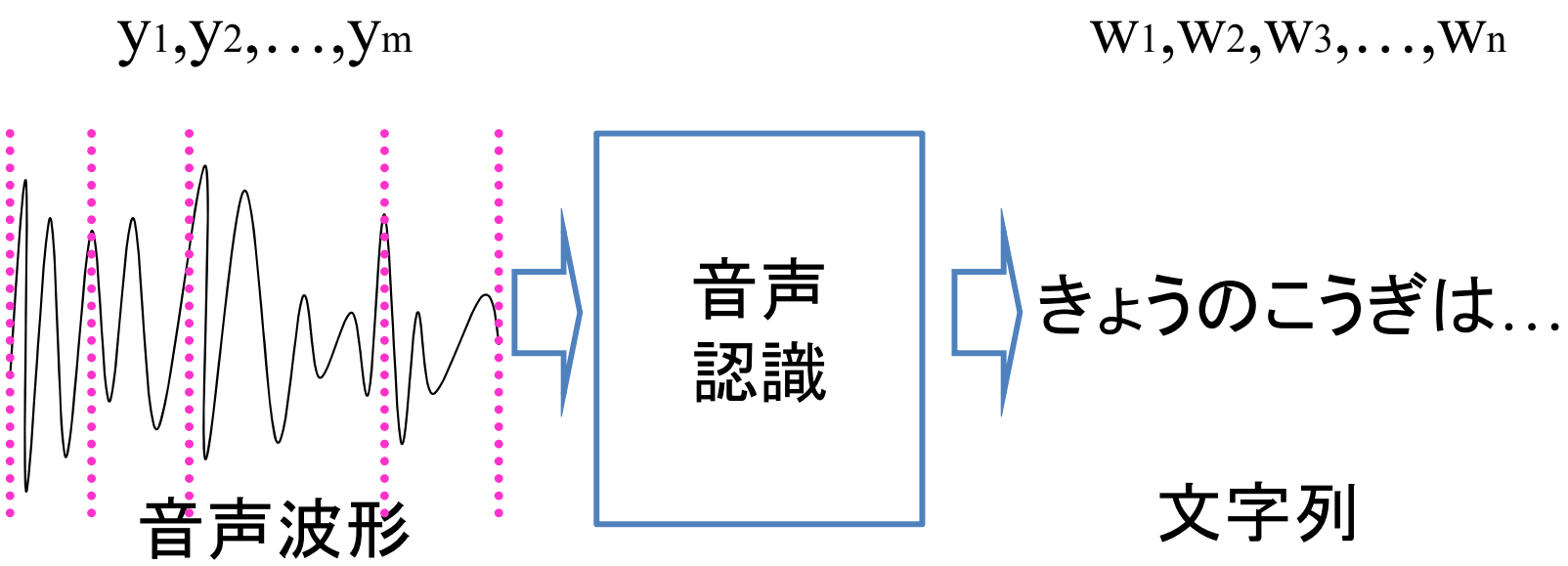
→ 送信信号の「自然さ」

$P(O|I)$: 通信路の確率モデル

→ 通信路にどのような

「ノイズ」が重畳するか

例　音声認識



音声認識の定式化

- 2-gramによる推定を考えると…

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|\mathbf{y})$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{w})P(\mathbf{w})/P(\mathbf{y})$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{w})P(\mathbf{w})$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1,\ldots,y_m|w_1,\ldots,w_n)P(w_1,\ldots,w_n)$$

$$\fallingdotseq \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1,..,y_{i_1}|w_1)\cdots P(y_{i_m},\ldots,y_m|w_n)\, P(w_1)P(w_2|w_1)\cdots P(w_n|w_{n-1})$$

言語モデル

文字(単語)が
どのように現れるか

音響モデル

文字(単語)が
どのように発音されるか

演習に関連する事項

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。

「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)、改行
であるが、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する
(¥を直前につける)。
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　ヨミ　見出し　品詞　活用型　活用形

出現形　基本形

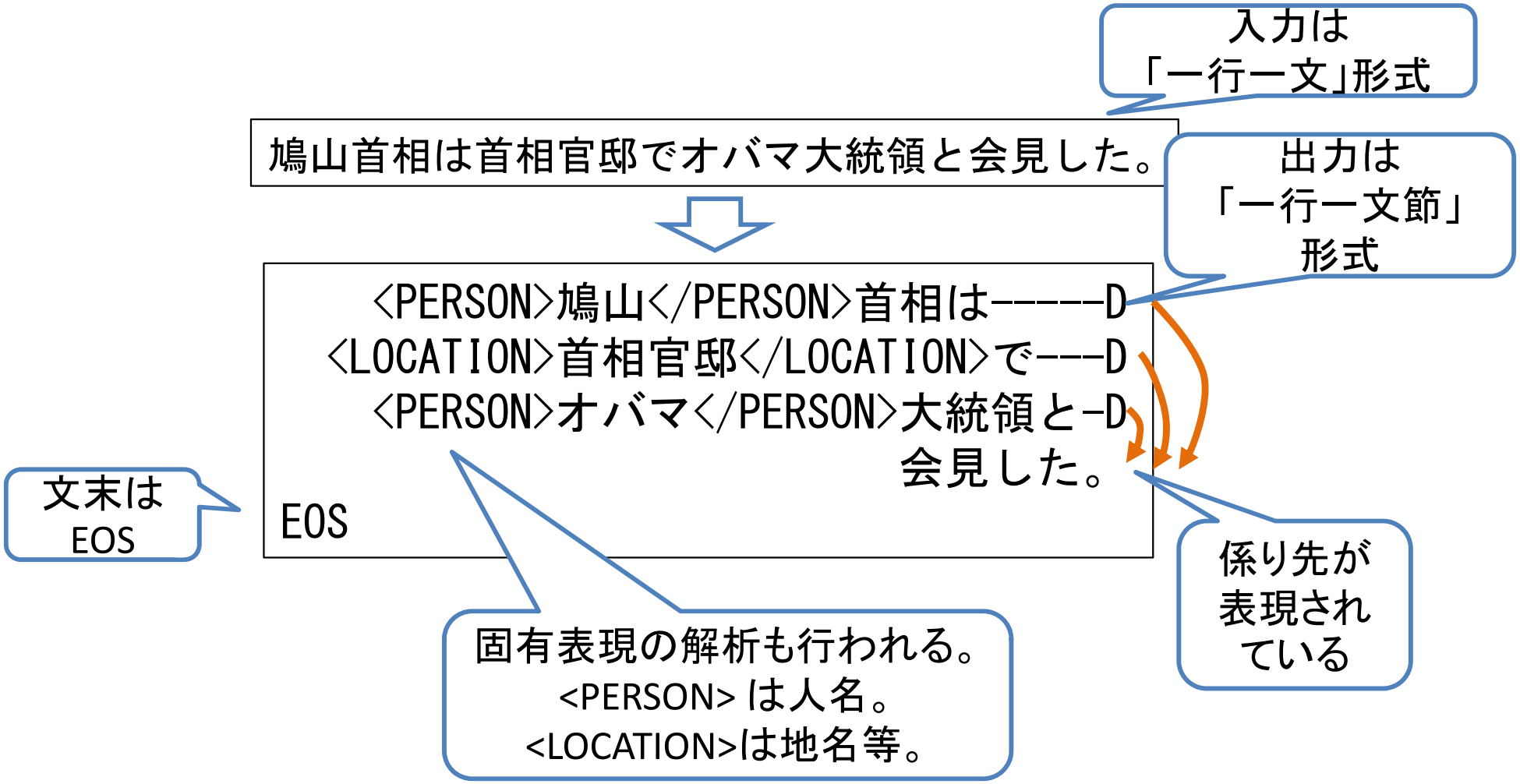
係り受け解析器CaboCha

- 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
係り先が
表現され
ている
固有表現の解析も行われる。
<PERSON> は人名。
<LOCATION>は地名等。

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

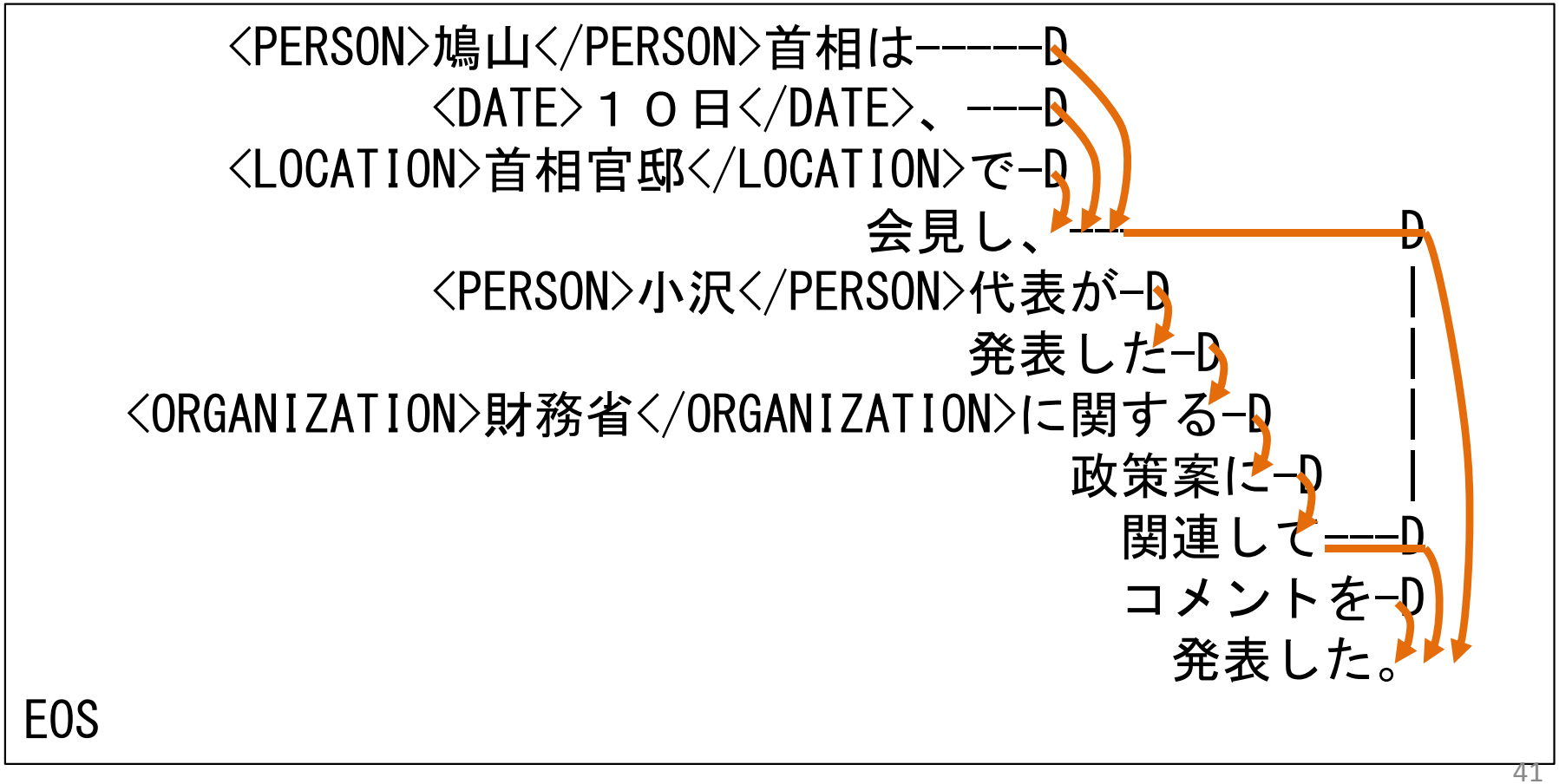
固有表現の出現位置をIOB2法で表現したもの。B-... が開始、I-... が継続

```
＊ 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
```

文節内の形態素の情報を一行一形態素形式で表示

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
```

目的に応じて実際に行いたい処理
⇒ 演習の内容

演習の内容

- 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。

演習1:テキストに対する前処理

- EUC-JPエンコーディングの一行一文形式に変換
- いくつかのフィルタプログラムを組み合わせることで実現することとして、以下の各点を考察。
  - 失敗事例に対する分析
    - 失敗箇所について、正しい解析結果の提示
    - 失敗箇所について、処理の問題点を指摘

演習2,3: ChaSen, CaboChaを用いた解析実験

- 既存の形態素解析器、係り受け解析器を用いて、実際にいくつかの文を解析してみる。
- 解析結果について、以下の各点を考察。
  - 失敗事例に対する分析
    - 失敗箇所について、正しい解析結果の提示
    - 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

- 与えられた新聞記事すべてを対象。
- 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以下の統計量を求める。
  - 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。
    - 新聞2紙を混ぜて行った場合と、それぞれ単独でおこなったときにどのような分布の違いがあるか調べ、考察する。
  - 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。
    - 新聞2紙を混ぜて行った場合と、それぞれ単独でおこなったときにどのような分布の違いがあるか調べ、考察する。

演習5: 簡単なフィルタプログラムの作成(1)

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以下の動作をするフィルタプログラムを作成してみよう。

- 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(2) : 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
```

```
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(3) : 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
```

演習5: 簡単なフィルタプログラムの作成(4) : 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
```

⇩

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

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)

- 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。
    - 新聞2紙を混ぜて行った場合と、それぞれ単独でおこなったときにどのような分布の違いがあるか調べ、考察する。
- 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。
    - 新聞2紙を混ぜて行った場合と、それぞれ単独でおこなったときにどのような分布の違いがあるか調べ、考察する。

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 (2)

- ある特定の表現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。
  - 新聞2紙を混ぜて行った場合と、それぞれ単独でおこなったときにどのような分布の違いがあるか調べ、考察する。
- ある特定の表現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。
  - 新聞2紙を混ぜて行った場合と、それぞれ単独でおこなったときにどのような分布の違いがあるか調べ、考察する。

- 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。
    - 新聞2紙を混ぜて行った場合と、それぞれ単独でおこなったときにどのような分布の違いがあるか調べ、考察する。

演習6: 簡単な情報抽出処理

- 新聞記事より、用語の定義・説明をしている文を抽出する
  - 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。
  - この手法で、当初の目的が達せられるかを調べ、失敗事例を分析するとともに、改善案を考察し、その効果について検討する。

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

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

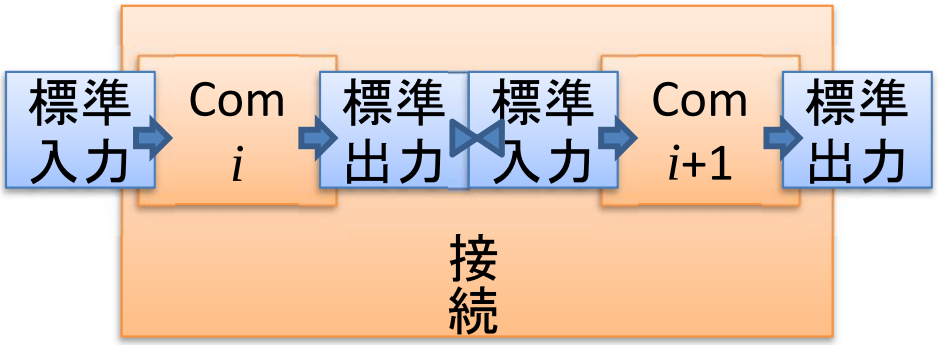
    command < *in_filename* > *out_filename*

※ 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。

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

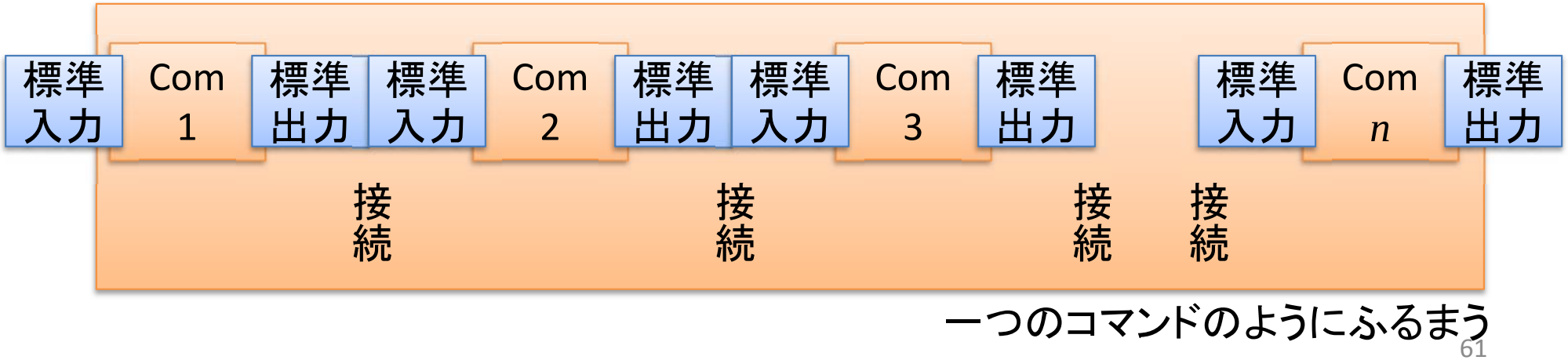


一つのコマンドのようにふるまう

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。