

第11回:2010年6月29日

IT教育ネットワーク論特論A (言語の入力と出力(2))

小嶋 秀樹

xkozima@myu.ac.jp

IT教育ネットワーク論特論A

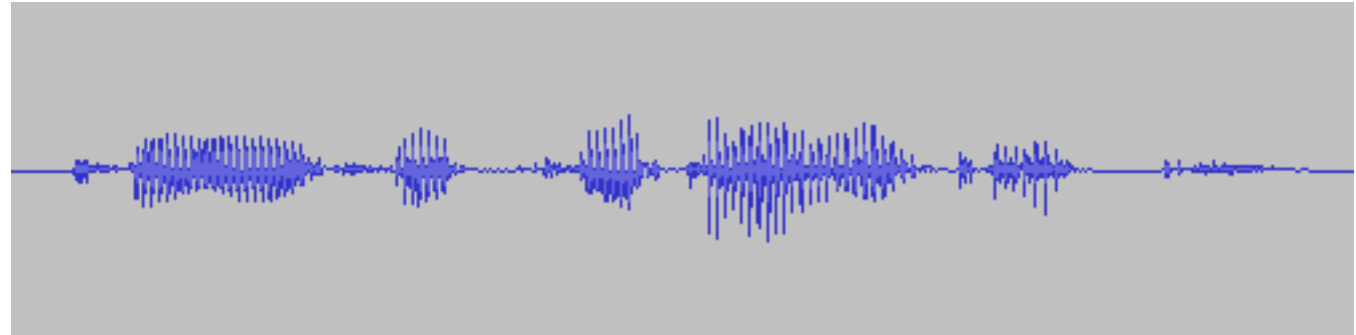
前回の復習

言語の入力と出力

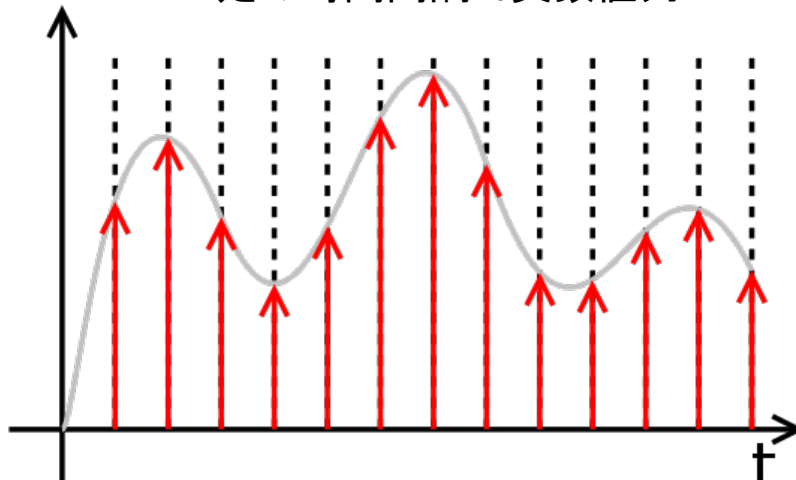
音声の入出力(すこしだけ)

音声のデジタル表現(標本化と量子化)

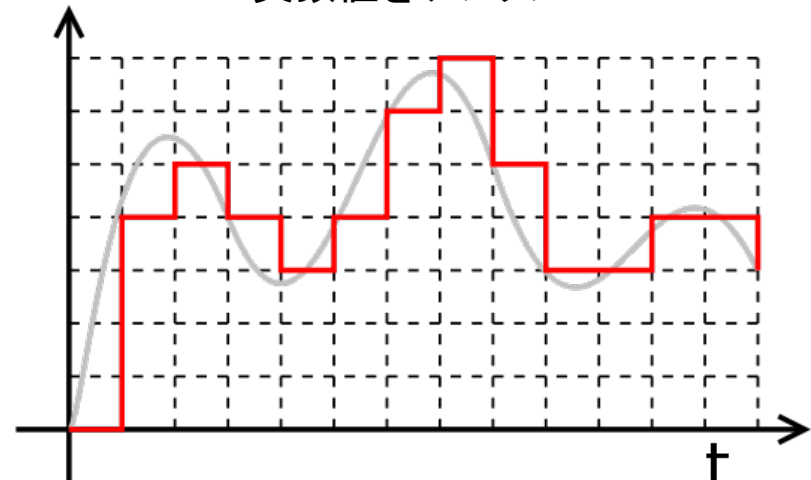
音の波形
(空気の
圧力変化)



標本化 (sampling)
一定の時間間隔で実数値列へ



量子化 (quantization)
実数値をデジタルへ

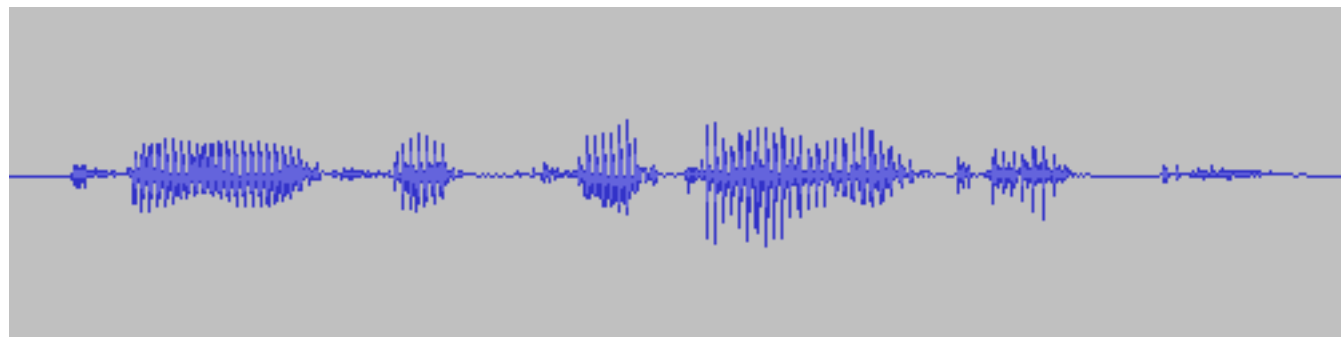


音声の入出力(すこしだけ)

音声認識(音声⇒テキスト)

Audacity(フリーソフト)を使用

音声信号



音響モデル

音素列
音韻列

t o: h o k u d a i g a kh
とー ほ く だい が く

言語モデル

単語列
テキスト

東北

大学

東北大学

音声の入出力(すこしだけ)

使える音声認識システム

Julius: 連続音声認識エンジン

PC上で**数万単語の連続音声認識を実時間で実行**

フリーソフト(京都大学ほか)

ウィンドウズ, Unix, Mac など

(試用する場合は「ディクテーションキット」がよい)

音響モデル・言語モデルを差し替え/カスタマイズ可能

(口語や他言語への適応が可能)

音声認識の精度を上げるコツ

アナウンサーのように喋る

(ハッキリと, 一定の強さ・速さで)

よいマイクを使う

(音響モデルの構築時に使ったマイクがベスト)

文字と文字コード

文字の情報処理

日本語の場合、「**小嶋**」はつぎのようなバイト列になる・・・

JIS : **3e 2e 45 68**

SJIS: **8f ac 93 88**

EUC : **be ae c5 e8**

UTF8: **e5 b0 8f e5 b6 8b** (すべて16進数)
(Unicode)



日本語を表現する方式が複数ある
(だから<文字化け>が起こる・・・)

文字と文字コード

ユニコード(Unicode)

ISO-10646 (1991) = JIS X 0221 (1995)

Windows-XP, Mac OS X, Java, XML(XHTML) などの標準に...

文字集合 UCS (Universal Coding System)

UCS-2 : 世界中の文字を16ビットで表現しようとしているが,

UCS-4 : 32ビット化しようとする動きもある.

符号化方式は UTF-8 (UCS Transferring System) が標準に...

0xxxxxxx (ASCII はそのまま)

OS内部ではUTF-16も使われる
(16ビット)

110xxxxx 10xxxxxx (欧州諸字は2バイト)

1110xxxx 10xxxxxx 10xxxxxx (日本語・韓国語・中国語は3バイト)

11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

Unicode ↔ JIS の相互変換には **変換表** が必要 !

文字と文字コード

ユニコード (Unicode)

世界のあらゆる文字を登録し、統一的な処理を可能にする。

იგი Unicode-ის ენაზე ლაპარაკობს.

ندم ا یریدال عالم أن یتکلّم، فهو یتحدّث بلغة یونیكود

Når verda ønskjer å snakke, talar ho Unicode.

Unicode が**世界標準**になりつつあるが、
まだ整理のつかない問題も多い・・・

符号化効率の不公平さ

CJK漢字の「Han unification」・・・「骨」と「骨」

ツギハギ ⇒ ソートしづらい

文字の統計学

文字はいくつあるか？

英語: アルファベット …… 26字

日本語: ひらがな …… 46字, 48字(≡ め・ゑ), 73字(が・ざ・だ・ば・ぱ)

カタカナ …… 46字, 48字(≡ ㇿ・ヱ), 73字(ガ・ザ・ダ・バ・パ)

漢字 …… 教育漢字(1,006字) ⊂ 常用漢字(1,945字)

⊂ JISX0208 第1(2,965字) + 第2(3,390字)

⊂ 諸橋轍次「大漢和辞典」(約50,000字)

(1955~1960)

韓国語: ハングルは理論上 11,172字

KSC5601のハングル(2,350字)で日常の99%以上

(1986~1990)

中国語: 簡体字(电气)は 2,235字. 「漢語大字典」(約56,000字)

GB2312-80 = 第1(3,755字) + 第2(3,008字)

歴史を遡れば「康熙字典」(1716年; 約49,000字)

文字と単語の統計学

単語(語彙)はいくつある？

日本語:「日本国語大辞典」(小学館), 約 450,000語

英語: 「Oxford English Dictionary」, 約 616,500語

中国語:「漢語大詞典」, 約 370,000語

日常的な単語(語彙)はいくつある？

国立国語研究所「現代雑誌九十種の用語用字」

異なり単語は 39,930個 (のべ438,000語の調査から)

(頻度1が 45.2%, 頻度2が 16.5%)

日常的には比較的少数(数万語)の語彙で十分

文字と単語の統計学

単語(語彙)はいくつ使われているか

語彙数とカバー率

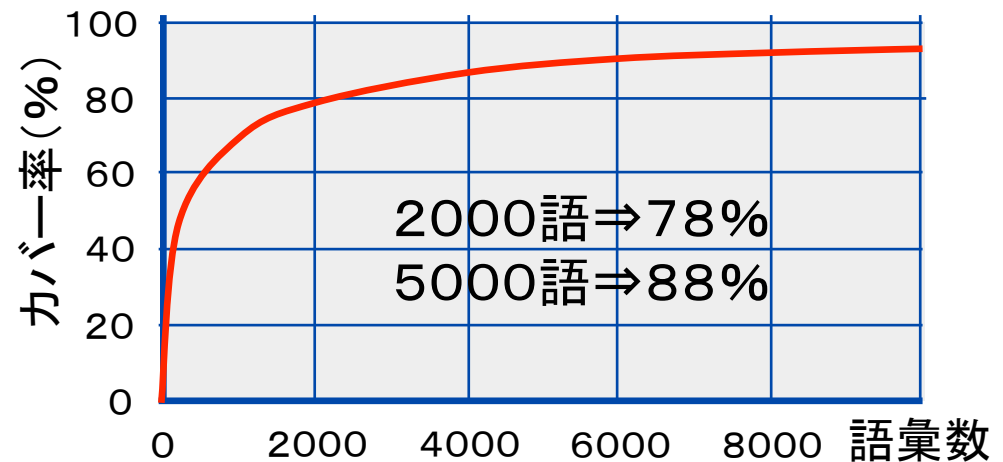
英語(LOBコーパス)

総語数(トークン数):

1,006,815

総語彙数(タイプ数):

47,888



日英仏の比較

	1000語	2000語	3000語	4000語	5000語
日本語	60.5%	70.0%	75.3%	N/A	81.7%
英語	80.5%	86.6%	90.0%	92.2%	93.5%
フランス語	83.5%	89.4%	92.8%	94.7%	96.0%

日本語は国立国語研究所「現代雑誌九十種の用語用字」による。
英語・フランス語はモスクワ国立言語研究所の調査による。

IT教育ネットワーク論特論A

前回の補足

Zipf の法則

文字と単語の統計学

単語(語彙)はいくつ使われているか

Zipfの法則 = 「頻度順位 × 出現確率 \div 一定」という経験則
ジップ

the	$1 \times 0.06887 = 0.06887$
of	$2 \times 0.03584 = 0.07168$
and	$3 \times 0.02840 = 0.08520$
to	$4 \times 0.02574 = 0.10294$
a	$5 \times 0.02300 = 0.11500$
he	$10 \times 0.00939 = 0.09390$
I	$20 \times 0.00510 = 0.10200$
if	$50 \times 0.00216 = 0.10800$
down	$100 \times 0.00088 = 0.08800$
took	$200 \times 0.00042 = 0.08400$
finally	$500 \times 0.00019 = 0.09500$
current	$1000 \times 0.00010 = 0.10000$

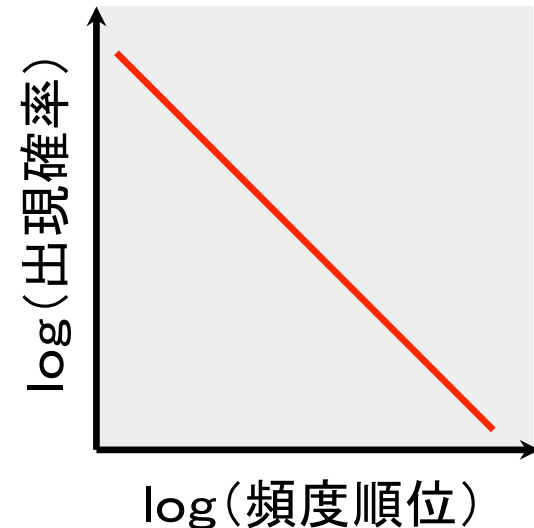
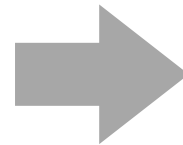
Brown
Corpus
(100万語)
で計算

文字と単語の統計学

単語(語彙)はいくつ使われているか

Zipfの法則 = 「頻度順位 × 出現確率 ≒ 一定」という経験則
ジップ

the	$1 \times 0.06887 = 0.06887$
of	$2 \times 0.03584 = 0.07168$
and	$3 \times 0.02840 = 0.08520$
to	$4 \times 0.02574 = 0.10294$
a	$5 \times 0.02300 = 0.11500$
he	$10 \times 0.00939 = 0.09390$
I	$20 \times 0.00510 = 0.10200$
if	$50 \times 0.00216 = 0.10800$
down	$100 \times 0.00088 = 0.08800$
took	$200 \times 0.00042 = 0.08400$
finally	$500 \times 0.00019 = 0.09500$
current	$1K \times 0.00010 = 0.10000$



Zipfの法則は多方面に当てはまる(らしい)

ウェブページのアクセス数/リンク数, 都市の人口,
商品の売り上げ, 日本人の苗字, などなど.

佐藤, 鈴木, 田中, 高橋, 渡辺, 伊藤, ...

IT教育ネットワーク論特論A

言語の入力と出力(2)

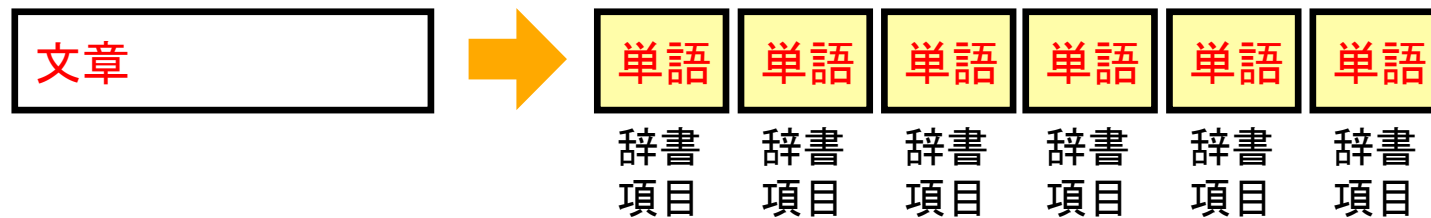
形態素解析と構文解析

形態素解析

morphological analysis

形態素解析とは何か

文章を形態素(=単語)に分割し, 辞書項目と対応づける.



「まずい朝食を食べさせられた」

まずい	朝食	を	食べ	させ	られ	た
形容詞	一般名詞	格助詞	動詞語幹	助動詞	助動詞	助動詞
「まずい」	「ちょうしょく」	「を」	「たべる」	「させ」	「られ」	「た」
〈tasteless〉	〈breakfast〉	〈OBJ〉	下一連用形	連用形	連用形	終止形
			〈eat〉	〈CAUSE〉	〈PASSIVE〉	〈PAST〉

形態素解析

形態素への分割は、じつは難しい...

欧文(英語など)場合, すでに「わかち書き」されているため
語尾変化や不規則動詞に対応すればよい.

日本語の場合
辞書を引くだけでは
済まされない

くるまではまった

来る まで は 待った

車で は 待った

車で ハマった

...

繰る 間 出 歯 舞った

もっと

尤もらしい(常識的な)結果 を選択しなければならない!

単語の切りかた
単語の繋げかた

形態素解析

日本語の形態素への分割はどうすればよいか...

辞書を引く

くるまではまった

(動詞)来る

(副助詞)まで

(係助詞)は

(動詞)待つ

(助動詞)た

(名詞)車

(格助詞)で

(動詞)ハマっ

(助動詞)た

単語(品詞)の出現頻度を吟味する

大学生協



大学

生協



大学生

協



大

学生

協

単語(品詞)の接続可能性を吟味する

動詞/未然

食べられ

助動詞/連用タ

なかつ

助動詞/基本

た

終助詞

ね

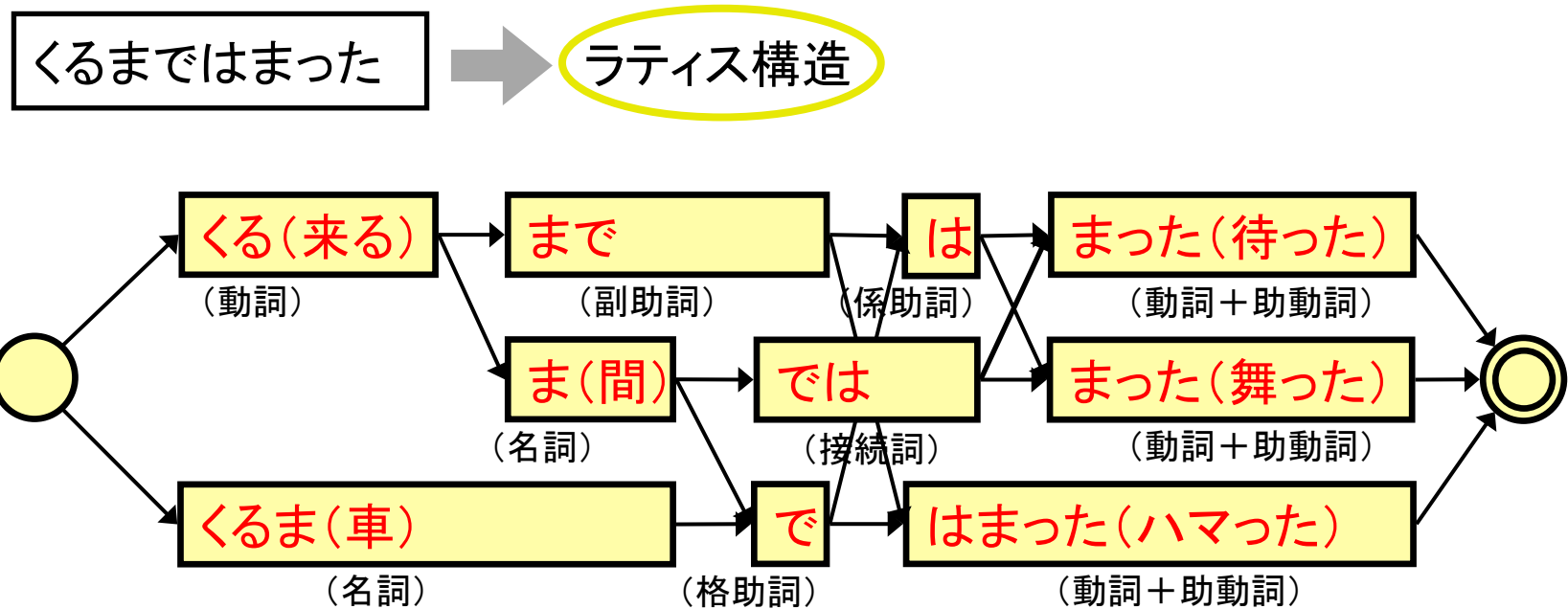
文末

。

形態素解析

日本語の形態素解析

単語分割 + 頻度チェック + 接続チェック の同時進行



形態素解析

接続チェックは「テーブル」で

接続可能性
テーブル

前⇒後
の可能性

前⇒後	名詞	助詞	形容詞	副詞	助動詞	動詞
名詞	✓	✓			✓	
助詞	✓		✓	✓		✓
形容詞	✓				✓	
副詞	✓					✓
助動詞	✓				✓	✓
動詞	✓				✓	✓

もっと

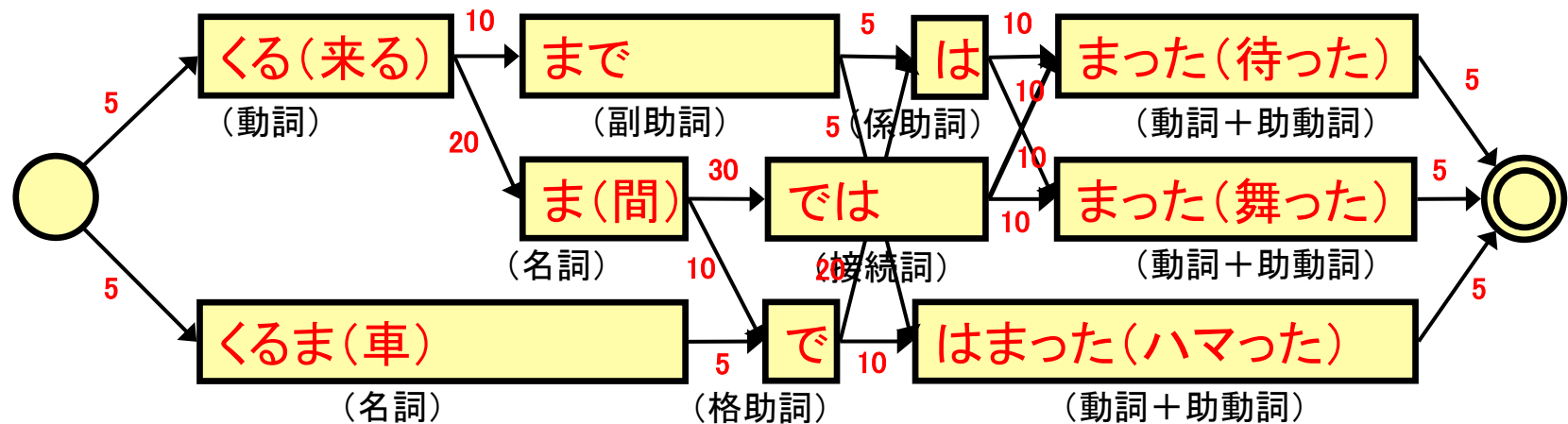
尤もらしい接続のしかたを選ぶ方法

- 形態素数/文節数の最小化（最長一致法）
- 接続コストの最小化（接続確率の最大化）

形態素解析

形態素解析は「分割＋チェック」というよりも「最適化」

接続コストが最小になる経路を見つける



状態遷移(矢印)にコストをつける (← 接続頻度や接続確率)

スタート ○からゴール ◎までの最小コスト経路を見つける！

形態素解析

形態素解析エンジン MeCab (和布蕪)

めかぶ

工藤拓氏 + NTT・CS研による研究用のフリーソフト
Unix (Linux 等), Mac OS X, Windows などで動作.

```
% mecab ↵
```

```
車で大学まで来ます ↵
```

```
車      名詞, 一般, *, *, *, *, 車, クルマ, クルマ  
で      助詞, 格助詞, 一般, *, *, *, で, デ, デ  
大学    名詞, 一般, *, *, *, *, 大学, ダイガク, ダイガク  
まで    助詞, 副助詞, *, *, *, *, まで, マデ, マデ  
来      動詞, 自立, *, *, 力変・来ル, 連用形, 来る, キ, キ  
ます    助動詞, *, *, *, 特殊・マス, 基本形, ます, マス, マス  
EOS
```

(ここで実演します)

形態素解析

プログラムから MeCab を利用するには

ライブラリ(API)を組み込む …… MeCab の場合

```
#include <mecab.h>
#include <stdio.h>

int main (int argc, char **argv)
{
    mecab_t *mecab;
    char *input = "私は新幹線で学会の準備をしています";
    const char *res_str;

    mecab = mecab_new(argc, argv);
    res_str = mecab_sparse_tostr(mecab, input);
    printf("%s\n", input);
    printf("%s\n", res_str);
    mecab_destroy(mecab);
}
```

mcctest1.c

MeCabオブジェクト

input を解析

リソース解放

形態素解析

プログラム(C言語)から MeCab を利用するには

ライブラリ(API)を組み込む・・・ MeCab の場合

```
% cc mcctest1.c -o mcctest1 -lmecab ↵
```

```
% mcctest1 ↵
```

私は新幹線で学会の準備をしています

私 名詞, 代名詞, 一般, *, *, *, 私, ワタシ, ワタシ

は 助詞, 係助詞, *, *, *, *, は, ハ, ワ

新幹線 名詞, 一般, *, *, *, *, 新幹線, シンカンセン, シンカンセン

で 助詞, 格助詞, 一般, *, *, *, で, デ, デ

学会 名詞, 一般, *, *, *, *, 学会, ガッカイ, ガッカイ

の 助詞, 連体化, *, *, *, *, の, ノ, ノ

準備 名詞, サ変接続, *, *, *, *, 準備, ジュンビ, ジュンビ

を 助詞, 格助詞, 一般, *, *, *, を, ヲ, ヲ

し 動詞, 自立, *, *, サ変・スル, 連用形, する, シ, シ

て 助詞, 接続助詞, *, *, *, *, て, テ, テ

い 動詞, 非自立, *, *, 一段, 連用形, いる, イ, イ

ます 助動詞, *, *, *, 特殊・マス, 基本形, ます, マス, マス

EOS

ライブラリ組み込み
(libmecab)

形態素解析

スクリプト言語 (Perl) から MeCab を利用するには

バインドイングを組み込む …… MeCab の場合

```
#!/usr/bin/perl  
use MeCab;  
$mecab = new MeCab::Tagger(""); # "起動オプション"  
$input = "新幹線でビールを飲みます";  
$tagged = $mecab->parse($input);  
printf("入力文: $input\n");  
printf("$tagged\n");
```

オマジナイ

結果は文字列として返る

Java, Python, Ruby についてもほぼ同様 !

構文解析

構文解析とは何か？

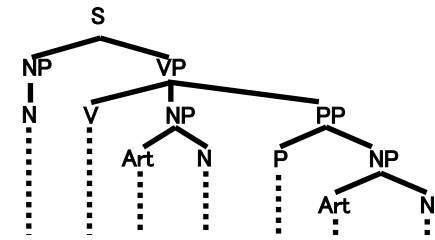
〈文の意味〉 ≠ \sum 〈単語/文節の意味〉

〈文の意味〉 $\stackrel{?}{=}$ \sum 〈単語/文節の意味〉 + 〈文の構造〉

I saw a man
with a telescope.

S → NP VP
NP → (Art) N (PP)
VP → V (NP) (PP)
PP → P NP

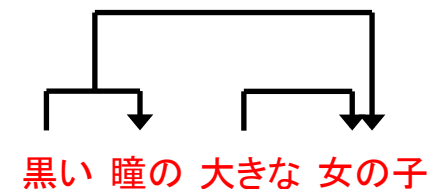
文法



I saw a man with a telescope.

黒い瞳の
大きな女の子

形容詞 + 名詞 ⇒ 名詞2
名詞 + の ⇒ 形容詞
名詞2 + の ⇒ 名詞
〈係り受けは交差しない〉



黒い 瞳の 大きな 女の子

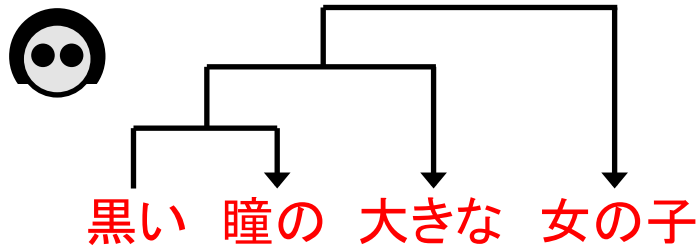
構文解析

syntactic analysis

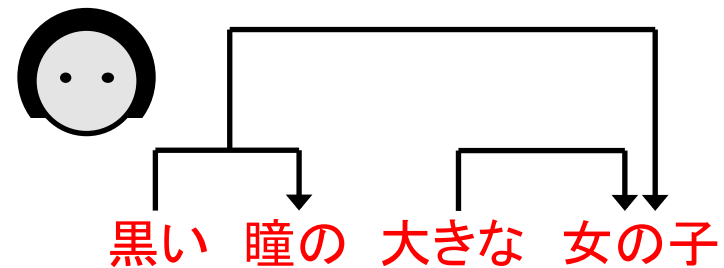
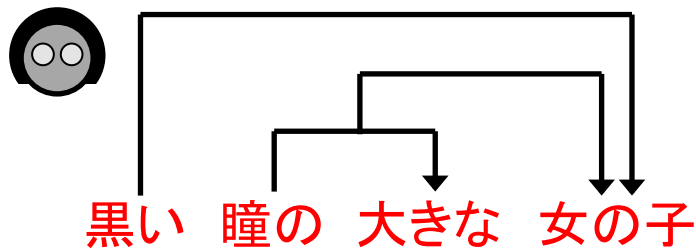
構文解析とは何か？

日本語の場合：文節列について〈係り受け構造〉をつくること

文節 = 意味と発音の点から文を自然に区切った最小単位
〈文節〉 → 〈接頭辞〉* 〈自立語〉+ 〈接尾辞・助詞・助動詞〉*



仙台/駅前/で
お/弁当/を
買い/ます/か

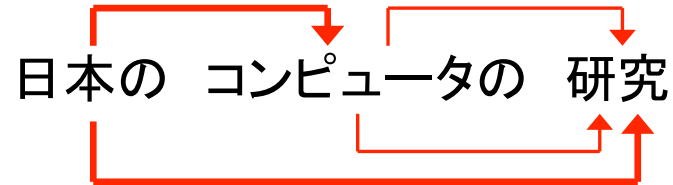


構文解析

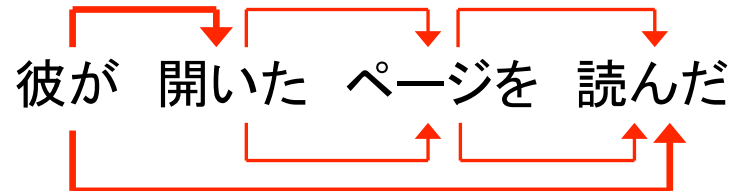
〈係り受け構造〉はどのようにつくればよいか？

文節間の〈係り受け〉の解析は難しい…

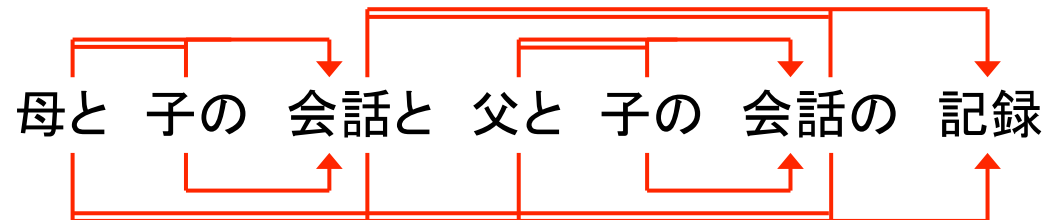
連続した
連体修飾



格要素と述語
の対応づけ



並列構造



構文解析

〈係り受け構造〉はどのようにつくればよいか？

言語には**暗黙のルール**がある

- ・ 係り受けは**交差しない**

例外) 彼女が 僕は 美しいと 思う.




- ・ **意味的に**係り受け可能な(近い)文節に係る

東北大の 通信の 研究 / 東北大の 学生の 研究 (?)




コーヒーを 歩きながら 飲んだ / 一番町を 歩きながら ...



- ・ 並列構造は**反復**や**意味的な類似性**から捉える

友人と 物理と数学を勉強する / 英語と物理と数学を ...



構文解析

〈係り受け構造〉はどのようにつくればよいか？

文節間の〈係り受け〉の解析

私は 朝の 新幹線で 学会の 準備を しました

〈名詞〉+「の」→〈名詞〉
〈名詞〉+「と」→〈名詞〉
⋮
〈名詞〉+「が」→〈述語〉
〈名詞〉+「を」→〈述語〉
〈名詞〉+「で」→〈述語〉
⋮

依存文法

CaboCha の出力

```
私は-----D
  朝の-D      |
  新幹線で----D
    学会の-D  |
      準備を-D
        しました
```

構文解析

簡単に使える構文解析ソフトがあります。

CaboCha/南瓜 (フリーソフト; Unix/Mac/Windows)

工藤拓氏 (奈良先端大 → Google)

形態素解析は MeCab が担当

```
unix> cabocha ↵
```

```
私は朝の新幹線で学会の準備をしました ↵
```

```
私は-----D  
朝の-D |  
新幹線で----D  
学会の-D |  
準備を-D  
しました
```

```
EOS
```

コーパスから
統計的学習により
抽出した規則

構文解析

CaboCha による係り受け構造解析例

```
user> cabocha -f1 ↵
私は朝の新幹線で学会の準備をしました ↵
* 0 5D 0/1 0.000000
私      名詞, 代名詞, 一般, *, *, *, 私, ワタシ, ワタシ 0
は      助詞, 係助詞, *, *, *, *, は, ハ, ワ      0
* 1 2D 0/1 1.683670
朝      名詞, 副詞可能, *, *, *, *, 朝, アサ, アサ      0
の      助詞, 連体化, *, *, *, *, の, ノ, ノ      0
* 2 5D 0/1 0.000000
新幹線 名詞, 一般, *, *, *, *, 新幹線, シンカンセン, シンカンセン 0
で      助詞, 格助詞, 一般, *, *, *, *, で, デ, デ      0
* 3 4D 0/1 1.649081
学会    名詞, 一般, *, *, *, *, 学会, ガッカイ, ガッカイ      0
の      助詞, 連体化, *, *, *, *, の, ノ, ノ      0
* 4 5D 0/1 0.000000
準備    名詞, サ変接続, *, *, *, *, 準備, ジュンビ, ジュンビ 0
を      助詞, 格助詞, 一般, *, *, *, *, を, ヲ, ヲ      0
* 5 -1D 0/2 0.000000
し      動詞, 自立, *, *, サ変・スル, 連用形, する, シ, シ      0
まし    助動詞, *, *, *, 特殊・マス, 連用形, ます, マシ, マシ      0
た      助動詞, *, *, *, 特殊・タ, 基本形, た, タ, タ      0
EOS
```

文節0

文節1

文節2

文節3

文節4

文節5

構文解析

CaboCha による係り受け構造解析例

```
unix> cabocha -f3  
朝の新幹線で学会の準備をしました
```

XML形式で出力

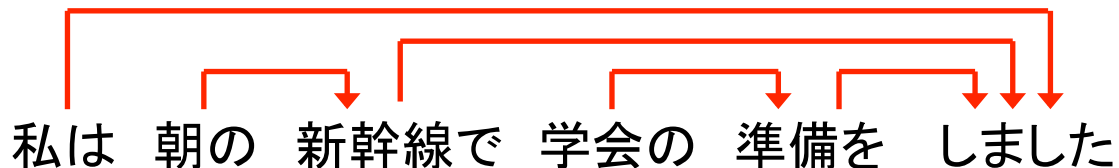
```
<sentence>  
  <chunk id="0" link="4" rel="D" score="0.000000" head="0" func="1">  
    <tok id="0" feature="名詞,代名詞,一般,*,*,*私,ワタシ,ワタシ" ne="0">私</tok>  
    <tok id="1" feature="助詞,係助詞,*,*,*,*は,ハ,ワ" ne="0">は</tok>  
  </chunk>  
  <chunk id="1" link="4" rel="D" score="0.000000" head="2" func="3">  
    <tok id="2" feature="名詞,一般,*,*,*,*新幹線,シンカンセン,シンカンセン" ne="0">新幹線</tok>  
    <tok id="3" feature="助詞,格助詞,一般,*,*,*,*で,デ,デ" ne="0">で</tok>  
  </chunk>  
  <chunk id="2" link="3" rel="D" score="1.649081" head="4" func="5">  
    <tok id="4" feature="名詞,一般,*,*,*,*学会,ガッカイ,ガッカイ" ne="0">学会</tok>  
    <tok id="5" feature="助詞,連体化,*,*,*,*,*の,ノ,ノ" ne="0">の</tok>  
  </chunk>  
  <chunk id="3" link="4" rel="D" score="0.000000" head="6" func="7">  
    <tok id="6" feature="名詞,サ変接続,*,*,*,*準備,ジュンビ,ジュンビ" ne="0">準備</tok>  
    <tok id="7" feature="助詞,格助詞,一般,*,*,*,*を,ヲ,ヲ" ne="0">を</tok>  
  </chunk>  
  <chunk id="4" link="-1" rel="D" score="0.000000" head="8" func="10">  
    <tok id="8" feature="動詞,自立,*,*,*サ変・スル,連用形,する,シ,シ" ne="0">し</tok>  
    <tok id="9" feature="助動詞,*,*,*,*特殊・マス,連用形,ます,マシ,マシ" ne="0">まし</tok>  
    <tok id="10" feature="助動詞,*,*,*,*特殊・タ,基本形,た,タ,タ" ne="0">た</tok>  
  </chunk>  
</sentence>
```

まとめ

形態素解析 = 単語分割 + 辞書項目との対応づけ

まずい	朝食	を	食べ	させ	られ	た
形容詞	一般名詞	格助詞	動詞語幹	助動詞	助動詞	助動詞
「まずい」	「ちょうしょく」「を」		「たべる」	「させ」	「られ」	「た」
〈tasteless〉	〈breakfast〉	〈OBJ〉	下一連用形	連用形	連用形	終止形
			〈eat〉	〈CAUSE〉	〈PASSIVE〉	〈PAST〉

構文解析 = 文節間の〈係り受け〉の解析



MeCab / CaboCha のような解析ツールを活用！

次回は 7月13日

以上ですが...

<http://www.myu.ac.jp/~xkozima/course/in-itnet.html>