

# 自然言語処理 第4回

中田尚

2025/5/13

# テキストと評価

- テキスト
  - 『自然言語処理の基礎』、奥村学著、コロナ社、ISBN：978-4-339-02451-7
  - 本日配布
- 学生に対する評価
  - 科目認定条件
    - ※出席率について80%以上であること。
    - ※定められた提出物が80%以上提出されていること。
  - 科目評価方法
    - 小テスト20%・グループワークの貢献度20%及び期末レポート60%
    - 小テスト及び期末課題レポートなどを総合して評価。

# 復習：動的計画法

- 前提
  - 今から小テストをしたい
  - 左右の座席は空席にしたい
    - 通路は存在しないものとし、両端以外は左右に必ず席が存在する
  - 列単位で座って良い席と座らない席を指定する
  - 前後には十分多くの席が続いているとする
- 問題
  - 何人かの学生に移動を命じる
    - 移動先は座って良い席の後方とする
  - 移動する学生の人数を最小にする方法を求め、その人数を答えよ


# 移動パターン

- 偶数奇数で分ける
  - 最適ではない
    - 奇数： $3+1+3+1=8$  または 偶数： $2+5+4=11$  よりも良いパターンがある

1            2            3            4            5            6            7

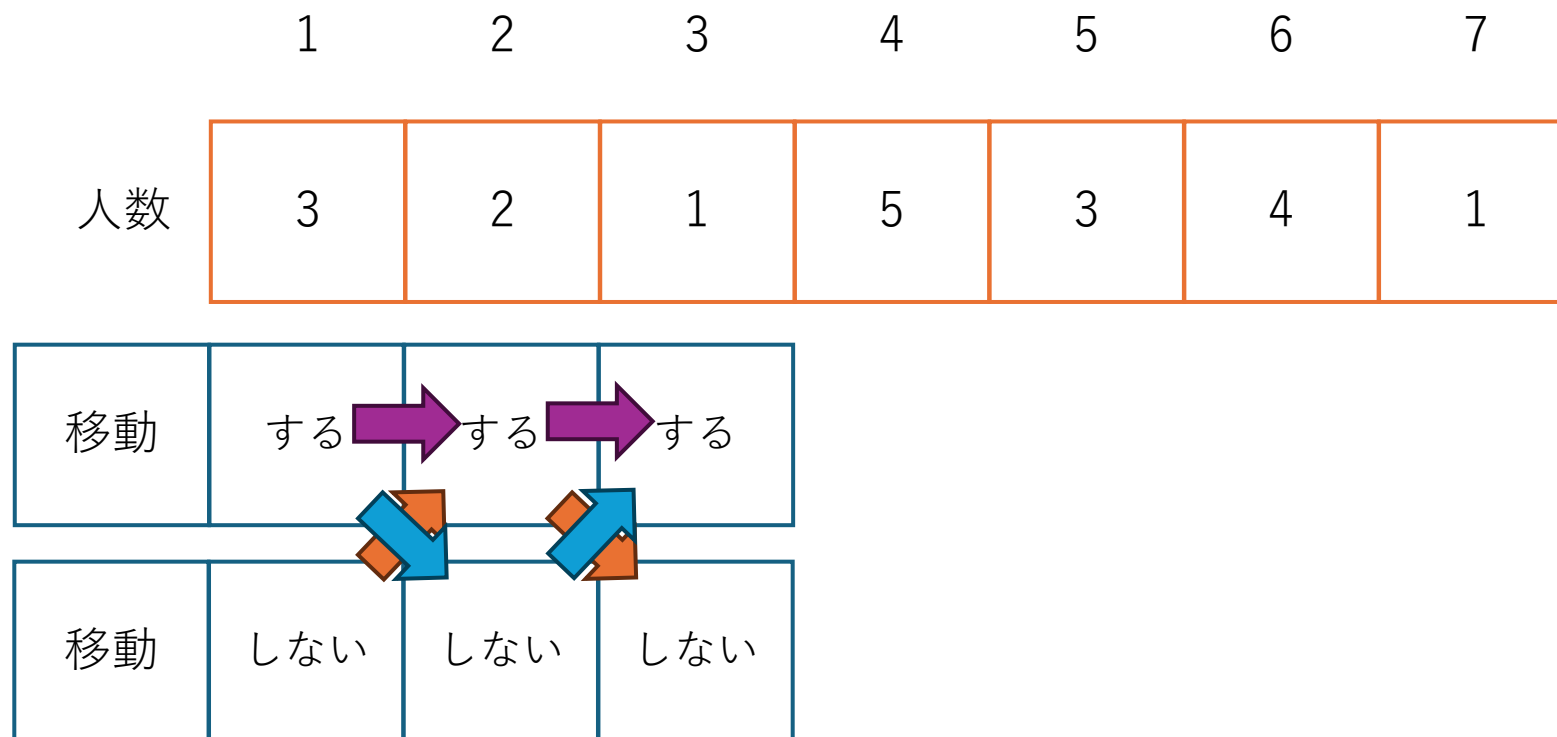
3	2	1	5	3	4	1
---	---	---	---	---	---	---

移動	する	する	する
移動	しない	しない	しない



# 移動パターン

- 2列連続で移動しても良い
  - 3列連続は意味が無い



# 移動パターンと移動人数の累積

- ➡ 連続で移動する：左と上を足す
- ↙ 前回移動したので、移動しない：左上からコピー
- ↗ 前回移動しなかったので移動する：左下と上を足す

	1	2	3	4	5	6	7
人数	3	2	1	5	3	4	1

移動 人数	する 足す	する 足す	する 足す
移動 人数	しない 足さない	しない 足さない	しない 足さない

合流点では  
小さい方を残す

# 実行結果

奇数： $3+1+3+1=8$  または 偶数： $2+5+4=11$  よりも  
良いパターンがある

- 全部埋めてみた
  - 右端上下の小さい方が求めたい答え
  - 7 (2,3,5,7列目が移動する)

	1	2	3	4	5	6	7
人数	3	2	1	5	3	4	1

する

0	3	2	3	7	6	10	7
0	0	3	2	3	7	6	10

# 実行結果

奇数： $3+1+3+1=8$  または 偶数： $2+5+4=11$  よりも  
良いパターンがある

- 全部埋めてみた
  - 右端上下の小さい方が求めたい答え
  - 7 (2,3,5,7列目が移動する)

		1	2	3	4	5	6	7
人数		3	2	1	5	3	4	1
する	0	3	2	3	7	6	10	7
	0	0	3	2	3	7	6	10

よく見ると上も下も  
同じ数列になっている



# 実行結果

奇数： $3+1+3+1=8$  または 偶数： $2+5+4=11$  よりも  
良いパターンがある

- 全部埋めてみた
  - 右端上下の小さい方が求めたい答え
  - 7 (2,3,5,7列目が移動する)

		1	2	3	4	5	6	7
人数		3	2	1	5	3	4	1
する	0	3	2	3	7	6	10	7
	0	0	3	2	3	7	6	10

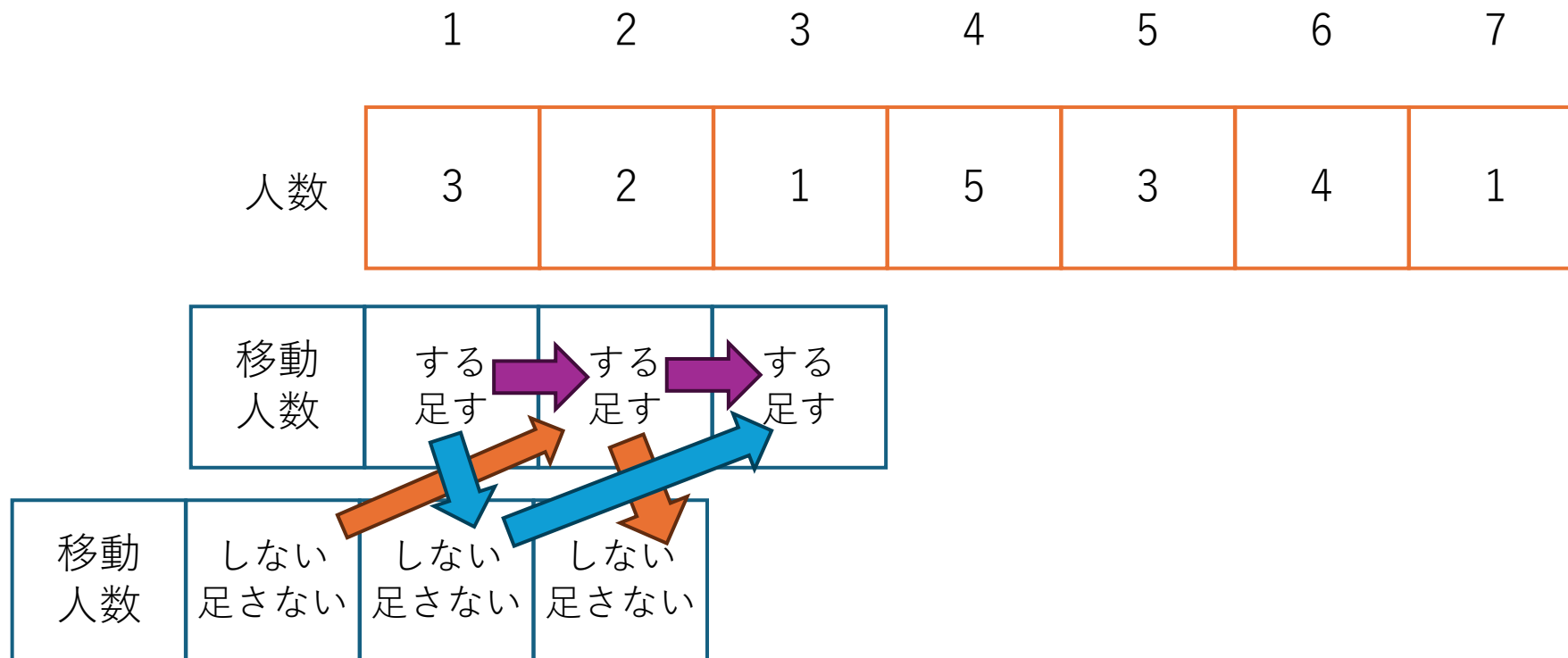
よく見ると上も下も  
同じ数列になっている

# 動的計画法

	1	2	3	4	5	6	7
人数	3	2	1	5	3	4	1

移動 人数	する 足す	する 足す	する 足す
移動 人数	しない 足さない	しない 足さない	しない 足さない

# 動的計画法



- つまり配列は1つで十分で、
- 「2つ前+今」と「1つ前+今」の小さい方を採用する



# 動的計画法

- `x=[3,2,1,5,3,4,1]`
- N要素のリスト `x` に人数が入っている
- `dp=[0]*(N+1)`
- N+1要素のリスト `dp` を準備する
- `dp[0]=0, dp[1]=x[0]` と初期化する
- `for i in range(2,N+2):`
  - `dp[i]=min(dp[i-2]+x[i-1], dp[i-1]+x[i-1])`
- `min(dp[N], dp[N+1])` が答え

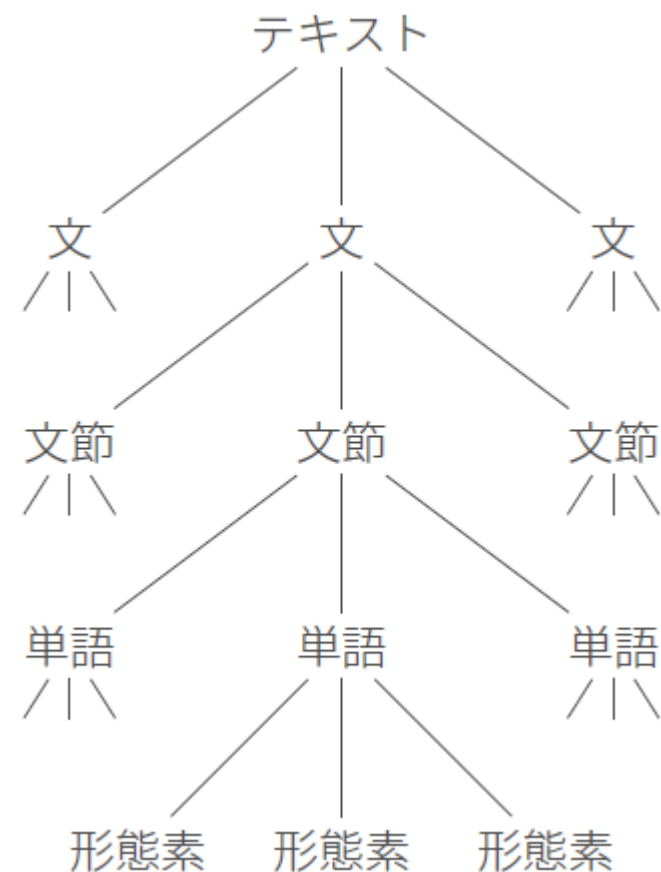
# 動的計画法の使い方

- その1：形で覚える
  - とりあえずでも使えないと話にならない
  - ~~100~~ 30回くらい使っていると、
    - ~~気持ちがわかるようになってくる~~
    - 気持ちを理解したくなってくる
- その2：理屈で考える
  - $i-1$ までの結果で $i$ が求められることに気づけるかどうか

# 形態素解析

# 形態素とは

- 形態素(Morpheme)
  - 意味を持つ最小の言語単位\*
    - 形態素には「語幹」と「接辞」がある
    - 接辞には語幹の前につく「接頭辞」と
    - 後につく「接尾辞」がある
- 単語(Word) はいくつかの形態素列から成る
  - 単語には必ず語幹が含まれる
  - 接辞は含まれる場合も含まれない場合もある
  - 例
    - illegally (違法に、不法に) → il + legal + ly  
(否定) 合法の (副詞化)
    - お茶 → お + 茶  
(敬語表現)
    - 【単語】 → (接頭辞+) 語幹 (+接尾辞)
  - 接頭辞: Prefix、語幹: Stem、接尾辞: Suffix



# 形態素解析とは

- 形態素解析(Morphological Analysis)
  - 形態素／単語を対象とした解析
  - 大きく以下の3つの処理がある
- 1. 単語分割(Word Segmentation)：入力された文を単語の並びに分解する
  - 英語のように「分かち書き」する言語では不要
- 2. 単語への品詞付与(Part-of-Speech Tagging)：単語の品詞を決定する
  - 英語は文全体から単語の品詞を判断する例：“like”の品詞：名詞（「好み」）、動詞（「好む」）、前置詞（「～に似た」）など
  - 日本語は単語だけでほぼ品詞が分かる例：好み【名詞】、好む【動詞】、好き【形容動詞】、好ましい【形容詞】
- 3. 単語の原形の復元(Stemming, Lemmatization)：語形変化、活用変化している単語を元に戻す
  - 例：flies→fly + s（複数形、三単現） taller→tall + er（比較級）
  - 例：聞いた→聞く + た（連用形） 美しければ→美しい + ば（仮定形）
- 日本語の場合処理1～3は同時に実行できる
  - 利用例：かな漢字変換（変換候補として妥当なものが挙げられる）や
  - 検索（文単位ではヒットが難しいので単語単位に分割する）がある

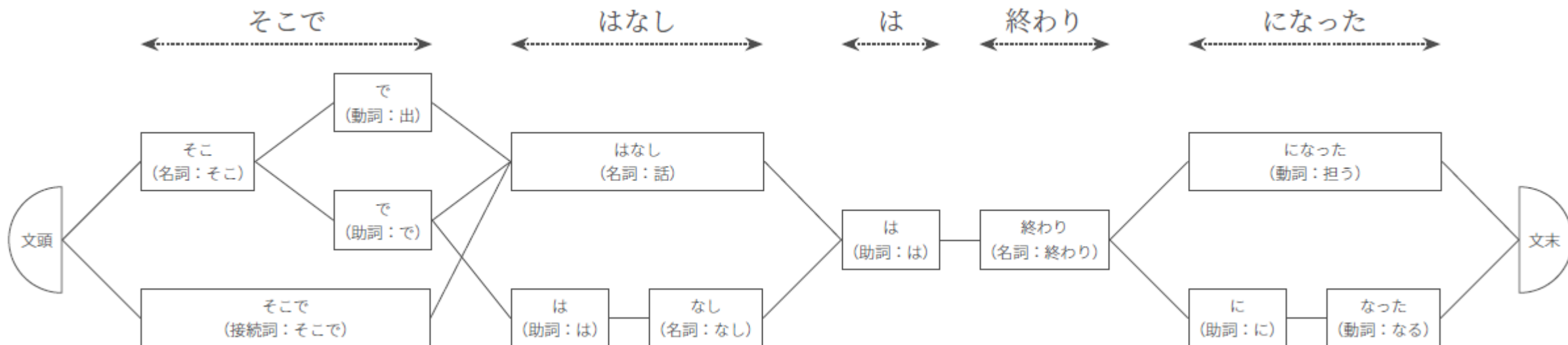


# 日本語の形態素解析

- 単語分割
- かな漢字変換
  - 自然言語の応用の一つ
  - 入力された「ひらがな文字列」を「漢字かな混じり」に変換する
  - 単語の区切りを見つけることが重要
  - 形態素解析（単語分割）
    - ていあんしたいけん → ていあん/し/たい/けん → 提案したい件  
→ ていあん/し/た/いけん → 提案した意見  
→ ていあん/し/たいけん → 提案し体験
  - 単語の分割に複数の可能性がある場合は、変換候補も複数

# 日本語の形態素解析

- 日本語の形態素解析
  - 日本語では単語分割、品詞付与、原形の復元をまとめて処理する
  - 特に単語分割が重要（単語に分割できれば品詞、原形の復元はほぼ一意に定まる）
  - 形態素解析の結果表現の1 つにラティス(Lattice) がある
    - 単語をノード、接続関係をリンクでつないだグラフの一種
    - 文頭ノードから文末ノードまでのパスが、出力単語の系列に相当する



# 日本語の形態素解析

- ラティスの構成準備

- 「単語辞書」と「接続可能性行列」を用意する
- 単語辞書：単語とその読み、品詞、活用型などが記載された辞書（データ）
- 接続可能性行列：ある単語種（主に品詞）の後にどのような単語種が出現しうるか（接続可能か）を表した行列
  - 接続可能であれば1、接続不可能であれば0で表す

- ラティスの構成方法

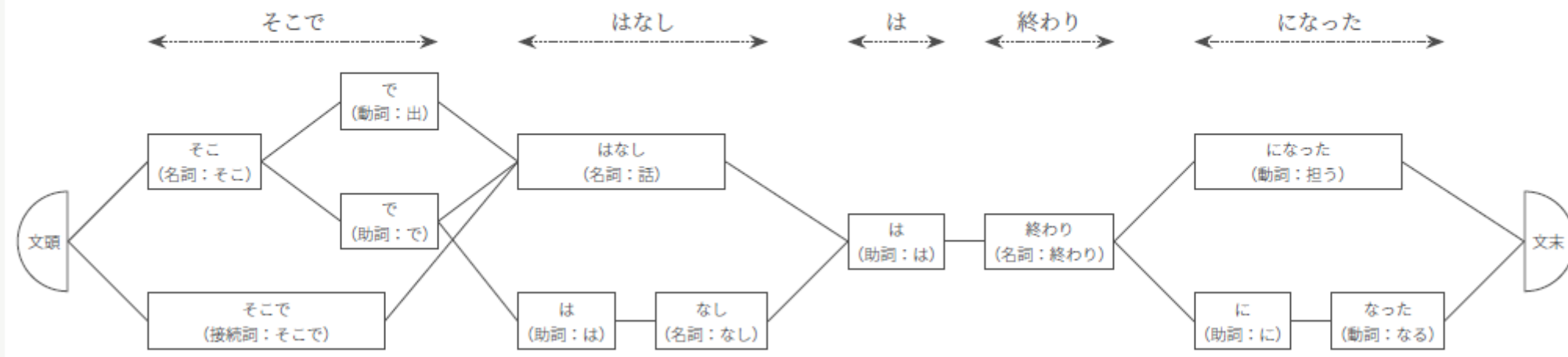
- 以下の2ステップを繰り返すことでラティスが得られる
- 1. 単語辞書を参照し、入力文字列の各文字の位置から始まる単語を取り出し、ノードとする
  - 単語辞書を参照する際に、品詞付与・原形の復元も同時に実行
- 2. 相前後する2単語について、接続可能性行列を参照し、接続可能な2単語をリンクで繋ぐ

単語辞書（例）

見出し語	読み	品詞	活用型	活用形	基本形
終わり	おわり	名詞			終わり
そこ	そこ	名詞			そこ
そこで	そこで	接続詞			そこで
で	で	助詞			で
で	で	動詞	一段	連用形	で
出	で	動詞	一段	連用形	出る
⋮					
話	はなし	名詞			話

接続可能性行列（例）

	文末	名詞	動詞	助詞	接続詞
文頭	0	1	1	0	1
名詞	1	1	1	1	0
動詞	1	1	1	0	1
助詞	1	1	1	1	1
接続詞	0	1	1	0	1



# 演習

- 文「くるまでまつ」と右の単語辞書について単語辞書・接続可能行列を用いて、形態素解析した結果をグラフを示せ
  - 単語をノード、その間の接続をリンクで表せ
  - 単語辞書を引く際には見出し語ではなく「読み」を用いる
- P.25の接続可能行列（右図）とP.147の接続可能行列が異なる
  - 動詞→助詞が0→1になっている

単語辞書		
見出し語	読み	品詞
来る	くる	動詞
車	くるま	名詞
間	ま	名詞
まで	まで	助詞
句	く	名詞
で	で	助詞
松	まつ	名詞
待つ	まつ	動詞

接続可能性行列（例）

	文末	名詞	動詞	助詞	接続詞
文頭	0	1	1	0	1
名詞	1	1	1	1	0
動詞	1	1	1	0	1
助詞	1	1	1	1	1
接続詞	0	1	1	0	1

# 日本語の形態素解析

- 最適単語列の導出
  - 選好：何らかの基準に基づいて計算し、もっとも評価の良い一つを選ぶ
  - 基準については絶対的なものではなく、経験論的に、あるいは統計的・学習的アプローチで定める
  - 最長一致法(Longest Match Method)
    - （前方から）できるだけ長い単語で構成される解析結果を選択
    - そこで/はなし/は/終わり/になった
  - 形態素数（文節数）最小法
    - ラティスを構成し、その中でできるだけ少ない形態素数で構成される解析結果を選択
    - そこで/はなし/は/終わり/になった
  - コスト最小法（実際の日本語形態素解析プログラムで用いられている）
    - ラティス中のノードとリンクにコストを与え、コスト最小のパスを最適解として選択
    - コスト最小のパスを探す際にはビタビアルゴリズム（動的計画法; DP の一種）がよく用いられる

# 日本語の形態素解析

- 単語コストは出現頻度が多い単語（品詞）ほど小さく設定される
- 接続コストは接続しやすい品詞間ほど小さく設定される
  - 名詞の次は動詞よりも助詞の方が接続する可能性が高い。
  - 例：「本買った」（話し言葉では許されるが書き言葉では不自然）よりも「本を買った」
- コストの具体的な値は人手や学習などで与えられる

# 日本語の形態素解析ツール

- MeCab (和布蕪) <https://taku910.github.io/mecab/>
  - 京都大学-NTT コミュニケーション共同研究プロジェクトで開発されたオープンソース形態素解析エンジン
  - macOS やiOS の日本語入力に利用されていると言われている
- ChaSen (茶筌) <https://chasen-legacy.osdn.jp/>
  - 奈良先端科学技術大学院大学で開発された形態素解析器
  - JUMAN をベースに開発された
- JUMAN++ <https://nlp.ist.i.kyoto-u.ac.jp/?JUMAN%2B%2B>
  - 京都大学で開発されたJUMAN の後継
  - 言語モデルとしてディープラーニングの技術(Recurrent Neural Network) を使用
- KAKASI <http://kakasi.namazu.org/index.html.ja>
  - オープンソースの形態素解析エンジン
  - 日本語の漢字仮名交じり文を平仮名やローマ字綴りの文に変換できる
- Janome (蛇の目) <https://pypi.org/project/Janome/>
  - オープンソースの形態素解析器
  - Pure Python で書かれており、辞書内包式アプリケーションに組み込みやすいシンプルなAPI を備えている



# 英語の形態素解析

- 英語の形態素解析
  - 分かち書きされているため、単語分割は不要
  - 品詞付与：「入力された単語列に対し、もっとも適切な品詞列を出力」することが重要
  - 品詞列を求める方法の1 つに統計的言語モデルを用いる方法がある
- 統計的言語モデルを用いた品詞付与問題
  - 単語列,  $W$ が入力されたとき、品詞列  $T$ を出力する。このとき、与えられた単語列  $W$ に対する品詞列  $T$ の条件付確率  $P(T/W)$ を最大とする  $T$ を求める
  - $P(T/W)$  は  $n$ 個の要素（単語、品詞）の共起に対する条件付確率
  - そのまま  $P(T/W)$ をコーパスから推定するのは現実的ではない  
→仮定をおいて近似する

# 条件付き確率

- $P(\text{品詞}X | \text{単語}Y)$

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)} = \frac{C(X \cap Y)}{C(Y)}$$

- 単語Yが品詞Xで出現する回数 ÷ 単語Yが出現する回数

# 英語の形態素解析

- 【仮定A】 品詞の出現する確率は前後の単語とは独立である
- $P(T/W)$  は以下のように近似できる

$$P(T|W) = P(t_1, \dots, t_n | w_1, \dots, w_n) = \prod_{i=1}^n P(t_i | w_i)$$

- $P(t_i | w_i)$  は単語  $w_i$  が品詞  $t_i$  である確率
  - 品詞タグ付コーパスがあれば計算できる
- 「各単語について、複数の品詞の中で、どの品詞が出てきやすく、どの品詞はほとんど出てこないか」
- n-gramと同様に「最も可能性の高い品詞」のみを採用する
  - 例えば「book は名詞」と決め打ちすることに相当
- このような単純なモデルでも90% くらいの精度
- “the”（冠詞）の次に名詞が出てきやすい、などは表現できない  
→より工夫したモデルを考える

# 例

$$\begin{aligned} P(T|W) &= P(\text{名詞、助詞、名詞} | \text{吾輩は猫}) \\ &= \prod_{i=1}^n P(t_i|w_i) = P(\text{名詞}|\text{吾輩})P(\text{助詞}|\text{は})P(\text{名詞}|\text{猫}) \end{aligned}$$

- 日本語は単語が決まると品詞が決まることが多いが、英語は単語の品詞を決めることが重要

# 英語の形態素解析

- ベイズの定理を使うと  $P(T|W)$  は以下のように変形できる

$$P(T|W) = \frac{P(T) P(W|T)}{P(W)}$$

- 分母は  $T$  を変化させても変化しないので無視して良い
- 分子を最大化する  $T$  を求めれば良い
- 【仮定B-1】 ある品詞が出現する確率は、直前の品詞だけに依存する
  - すなわち、品詞列の  $i$  番目の品詞がどれかは  $(i-1)$  番目の品詞（のみ）に依存するので、次のように近似できる
  - $P(T) = \prod_{i=1}^n P(t_i | t_{i-1})$  （ただし  $t_0 = \phi$ （文頭を表す仮想的な品詞）とする）
- 【仮定B-2】 各品詞に対してある単語が出力される確率は前後の品詞とは独立である
  - すなわち、品詞列の  $i$  番目の品詞がどれかは単語列の  $i$  番目の単語（のみ）に依存するので、次のように近似できる
  - $P(W|T) = \prod_{i=1}^n P(w_i | t_i)$

# 英語の形態素解析

$$P(T|W) = P(t_1, \dots, t_n | w_1, \dots, w_n) = \prod_{i=1}^n P(t_i | w_i)$$

これと逆になって  
いる点に注意

- よって  $P(T|W)$  は以下のように近似できる

$$P(T|W) \approx \prod_{i=1}^n P(t_i | t_{i-1}) P(w_i | t_i)$$

- このとき  $P(T|W)$  が最大となるパスを求めれば良い
  - (品詞の並び)  $\times$  (その品詞でその単語が使われる割合) を最大化
  - I(名詞) book(名詞) よりも I(名詞) book(動詞) が選ばれるようになることを期待
- 日本語のときと同様にビタビアルゴリズムを適用して求めることができる
- 統計的言語モデルを用いた、英語の品詞付与は98～99 %程度の精度

# 英語の形態素解析ツール

- Tree Tagger <https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/>
  - シュトゥットガルト大学で開発された品詞付与ツール
  - ドイツ語、英語、フランス語など多くの言語に対等している
- Porter Stemmer <https://tartarus.org/martin/PorterStemmer/>
  - Martin Porter 氏の発表した論文のアルゴリズムに基づいた英語の語幹 (stem) を取り出すツール
  - 各種プログラミング言語版が配布されている

# mecab-python

- ライブラリと辞書のインストール
  - `pip install mecab mecab-python unidic_lite`



# サンプルプログラム解説

- **analyze\_sentence(sentence, N=10)**
  - sentenceのすべての部分文字列を解析する
    - 「わたしは」ならば「わ」「わた」「わたし」「わたしは」「た」「たし」「たしは」「し」「しは」「は」
- **get\_pos\_candidates\_for\_substring(substring, N)**
  - 部分文字列に対応する品詞をN個まで取得する
  - **mecab.parseNBest(N, substring)**
    - 結果が文字列で帰ってくるので、手動で解析する
      - 例「まつ マツ マツ 待つ 動詞-一般 文語四段-タ行 連体形-一般」
    - 今回は品詞だけを保存する
    - 一般的ではない用法も含まれるのでかなり多い
- ここまでの結果を**result**に保存

# サンプルプログラム解説

- `display_connections_from_text(text, result)`
  - 一覧表示
- `connection_matrix = {"名詞":{"名詞": 0.6, "動詞": 0.7, "形容詞": 0.0, ...}`
  - 連結行列、品詞の並びにスコアを付ける。0.0はあり得ない組み合わせ
  - 特に「記号」は品詞ではないのですべて0.0にした
- `enumerate_valid_sequences_complete(text, result, connection_matrix)`
  - 連結行列を0またはそれ以外として、あり得るパターンを全部表示
    - かなり長くなる。

# サンプルプログラム解説

- `dp_enumerate_sequences_top_k(text, result, connection_matrix, k)`
  - 常に中間結果を上位**k**個に限定して探索
    - **k**が小さすぎると真の解を見逃す可能性がある
  - 最後にスコアの上位**10**個を表示
    - 1: `-0.2107, [( 'くるま', '名詞'), ('で', '助詞'), ('まつ', '名詞')]`
    - 2: `-0.2107, [( 'くるま', '動詞'), ('で', '助詞'), ('まつ', '名詞')]`
    - 3: `-0.2107, [( 'くる', '動詞'), ('まで', '助詞'), ('まつ', '名詞')]`
- 以下の値を最大化した（品詞の並びのみを最大化）

$$P(T) = \prod_{i=1}^n P(t_i | t_{i-1})$$

# 次回予告

- 文法を用いた構文解析
- 品詞だけを用いた接続判定には限界がある
  - 走る＋です
    - 動詞＋助動詞
  - 走ら＋ない
    - 動詞＋助動詞

# 文法の役割

- 日本語は語順に寛容
  - 私は彼を助けた
  - 彼を私は助けた
  - 意味としては同じ（ニュアンスや強調は変わる）
  - 助詞の役割が大きい
- 英語は語順が大切
  - I helped him.
  - He helped me.
- 英語には前置詞（in, on, at, …）があるが助詞ほどの柔軟な役割はない