日本語形態素解析入門

Version 0.9.1

山下達雄
tatuo-y@is.aist-nara.ac.jp
奈良先端科学技術大学院大学 自然言語書理学講座

1998年5月14日

目 次

1 形態素解析とは **2**
1.1 曖昧性 . . . 2
1.2 役割 . . . 3
1.3 将来 . . . 3

2 仕組み **3**
2.1 辞書検索 . . . 3
2.1.1 Common Prefix Search . . . 4
2.1.2 AC法 . . . 7
2.2 接続可能性確認 . . . 7
2.3 最適解選択 . . . 8
2.3.1 様々な方法 . . . 8
2.3.2 コスト最小法 . . . 8
2.3.3 確率モデル . . . 9
2.4 その他 . . . 11
2.4.1 活用語尾の扱い . . . 11
2.4.2 未定義語の扱い . . . 11

3 解析精度の向上 **12**
3.1 解析精度の測定 . . . 12
3.2 連語登録 . . . 12
3.3 可変長連接規則 . . . 13
3.4 人手によるコスト体系と確率モデルの統合 . . . 14
3.5 誤り駆動学習 — Error-Driven Learning — . . . 14
3.6 属性/クラスの決定 . . . 15

参考文献 **16**

1 形態素解析とは

形態素解析とは、与えられた文を形態素単位に区切り処理である。形態素 (morpheme) とは「意味をもつ最小の言語単位」と定義される。工学的な立場では、形態素 (lexeme) を「辞書にある項目 (文字列) とそれに付与されているタグ (品詞など) の組み」として扱う。

しかし、日本語では、形態素とは何かという絶対的なものが確立されていないので、人によって品詞体系が異なる。例えば、「待って/動詞」を一語とみなさず「待/動詞」+「って/活用語尾」とみなしたり、「僕ら/名詞」を「僕/名詞」+「ら/接尾」とみなしたり。工学的な立場での共通の形態素概念 (品詞体系) の整備が待たれるところである。

例 :「魚は僕らを待っている」という文は、以下のように解析される。

魚 [普通名詞]
は [副助詞]
僕ら [普通名詞]
を [格助詞]
待って [動詞-子音動詞タ行-タ系連用テ形]
いる [動詞性接尾辞-母音動詞-基本形]

形態素解析の三つの機能

1. tokenization

I am a pen → I | am | a | pen

わかち書きされる言語 (英語など) とわかち書きされない言語 (日本語など) で捉え方が異なる [山下ら 98]。

2. lemmatization

goes → go, am → be, cats → cat,

活用のある言語のみ

3. part-of-speech tagging

I am a pen → I[PRP] am[VBP] a[DT] pen[NN]

形態素解析システムの最終的な出力

1.1 曖昧性

英語などのわかち書きされている言語では、各単語にどんなタグをふるかという品詞付与の曖昧性だけを考慮すれば良い (本当はそうとも言えない)。これに対して日本語のようなわかち書きされていない言語では、品詞付与の曖昧性に加え区切りの曖昧性も考慮する必要がある。

- 品詞付与の曖昧性
  “今日” = 名詞 or 副詞 ？　“live” = 動詞 or 名詞 or 形容詞 ？
- 区切りの曖昧性
  - 組合せ曖昧性

暴　力　団

暴　力	団

– 交差曖昧性

図　書　館	員
図　書	館　員

1.2　役割

形態素解析システムの利用方法には大きく分けて次の二つがあります。

より高度な自然言語処理のための前処理

解析結果を係り受け・構文解析を通して、意味解析、機械翻訳、要約などのより高度な自然言語処理に利用するというのが、コンピュータによる言語の理解を目標とするこの分野での形態素解析の役割です。形態素解析処理だけでは完全に解消できないような曖昧性はそのまま後の処理に渡してしまうというのが合理的です。

求められる機能は、正確さ (解の候補に必ず正解が含まれていて欲しい)、未定義語の認識、上位処理との連携などです。研究のために利用することがほとんどで精度重視のシステムが求められます。

形態素解析結果をそのまま利用

形態素解析は他の自然言語処理技術と比べかなり実用化されていると言えます。より高度な自然言語処理システムの完成まで待ってられないので、形態素解析それ自体を完結した自然言語処理システムとして利用したいという要求があります。例えば、検索のためのインデックス作成、読み上げのための読み付与、日本語入力支援、言語統計のための前処理などが挙げられます。

求められる機能は、速い、使いやすい、コンパクト、動く (安定) などで、とにかく使えるシステムが求められます。

1.3　将来

現在、形態素解析に関する技術は成熟段階に来ているといっても言いすぎではないでしょう。形態素解析処理に関する大きな枠組・理論はほぼ出尽くしているので、今後の課題として些細な問題を一つ一つ解決していくということが重要になっていきます。例えば...。実用に耐え得るシステムの構築を目指すためには不可欠なことです。

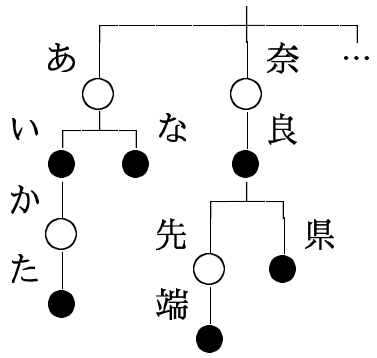
2　仕組み

処理はおおまかに、**形態素辞書の検索**、**形態素間の接続可能性の確認**、**最適解選択**、の三つに分けられる。効率化のためにこれらの処理を統合することが多いが、ここでは、これらの処理が完全に独立で順次実行されるものとして説明する。

2.1　辞書検索

まず最初に文中の形態素を認識する必要がある。文のあらゆる部分文字列で形態素辞書を検索する[1]。ここでは、この処理をいかに効率的に行なうかを解説する。

[1]字種 (と少量の辞書) によって形態素を認識する方法もあるが、ここではとりあげない。



ノード●は単語の終わりを表す (実際は品詞等の情報が格納されている)。このトライに格納されている単語は「あい」「あいかた」「あな」「奈良」「奈良先端」「奈良県」。例えば「あいかたが…」で Common Prefix Search を行なうと、「あい」「あいかた」が検索結果となる。

図 1: トライによる辞書検索

2.1.1 Common Prefix Search

Common Prefix Search とは、検索キーの prefix (接頭文字列) になっている辞書エントリを取り出すアルゴリズムである。例えば、下に示す辞書に対して、検索キー「東海道新幹線に乗った」で Common Prefix Search を行なうと結果として「東」「東海」「東海道」「東海道新幹線」が返って来る。

東 : :名詞
東海 : :名詞
東海道 : :名詞
東海道新幹線 : :名詞
東京 : :名詞
東京都 : :名詞

つまり、形態素解析システムへの入力文に対して、一文字ずつずらしながら Common Prefix Search を行なえば、文中の全ての形態素を取り出すことができる。

実装に一番適しているのはトライと呼ばれるデータ構造である。1 回の走査で探索できる。トライ (trie) とはディジタル検索のためのデータ構造の一つで、木構造で表現される [情報科学辞典]。詳しいアルゴリズムに関しては [Sedgewick 92] を参照。形態素解析では文字単位のトライを使用する (図 1, 図 2)。素直に木構造を実装するとデータ効率が悪いので、様々な工夫が凝らされている。

青江 [青江順一 88] は二つのトライを用いるダブル配列という検索アルゴリズムを提案しており、形態素解析システム Breakfast[颯々野ら 96] ではそれを用いて Common Prefix Search を実装している。

形態素解析システム JUMAN2.0[松本ら 94] では、ハッシュデータベース (NDBM) を用いてトライを効率的に実装している [Kurohashi et al 94]。ある長さの文字列で検索したとき次に何文字で検索すれば良いかという情報を付加することにより、無駄な検索を防いでいる。例えば「コンビニエンス」という文字列を Common Prefix Search することを考える。実際には「コンビ」「コンビニ」「コンビニエンス」しか辞書にないことが分かっているならば、図 3のように次に何文字で検索するかがわかれば、4 回のハッシュ操作ですむ。単純にトライを辿ると、文字数と同じ回数の操作が必要になる。

形態素解析システム「茶筌」[松本ら 97] では JUMAN2.0 の検索部分にパトリシア木を用いてトライの実装を行なっている (文献 [山下 96])。パトリシア木とは，ビット単位のトライ (バイナリトライ) 中で枝が

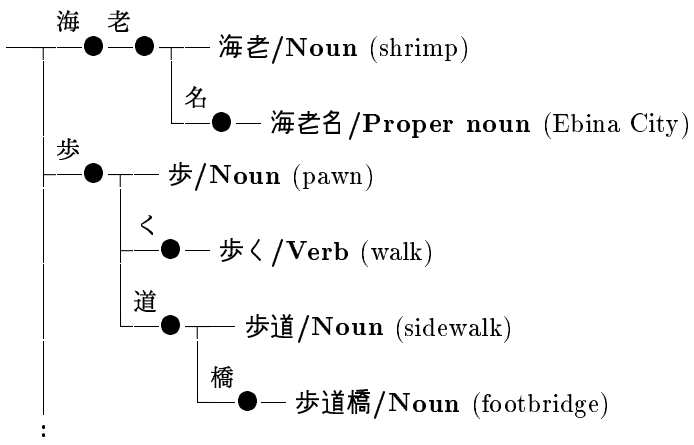


図 2: Japanese TRIE Structure Dictionary

辞書検索する文字列	引き出された情報	
	形態素情報 (一部)	次に何文字で検索するか？
1. コ	なし	3 文字で検索せよ
2. コンビ	普通名詞	4 文字で検索せよ
3. コンビニ	普通名詞	7 文字で検索せよ
4. コンビニエンス	普通名詞	これでおしまい

図 3: JUMAN2.0 の辞書検索方法

1 本しかない無駄なノードをまとめて圧縮し、効率化を図った木構造である (図 4)．図中のパトリシア木のノード上の数字は，何ビット目をチェックするかを表している．そのビットが 1 か 0 かによって辿る枝が異なる．バイナリトライでは全てのビットをチェックするが，パトリシア木ではノード上で指定されたビットしかチェックしないため，最終ノードに到着後にキー全体をマッチングして成否を確認する必要がある．パトリシア木に関しては [Sedgewick 92] を参照されたい．

さて，パトリシア木を用いて日本語形態素辞書 (EUC コード) を実装すると図 5に示すように，敷居ビット (16 の倍数) をチェックするノードの左 (0) 側のノードを見ていくことにより，Common Prefix Search が実現できる。図の例では「あえくれ…」で Common Prefix Search を行なって、「あ」「あえ」「あえくれ」が結果として返される。以下に挙げるの 3 つの事実により，16 の倍数の位置のビットをチェックしているノードがあれば，その長さ (ビット長) の単語がその左 (0) 側に存在するということが保証されている．

- 全角文字は 1 文字が 16 ビットで構成される．
- EUC では全角文字は 0 ビット目 (一番左) が必ず 1 になる．つまり全角文字列 (単語) の内部では 16 の倍数の位置のビットは必ず 1 になる．

  あい = 1010010010100010100100101000011

- 文字列 (単語) は後ろに 0 が続く無限ビット列としてパトリシア木に挿入される．つまり全角文字列 (単語) の外では 16 の倍数の位置のビットは必ず 0 になる．

  あい = 10100100101000101010010010100011000000...

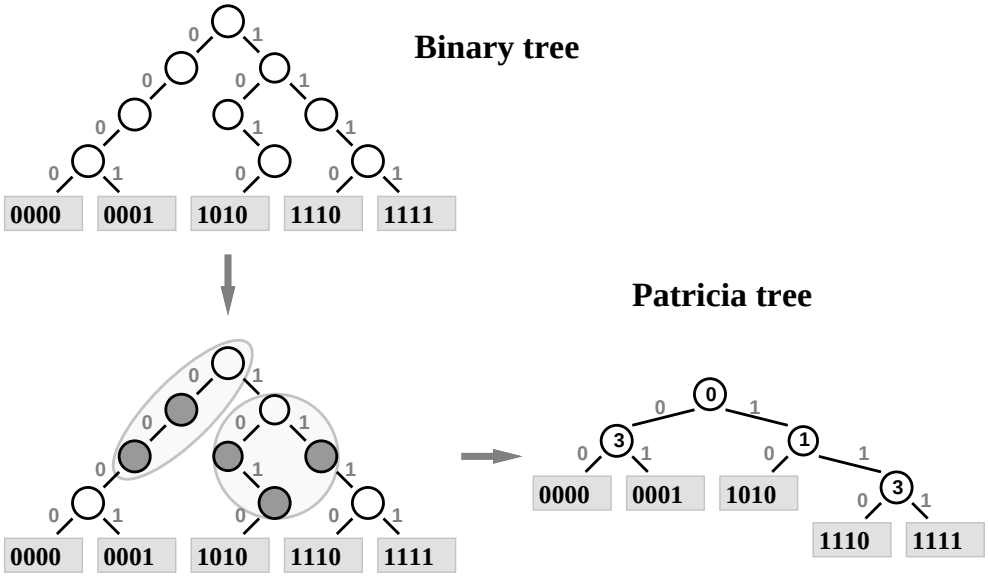


図 4: パトリシア木とは？

しかし、全角文字から成る単語しか登録できないという欠点がある。

suffix array を用いた高速文字列検索ライブラリ SUFARY[NAIST CL-Lab. 98] には Common Prefix Search を行なうための関数が用意されている。suffix array に関しては SUFARY 付属のドキュメントを参照されたい。suffix array はそのまま疑似的なトライとみなせるので、Common Prefix Search の実装は非常に容易である。suffix array とトライの関係については [山下ら 97-2] を参照されたい。

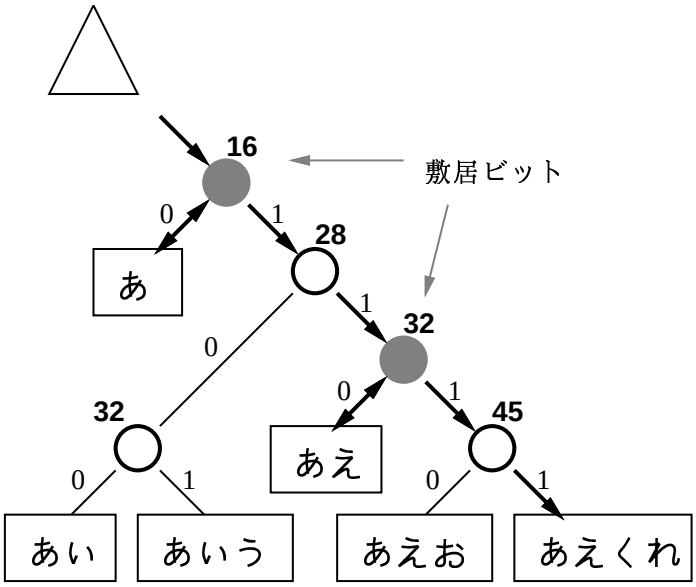
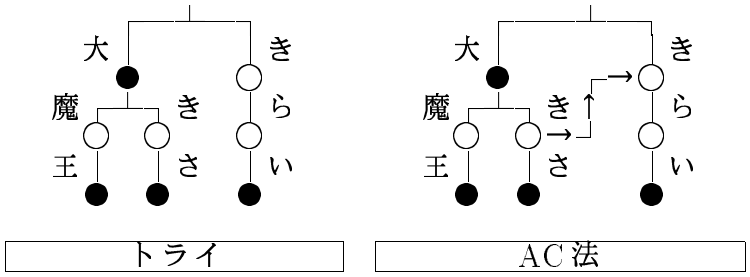


図 5: パトリシア木による辞書引き



ノード●は単語の終わりを表す (実際は品詞等の情報が格納されている)。格納されている形態素は「大」「大魔王」「大きさ」「きらい」。矢印はあらかじめ作成された失敗関数を表す。
入力文「大きらい」を例に解説する。普通のトライでは「大きらい」「きらい」「らい」「い」と入力文を一文字ずつずらしてトライを探索する。しかし、AC 法では「大き」まで行って次の文字が異なっても失敗関数によって検索が続けられるので、無駄が省ける。

図 6: AC 法による辞書検索

2.1.2 AC 法

入力文をずらしながら毎回トライを検索するのは複数回読まれる文字があるので無駄である。

丸山 [Maruyama 94] は、Aho と Corasick が提案した文字列マッチングの方法 [Aho 90] を利用して、入力文を一回スキャンするだけで、入力文に含まれる全ての形態素を取り出している (図 6)。

森 [森 96] は AC 法を決定性オートマトン (DFA) で実装し、検索速度を高速化させた。

AC 法は前節の Common Prefix Search と比べ検索時間が少ないという利点を持つが、大きい記憶域を必要とするという欠点をも合わせ持つ。今後の計算機記憶媒体の発展が期待される。

2.2 接続可能性確認

形態素認識の結果中の隣接する形態素同士が、接続するか否か、どれくらい連接しやすいかを調べる処理である。一般的には、二次元の表や二つの引数を取る関数として実装される。二つの値は、左の (直前の) 形態素と右の (現在の) 形態素の属性 (品詞など) やオートマトンの状態番号と右の形態素の属性だったりする。形態素が右にあるときと左にあるときでは違う属性として扱う場合が一般的である。例えば、動詞は左から見るときの属性は「動詞」で十分だが、右から見るときには「動詞 命令形」のように活用形の情報も欲しい。

さて、連接するか否かは『動詞の基本形と格助詞は連接できない』といった規則で定義される。ただし「連接できない」には 2 種類あることに注意が必要である。

非常に連接しにくい 普通の文では文頭に「を/助詞」は来ないと思われる。つまり、『「文頭」には「を/助詞」は連接しない』という規則を書きたくなる。しかし、絶対とは言い切れない。対話文であらわれる可能性がある。コーパスから推定した連接確率値が 0 の場合も絶対連接しないとは言い切れない。『連接しないことも無いとは言えないが非常に連接しにくい』規則として定義するべきである。

絶対連接できない 活用語の語幹と語尾をそれぞれ別個の品詞としている品詞体系もある。この場合『助詞には連体形語尾「る」は絶対連接しない』と言っても問題はない。

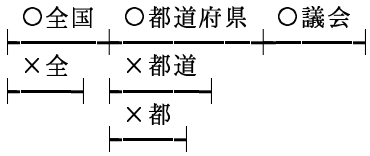
どれくらい連接しやすいかはコストや確率値などで表される。例えば『名詞と動詞は連接しにくさは 100』など。

2.3 最適解選択

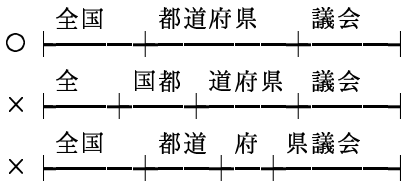
2.3.1 様々な方法

多くの形態素解析システムで用いられている優先規則を以下に示す [人工知能ハンドブック][長尾 96]。

左最長一致法 文を左から見て最も長い形態素から優先して切り出していく。



形態素数最小法 入力された文字列に対して可能な形態素の分割を列挙して、そのうち最も形態素数の少ない解を優先させる。



2文節最長一致法 文を左から見て2文節毎の長さが長い解を優先する。

文節数最小法 入力された文字列に対して可能な形態素の分割を列挙して、そのうち最も文節数の少ない解を優先させる。

コスト最小法 各形態素間の接続と形態素そのものにコストを与えて、その合計が最小のものに高い評価を与える。

確率モデル コーパスなどから形態素の接続確率や出現確率を推定し、それを用いて最適解を選択する。解析時に各形態素間の接続と形態素そのものに先に推定した確率値を与えて、その積の合計が最大のものに高い評価を与える。確率値を変換すれば、コスト最小法のコストして利用可能。

以降、コスト最小法と確率モデルについて詳しく説明する。

2.3.2 コスト最小法

コスト最小法とは、図 7のようなラティス状のグラフの全てのノード (語) とリンク (語と語の連接) に適当なコスト (図中の数値) を与え (前者を形態素コスト、後者を連接コストと呼ぶことにする)、コストの合計値が最小なパスを最適解として選択する方法である。

例えば、図 7の例では最適解としてコストの合計値が最小なパス

さかな/名詞 ― だ/判定詞 ― よ/助詞

が選択されている。

日本語の場合、人手によってコストを設定することが多かったが、近年、日本語品詞タグ付きコーパスの整備が進んだため、コーパスを用いて自動的にコストを設定するという研究が多くなって来ている。小松ら [小松ら 95] は、品詞タグ付きコーパスを評価用データとして用いて、コスト決定用のルールの確からしさを、数理計画法の手法で決定するという研究を行っている。コーパスから推定した確率値を変換してコストとして用いる方法については次節で詳しく述べる。

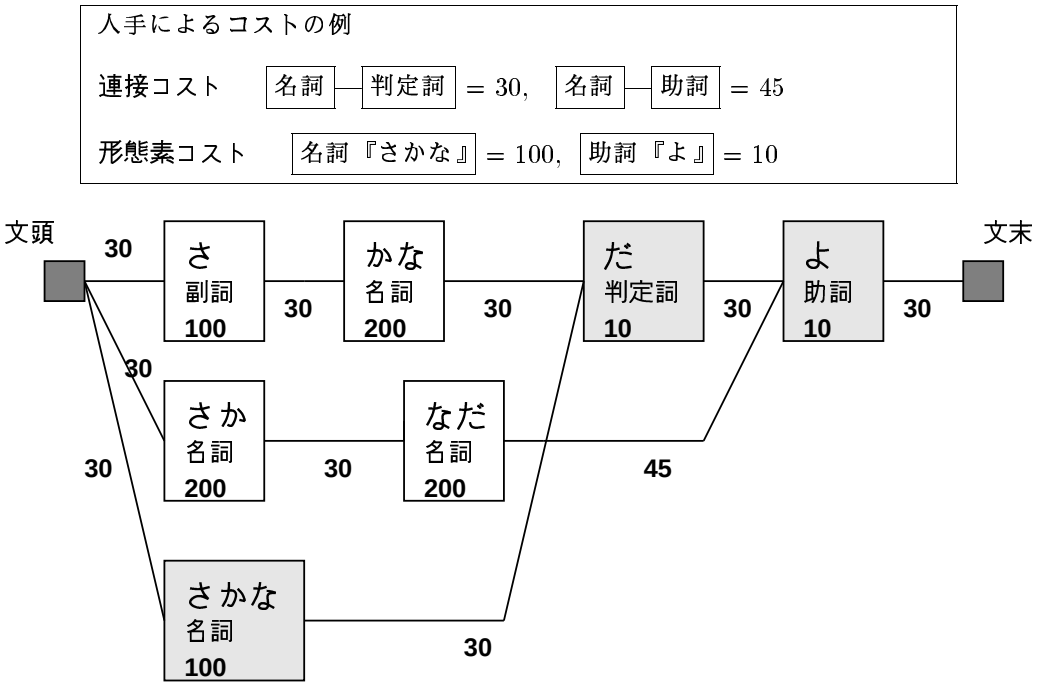


図 7: 最適解の選択

2.3.3 確率モデル

文が単語列 $W = w_1 \cdots w_n$、品詞列 $T = t_1 \cdots t_n$から構成されているものとすると、形態素解析は、単語列と品詞列の同時確率 $P(W,T)$ を最大化する品詞列$\hat{T}$を求める問題に帰着される [北, 中村, 永田 96]。

$$\hat{T} = \arg\max_{T} P(W,T)$$

$\arg\max_{x} f(x)$ は、$f(x)$ を最大にする x を返す関数である。

$P(W,T)$ を計算するための確率モデルを品詞付けモデルと呼ぶ。どのような品詞付けモデルを使用するかが問題となるが、一般に式 (1) に示すような品詞 N-gram モデルが用いられる。

$$P(W,T) = \prod_{i=1}^{n} p(w_i \mid t_1, ..., t_i) p(t_i \mid t_1, ..., t_{i-1}) \tag{1}$$

N = 2 のときを品詞 bigram モデル、N = 3 のときを品詞 trigram モデルと呼び、どちらもよく用いられる。$P(W,T)$ を品詞 bigram モデルで近似すると式 (2) のようになる。

$$P(W,T) = \prod_{i=1}^{n} p(w_i \mid t_i) p(t_i \mid t_{i-1}) \tag{2}$$

単語列を観測可能なシンボルの系列、品詞列を観測不能な状態系列と考えれば、$P(W,T)$ は隠れマルコフモデルにより定式化できる。品詞二つ組確率 $p(t_i \mid t_{i-1})$ と品詞別単語出現確率 $p(w_i \mid t_i)$ は、それぞれ、状態遷移確率とシンボル出力確率に相当する。

品詞タグ付きコーパスがあれば、品詞二つ組や単語の出現頻度を調べることにより、以下に示す式を用いて、これらの確率値を推定することができる。$C(x)$ は、x の出現頻度を表す。

コーパスから推定した確率値の例

連接確率　p(判定詞 | 名詞) $= 0.3$，　p(助詞 | 名詞) $= 0.02$

形態素出現確率　p(さかな | 名詞) $= 0.05$，　p(だ | 判定詞) $= 1$

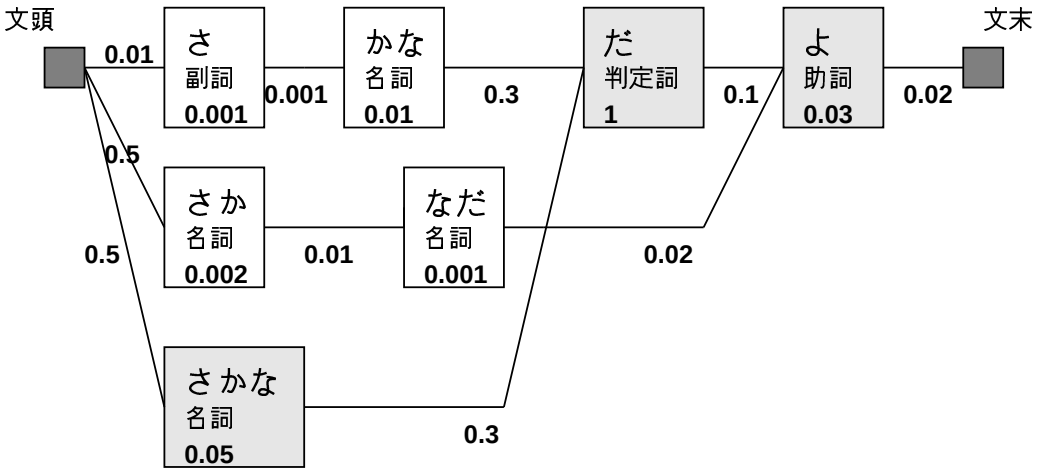


図 8: 最適解の選択 (確率モデル)

$$
\begin{aligned}
p(w_i \mid t_i) &= \frac{C(w_i, t_i)}{C(t_i)} \\
p(t_i \mid t_{i-1}) &= \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}
\end{aligned}
$$

例えば、ある品詞タグ付きコーパス中で、名詞が 100 回現れたとすると C(名詞) $= 100$ となる。そして、名詞の直後に動詞が 30 回現れたとすると C(名詞,動詞) $= 30$ となり、

$$
p(\text{動詞} \mid \text{名詞}) = \frac{C(\text{名詞}, \text{動詞})}{C(\text{名詞})} = \frac{30}{100} = 0.3
$$

と推定できる。また、名詞 100 回中、「さかな」が 5 回現れたとすると C(さかな、名詞) $= 5$ となり、

$$
p(\text{さかな} \mid \text{名詞}) = \frac{C(\text{さかな}, \text{名詞})}{C(\text{名詞})} = \frac{5}{100} = 0.05
$$

と推定できる。図 8に確率モデルに基づく形態素解析結果の例を挙げておく。数値は確率値を表す。

品詞タグ付きコーパスが無い場合でも普通のテキストから Baum-Welch アルゴリズム (Forward-Backward アルゴリズム)[Baum 72] を用いて確率値を推定することができる。このような推定方法は、品詞タグ付きコーパスを用いる方法と比べ、高い解析精度は得られないが、大規模な品詞タグ付きコーパスが不要な点が嬉しい。

このようにして全ての確率値が求まれば、入力単語列に対して式 (2) の値を最大にする品詞列は、動的計画法の一種である Viterbi アルゴリズムによって求めることができる。Viterbi アルゴリズムとは、ラティス状のグラフの各ノードに、始めのノードからそのノードまでの最大確率を保存しておくという処理を順

次行い、最後のノードにたどり着いたときに、最終的に最大確率を得ることのできるパスが判るというものである。形態素解析処理における Viterbi アルゴリズムの利用については [Allen 95] に詳しい。

式 (2) の対数 (log) を取った関数 G は、コスト最小法のコスト計算に用いる目的関数とみなせる。

$$\begin{aligned} G(W,T) &= -\log \prod_{i=1}^{n} p(w_i \mid t_i) p(t_i \mid t_{i-1}) \\ &= \sum_{i=1}^{n} \Big(-\log p(w_i \mid t_i) \Big) + \sum_{i=1}^{n} \Big(-\log p(t_i \mid t_{i-1}) \Big) \end{aligned}$$

$-\log p(w_i \mid t_i)$ が形態素コスト、$-\log p(t_i \mid t_{i-1})$ が連接コストに対応する。コスト合計最小のパスを求めることは、確率積和最大のパスを求めることに等しい。

2.4 その他

2.4.1 活用語尾の扱い

動詞、助動詞、形容詞などの活用する語の語尾の扱いには 2 種類ある。

語幹+活用語尾で一形態素とみなす 実装には、活用形を全て展開したものを辞書登録する方法と解析時に内部で語幹に活用語尾を付与して展開する方法がある。前者は処理系は単純になるが辞書項目が膨大になるという欠点がある。後者は JUMAN[松本ら 94]、茶筌 [松本ら 97] で採用されている。

語幹を活用語尾をそれぞれ一形態素とみなす 語幹と活用語尾の連接は絶対的なもので、活用語尾は他のいかなる形態素の後にも接続しない。活用語尾は一文字二文字のひらがなで構成されるものが多く、辞書から取り出される形態素が増えてしまい、解析速度に影響をあたえるという問題点がある。

久光ら [久光ら 94] の実験によると、活用語の活用形を全て展開して辞書登録する方が、語幹と活用語尾を分割して辞書登録するよりも解析速度が速い (後者を 1 とすると前者は 0.7)。ただし、品詞体系にかなり依存すると思われる。

また、JUMAN/茶筌のように内部で活用形を展開する方法は他の方法と比べかなり速いとの噂がある。格助詞などを名詞の活用にすることによって解析速度が向上するという噂もある。

2.4.2 未定義語の扱い

形態素辞書に存在しない文字列を未定義語とみなすのは自然な考え方である。しかし、日本語には一文字二文字からなる形態素も多いので、無理矢理細かく分割してしまう傾向にある。例えば、「チャイ/普通名詞」「スキー/サ変名詞」は登録されているが「チャイコフスキー/固有名詞」が登録されていない辞書での解析結果「チャイ/名詞」「コフ/未定義語」「スキー/名」など。このような場合、同一字種文字列を一つの単語とみなすといった字種情報に基づき未定義語を認識するのが現実的である。つまり、解析結果として「チャイコフスキー/未定義語」を出力する。

こういった未定義語にどういうコストを与えたら良いのかが問題となる。JUMAN[松本ら 94]、茶筌 [松本ら 97] ではユーザが全ての未定義語に対して同じコストを設定するようになっている。

3　解析精度の向上

3.1　解析精度の測定

当り前のことであるが、形態素解析の精度はどのように測定されたかが非常に重要である。品詞体系が異なればそれだけで精度や解析速度に差が出る。また、区切り精度、品詞まで見ての精度かも重要である。さらに、品詞の粒度にも注意しなければならない。第一解での評価か、複数の解の中に正解が含まれているかという評価というのも重大である。とにかく、数字だけが一人歩きしてしまってはいけないのである。

ここでは、一般的な形態素解析の評価尺度である再現率 (recall)・適合率 (precision) について解説する。評価には正解データとシステムの出力データが必要である。再現率・適合率は、正解データに含まれる形態素の数、システムの出力に含まれる形態素の数、両方に共通の形態素 (システムの出力中の正しい形態素) の数を数え、図 9の式で求める。

具体的な例 (図 10) を用いて説明する。文「前年度に比べ、八割増となった。」に対し、正解データに含まれる形態素の数は 10 個、システムの出力に含まれる形態素の数は 9 個である。両者の中で、形態素の持つ全ての情報が等しいものは 7 個である。従って、再現率と適合率は 7/10 と 7/9 である。これは、形態素の持つ全ての情報まで見ての評価であるが、区切り精度を求める場合は、「比べ (動詞/名詞)」は両者に共通な形態素とみなせ、共通形態素の数は 8 となり、再現率と適合率は 8/10 と 8/9 となる。

3.2　連語登録

山地ら [山地ら 96] は形態素解析システム JUMAN[松本ら 94] にて、連語 (複数の形態素から成るが一つの形態素であるかのように扱ってもいいもの) を登録し解析時に優先させる方法を提案している。

連語は、以下のように形態素の組みと連語コストで構成される。

こと/名詞 ― に/助詞 ― なった/動詞 : 0.5

ラティス構造中に登録されている連語があれば、それに関わる形態素の連接/単語コストを連語コスト倍する。これにより、最適解選択時に連語に関わる形態素が優先される。

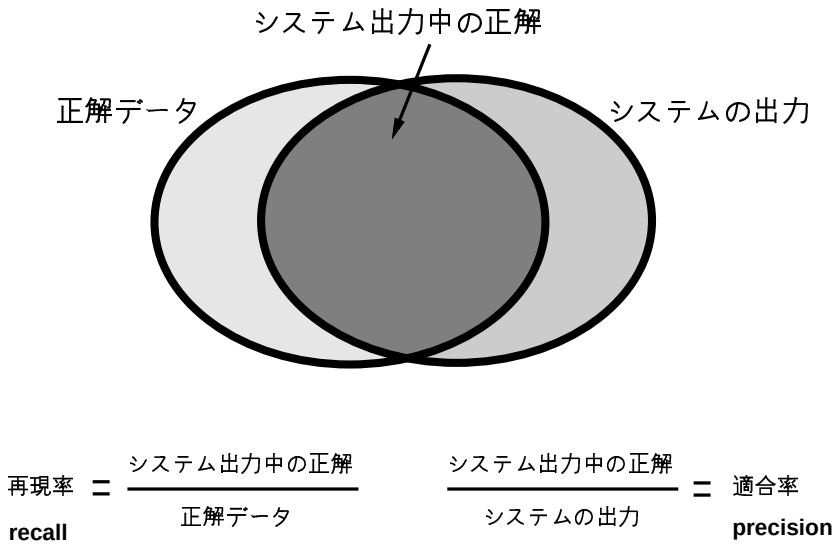


図 9: 再現率・適合率

正解データ	システムの出力
前年度 ぜんねんど 前年度 名詞 普通名詞 * *	前年度 ぜんねんど 前年度 名詞 普通名詞 * *
に に に 助詞 格助詞 * *	に に に 助詞 格助詞 * *
＋ 比べ くらべ 比べる 動詞 * 母音動詞 基本連用形	＋ 比べ くらべ 比べ 名詞 普通名詞 * *
、 、 、 特殊 読点 * *	、 、 、 特殊 読点 * *
八 はち 八 名詞 数詞 * *	八 はち 八 名詞 数詞 * *
＋ 割 わり 割 接尾辞 名詞性名詞助数辞 * *	＋ 割増 わりまし 割増 名詞 普通名詞 * *
＋ 増 ぞう 増 名詞 普通名詞 * *	と と と 助詞 格助詞 * *
と と と 助詞 格助詞 * *	なった なった なる 動詞 * 子音動詞ラ行 タ形
なった なった なる 動詞 * 子音動詞ラ行 タ形	。 。 。 特殊 句点 * *
。 。 。 特殊 句点 * *	

フォーマット：見出し語 読み 基本形 品詞 品詞細分類 活用型 活用形
“＋” は正解データとシステムの出力とで異なる部分を示している。

図 10: 形態素解析精度の測定

連語登録により解析誤りの 3 分の 1 近くが改善され、副作用は改善形態素 980 に対して 18 と非常に少ない。しかし、人手による連語登録作業を継続して行なう必要があるのと、連接/単語コスト体系を変更した場合、新たに登録すべき連語が大量に出現する可能性があるのが問題になるかもしれない。

3.3 可変長連接規則

北内ら [北内ら 97] は形態素解析システム「茶筌」[松本ら 97] に可変長連接規則を実装した。

可変長連接規則は bi-gram, tri-gram といった固定長連接のモデルに比べ柔軟な記述が可能である。言語現象の複雑さに合わせて連接数を変更でき、「特異な振舞には連接数を増やしたい」「一般的な振舞には bi-gram で十分」といった要求に対応できる。

例えば、特異な振舞として、

助動詞 * ダ列タ系連用テ形
判定詞 * ダ列タ系連用テ形
形容詞 ナ形容詞 ダ列タ系連用テ形
形容詞 イ形容詞アウオ段 基本連用形
形容詞 ナ形容詞 ダ列タ系連用テ形
形容詞 ナノ形容詞 ダ列タ系連用テ形

の次に

助詞 副助詞 * * さえ
助詞 副助詞 * * など
助詞 副助詞 * * は
助詞 副助詞 * * も

が続いて、その後に

接尾辞 形容詞性述語接尾辞 イ形容詞アウオ段 * ない

が来るときのコストが 10 と定義すると、図 11のように解析される。丸で囲んだ部分は「で/判定詞-ダ列タ系連用テ形」の次に「は/副助詞」が来た (オートマトンでいうところの) 状態を表している。

連語登録 [山地ら 96] と異なる点について述べる。連語登録では辞書登録の延長上にあるので、語彙表層 (見出し語) まで記述する必要があるが、可変長連接規則は連接処理なので任意のレベルで記述できる。また、可変長連接規則を決定性オートマトン (DFA) へ変換することにより、処理速度に影響を与えずに、確率モデルの枠組で利用できる。つまり、品詞タグ付きコーパスからの統計的学習により、今後の精度向上が期待できるのである。その方法として、春野ら [春野ら 96] は品詞タグ付きコーパスに対し、文脈木を用いた個別の連接規則における最適な連接数の決定法を提案している。

しかし、品詞タグ付きコーパスが大量に蓄積されるまでは、結局は連語登録のように人手により規則を

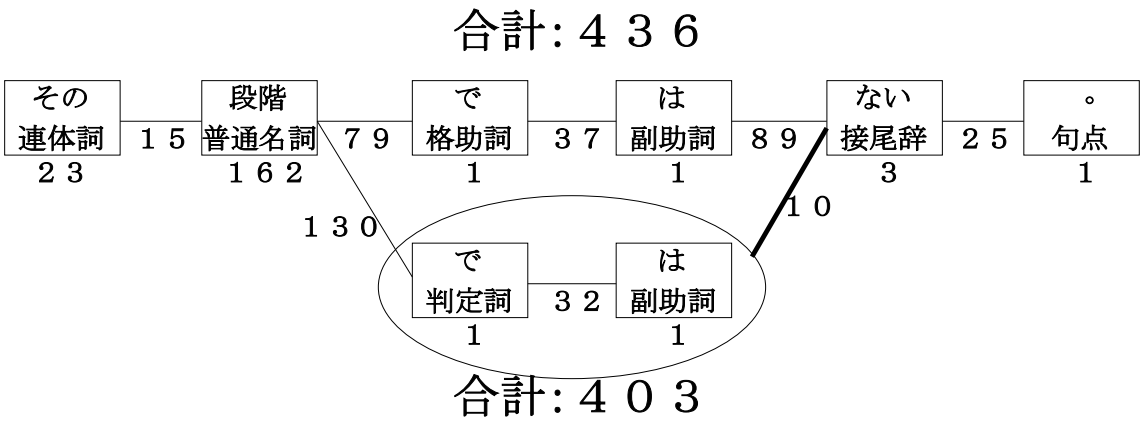


図 11: 可変長連接規則を用いた解析例

書いていかなくてはならない。

3.4 人手によるコスト体系と確率モデルの統合

山下 [山下 97][山下ら 97-1] は、未開拓な分野で高い解析精度を得るために、少量の品詞タグ付きコーパスから学習された確率パラメータと有用な言語資源である人手により作成された制約や優先規則の統合を行なった。

つまり、人手によるコスト体系とコーパスから学習された確率パラメータをある比率で混ぜ合わせ新たな値をつくり出し、それを最適解の選択に用いる。

そのために両者を確率として扱うことにする。コストを適当にスケール変換したのち指数関数を用いて確率値 (らしきもの) に変換する。それに対応するコーパスから推定した確率値を適当な割合で足し合わせ、それをコストに変換する。図 12に例を示す。例では、名詞と助詞の人手による連接コスト 45 を確率値 (らしきもの) p(助詞 | 名詞) = 0.04 に変換し、これに対応するコーパスから推定した確率値 p(助詞 | 名詞) = 0.02 と 1:3 の割合で足し合わせコストに変換している。

実験により統合手法が優位であることが示された。特に学習に用いる品詞タグ付きコーパスが小規模の場合に優れている。しかし、人手により整えられたコスト体系がない品詞体系では有用とは言えない。

3.5 誤り駆動学習 — Error-Driven Learning —

Brill[Brill 95] は変形規則と呼ばれる操作によって形態素解析を行なうシステムで、解析誤りを最も減少させる規則を追加していく操作を繰り返し、解析精度を向上させている。最終的には規則の最適な適用順序が得られる。解析誤りの減少は、解析結果と品詞タグ付きコーパス (Penn Treebank) の比較により調べる。

具体的な手順：

1. まず入力テキストに対してとりあえず形態素解析を行なう。
2. その解析結果に「直前の品詞が A なら現在位置の品詞 B を A に変えよ」といった変形規則を適用してみる。こういった変形規則は複数あり、適用結果も複数となる。
3. その中で解析誤りの一番少ない適用結果に適用した変形規則を採用する。
4. また様々な変形規則を適用してみる。 → 3

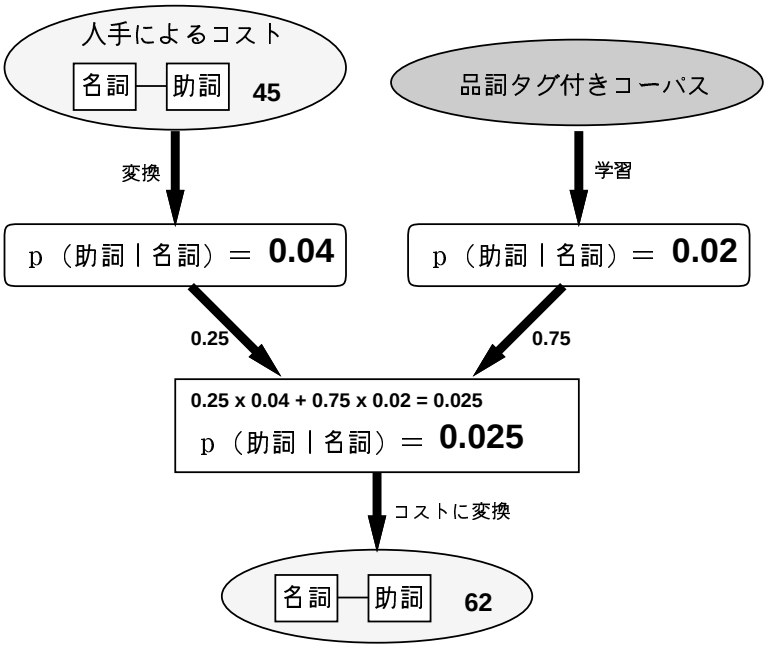


図 12: 統合処理の流れ

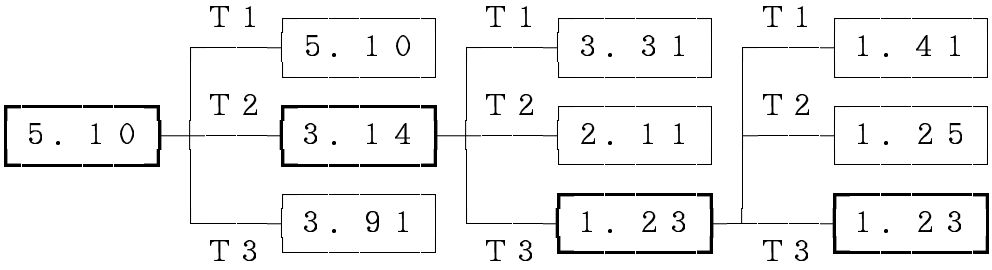
図 13に例を示す。

このような誤り駆動による精度向上は変形規則だけでなく、コスト体系、属性、使用するコーパスなどの調整に応用できる。精度の向上に非常に有用な手法であると言える。

3.6 属性/クラスの決定

状況によって品詞のどのレベルまで考慮するかを変更したい場合がある。例えば、「前が格助詞だったら、動詞は活用形まで考慮して連接確認したい」など。または、新たな分類グループ (クラス) を定義したい場合がある。

人手によって記述していくのは大変であり、品詞タグ付きコーパスを用いる方法が現実的である。



Ｔ１、Ｔ２、Ｔ３は追加する変形規則を、ボックス内の数値は解析誤りを表す。解析誤りが最も少なくなるように変形規則を選んでいく。この例の場合、最適な適用順序はＴ２→Ｔ３となる。Ｔ１は適用すると却って誤りが増すので不要。

図 13: Transformation-Based Error-Driven Learning

北内ら [北内ら 98] は誤り駆動学習により、状況に応じた適切な品詞レベルの選択を行なっている。
藤本ら [藤本ら 98] は品詞枝分かれ構造というものを用いて最適な品詞レベルの選択を行なっている。
森ら [森ら 98] はクロスエントロピーを用いて形態素のクラスタリングを行なっている。

参考文献

[人工知能ハンドブック] 人工知能学会編. “人工知能ハンドブック”, p.226-227, オーム社, 1990

[Allen 95] James Allen. “Natural Language Understanding (Second Edition)”, Benjamin/Cummings Publishing, 1995

[Baum 72] L. E. Baum. “An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Function of a Markov Process”, Ineqealities, Vol.3, pp.1-8, 1972

[北, 中村, 永田 96] 北研二, 中村哲, 永田昌明. “音声言語処理 —コーパスに基づくアプローチ—”, pp.91-98, 森北出版, 1996

[小松ら 95] 小松英二, 安原宏. “コスト最小法形態素解析のコストルール作成実験”, 情報処理学会研究報告, 95-NL-105, pp.1-6, Janualy 1995

[松本ら 94] 松本裕治, 黒橋禎夫, 宇津呂武仁, 妙木裕, 長尾真. “日本語形態素解析システム JUMAN 使用説明書 version 2.0”, NAIST Technical Report, NAIST-IS-TR94025, July 1994

[松本ら 97] 松本裕治, 北内啓, 山下達雄, 今一修, 今村友明. “日本語形態素解析システム『茶筌』 version1.0 使用説明書”, NAIST Technical Report, NAIST-IS-TR97007, February 1997

[Kurohashi et al 94] Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, Makoto Nagao. “Improvements of Japanese Morphological Analyzer JUMAN”, SNLR: Proceedings of the International Workshop on Sharable Natural Language Resources, pp.22-28, Aug 1994.

[Sedgewick 92] Robert Sedgewick(著), 野下浩平・星守・佐藤創・田口東 (訳). “アルゴリズム (Algorithms) 原書第 2 版 第 2 巻 探索・文字列・計算幾何”，近代科学社，1992.

[情報科学辞典] 長尾真, 石田晴久, 稲垣康善, 田中英彦, 辻井潤一, 所真理雄, 中田育夫, 米沢明憲 編. “岩波情報科学辞典”, 岩波書店, 1990.

[NAIST CL-Lab. 98] NAIST CL-Lab. “SUFARY Version 2.0”, http://cl.aist-nara.ac.jp/lab/nlt/ss/, June 1998.

[山下ら 96] 山下達雄, 松本裕治. “形態素解析結果の視覚化システム ViJUMAN とその学習機能”, 情報処理学会研究会報告, 96-NL-110, pp.71-78, September 1996.

[山下 96] 山下達雄. “パトリシア木を用いた形態素解析のための辞書検索 第 1 版”, ChaSen Technical Report, 茶筌パッケージ付属, October 1996.

[山下 97] 山下達雄. “規則と確率モデルの統合による形態素解析”, 奈良先端科学技術大学院大学修士論文, NAIST-IS-MT9551119, March 1997.

[山下ら 97-1] 山下達雄, 松本裕治. “コスト最小法と確率モデルの統合による形態素解析”, 情報処理学会研究会報告, 97-NL-119, pp.85-90, May 1997.

[山下ら 97-2] 山下達雄, 松本裕治. “Suffix Array を用いたフルテキスト類似用例検索”, 情報処理学会研究会報告, 97-NL-121, pp.83-90, September 1997.

[山下 98] 山下達雄 編. “SUFARY ガイド”, SUFARY Version 2.0 パッケージ付属, June 1998.

[山下ら 98] 山下達雄, 松本裕治. “言語に依存しない形態素解析ツールキットの開発”, 情報処理学会研究会報告, 98-NL-128, pp.?-?, November 1998.

[颯々野ら 96] 颯々野学, 難波功. “利用者による調整が可能な高速日本語形態素解析”, 情報処理学会第 52 会全国大会, 第 2 巻, 1996.

[青江順一 88] 青江順一. “ダブル配列による高速ディジタル検索アルゴリズム”, 電子情報通信学会論文誌, Vol.J71-D, No.9, pp.1592-1600, 1988.

[Maruyama 94] Hiroshi Maruyama. “Backtracking-Free Dictionary Access Method for Japanese Morphological Analysis”, COLING: Proceedings of the 15th International Conference on Computational Linguistics, pp.208-213, 1994

[Aho 90] Alfred V. Aho. “Algorithms for Finding Patterns in String”, In Handbook of Theoretical Computer Science, Vol.A: Algorothms and Complexity, pp.208-213, 1994.

[森 96] 森信介. “DFA による形態素解析の高速化”, 情報処理学会研究会報告, 96-NL-114, pp.101-107, July 1996.

[山地ら 96] 山地治, 黒橋禎夫, 長尾眞. “連語登録による形態素解析システム JUMAN の精度向上”, 言語処理学会第 2 回年次大会発表論文集, pp.73-76, March 1996

[春野ら 96] 春野雅彦, 松本裕治. “文脈木を利用した形態素解析”, 情報処理学会研究会報告, 96-NL-112, pp.31-36, March 1996.

[北内ら 97] 北内啓, 山下達雄, 松本裕治. “日本語形態素解析システムへの可変長連接規則の実装”, 言語処理学会第 3 回年次大会発表論文集, pp.437-440, March 1997.

[Brill 95] Eric Brill. “Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging”, Computational Linguistic, Vol.21, No.4, pp.543-565, 1995.

[北内ら 98] 北内啓, 宇津呂武仁, 松本裕治. “誤り駆動型の確率モデル学習による日本語形態素解析”, 情報処理学会研究会報告, 98-NL-124, pp.41-48, March 1998.

[藤本ら 98] 藤本浩司, 乾伸雄, 小谷善行. “枝分かれ構造を持つ同時確率モデルによる形態素解析”, 情報処理学会研究会報告, 98-NL-123, pp.1-8, January 1998.

[森ら 98] 森信介, 長尾眞. “形態素クラスタリングによる形態素解析精度の向上”, 自然言語処理, Vol.5, No.2, April 1998.

[久光ら 94] 久光徹, 新田義彦. “日本語形態素解析における効率的な動詞活用処理”, 情報処理学会研究会報告, 94-NL-103, pp.1-7, September 1994.

[長尾 96] 長尾真編. “自然言語処理”, pp.117-137, 岩波書店, 1996

★形態素解析は主に JUMAN の話...。

[言語の科学 3] 松本裕治, 影山太郎, 永田昌明, 齋藤洋典, 徳永建伸. “単語と辞書”, 岩波講座 言語の科学 3, 岩波書店, 1997.

★永田さんによる第 2 章「形態素解析」は要チェック!