# 日本語形態素解析システムへの可変長連接規則の実装

北内 啓　山下 達雄　松本 裕治
奈良先端科学技術大学院大学 情報科学研究科
{akira-k,tatuo-y,matsu}@is.aist-nara.ac.jp

## 1 はじめに

本研究では、可変長連接規則を記述できるよう日本語形態素システム「茶筌」[6] の機能を拡張した。これにより、言語現象の複雑さにあわせて連接規則で扱う品詞の連接数を変更することができるようになり、固定長連接規則に比べて柔軟な記述が可能になった。例えば、bi-gram モデルにおける連接規則の記述力不足、tri-gram 以上の固定長モデルにおけるデータスパースといった問題を克服することができる。また、可変長連接規則の実装における問題点とその実装方法を述べ、tri-gram 以上の連接規則が有効であることを簡単な実験により示す。

## 2 固定長連接規則の問題点

現在公開されている多くの日本語形態素解析システムは、主として人手によって作成された制約や優先規則に基づいている。従来、最長一致法、文節数最小法、コスト最小法などさまざまな方法が提案されてきた。特に、品詞の並びに関する制約を扱った研究においには、隣接する2つの品詞間の関係のみに着目しているものが多い。

また、近年電子化された日本語コーパスが大量に整備されつつあり、確率モデルに基づく手法も盛んに研究されてきている。その際、bi-gram や tri-gram といった固定長の品詞並びを扱うモデルが主流である [1]。

隣接する2つの形態素間の連接規則は記述が比較的容易で、人手で記述することも十分可能であり、小規模のコーパスによる統計的学習でも、ある程度の解析精度が得られる。しかし、「隣接する2つ」という制限のため、原理的に複雑な言語現象を捉えることができない。これに対し、tri-gram 以上のモデルでは、複雑な言語現象を捉えることができるが、連接規則を人手で記述するのはきわめて困難で、ほとんどの場合統計的学習により推定したものを用いる。しかし、データスパースの問題が発生しやすく、学習には大量のコーパスが必要となってしまう。

## 3 可変長連接規則

前節で述べたように、bi-gram や tri-gram のような固定長の連接規則を用いた確率モデルにはそれぞれ問題がある。それらを解決するために、言語現象の複雑さにあわせて連接数を変更する必要があり、そのようなモデルとして variable-gram モデルがある [2]。これにより、2つあるいは3つといったあらかじめ決められた長さだけでなく、任意の長さの連接規則を利用することができる。

例えば、人手で連接規則を記述する場合、まず隣接する2つの形態素間の規則 (bi-gram) を記述し、その次に bi-gram では記述できないような3つ以上の形態素間の連接規則を記述するといった手法を取ることができる。

また、コーパスからの学習に関して、春野ら [5] は文脈木を用いて個別の連接規則における最適な連接数の学習を行う方法を提案している。この方法により多様な言語現象の獲得が可能になるだけでなく、解析精度を下げることなくパラメータ数をコンパクトに保ち、データスパースの問題をも解決することができる。

## 4 可変長連接規則の実装

本研究では、コスト最小法 (bi-gram モデル) に基づく日本語形態素解析システム「茶筌」version

| 規則 | 連接する形態素列 | 連接コスト |
|---|---|---|
| $r_1$ | $(a, b)$ | 10 |
| $r_2$ | $(a, b, c)$ | 20 |
| $r_3$ | $(b, c, d, e)$ | 30 |
| $r_4$ | $(d, e)$ | 40 |
| $r_5$ | $(e, d)$ | 50 |

図 1: 可変長連接規則の例

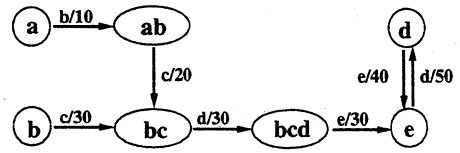


図 2: 可変長連接規則を変換して得られた DFA

1.0 [1]をもとに、variable-gram モデルに基づくシステムを開発した。

## 4.1 連接規則から DFA への変換

可変長連接規則を「茶筌」に実装するために連接規則を決定性有限オートマトン (DFA) に変換する。これにより、bi-gram モデルの時と同じように Viterbi アルゴリズムを用いて、処理速度に影響を与えることなく解析を行うことができる。この節ではその変換アルゴリズムについて説明する。

はじめに連接規則の記法について述べておく。連接規則は形態素の列であり、ここでは $(a_1, ..., a_n)$ のように記す。$a_i$は一つの形態素 (品詞や語彙など) を表している。ただし、「茶筌」では可変長連接規則を形態素の集合の列 $(A_1, ..., A_n)$ を記述する (次節参照)。

まず連接規則の補完を行う。ある連接規則は、その prefix も規則として含んでいると考え、連接規則の prefix をそれぞれ 1 つの連接規則として追加する。ただし、prefix の表す規則が連接規則としてすでに存在しているときは規則を追加しない。追加した連接規則の連接コストはもとの連接規則の連接コストと同じにする。規則を補完せずにそのままオートマトンに変換しようとすると、連接規則に対応する遷移先の状態が見つからず、オートマトンを作ることができない場合があるため規則の補完が必要となる。

例えば $(a, b, c, d)$ という連接規則に対しては、その prefix である $(a, b)$, $(a, b, c)$ という連接規則を含んでいると考え、もし $(a, b)$ や $(a, b, c)$ 自身が連接規則として存在していなければそれぞれを連接規則の集合に追加する。

次に、この補完された連接規則の集合 $R$に対して、オートマトンの状態集合とその状態遷移を求める。連接規則の形態素列を $r_i = (a_1, ..., a_n)$, $r_i$から最後の形態素を取り除いた形態素列を $prefix(r_i) = (a_1, ..., a_{n-1})$, $r_i$の最後の要素を $last(r_i) = a_n$, 連接コストを $c_i = C(a_n|a_1, ..., a_{n-1})$ とする。オートマトンの状態集合 $S$と入力記号の集合$\Sigma$は次のように定義される。[2]

$$S = \{prefix(r_i) | r_i \in R\}$$

$$\Sigma = \{last(r_i) | r_i \in R\}$$

規則の長さが短い順に遷移を作成する。各連接規則 $r_i$について、状態遷移の現状態 $s_i$、入力記号$\sigma_i$、次状態 $s'_i$、出力 $o_i$を以下のようにして求める。

**現状態** $s_i$ その suffix が $prefix(r_i)$ と等しいような状態。$prefix(r_i) \in S$であるから、$s_i$は少なくとも 1 つ存在する。2 つ以上存在する場合は、その数だけ遷移を作る。

**入力記号**$\sigma_i$ 連接規則の最後の要素 $last(r_i)$

**次状態** $s'_i$ $s_i\sigma_i$の suffix と等しい状態の中で最も長いもの。$s_i\sigma_i$の suffix と等しい状態が存在しない場合はそのような遷移先はないということであり、状態遷移を追加しない。

**出力** $o_i$ 連接コスト $c_i$

図 1で示される連接規則を例にとると、DFA の状態集合は $S = \{a, ab, b, bc, bcd, d, e\}$ であり、連接規則を DFA に変換すると図 2のようになる。

また、$r_2 = (b, c, d, e)$ という連接規則の prefix として 2 つの規則 $(b, c)$,$(b, c, d)$ が追加され、$b$,$c$ とい

[1] 「茶筌」version 1.0 は JUMAN2.0[7] の上位互換で、機能的には大きな違いはない。可変長連接規則を実装したシステムは「茶筌」version 2.0 として公開予定である。

[2] 詳細は省略するが、実際の連接規則には、初期状態に相当する「文頭」という特殊な形態素から始まる連接規則と、終了状態に相当する「文末」という特殊な形態素で終わる連接規則が少なくとも 1 つ含まれており、これをオートマトンに変換すれば数学的な定義を満たすオートマトンを得ることができる。

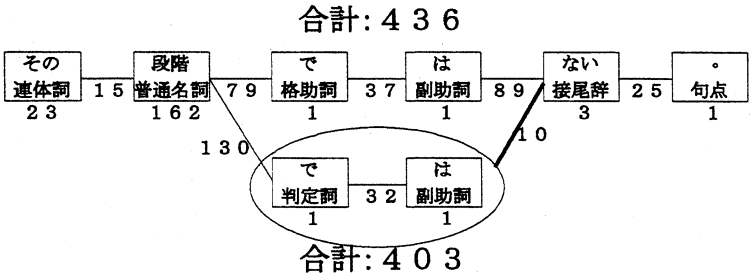


図 4: 可変長連接規則を用いた解析例

```
(((((形容詞) ナ形容詞 語幹)
   ((形容詞) ナノ形容詞 語幹))
  (((助詞 格助詞) * * に))
  (((動詞) 子音動詞ラ行 * なる)))
 5)
```

図 3: 可変長連接規則の記述例

う状態や $b \xrightarrow{c/30} bc$, $bc \xrightarrow{d/30} bcd$ という状態遷移が作成されていることが分かる。実際には、$(b,c)$ は連接しにくいが、$(b,c,d)$ は連接しやすいということもある。その場合は $(b,c)$ の連接規則を別に記述し、それに高い連接コストを与えておけばよい。

## 4.2 連接規則の記述

可変長連接規則には、連接する形態素群の列とその連接コストを指定する。実際には図3のようにS式を記述する。S式は連接する形態素群の列とその連接コストからなる2つの要素のリストによって記述される。形態素群の列は形態素群のリストからなり、さらに形態素群は形態素のリストから構成される。形態素には、品詞分類、活用型、活用形、語彙を自由に指定することができる。連接コストはその値が小さいほど連接しにくいことを表している。

図3の連接規則は、ナ形容詞の語幹あるいはナノ形容詞の語幹、格助詞「に」が出現しているという状態のもとで、動詞「なる」がコスト5で連接することを表している。

## 4.3 解析例

可変長連接規則によって形態素解析を行った例を図4に示す。bi-gram の連接規則の他に、図5のtri-gram の連接規則を追加して解析を行った。図4で太線で示したものがその連接規則によるもので、判定詞「で」—副助詞「は」の後ろに、接尾辞「ない」がコスト 10 で連接することを表している。この連接規則により、上段のパスのコストよりも下段のパスのコストの方が小さくなり、下段のパスが最適パスとして選ばれることが分かる。

もし図5の tri-gram の連接規則なしで解析を行うと、太線の部分の連接コストは上段と同じ 89 になるので、下段のコストの方が大きくなり、上段の間違ったパスが選択されてしまう。

```
(((((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))
  (((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))
  (((接尾辞 形容詞性述語接尾辞) イ形容詞アウオ段 * ない)))
 10)
```

図 5: tri-gram の連接規則

## 5 実験

可変長連接規則を用いた日本語形態素解析システムの性能を評価するために、品詞タグつきコーパスを用いた簡単な実験を行った。日経新聞の新聞記事から取り出した1000文に人手でタグ付けしたコーパスを評価用の実験データとして用いた。1000文のうち、トレーニングデータとして900文を用いて学習させ、残りの100文をテストデータとして解析を行い、解析精度を測定した。ten-fold cross validationによって学習と評価を10回行い、再現率、適合率の平均を求めた。形態素辞書については、学習によって得られた単語の他に「茶筌」付属の辞書に登録されている各形態素のコストを最大の値にしたものを用いた。

実験ではまず、bi-gramの連接規則を最尤推定法により統計的に学習させ、その結果を用いて解析を行い、誤りについて調べた。次に、bi-gramの連接規則で対処できる誤りに関しては人手で連接規則を書いた。その後、bi-gramの規則で対処できない誤りについてtri-gramの連接規則を試験的に図3、図5の規則を含め計3個だけ人手で記述し、bi-gramと同様の実験を行った。

その結果、tri-gramの連接規則で記述した単語連接についてはすべて正解となった。また、連接規則を追加したことによる副作用も全く起きなかった。

## 6 おわりに

本論文では、日本語形態素解析システムに可変長連接規則を実装することにより、従来の固定長連接規則での問題点を解消した。新聞記事1000文の品詞タグ付きコーパスを用いた実験を行い、tri-gramの連接規則の効果を確認した。

今後はさらにより多くの可変長連接規則を獲得し、解析精度の向上を目指すつもりである。獲得の手法としては以下に挙げる二通りの手法を考えている。

ひとつは実験で行なったような人手による獲得手法で、今後、形態素解析用のGUIであるVi JUMAN[3]の学習機能を用いて、品詞タグ付きコーパス作成作業の際に作業者による解析誤りの修正データから可変長連接規則を半自動的に抽出するという方法を検討している。

もうひとつは統計的な手法による獲得である。これには、前述した文脈木を用いて最適な連接数の規則を学習する手法などがあり、大規模な品詞タグ付きコーパスがあれば、最適な可変長規則の集合を自動的に求めることができる。

また、関連研究の一つとして、JUMAN3.0[4]では形態素解析の精度を向上させるために、連語登録による手法を用いている。この手法では連語登録の際単語の語彙表層まで記述する必要があるのに対し、本論文で提案している可変長連接規則では任意の品詞レベルを記述することができる。また、JUMAN3.0では連語内の形態素コストと連語コストを$\alpha$倍 $(0 < \alpha < 1)$ することによって連語が優先的に解析結果に現れるようにしているが、この手法は厳密な確率モデルに基づくものではない。これに対し、本研究の手法はvariable-gramの確率モデルに基づいているため、統計的学習に適しているという利点がある。

なお、「茶筌」に関する情報については以下のURLを参照されたい。

http://cactus.aist-nara.ac.jp/lab/nlt/chasen.html

## 参考文献


[1] Masaaki Nagata. A stochastic Japanese morphological analyzer using a forward-DP backward-A* N-best search algorithm. *Proceedings of the 15th COLING*, pp. 201–207, 1994.

[2] Dana Ron, Yoram Singer, and Naftali Tishby. Learning probabilistic automata with variable memory length. *Proceedings of the 7th Annual ACM Conference on Computational Learning Theory*, pp. 35–46, 1994.

[3] 山下達雄, 松本裕治. 形態素解析視覚化システム Vi JUMAN 使用説明書 version 1.0. NAIST Technical Report NAIST-IS-TR96005, Feb 1996.

[4] 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, Mar 1996.

[5] 春野雅彦, 松本裕治. 文脈木を利用した形態素解析. 情報処理学会研究報告(自然言語処理研究会) NL-112-5, pp. 31–36, 1996.

[6] 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』version 1.0 使用説明書. NAIST Technical Report NAIST-IS-TR97007, Feb 1997.

[7] 松本裕治, 黒橋禎夫, 宇津呂武仁, 妙木裕, 長尾眞. 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, 1994.