原著論文 Original Paper

形態素解析における高速な単語ラティス生成

Fast Word Lattice Generation in Morphological Analysis

鍛治 伸裕 Nobuhiro Kaji
東京大学 生産技術研究所
Institute of Industrial Science, University of Tokyo
kaji@tkl.iis.u-tokyo.ac.jp

喜連川 優 Masaru Kitsuregawa
国立情報学研究所，東京大学 生産技術研究所
National Institute of Informatics, Institute of Industrial Science, University of Tokyo
kitsure@tkl.iis.u-tokyo.ac.jp

keywords: Morphological analysis, Unknown words, Word lattice, Fast algorithm

Summary

This paper proposes a fast word lattice generation algorithm for Japanese morphological analysis. We conducted experiments on three Japanese data sets to demonstrate that the previously proposed pruning-based algorithm is in fact not efficient enough, and that the pipeline algorithm, which is introduced in this paper, achieves considerable speed-up without loss of accuracy. Moreover, the compactness of the lattice generated by the pipeline algorithm was investigated from both theoretical and empirical perspectives.

1. はじめに

現在，形態素解析処理を実現するための手法として，単語ラティス上の経路の探索または再順位付け [Jiang 08] に基づくものが広く用いられている [Kurohashi 98, Asahara 00, Kudo 04, Kruengkrai 06, Jiang 08, Hagiwara 13]．これらは，まず入力文に対して単語ラティスを生成し，次に単語ラティス上の最適な経路を選択するという，2 段階の処理を経て形態素解析を行う仕組みになっている．

この枠組みにおいて，未知語を考慮した形態素解析を行おうとした場合，単語ラティスを高速に生成することが技術的な課題となる．今，入力文の長さが n 文字であったとすると，そこには ${}_{n+1}C_2$ 個の単語候補が存在することになる*1．それら全ての候補に対して，単語ラティスに含めるか否かを判断する処理を行ったとすると，$O(n^2)$ の時間計算量が発生し，解析速度が大きく低下してしまう．そのため，何らかの方法によって，これを高速化することが実用上重要となる．

単語ラティスの生成においては，速度以外にも考慮するべき要因が存在しており，その高速化は決して単純な課題ではない．まず，高精度な形態素解析を実現するためには，正しい解析結果の漏れが少ない単語ラティスを生成する必要がある．これに加えて，単語ラティスが巨大になると，経路探索に必要な計算時間が深刻なものとなってしまうため，可能な限り小さな単語ラティスを生成することが望ましい．

*1 単語の最大長に制限を設けていない場合の単語候補の数である．

このため，どのような方法で単語ラティスの生成を行うのが効果的であるのかは，少なくとも自明に分かることではない．それにも関わらず，従来の研究において，単語ラティスの生成方法に関する技術的検討は十分に行われてこなかった．よく知られているアプローチとしては，辞書引きと字種に基づくヒューリスティクスの組み合わせによるものを挙げることができる [Kudo 04]．この方法は確かに高速であるものの，未知語に対して脆弱であるという問題が従来より指摘されている [Uchimoto 01, 岡野原 08]．Jiang らは，単語ラティス生成のための枝刈りアルゴリズムを提案しているが，計算時間に関する報告は行われていない [Jiang 08]．一方，最大単語長に制限を設けた上で，全ての候補を考慮した単語ラティスを生成するという方法も考えられる [Sarawagi 04]．この方法は確かに高速に単語ラティスを生成可能である．しかし，形態素解析では，片仮名語などをうまく扱うために，最大単語長を例えば 10 文字などの大きな値に設定する必要がある [持橋 11]．さらに，品詞タグの数は数百にオーダになることが一般的である [Kudo 04]．そのため，生成されるラティスが巨大化してしまい，ラティスの探索に多大な時間が発生してしまう．

こうした研究状況を踏まえ，本論文では，未知語を考慮した形態素解析において，効率的に単語ラティスを生成する方法を明らかにする．まず，Jiang らの枝刈りアルゴリズム [Jiang 08] は，実際のところ十分に効率的ではなく，形態素解析を行う際のボトルネックとなることを示す．次に，枝刈りアルゴリズムに代わる手法として，段

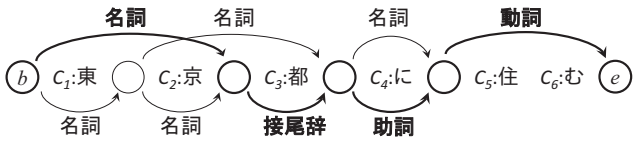


図 1 単語ラティスの例 [Kudo 04]．太字で強調されている経路が，この入力文に対する正しい形態素解析結果である．

階的な単語ラティス生成アルゴリズムを提案する．そして，このアルゴリズムが，解析精度を犠牲にすることなく，処理速度を 10 倍程度高速化できることを実験によって示す．さらに，段階的アルゴリズムによって生成される単語ラティスの大きさに関して理論的分析を行い，その妥当性を実験によって検証する．

本研究の主たる貢献は，(1) 単語ラティス生成が形態素解析の処理時間に大きな影響を与えるという問題を初めて明確に指摘し，(2) 段階的アルゴリズムが持つ優位性を実証的に明らかにしたことである．提案アルゴリズム自体は，既存技術 [Peng 04, Shi 07, Neubig 11] の素直な応用である．しかし，従来の研究において，単語ラティス生成における有用性という観点からの議論は見られない．また，既存手法である枝刈りアルゴリズム [Jiang 08] との優劣も明らかにされていない．

2. 予 備 知 識

本節ではいくつかの予備知識を導入する．まず，形態素解析結果の圧縮表現である単語ラティスについて述べ，その経路の再順位付けにもとづく形態素解析手法を説明する．そして，従来提案されている単語ラティス生成手法として，枝刈りに基づくアルゴリズムを紹介する [Jiang 08]．

2·1 単 語 ラ ティス

単語ラティス [Jiang 08] は，指数関数的な数の形態素解析結果の候補を効率的に保持するための圧縮データ構造であり，形式的には非循環有向グラフの一種と見ることができる．図 1 に，入力文「東京都に住む」に対する単語ラティスの例を示す．単語ラティスにおけるノードは単語境界を，エッジは単語と品詞の組を表現している (図 1 において，ノードとエッジはそれぞれ円と矢印で表されている)．なお，文頭と文末は常に単語境界に対応するため，特殊ノード b と e が必ず配置される．

単語ラティス上のノード b から e に至る経路は，入力文に対する形態素解析結果に対応している．そのため，形態素解析処理は，単語ラティス上の最適経路を探索する問題と考えることができる．このときの探索アルゴリズムとしては動的計画法が代表的である．

2·2 単語ラティスの再順位付け

Jiang らは，高速に形態素解析を行うための方法として，単語ラティスの再順位付け (word lattice reranking) という枠組みを提案している [Jiang 08]．これは，まず入力文に対する単語ラティスを生成し，次に単語ラティス上の最適な経路を選択するという，2 段階の処理を経て形態素解析を行うというものである．

このアプローチの利点は，単語ラティスによって解析候補が絞り込まれ，その結果として処理が高速化されることである．これは，いわゆる再順位付け [Collins 00] と同様の考え方であり，単語ラティスの再順位付けと呼ばれるのもこのためである．通常の再順位付けにおいて解析候補はリスト形式で表現されるが，ここでは圧縮構造である単語ラティスが用いられている．このように圧縮構造を利用して再順位付けを行うというテクニックは，構造予測問題において有効であることが知られている [Huang 08]．

単語ラティスの再順位付けにもとづいた形態素解析処理は以下のように定式化できる．

$$\hat{y} = \arg\max_{y \in L(x)} \textsc{Score}(x, y), \tag{1}$$

ここで，$L(x)$ は入力文 x に対する単語ラティス，$y \in L(x)$ は単語ラティス上のある経路，$\hat{y}$ は最適経路，$\textsc{Score}(x, y)$ は経路 y の再順位付けを行うスコア関数である．

本論文における議論の対象は，単語ラティス $L(x)$ の生成方法である．1 章でも述べたように，単純な方法で単語ラティスの生成を行うと，$O(n^2)$ の時間計算量が発生してしまうため，これをいかに高速化するかということが技術的な課題となる．このとき，厳密には，最適経路 $\hat{y}$ の探索に要する計算コストも考慮する必要があるが，この枠組みでは通常小さな単語ラティスが生成されるため，最適経路の探索に必要な計算コストは実際には無視することができる．

2·3 枝刈りに基づく単語ラティス生成アルゴリズム

これまでにも，未知語を考慮した形態素解析のための単語ラティス生成手法としては，Jiang らの提案したアルゴリズムが知られている [Jiang 08]．以下では，彼らのアルゴリズムの概要を説明する．詳細については文献 [Jiang 08] を参照されたい．

Jiang らのアルゴリズムは，入力文を左から右へと走査しながら，単語ラティス (より正確にはエッジ集合 E) を生成する (アルゴリズム 1)．今，先頭から i 番目の文字 c_i に注目している場合を考える．このときアルゴリズムは，文字 c_i が終端となるようなエッジ (すなわち，単語 $w = c_{i-l+1}c_{i-l+2}\ldots c_i$ と品詞タグ t の組) の候補集合 C を生成する (5-10 行目)．ただし，計算効率を考えて，単語の長さ l は高々 K 文字に制限する (5 行目)．これによって時間計算量を $O(Kn)$ に抑えることが可能となる．そして，得られた候補集合 C 中のスコア上位 k 件をエッ

ジ集合 E に追加する．この処理を各文字 c_i について順に行い，最終的にエッジ集合 E から成る単語ラティスを出力する．

このアルゴリズムは，単語長を最大 K 文字に制限することによって，$O(n^2)$ 個の候補の枝刈りを行いながら単語ラティスを生成している．このようなアプローチは，全ての単語候補を網羅的に考慮するような方法と比べると確かに効率的である．しかしながら，6 章の実験結果が示すように，このアルゴリズムには依然として大きな計算コストが必要となり，形態素解析処理の速度低下を招く．

枝刈りアルゴリズムに関するもう一つの問題点として，閾値 K の決定方法が自明ではないことを指摘できる．K の値を小さくすれば，探索範囲がより狭くなるため，単語ラティス生成の速度は向上する．しかし，その一方で，正しい単語候補が探索範囲から外れ，形態素解析の精度が低下する危険性が高まる．Jiang らはこのトレードオフについて考察を行っていないが，我々の実験では K の値が精度と速度に与える影響について調査する．

3. 関 連 研 究

単語ラティスを使って形態素解析を行うという方法自体は，日本語処理の研究において，これまでにも広く用いられている [Kurohashi 98, Asahara 00, Kudo 04]．これらの研究では，単語ラティスを辞書引きによって構築しているが，辞書に登録されていない未知語の扱いに関しては十分な議論が行われていない．例えば，Kudo ら [Kudo 04] は，字種と単語長に基づくヒューリスティクスを用いているが，扱いが不十分であることはこれまでにも指摘されている [岡野原 08]．一方，頑健な未知語処理を目指した研究も多く見られるが，後述する岡野原らの研究を除いては [岡野原 08]，解析速度に関する議論は見られない [Uchimoto 01, 東 06, Hagiwara 13]．

Nakagawa らや Kruengkrai らは，単語ラティスと文字単位のラベリングモデルを組み合わせたハイブリッドモデルを提案している [Nakagawa 07, Kruengkrai 09]．こうした方法と比較すると，単語ラティスの再順位付けは，未知語の解析に単語単位の素性を利用できることが利点であり，より豊富な素性を取り入れた解析を指向した枠組みであると言える．

形態素解析の高速化に関する先行研究としては，Zhang ら [Zhang 08, Zhang 10] や岡野原ら [岡野原 08] の研究がある．これらはいずれも，解探索アルゴリズムの高速化を図るものであり，単語ラティス生成を扱っている本研究とは相補的な試みである．そのため，上記の先行研究で提案されている探索アルゴリズムは，我々の提案する単語ラティス生成と組み合わせて使うことが可能である．今回，我々は最適経路の探索に動的計画法を用いたが (5 章参照)，例えば非局所素性を用いる場合などには，Zhang らや岡野原らの提案するような探索アルゴリズムが有用であると考えられる．

Algorithm 1 枝刈りに基づく単語ラティス生成アルゴリズム

```
1: T ← 全ての品詞タグ集合
2: E ← ∅   // 出力されるエッジ集合．初期値は空集合．
3: for i = 1...n do
4:     C ← ∅
5:     for l = 1...min(i, K) do
6:         w ← c_{i-l+1} c_{i-l+2} ... c_i
7:         for t ∈ T do
8:             C ← C ∪ (w, t)
9:         end for
10:    end for
11:    候補集合 C のスコア上位 k 件を E に追加
12: end for
13: return E
```

4. 段階的単語ラティス生成アルゴリズム

本節では，枝刈りアルゴリズムに代わる単語ラティス生成手法として，段階的アルゴリズムを提案する (アルゴリズム 2)．このアルゴリズムは，単語ラティスを構成する単語と品詞タグを独立に生成する．まず，入力文 x に対して，単語ラティスを構成する単語集合 W を生成する (2 行目)．例えば，図 1 の単語ラティスを構成する単語集合は $W = \{$ 東，京，東京，京都，都，に，住む $\}$ となる．ついで，各単語 $w \in W$ に対して適切な品詞集合を割り当てることによって (4 行目)，エッジ集合 E の生成を行う．

段階的アルゴリズムの利点は，単語集合 W を生成する際に，文字単位の単語分割モデル [Xue 03, Peng 04, Neubig 11] を利用することによって，$O(n^2)$ の時間計算量が発生するのを防ぐことができることである．この結果，入力文長[*2]に対して線形の時間で，単語ラティスを生成することが可能になる．それだけではなく，枝刈りアルゴリズムとは異なり，任意の長さの単語を生成することもできる．

以下では，単語生成と品詞タグ生成について，それぞれ 4·1 節と 4·2 節で説明する．そして，4·3 節では，段階的アルゴリズムの時間計算量を議論する．段階的アルゴリズムによって生成された単語ラティスの経路探索を行うために必要な空間計算量 (単語ラティスの大きさ) と時間計算量についても議論を行う．

4·1 単 語 生 成

入力文 x に対して単語集合 W を生成するために (アルゴリズム 2 の 2 行目)，文字単位の単語分割モデル [Peng 04] を用いる:

$$b = \arg\max_b \mathbf{\Lambda}_w \cdot \mathbf{F}_w(x, b)$$

*2 以下では，入力文長と言った場合，入力文の文字数を意味するものとする．

名称	素性テンプレート
文字 n-gram	$\langle c_{i-1}, b\rangle$, $\langle c_i, b\rangle$, $\langle c_{i+1}, b\rangle$, $\langle c_{i-2}, c_{i-1}, b\rangle$, $\langle c_{i-1}, c_i, b\rangle$, $\langle c_i, c_{i+1}, b\rangle$, $\langle c_{i+1}, c_{i+2}, b\rangle$, $\langle c_{i-3}, c_{i-2}, c_{i-1}, b\rangle$, $\langle c_{i-2}, c_{i-1}, c_i, b\rangle$, $\langle c_{i-1}, c_i, c_{i+1}, b\rangle$ $\langle c_i, c_{i+1}, c_{i+2}, b\rangle$, $\langle c_{i+1}, c_{i+2}, c_{i+3}, b\rangle$
文字種 n-gram	$\langle c'_{i-1}, b\rangle$, $\langle c'_i, b\rangle$, $\langle c'_{i+1}, b\rangle$, $\langle c'_{i-2}, c'_{i-1}, b\rangle$, $\langle c'_{i-1}, c'_i, b\rangle$, $\langle c'_i, c'_{i+1}, b\rangle$, $\langle c'_{i+1}, c'_{i+2}, b\rangle$, $\langle c'_{i-3}, c'_{i-2}, c'_{i-1}, b\rangle$, $\langle c'_{i-2}, c'_{i-1}, c'_i, b\rangle$, $\langle c'_{i-1}, c'_i, c'_{i+1}, b\rangle$, $\langle c'_i, c'_{i+1}, c'_{i+2}, b\rangle$, $\langle c'_{i+1}, c'_{i+2}, c'_{i+3}, b\rangle$
辞書情報	$\langle$BEGIN$, b\rangle$, $\langle$END$, b\rangle$, $\langle$INSIDE$, b\rangle$, $\langle$BEGIN$, s, b\rangle$, $\langle$END$, s, b\rangle$, $\langle$INSIDE$, s, b\rangle$

表 1 単語分割モデルのための素性テンプレート．b は出力タグ，c_i と c'_i はタグ付与対象の文字およびその文字種を表す．文字種には (1) アルファベット，(2) 漢字，(3) 平仮名，(4) 片仮名，(5) 数字，(6) その他，の 6 種類を用いる．c_{i-1} や c'_{i+1} は対象文字の周辺に出現する文字およびその文字種を表す．BEGIN(END) は，c_i から始まる (の直前で終わる) 文字列が辞書に登録されていることを示す．INSIDE は，c_i を内部に含む文字列が辞書に登録されていることを示す．s は，そのときに辞書に登録されてる文字列の文字数 (1，2，3，4 または 5 以上) である．

名称	素性テンプレート
単語	$\langle w, t\rangle$
単語長	$\langle$LENGTH$(w), t\rangle$
接辞文字列	$\langle c_i, t\rangle$, $\langle c_i, c_{i+1}, t\rangle$, $\langle c_{j-1}, t\rangle$, $\langle c_{j-2}, c_{j-1}, t\rangle$
周辺文字列	$\langle c_{i-1}, t\rangle$, $\langle c_{i-2}, c_{i-1}, t\rangle$, $\langle c_{i-3}, c_{i-2}, c_{i-1}, t\rangle$, $\langle c_j, t\rangle$, $\langle c_j, c_{j+1}, t\rangle$, $\langle c_j, c_{j+1}, c_{j+2}, t\rangle$
辞書情報	$\langle$DICT$(w, t)\rangle$, $\langle$DICT$(w, t), t\rangle$

表 2 品詞タグ付与モデルのための素性テンプレート．$w = c_i c_{i+1} \ldots c_{j-1}$ は品詞タグを付与する対象となる単語，t は品詞タグを表す．LENGTH(w) は単語 w の文字数を返す関数であり，その返り値は 1，2，3，4 または 5 以上のいずれかである．DICT(w, t) は，単語 w と品詞タグ t の組が辞書に登録されていることを表す．

Algorithm 2 段階的単語ラティス生成アルゴリズム

```
1: E ← ∅
2: W ← WordGenerator(x)
3: for w ∈ W do
4:     T ← PosTagGenerator(x, w)
5:     for t ∈ T do
6:         E ← E ∪ (w, t)
7:     end for
8: end for
9: return E
```

ここで $b = b_1 \ldots b_n$ は各文字に付与されるラベル系列であり，単語分割結果を表現している．b_i は B または I の 2 値をとり，それぞれ文字 c_i が単語の先頭またはそれ以外に対応していることを表す．$\mathbf{\Lambda}_w$ と $\mathbf{F}_w(x, b)$ は重みベクトルと素性ベクトルである．

重みベクトル $\mathbf{\Lambda}_w$ の学習には構造化パーセプトロンを用いる [Collins 02]．素性としては，表 1 にあるものとタグ 2-gram を用いる．表の先頭 2 行は，注目している文字 c_i の周辺に出現する文字 n-gram と文字種 n-gram である．最後の行は，Neubig らが提案した辞書素性である [Neubig 11]．この素性は，入力文に含まれる文字列が辞書に単語として登録されている場合に発火し，その文字列と注目している文字の相対的な位置関係，およびその文字列の長さを符号化している．実験において使用した辞書については 6 章で後述する．

この単語分割モデルを用いて，スコア上位 α 件の単語分割結果を取得し，それらの和集合を W とする:

$$W = \cup_{i=1\ldots\alpha} W_i$$

ここで W_i は，i 番目に大きなスコアを持つ単語分割結果に含まれる単語の集合である．α はパラメータであり，開発データを用いて決定する．

4·2 品詞タグ生成

各単語 w に対して品詞タグ集合 T を生成するために (アルゴリズム 2 の 4 行目)，多クラス線形モデルを構築し，そのスコアの上位 β 件を T とする．具体的には，入力文 x における単語 w が与えられたとき，各品詞タグ t に以下のようなスコアを与えるようなモデルを構築する [Neubig 11]:

$$\mathbf{\Lambda}_t \cdot \mathbf{F}_t(x, w, t)$$

$\mathbf{F}_t(x, w, t)$ は素性ベクトルである．ここでの素性としては，[Neubig 11] と同様に，単語の表層形，単語長 (文字数)，接辞文字列，周辺文字列を用いる (表 2)．また，単語 w と品詞タグ t の組が，外部辞書に登録されているかどうかを示す 2 値素性も合わせて用いる．$\mathbf{\Lambda}_t$ は重みベクトルであり，平均化パーセプトロンを用いて学習する．パラメータ β は，α と同様に開発データを用いて決定する．

4·3 計　算　量

段階的アルゴリズムは，枝刈りアルゴリズムとは異なり，任意の長さの単語を生成することが可能である．しかし，その一方で，枝刈りアルゴリズムと同じく，入力文長 n に対して線形時間で単語ラティスを生成することができる．

このことは次のように証明できる．まず，単語集合 W の生成には動的計画法を用いることができるため，これに必要な時間は $O(n)$ である．さらに，

$$|W| = |\cup_{i=1\ldots\alpha} W_i| \leq \sum_{i=1\ldots\alpha} |W_i| \leq \alpha\, n$$

により，アルゴリズム 2 の外側のループ (3 行目) を実行するために必要な時間も $O(\alpha n)$ である．さらに，$|T| = \beta$ であるため，内側のループ (5 行目) 処理に必要な時間は $O(\beta)$ である．これらのことより，段階的アルゴリズムに必要な計算時間は $O(\alpha\beta n)$ であることが分かる．

同様の議論によって，段階的アルゴリズムによって生成された単語ラティスの探索に必要な空間計算量 (単語ラティスに含まれるエッジ数) も $O(|E|) = O(\alpha\beta n)$ であることが確認できる．このことに加え，あるノードに入ってくるエッジの数は高々α であって n に依存しないことから，動的計画法を用いて単語ラティスの最適経路を探索する際に必要な時間計算量は $O(\alpha^2\beta n)$ であることが分かる．

5. 再順位付け

再順位付けの方法としては，Huang の提案したものを用いる [Huang 08]．すなわち，スコア関数 $\textsc{Score}(x,y)$ に線形モデルを用いて，以下のように最適経路 $\hat{y}$ を求める．

$$\begin{aligned}\hat{y} &= \arg\max_{y \in L(x)} \textsc{Score}(x,y) \\ &= \arg\max_{y \in L(x)} \mathbf{\Lambda} \cdot \mathbf{F}(x,y)\end{aligned}$$

$\mathbf{F}(x,y)$ と $\mathbf{\Lambda}$ は，それぞれ素性ベクトルと重みベクトルである．素性としては，まず表 1 の素性テンプレートに対して，出力タグを BIES の 4 種類 [Nakagawa 04] に拡張したものを用いる．これに加えて，表 2 と同一の素性，品詞タグ 2-gram を用いる．最適経路の探索には，動的計画法の一種であるビタビアルゴリズム [Kudo 04, 萩原 10] を用いる．

5·1 訓　　練

重みベクトル $\mathbf{\Lambda}$ の訓練には，[Huang 08] と同様に平均化パーセプトロンを用いる．ただし，以下の 2 つの工夫を行う．

まず，生成された単語ラティス $L(x)$ は，必ずしも正解の経路を含んでいるとは限らない．そこで，訓練時には，正解経路に出現する全てのエッジをあらかじめ $L(x)$ に追加しておく．

次に，再順序付けの重みベクトル $\mathbf{\Lambda}$ の訓練と，単語ラティス生成器 (4·1 節と 4·2 節で説明した単語分割モデルと品詞タグ付与モデルのことを指す) の訓練には別のデータを使う．この理由は，もし同じ訓練データを使用したとすると，実際のテスト時よりも良い単語ラティスを使って再順序付けの訓練を行うことになってしまうと予想されるためである．この問題を避けるため，以下の手順で訓練を行う．まず，訓練データを 10 個のサブデータに分割する．そして，そのうち 9 つのサブデータを用いて単語ラティス生成器の学習を行い，それを用いて残り 1 つのサブデータに対して単語ラティスを生成する．なお，テスト時には，全訓練データから学習した単語ラティス生成器を用いる．

	訓練	開発	評価
KC	30,608	4,028	3,764
KNBC	3,453	385	348
BCCWJ	47,547	6,144	5,741

表 3 3 つのデータセットに含まれる文数.

6. 実　　験

本節では実験結果について報告する．6·1 節，6·2 節，6·3 節では，それぞれデータセット，比較するラティス生成アルゴリズム，パラメータの調整方法について述べる．そして，6·4 節において実験結果を詳細に述べる．最後に，6·5 節では既存の形態素解析ソフトウェアとの比較を行う．

6·1 データセット

評価には，京都大学テキストコーパス version4.0，京都大学 NTT ブログコーパス version1.0，現代日本語書き言葉均衡コーパスの 3 つのデータセットを用いた [Kurohashi 98, Hashimoto 11, Maekawa 08]．以下ではそれぞれを KC，KNBC，BCCWJ と表記する．各コーパスは訓練，開発，評価用の 3 つに分割して利用した (表 3)．

素性抽出に必要な辞書としては，JUMAN 辞書 version 7.0 [*3] と UniDic version 1.3.12[*4] を用いた．単語分割基準および品詞体系の違いから，KC と KNBC での実験には JUMAN 辞書を，BCCWJ での実験には UniDic を用いる．

6·2 単語ラティス生成アルゴリズム

枝刈りアルゴリズムと段階的アルゴリズムの 2 種類の単語ラティス生成アルゴリズムに基づく形態素解析器を実装し，それらの比較を行った．再順位付けについては，どちらも 5 章で説明した手法を用いた．

Jiang らは枝刈りアルゴリズムの閾値 K を 20 に固定して実験を行っていたが [Jiang 08]，我々は，異なる閾値の効果を調べるために K = 5, 10, 20 の 3 種類を試した．

6·3 パラメータの調整

枝刈りアルゴリズムのパラメータ k は開発データを用いて調整した．具体的には，$\{1,2,4,8,\ldots 256\}$ の中から，正解エッジの少なくとも θ% を含む単語ラティスを生成し，なおかつ，エッジ数が最も少ない単語ラティスを生成した値を採用した．段階的アルゴリズムについても，$\{(\alpha,\beta)|\alpha,\beta \in \{1,2,4,8,\ldots,256\}\}$ の中から，上記と同じ要領でパラメータ (α,β) の値を決定した．

θ の値は，KC と BCCWJ については 99，KNBC については 97 とした．KNBC の場合に小さな値を用いたのは，訓練データが小さく，99%の正解エッジを含むような単語ラティスが生成できなかったためである．KC, KNBC,

*3 http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?NLPresources
*4 http://www.tokuteicorpus.jp/dist

	KC					KNBC					BCCWJ				
	Time		#Cand.	F_1	Size	Time		#Cand.	F_1	Size	Time		#Cand.	F_1	Size
枝刈り (K=5)	18	20	212	†97.17	356	1.1	1.2	137	93.19	235	20	21	163	†97.33	276
枝刈り (K=10)	27	28	400	**97.89**	88.9	1.6	1.7	250	93.24	118	30	32	301	†98.08	138
枝刈り (K=20)	59	60	702	97.88	88.9	3.2	3.3	413	93.39	118	66	67	516	**98.18**	138
段階的	**1.6**	**2.4**	**30.4**	97.87	**60.9**	**0.11**	**0.15**	**24.8**	**93.64**	**99.1**	**2.1**	**3.1**	**23.3**	98.09	**46.6**

表 4 異なる単語ラティス生成アルゴリズムの比較．それぞれの指標において最も良い結果は数字を太字で強調している．

BCCWJ の各データに対して選択された k の値は，8，8，2 (K=5)，2，4，2 (K=10)，2，4，2 (K=20) であった．一方，(α, β) の値は，$(4,2), (8,4), (2,2)$ であった．

6·4 実 験 結 果

表 4 に実験結果を示す．表中の Time は，単語ラティス生成に要した時間と形態素解析全体に要した時間を秒単位で示している[*5]．#Cand は，1 文当たりの単語候補数 (詳細は後述) の平均である．F_1 は単語単位での適合率と再現率の調和平均である．このとき，単語が適切に分割され，なおかつ正しい品詞タグが付与された場合を正解として数えている．最後に，Size は 1 文当たりの単語ラティスの平均サイズ[*6]である．

単語候補数とは，単語ラティスを生成する過程で考慮された単語数のことを指す．枝刈りアルゴリズムの場合は，アルゴリズム 1 の 6 行目で生成された単語 w の数であり，段階的アルゴリズムの場合は，単語集合の大きさ $|W|$ である (アルゴリズム 2 の 2 行目)．ここで単語候補数を調べているのは，この数字から，単語ラティスの生成に必要な時間を見積ることができるためである．すなわち，単語候補数を調べることによって，実装の詳細に左右されることなく，アルゴリズムの効率性を調査することができる．

各データセットについて，最も高い F_1 スコアを達成したアルゴリズムと，それ以外のアルゴリズムの差が統計的に有意にであるかどうかを調査した ($p < 0.01$)．統計的な有意差を調べるためには，bootstrap resampling 法を用いた．その際のサンプル数は 1000 とした．この結果，最も F_1 スコアの高かったアルゴリズムよりも，F_1 スコアが有意に低いと判断された結果は，表中で † によって示している．

§1 処理時間

表 4 の結果から，枝刈りアルゴリズムを用いた形態素解析システムは，単語ラティスの生成に処理時間の大部分を費やしていることが分かる．このことから，枝刈りアルゴリズムは効率性に難点があると言える．このことは，これまでの研究において，全く指摘されていないことである [Zhang 10, Sun 11]．一方，段階的アルゴリズムは，枝刈りアルゴリズムを用いた場合よりも 10 倍から 30 倍程度高速であり，効率性という観点から大きな優位性を持っていることが確認できる．

一方，処理時間は，単語候補数 (#Cand) におおよそ比例している．このことから，処理時間の短縮は，主に考慮する単語候補数を削減することによって達成できていると結論づけることができる．

§2 解析精度

次に解析精度について考察する．段階的アルゴリズムを用いた場合の F_1 スコアは，枝刈りアルゴリズムを用いた場合とほぼ同等であるか，または高いという結果になった．この結果から，段階的アルゴリズムの高速性は，解析精度を犠牲にすることによって得られたものではないことが分かる．

また，この実験の結果からは，枝刈りアルゴリズムにおいては，適切な閾値 K を設定することが重要であることも分かる．まず，K の値が小さすぎると，F_1 スコアが大きく低下することが確認できる．例えば K=5 の場合，全てのデータセットにおいて，最も F_1 スコアが低くなった．一方，K の値が大きすぎると，F_1 スコアは改善されるものの，速度が大きく低下してしまうことが確認できる (K=20)．K=5 の場合に解析できない典型的な事例としては，長い片仮名語 (例:「ノーマライゼーション」「アイデンティティー」)，長い活用語尾を持つ語 (例:「同じだったろう」[*7])，複合動詞 (例:「行き詰まって」) が見られた．

§3 単語ラティスの大きさ

表 4 の単語ラティスのサイズからは，段階的アルゴリズムは枝刈りアルゴリズムよりも，一貫して小さな単語ラティスを生成していることが分かる．このことは，枝刈りアルゴリズムが，単語ラティスのノードを刈り込むことができないことに一因があると考えられる．すなわち，枝刈りアルゴリズムは，入力文長が n であった場合，必ず $n+1$ 個のノードを持つ単語ラティスを生成するため，単語ラティスのサイズが大きくなりやすい．一方，段階的アルゴリズムにはそのような問題は見られない．

さらに，パラメータを変化させた際に，単語ラティスのサイズと，正解エッジに対するカバー率[*8]の関係がどのように変化するのかを調査した (図 2)．枝刈りアルゴリズムは，パラメータ k を $\{1,2,4,8,16\}$ の範囲で変化させた．ただし，K が 10 と 20 の場合では，ほぼ同じサイズの単語ラティスが得られたので，K=10 の結果は省略している．一方，段階的アルゴリズムの場合はパラ

∗5 実験には 3.2 GHz Intel® Xeon™ CPU を用いた．
∗6 単語ラティスのサイズとはエッジ数のこととする．
∗7 JUMAN 辞書の基準では「だったろう」が活用語尾となっている．
∗8 正解エッジ全体に対する，単語ラティスに含まれている正解エッジの割合．

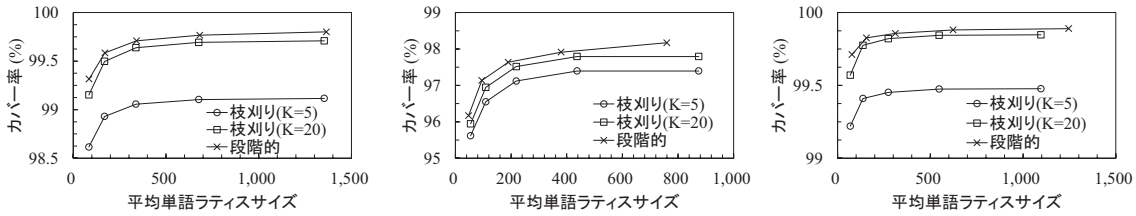


図 2 正解エッジに対するカバー率と単語ラティスの平均サイズの関係 (左: KC，中央: KNBC，右: BCCWJ).

メータを 2 つ持っているため，$\alpha = 32$ と固定して，β を $\{1, 2, 4, 8, 16\}$ の範囲で変化させることによって，2 次元平面上にグラフを描画した (ここでの α は，6.3 節で採用されたものよりも大きな値を用いた．これは，α の値が小さいと β を変化させてもカバー率がほとんど向上せず，手法間の差を図示することが難しかったためである．). 図 2 より，生成される単語ラティスのサイズが同等であるときには，段階的アルゴリズムは枝刈りアルゴリズムよりも高いカバー率を実現可能であることが分かる．

4·3 節では，単語候補数 $|W|$ が入力文長に対して線形であることを示した．このことを実験的に確認するため，入力文長と単語候補数の関係についても調査を行った (図 3)．この図から，単語候補数は入力文長に対して線形に増加することが確認できる．

6·5 既存ソフトウェアとの比較

最後に，提案する単語ラティス生成アルゴリズムにもとづく再順位付けシステムと，既存のソフトウェアとの比較を行った．ここでは代表的なソフトウェアとして，JUMAN version 7.0[*9]，MeCab version 0.993 [Kudo 04][*10]，Kytea version 4.6 [Neubig 11][*11]を用いた．ただし，これらのソフトウェアと我々のシステムとでは，解析モデルや実装の詳細が大きく異なっているため，これらの公平な比較は難しく，以下で述べる比較結果はあくまでも補足的な情報である．

表 5 に F_1 スコアと未知語再現率 (辞書に登録されていない単語に対する再現率) の比較結果を示す．表中の再順位付けとあるのが我々のシステムであり，KNBC における JUMAN の結果を除いては，既存ソフトウェアを上回る F_1 スコアを達成している．また，bootstrap resampling により，F_1 スコアに統計的有意差 ($p < 0.01$) が確認されるかどうを調べたところ，F_1 スコアの向上は統計的に有意であることが確認された (検定結果は表 4 と同様に † で表している).

一方，未知語再現率においても，提案手法は既存ソフトウェアよりも良好な結果を示した．特に，JUMAN や MeCab といった，辞書引き結果に強く依存したラティス生成方法を採用しているソフトウェアと比較した場合に，未知語再現率の差が顕著であることが分かる．また，提案手法は Kytea よりも高い未知語再現率が達成できているが，この結果は，単語単位の素性や単語 2-gram 素性など，豊富な素性を使用することのできるラティスに基づく解析手法の有効性を示す結果であると言える．

	KC	KNBC	BCCWJ
JUMAN	†95.4 (40.3)	**93.8** (30)	N/A
MeCab	†95.2 (35.7)	†91.6 (13)	†93.2 (74.8)
Kytea	†96.9 (66.9)	†91.0 (32)	†97.1 (84.6)
再順位付け	**97.9** (76.2)	93.6 (38)	**98.1** (87.0)

表 5 既存ソフトウェアとの F_1 スコアの比較．括弧内の数字は未知語再現率を表す．KNBC における未知語再現率は，未知語の数が 242 個と少なかったため有効数字 2 桁を記載している．その他については有効数字 3 桁としている．JUMAN は BCCWJ のアノテーション基準に沿った解析結果を出力できないため，当該部分の結果は N/A としている．

さらに，これらのソフトウェアとの解析速度の比較を行った．我々のシステムの解析速度は 1400 文/秒であった．これに対して，JUMAN，MeCab，Kytea の解析速度は，それぞれ 2100 文/秒，29000 文/秒，3200 文/秒であった．この結果から，我々のシステムが再順位付けによって高い F_1 スコアを達成する一方，JUMAN や Kytea と比較しても遜色のない解析速度を実現していることが分かる．

7. お わ り に

本論文では，未知語を考慮した形態素解析において，効率的に単語ラティスを生成する方法について実験的検証を行った．その結果，単語と品詞タグを独立に生成する段階的アルゴリズムの有効性を確認した．このアルゴリズムは，従来研究で利用されていた枝刈りにもとづくものと比べて 10 倍以上高速であり，なおかつ精度の低下も見られなかった．提案アルゴリズムは，今後，未知語に頑健な形態素解析システムの構築していく上で，有用なコンポーネントになると考えられる．提案するラティス生成方法は，系列ラベリングの多層化とも関係が見られるが，そのようなタスク設定においては従来にも様々な手法が提案されているため [Sutton 07, 東 10]，今後そうした手法を適用することも可能であろう．

今後の課題としては，本論文で提案した単語ラティス生成方法を用いて，より高精度な未知語処理の実現に取り組みたい．一例として，近年では翻字や言い換えを利用して単語分割の精度を向上させる試みが報告されているが [Kaji 11, Hagiwara 13]，そうした素性を取り組むこ

∗9 http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN
∗10 http://code.google.com/p/mecab
∗11 http://www.phontron.com/kytea

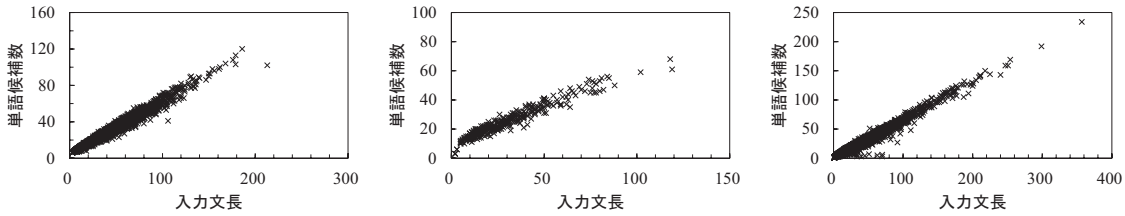


図 3 単語候補数と入力文長の関係 (左: KC，中央: KNBC，右: BCCWJ).

となどが考えられる．

謝　　辞

本研究は，最先端研究開発支援プログラム「超巨大データベース時代に向けた最高速データベースエンジンの開発と当該エンジンを核とする戦略的社会サービスの実証・評価」の支援を受けている．本原稿は，IJCNLP2013 において発表したものを日本語に再構成したものである．

◇ 参 考 文 献 ◇

[Asahara 00] Asahara, M. and Matsumoto, Y.: Extened Models and Tools for High-performance Part-of-speech Tagger, in *Proceedings of COLING*, pp. 21–27 (2000)

[Collins 00] Collins, M.: Discriminative Reranking for Natural Language Parsing, in *Proceedings of ICML*, pp. 175–182 (2000)

[Collins 02] Collins, M.: Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms, in *Proceedings of EMNLP*, pp. 1–8 (2002)

[Hagiwara 13] Hagiwara, M. and Sekine, S.: Accurate Word Segmentation using Transliteration and Language Model Projection, in *Proceedings of ACL (Short Papers)*, pp. 183–189 (2013)

[Hashimoto 11] Hashimoto, C., Kurohashi, S., Kawahara, D., Shinzato, K., and Nagata, M.: Construction of a Blog Corpus with Syntactic, Anaphoric, and Semantic Annotations (In Japanese), *Journal of Natural Language Processing*, Vol. 18, No. 2, pp. 175–201 (2011)

[Huang 08] Huang, L.: Forest Reranking: Discriminative Parsing with Non-local Features, in *Proceedings of ACL*, pp. 586–594 (2008)

[Jiang 08] Jiang, W., Mi, H., and Liu, Q.: Word Lattice Reranking for Chinese Word Segmentation and Part-of-Speech Tagging, in *Proceedings of Coling*, pp. 385–392 (2008)

[Kaji 11] Kaji, N. and Kitsuregawa, M.: Splitting Noun Compounds via Monolingual and Bilingual Paraphrasing: A Study on Japanese Katakana Words, in *Proceedings of EMNLP*, pp. 959–969 (2011)

[Kruengkrai 06] Kruengkrai, C., Sornlertlamvnich, V., and Isahara, H.: A Conditional Random Field Framework for Thai Morphological Analysis, in *Proceedings of LREC*, pp. 2419–2424 (2006)

[Kruengkrai 09] Kruengkrai, C., Uchimoto, K., Kazama, J., Wang, Y., Torisawa, K., and Isahara, H.: An Error-Driven Word-Character Hybrid Model for Joint Chinese Word Segmentation and POS Tagging, in *Proceedings of ACL*, pp. 513–521 (2009)

[Kudo 04] Kudo, T., Yamamoto, K., and Matsumoto, Y.: Applying Conditional Random Fields to Japanese Morphological Analysis, in *Proceedings of EMNLP*, pp. 230–237 (2004)

[Kurohashi 98] Kurohashi, S. and Nagao, M.: Building a Japanese Parsed Corpus while Improving the Parsing System, in *Proceedings of LREC*, pp. 719–724 (1998)

[Maekawa 08] Maekawa, K.: Balanced corpus of contemporary written Japanese, in *Proceedings of the 6th Workshop on Asian Language Resources*, pp. 101–102 (2008)

[Nakagawa 04] Nakagawa, T.: Chinese and Japanese Word Segmentation Using Word-Level and Character-Level Information, in *Proceedings of Coling*, pp. 466–472 (2004)

[Nakagawa 07] Nakagawa, T. and Uchimoto, K.: A Hybrid Approach to Word Segmentation and POS Tagging, in *Proceedings of ACL, Demo and Poster Sessions*, pp. 217–220 (2007)

[Neubig 11] Neubig, G., Nakata, Y., and Mori, S.: Pointwise Prediction for Robust Adaptable Japanese Morphological Analysis, in *Proceedings of ACL*, pp. 529–533 (2011)

[Peng 04] Peng, F., Feng, F., and McCallum, A.: Chinese Segmentation and New Word Detection using Conditional Random Fields, in *Proceedings of Coling*, pp. 562–568 (2004)

[Sarawagi 04] Sarawagi, S. and Cohen, W. W.: Semi-Markov Conditional Random Fields for Information Extraction, in *Proceedings of NIPS*, pp. 1185–1192 (2004)

[Shi 07] Shi, Y. and Wang, M.: A Dual-layer CRFs Based Joint Decoding Method for Cascaded Segmentation and Labeling Tasks, in *Proceedings of IJCAI*, pp. 1707–1712 (2007)

[Sun 11] Sun, W.: A Stacked Sub-Word Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging, in *Proceedings of ACL*, pp. 1385–1394 (2011)

[Sutton 07] Sutton, C., McCallum, A., and Rohanimanesh, K.: Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequ, *JMLR*, Vol. 8, pp. 693–723 (2007)

[Uchimoto 01] Uchimoto, K., Sekine, S., and Isahara, H.: The Unknown Word Problem: a Morphological Analysis of Japanese Using Maximum Entropy Aided by a Dictionary, in *Proceedings of EMNLP*, pp. 91–99 (2001)

[Xue 03] Xue, N.: Chinese Word Segmentation as Character Tagging, *Computatioinal Linguistics and Chinese Language Processing*, Vol. 8, No. 1, pp. 29–48 (2003)

[Zhang 08] Zhang, Y. and Clark, S.: Joint Word Segmentation and POS Tagging Using a Single Perceptron, in *Proceedings of ACL*, pp. 888–896 (2008)

[Zhang 10] Zhang, Y. and Clark, S.: A Fast Decoder for Joint Word Segmentation and POS Tagging Using a Single Discriminative Model, in *Proceedings of EMNLP*, pp. 843–852 (2010)

[岡野原 08] 岡野原 大輔, 辻井 潤一：Shift-Reduce 操作に基づく未知語を考慮した形態素解析, 言語処理学会第 14 回年次大会論文集 (2008)

[持橋 11] 持橋大地, 鈴木潤, 藤野昭典：条件付確率場とベイズ階層言語モデルの統合による半教師あり形態素解析, 言語処理学会第 17 回年次大会発表論文集 (2011)

[東 06] 東 藍, 浅原 正幸, 松本 裕治:条件付確率場による日本語未知語処理, 情報処理学会研究報告 自然言語処理研究会 2006-NL-175, pp. 67–74 (2006)

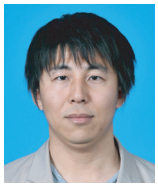
[東 10] 東 藍, 松本 裕治：系列ラベリングの多層化, 電子情報通信学会情報論的学習と機械学習研究会 (2010)

[萩原 10] 萩原正人, 中山敬広, 水野貴明 (訳)：入門 自然言語処理, オライリー・ジャパン (2010)

〔担当委員：橋本 泰一〕

2013 年 9 月 9 日　受理

著　者　紹　介



鍛治　伸裕

2005 年，東京大学情報理工学系研究科博士後期課程修了．博士 (情報理工学)．2007 年東京大学生産技術研究所特任助教を経て，現在，同大学生産技術研究所特任准教授．自然言語処理の研究に従事．言語処理学会，人工知能学会，情報処理学会 各会員．

喜連川　優

1983 年東京大学工学系研究科情報工学専攻博士課程修了，工学博士．東京大学生産技術研究所教授，東京大学地球観測データ統融合連携研究機構長，2013 年 4 月より国立情報学研究所所長．データベース工学の研究に従事．2013 年 6 月より情報処理学会会長．情報処理学会功績賞，ACM SIGMOD E.F Codd Innovations Award 受賞．平成 25 年紫綬褒章．ACM，IEEE，電子情報通信学会 ならびに情報処理学会フェロー．