Vol.2013-NL-211 No.12
Vol.2013-SLP-96 No.12
2013/5/24

未知語を考慮した形態素解析のための単語ラティスの効率的な生成方法

鍛治 伸裕[1,a)]　喜連川 優[1]

概要：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再順位付けにもとづくものが広く用いられている．しかし，この枠組みにおいて未知語を考慮した形態素解析を行う場合，入力文長に対して 2 乗のオーダの計算量が単語ラティスの生成に必要となってしまう．そのため，何らかの方法で，これを高速化することが実用上重要となる．本論文では，単語ラティス生成のために従来提案されていた枝刈りアルゴリズムは十分に効率的ではないこと，および，我々が考案した段階的アルゴリズムによって形態素解析処理がおよそ 10 倍高速化されることを示す．さらに，段階的アルゴリズムが生成する単語ラティスの大きさについて理論的な考察を行い，その妥当性を実験によって明らかにする．

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再順位付け [6] にもとづくものが広く用いられている [1], [6], [8], [10], [11]．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とい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う場合，単語ラティスを効率的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．それら全ての候補について，単語ラティスに含めるか否かの判断を行ったとすると，$O(n^2)$ の計算量が発生し，処理のボトルネックとなってしまう．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．

しかしながら，単語ラティスの生成を高速化することは決して単純な課題ではない．なぜなら，単語ラティスの生成には，速度以外にも考慮すべき要因がいくつか存在しているからである．例えば，高い精度での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ような単語ラティスを生成する必要がある．また，単語ラティスが巨大になると，経路探索に必要な計算コストが深刻なものとなってしまう．そのため，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．

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，過去の研究において，単語ラティスの生成に関する技術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，辞書引きによるものを挙げることができる [10]．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 [19], [23]．一方，文献 [6] では，単語ラティス生成のための枝刈りアルゴリズムが提案されているが，計算時間に関する報告はなされていない．

こうした研究状況を踏まえ，我々は，未知語を考慮した形態素解析における単語ラティスの生成方法について検討を行った．本論文では，そこから得られた知見の報告を行う．主な内容は以下の通りである．

- 文献 [6] に提案されている枝刈りアルゴリズムは，実際のところ十分に効率的ではなく，形態素解析を行ううえでボトルネックとなることを見つけた．
- 枝刈りアルゴリズムに代わる手法として，段階的な単語ラティス生成アルゴリズム (3 節を参照) を考案した．そして，このアルゴリズムが，精度を犠牲にすることなく，形態素解析処理を 10 倍程度高速化することを確認した．
- 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した．

本論文の貢献は，(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うるという問題を指摘し，(2) 段階的アルゴリズムの有効性を実証的に明らかにしたことである．3 節で述べるように，段階的アルゴリズム自体は，

[1] 〒 153-8505 東京都目黒区駒場 4-6-1 東京大学生産技術研究所
[a)] kaji@tkl.iis.u-tokyo.ac.jp

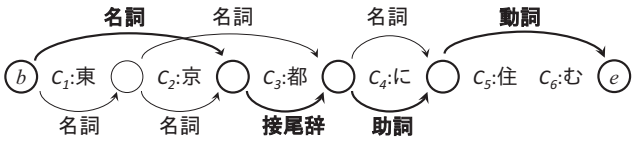

図 1 単語ラティスの例 [10]．太字で強調されている経路が，この入力文に対する正しい形態素解析結果である．

よく知られた技術 [15], [16], [17] の素直な拡張である．しかし，そのような単語ラティス生成アルゴリズムを再順位付けの中で用いた場合の有効性は，これまでに検証されてこなかった．また同様に，既存の枝刈りアルゴリズム [6] との優劣も明らかにされていない．これらの点に対して，実証的な回答を提示したところが本論文の成果である．実験の結果，段階的アルゴリズムは枝刈りアルゴリズムよりも 10 倍以上高速であり，なおかつ精度の低下も引き起こさないことが分かった．これは，実用上，有用な知見になると考えられる．

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，Jiang ら [6] によって提案された，枝刈りに基づく単語ラティス生成アルゴリズムを紹介する．

2.1 単語ラティス

単語ラティス [6], [10] とは，指数関数的な数の形態素解析結果の候補を効率的に表現するための圧縮データ構造であり (図 1)，形式的には有向グラフの一種と見ることができる．ここでノード (図 1 では円で表されている) は単語境界を表現しており，すべての文字の間にノードを配置することができる．また，文の先頭と末尾には，特殊ノード b と e を必ず配置するものとする．一方，エッジは単語と品詞タグの組を表現している (図 1 では矢印で表されている)．

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を表現している．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 [6] 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．これは，まず入力文に対して小さな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．

```
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
```

図 2 枝刈りに基づく単語ラティス生成アルゴリズム

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効率化されることである．これは，いわゆる再順位付け [2] 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 [5]．

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，$\hat{y}$ はその最適経路，$\textsc{Score}(x, y)$ は経路 y の再順位付けを行うスコア関数である．なお，表記を簡略化するため，$L(x)$ は単語ラティスに含まれる全経路の集合を表すものとする．

本論文では，上記のような定式化にもとづき，単語ラティス $L(x)$ の生成方法について議論を行う．節においても述べたように，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が技術課題となる．厳密には，最適経路 $\hat{y}$ の探索にかかる計算コストも考慮する必要があるが，この枠組みにおいて生成される単語ラティスは通常小さなものであるため，最適経路の探索コストは実質的に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における単語ラティス生成手法としては，Jiang ら [6] が提案した枝刈りアルゴリズムが知られている．以下では，この枝刈りアルゴリズムについて簡単な説明を行う．詳細については文献 [6] を参照されたい．

Jiang らの枝刈りアルゴリズムは，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 (図 2)．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 (i.e., 単語 $w = c_{i-l+1}c_{i-l+2}\dots c_i$ 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)．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 について順次行い，最終的にエッジ集合 E 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，我々の実験結果 (5 節) 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，形態素解析処理のボトルネックとなる．

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．Jinag ら [6] 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. 段階的単語ラティス生成アルゴリズム

我々は，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考案した (図 3)．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，単語集合 W を生成する (2 行目)．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 [15], [16], [20] を利用することによって，$O(n^2)$ の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[*1]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可能になる．

以下では，単語生成と品詞タグ生成について，それぞれ 3.1 節と 3.2 節で説明する．そして， 3.3 節では，段階的アルゴリズムの計算量について考察を行う．

3.1 単語生成

入力文 x に対して単語集合 W を生成するためには (図 3 の 2 行目)，文字単位の単語分割モデル [16] を用いた:

[*1] 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．

```
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
```

図 3 段階的単語ラティス生成アルゴリズム

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

ここで $b = b_1 \dots b_n$ 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している．$\mathbf{\Lambda}_w$ と $\mathbf{F}_w(x, b)$ はそれぞれ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た [3]．素性には，表 1 にあるものとタグ 2-gram を用いた．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 [15] が提案した辞書素性である．この素性は，入力文中のあ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．

この単語分割モデルに基づき，入力文に対するスコア上位 α 件の単語分割結果から単語集合 W を生成する:

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

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 は超パラメータであり，後述するように，開発データを用いて決定する．

3.2 品詞タグ生成

各単語 w に対して品詞タグ集合 T を生成するためには (図 3 の 4 行目)，多クラス線形モデルを構築して，そのスコアの上位 β 件を T とした．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した [15]:

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

$\mathbf{F}_t(x, w, t)$ は素性ベクトルである．ここでの素性としては，単語の表層形，単語長 (文字数)，接辞文字列，周辺文字列を用いた (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 値素性を用いた．$\mathbf{\Lambda}_t$ 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．β は超パラメータであり，α と同様に開発データを用いて決定する．

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

表 1 単語分割モデルのための素性テンプレート．c_i と c'_i は対象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 種類を用いた．c_{i-1} や c'_{i+1} は対象文字の周辺に出現する文字を表す．r(または l) は，対象語から始まる (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．また，i は，対象を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	$\langle \textsc{length}(w), t\rangle$
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	$\langle d\rangle$, $\langle d, t\rangle$

表 2 品詞タグ付与モデルのための素性テンプレート．このとき $w = c_i c_{i+1} \ldots c_{j-1}$ 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．$\textsc{length}(w)$ 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．d 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．

3.3 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，$O(n)$ の計算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 $O(|W|) = O(n)$ となるため，図 3 の外側のループ (3 行目) を実行するために必要な時間も $O(n)$ である．最後に，内側のループ (5 行目) の実行時間は，入力文の長さ n に依存しない．これらのことから，段階的アルゴリズムに必要な計算時間は $O(n)$ であることが分かる．

同様の議論によって，単語ラティスに含まれるエッジの数もまた入力文長に対して線形である (すなわち $O(|E|) = O(n)$ 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計算量も $O(n)$ であることが分かる．

4. 再順位付け

再順位付けの方法としては，Huang[5] が提案したものを用いる．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．

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

ここで $\mathbf{F}(x, y)$ と $\boldsymbol{\Lambda}$ は，それぞれ素性ベクトルと重みベクトルである．また，以下で述べるように，素性はすべて局所素性のみを用いているため，最適経路の探索には動的計画法を用いる．

4.1 素性

素性には次のようなものを用いた．まず，表 1 の素性テンプレートに対して，文字の単語内の位置情報 (Nakagawa[13] と同様に BIES の 4 種類の位置情報を用いた) を組み合わせたものを用いた．この素性テンプレートは，例えば，$\langle c_i, c_{i+1}, B\rangle$ や $\langle c'_i, I\rangle$ のようになる．前者は文字 c_i が単語の先頭である場合の文字 2-gram $c_i c_{i+1}$，後者は文字 c_i が単語の内部に位置する場合の文字種 1-gram c'_i を表す．こうした素性に加えて，表 2 と同一の素性，および品詞タグ 2-gram を用いた．

4.2 訓練

重みベクトル $\boldsymbol{\Lambda}$ の訓練には平均化パーセプトロンを用いた．ただし，以下の 2 つの点に工夫をする．

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．そのような場合，パーセプトロンアルゴリズムに従って重みベクトルを更新することができないことが問題となる．この問題を回避するため，訓練時には，正解経路に出現する全てのエッジを $L(x)$

	訓練	開発	評価
KC	30,608	4028	3764
KNBC	3453	385	348
BCCWJ	47,547	6144	5741

表 3 3 つのデータセットに含まれる文数.

に追加しておくこととした.

次に，重みベクトル $\mathbf{\Lambda}$ の訓練と単語ラティス生成器 (3.1 節と 3.2 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必要がある．もし，これらを同じデータを使って訓練した場合，単語ラティスはクローズドテストと同じ状況で生成されることになってしまい望ましくない．そこで以下のような手順で訓練を行った．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．

5. 実験

本節では実験結果について報告する．5.1 節，5.2 節，5.3 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，超パラメータの調整方法について述べる．そして，5.4 節において実験結果を詳細に述べる．最後に，5.5 節では既存の形態素解析ソフトウェアとの比較を行う．

5.1 データセット

アルゴリズムの訓練や評価には, 京都大学テキストコーパス version 4.0 [11]，京都大学 NTT ブログコーパス version 1.0 [4]，現代日本語書き言葉均衡コーパス [12] の 3 つを用いた．以下では各々を KC，KNBC，BCCWJ と表記する．各コーパスは訓練，開発，評価用に分割して利用した (表 3).

素性抽出に必要な辞書は JUMAN 辞書 versiono 7.0 [*2] と UniDic version 1.3.12[*3] 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 については JUMAN 辞書を，BCCWJ については UniDic を用いた.

5.2 単語ラティス生成アルゴリズム

実験では，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た．再順位付けの方法については，全て 4 節で説明したものを用いる.

Jiang ら [6] 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，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.

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

5.3 超パラメータの調整

枝刈りアルゴリズムの超パラメータ k は開発データを用いて調整した．具体的には，$\{2, 4, 8, 16 \ldots 256\}$ 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，$\{(\alpha, \beta) | \alpha, \beta \in \{2, 4, 8, \ldots, 256\}\}$ の中から，上記と同じ要領で超パラメータ (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.

5.4 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．#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[*4]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，図 2 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 (図 3 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，アルゴリズムの速度を評価することができる.

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には † を付与している.

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，効率性という観点から問題を抱えていると言える．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 [18], [22]．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

[*4] 単語ラティスのサイズとはエッジ数のこととする.

	KC					KNBC					BCCWJ				
	Time		#Cand.	F_1	Size	Time		#Cand.	F_1	Size	Time		#Cand.	F_1	Size
枝刈り (K=5)	20	22	212	†97.25	356	1.2	1.3	137	†92.72	235	25	26	163	†97.33	276
枝刈り (K=10)	31	32	400	97.92	88.9	2.0	2.1	250	93.48	235	43	44	301	†98.08	69.0
枝刈り (K=20)	62	63	702	**97.94**	88.9	3.6	3.8	413	93.42	235	88	90	516	**98.18**	69.0
段階的	**1.8**	**2.6**	**30.4**	**97.94**	**60.8**	**0.12**	**0.18**	**24.8**	**93.92**	**99.2**	**2.3**	**3.1**	**23.3**	98.10	**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．

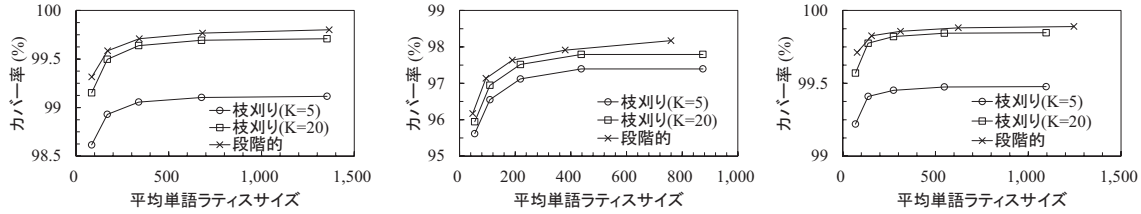

図 4 正解エッジに対するカバー率と単語ラティスの平均サイズの関係 (左: KC, 中央: KNBC, 右: BCCWJ).

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た．

一方，単語候補数 (#Cand) を見ると，処理時間と相関していることが分かる．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ことが確認できる．

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．これらの結果から，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確認できた．

また，実験の結果から，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が分かった．例えば，K の値が小さ過ぎた場合，F_1 スコアが大きく低下することが表から分かる (K=5)．実際，K=5 の場合，全てのデータセットにおいて，最も F_1 スコアが高かったアルゴリズムと比べて，F_1 スコアの差に統計的な有意差が見られた．一方，K の値が大き過ぎると，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た (K=20)．

単語ラティスの大きさ

単語ラティスのサイズについて調査を行ったところ，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った．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，超パラメータの変化に対して，単語ラティスのサイズと正解エッジに対するカバー率[*5]の関係性がどのように変化するのかを調査した (図 4)．枝刈りアルゴリズムは，超パラメータ 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，ほぼ同じ単語ラティスを生成したので，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超パラメータを 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．図 4 からは，全体的な傾向として，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していることが分かる．

3.3 節において我々は，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．図 5 に，入力文長と単語候補数の関係を示す．この図から，単語候補数が入力文長に対して線形に増加していることを実際に確認することができる．

5.5 既存ソフトウェアとの比較

最後に，我々の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[*6]，MeCab [10][*7]，Kytea [15][*8]を用いた．

表 5 に，3 つのソフトウェアとの F_1 スコアの比較結果を示す．表中の再順位付けとあるのが我々のシステムであり，全てのデータセットにおいて，既存ソフトウェアを上

[*5] 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．
[*6] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN
[*7] http://code.google.com/p/mecab
[*8] http://www.phontron.com/kytea

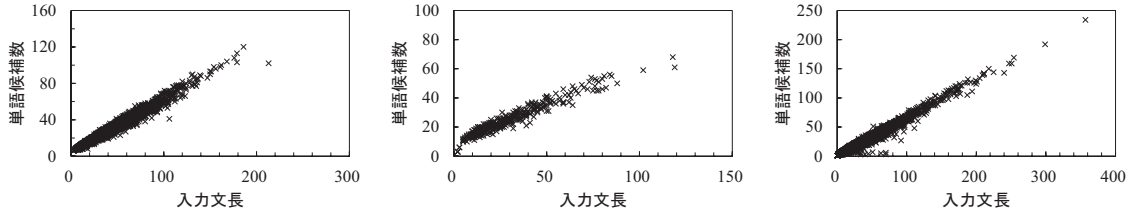

図 5 単語候補数と入力文長の関係 (左: KC，中央: KNBC，右: BCCWJ).

	KC	KNBC	BCCWJ
JUMAN	†95.37	93.85	N/A
MeCab	†95.45	†91.60	†96.31
Kytea	†96.95	†90.91	†97.10
再順位付け	**97.94**	**93.92**	**98.10**

表 5 既存ソフトウェアとの F_1 スコアの比較.

回る F_1 スコアを達成した．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かを調べ，その結果を表 4 と同様に † で表す．KNBC における JUMAN の結果を除いては，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．

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, 既存のソフトウェア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．

6. 関連研究

Nakagawa ら [14] や Kruengkrai ら [9] 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．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しやすいことが利点となる．

形態素解析の高速化を主眼においた研究としては, Zhang ら [21], [22] や岡野原ら [23] 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 (= 解空間の削減) を扱っている本研究と相補的な試みと考えられる．そのため，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考えられる．今回，我々は最適経路の探索に動的計画法を用いたが，例えば非局所素性を用いる場合などは，彼らの提案するような探索アルゴリズムが有効になると考えられる．

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．

今後は，ここで述べたような枠組みを利用して，未知語の解析精度の向上に取り組んでいきたい．一例として，近年では翻字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 [7], [24]，そうした素性を取り組むことなどが考えられる．

参考文献

[1] Asahara, M. and Matsumoto, Y.: Extened Models and Tools for High-performance Part-of-speech Tagger, *Proceedings of COLING*, pp. 21–27 (2000).
[2] Collins, M.: Discriminative Reranking for Natural Language Parsing, *Proceedings of ICML*, pp. 175–182 (2000).
[3] Collins, M.: Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms, *Proceedings of EMNLP*, pp. 1–8 (2002).
[4] 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).
[5] Huang, L.: Forest Reranking: Discriminative Parsing with Non-local Features, *Proceedings of ACL*, pp. 586–594 (2008).
[6] Jiang, W., Mi, H. and Liu, Q.: Word Lattice Reranking for Chinese Word Segmentation and Part-of-Speech Tagging, *Proceedings of Coling*, pp. 385–392 (2008).
[7] Kaji, N. and Kitsuregawa, M.: Splitting Noun Compounds via Monolingual and Bilingual Paraphrasing: A Study on Japanese Katakana Words, *Proceedings of EMNLP*, pp. 959–969 (2011).
[8] Kruengkrai, C., Sornlertlamvnich, V. and Isahara, H.: A Conditional Random Field Framework for Thai Morphological Analysis, *Proceedings of LREC*, pp. 2419–2424 (2006).
[9] 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, *Proceedings of ACL*, pp. 513–521 (2009).
[10] Kudo, T., Yamamoto, K. and Matsumoto, Y.: Applying Conditional Random Fields to Japanese Morphological Analysis, *Proceedings of EMNLP*, pp. 230–237 (2004).
[11] Kurohashi, S. and Nagao, M.: Building a Japanese Parsed Corpus while Improving the Parsing System, *Pro-*

ceedings of LREC, pp. 719–724 (1998).
[12] Maekawa, K.: Balanced corpus of contemporary written Japanese, *Proceedings of the 6th Workshop on Asian Language Resources*, pp. 101–102 (2008).
[13] Nakagawa, T.: Chinese and Japanese Word Segmentation Using Word-Level and Character-Level Information, *Proceedings of Coling*, pp. 466–472 (2004).
[14] Nakagawa, T. and Uchimoto, K.: A Hybrid Approach to Word Segmentation and POS Tagging, *Proceedings of ACL, Demo and Poster Sessions*, pp. 217–220 (2007).
[15] Neubig, G., Nakata, Y. and Mori, S.: Pointwise Prediction for Robust Adaptable Japanese Morphological Analysis, *Proceedings of ACL*, pp. 529–533 (2011).
[16] Peng, F., Feng, F. and McCallum, A.: Chinese Segmentation and New Word Detection using Conditional Random Fields, *Proceedings of Coling*, pp. 562–568 (2004).
[17] Shi, Y. and Wang, M.: A Dual-layer CRFs Based Joint Decoding Method for Cascaded Segmentation and Labeling Tasks, *Proceedings of IJCAI*, pp. 1707–1712 (2007).
[18] Sun, W.: A Stacked Sub-Word Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging, *Proceedings of ACL*, pp. 1385–1394 (2011).
[19] Uchimoto, K., Sekine, S. and Isahara, H.: The Unknown Word Problem: a Morphological Analysis of Japanese Using Maximum Entropy Aided by a Dictionary, *Proceedings of EMNLP*, pp. 91–99 (2001).
[20] Xue, N.: Chinese Word Segmentation as Character Tagging, *Computatioinal Linguistics and Chinese Language Processing*, Vol. 8, No. 1, pp. 29–48 (2003).
[21] Zhang, Y. and Clark, S.: Joint Word Segmentation and POS Tagging Using a Single Perceptron, *Proceedings of ACL*, pp. 888–896 (2008).
[22] Zhang, Y. and Clark, S.: A Fast Decoder for Joint Word Segmentation and POS Tagging Using a Single Discriminative Model, *Proceedings of EMNLP*, pp. 843–852 (2010).
[23] 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).
[24] 萩原正人，　関根聡：翻字と言語モデル投影を用いた高精度な単語分割，言語処理学会第 19 回年次大会論文集，pp. 908–911 (2013).