# PPM* 言語モデルを用いた日本語単語分割

小田裕樹[†,☆] 北 研二[†]

日本語処理において，単語の同定，すなわち文の単語分割は，最も基本的かつ重要な処理である．本論文では，確率的な文字言語モデルに基づく新しい単語分割手法を提案する．まず，本研究の基本法である文字 $n$-gram モデルに基づく単語分割法を提案する．さらに，単語分割の精度向上のために，文字 $n$-gram モデルに代わり，データ圧縮アルゴリズム PPM* を用いた言語モデルを適用した単語分割法を提案する．PPM* は，確率・統計的テキスト圧縮技法として最も性能の優れた PPM（Prediction by Partial Matching）の一種であり，無限長文脈を取り扱うことができる．ADD（ATR Dialogue Database）コーパスを用いた評価実験において，文字 $n$-gram モデルを用いた場合と PPM* モデルを用いた場合との比較を行った結果，PPM* モデルの方が高い単語分割精度を達成した．PPM* モデルを用いた本手法はオープンテストにおいて再現率 97.67%，適合率 98.27%を達成し，文字 $n$-gram モデルよりも頑健な未知語モデルとして機能した．

# A Japanese Word Segmenter Using a PPM*-based Language Model

Hiroki Oda[†,☆] and Kenji Kita[†]

Word segmentation, which segments an input sentence into words, is the most fundamental process of Japanese language processing. In this paper, we present a new method for segmenting the input sentence into words, which is suitable for those languages that have no delimiter between words, such as Japanese and Chinese. First, we present a word segmentation model using a character-based $n$-gram model, which is our baseline method. Next, we apply the PPM* compression algorithm to the problem of word segmentation. PPM (Prediction by Partial Matching) is a lossless compression algorithm based on a finite-context probabilistic modeling technique and PPM* is a variant of PPM, in which there is no *a priori* bound on context length. As the result of experiments on the ADD (ATR Dialogue Database) corpus, the proposed Japanese word segmenter using the PPM*-based language model marked a higher accuracy than the character-based $n$-gram model. In particular, the proposed method using the PPM*-based language model achieved 97.67% recall and 98.27% precision for open text.

## 1. はじめに

日本語や中国語等においては，単語間に空白を入れる習慣がないため，これらの言語の計算機処理では，まず文を単語列に分割する処理が必要となる．単語分割は日本語処理における最も基本的かつ重要な技術であり，精度・速度ともに高い水準の性能が要求される．単語分割と品詞付けから成る日本語形態素解析法の多くは，単語辞書の登録語との照合を行い，複数の形態素解析候補がある場合には，ヒューリスティクス（heuristics）を用いて候補間の順位付けを行うというものである．しかし，実際に，辞書中にすべての単語を網羅するのは不可能であるため，未知語（辞書未登録語）という重大な問題が生ずる．また，ヒューリスティクスでは扱うことのできない例外的な言語現象の存在や，例外現象に対処するための規則の複雑化が問題となる．その結果，一部の規則修正が全体に与える影響を人間が把握することが困難になり，規則の保守・管理に大きな労力を必要とすることとなる．

一方，英語の品詞付けでは，タグ付きコーパスを用いた確率的手法が確立されている[1)~3)]．言語表現の出現頻度に基づく確率的言語モデルを用いる方法には，対象領域のテキストからモデルのパラメータを学習する方法が存在するという大きな利点があり，タグ付きコーパスが整備されている領域では，実験的に最も高い精度が報告されている．英語の正書法は単語間で分かち書きするため，これらの手法は，単語モデル（word-based model）を用いている．英語の品詞付け

† 徳島大学工学部
Faculty of Engineering, Tokushima University
☆ 現在，NTT ソフトウェア株式会社
Presently with NTT Software Corporation

は，日本語の単語分割と技術的に似ているため，英語の品詞付け手法の多くは日本語の単語分割にも適用可能となる[4),5)]．しかし，単語モデルを日本語に適用する場合，未知語の存在が単語の同定にまで影響を与えるという英語の品詞付けにはない問題が生じてしまう．

実際に新語・造語は絶えず増え続けているため，以上の未知語問題は重大な問題となる．たとえば，新しい概念が記述された文書からその概念を学習することを考えた場合，文書を特徴付ける単語（専門用語）が新語である可能性は高い．ほかにも，話し言葉で書かれた文書を対象として解析を行う場合，断片的で不完全な表現や省略が問題となることも多い．情報検索等の分野では，そのような文書を対象とした場合でも，高い精度で単語の同定を行うことが望まれている．

以上の問題点から，本論文では，未知語に対して頑健な日本語単語分割手法として，文字モデル（character-based model）に基づく新しい手法を提案する．山本ら[6)]が指摘しているように，文字モデルには次のような利点がある．

(1) 日本語において，一般的に用いられる文字は約3,000 ～ 6,000 種類に限られる．これは，単語数に比べるとはるかに少ない．したがって，文字モデルでは，確率モデルのパラメータ数が少なくなり，頑健な推定を行うことができる．

(2) 漢字は表意文字であり，各文字が何らかの意味を担っている．さらに，平均単語長は約 2〜3 文字であるので，1 文字は単語に近い情報量を持つと考えられる．

(3) 文字モデルでは辞書を使用しない．したがって，未知語の概念そのものがなくなり，未知語問題を回避することができる．

文字モデルとしては，可変長 $n$-gram モデルの一種である PPM* モデル[7)]を用いる．その基礎となる PPM（Prediction by Partial Matching）モデル[8),9)]は，損失のない（lossless）データ圧縮アルゴリズムの一種であり，Canterbury Corpus を用いた実験で他の確率・統計的圧縮技法より優れた性能を達成している．データ圧縮は確率的言語モデルの最も直接的な応用であり，使用しているモデルのエントロピーが小さいほど，高い圧縮率を示す．実際に，PPM モデルの自然言語のモデル化への適用も報告されている[10)]．PPM* モデルは，PPM モデルを無限長の文脈を取り扱うことができるように拡張したものであり，事前にモデルの次数の上限値を決める必要がない．PPM* を自然言語のモデル化へ適用することにより，次数に上限のない可変長 $n$-gram モデルを構築することができる．

以下，本論文では，文字モデルに基づく日本語の単語分割手法を提案する．まず，本研究の基本法である文字 $n$-gram モデルとビタビ・アルゴリズム（Viterbi algorithm）から成る単語分割法を示す．さらに，文字モデルとして，文字 $n$-gram モデルに代わり，PPM* モデルを用いる単語分割法を提案する．ADD（ATR Dialogue Database）コーパスを用いた評価実験において，文字 $n$-gram モデルを用いた場合と，PPM* モデルを用いた場合の単語分割精度を比較し，提案した手法の評価を行う．

## 2. 文字 $n$-gram モデルに基づく単語分割法

本章では，文字 $n$-gram モデルに基づく単語分割法を提案する．この単語分割法が本研究の基本となる．文字 $n$-gram モデルでは，言語の文字生起は，$(n-1)$ 重マルコフモデルで近似される．長さ $l$ の文字列 $c_1^l = c_1 c_2 \cdots c_l$ において，直前の $(n-1)$ 文字のみが次の文字の生起確率に影響する．実際によく用いられるモデルは，$n=2$ あるいは $n=3$ のモデルであり，これらは bigram モデル，trigram モデルと呼ばれている．以下では，$n=3$ の文字 trigram モデルを用いることで，単語分割法の定式化を行う．

### 2.1 単語分割のモデル化

単語分割法の学習データとしては，単語境界位置の付与されたデータを用いる．**図 1** に学習データの例を示す．記号 ⟨d⟩ は単語境界を表す特殊記号であり，⟨s⟩ と ⟨/s⟩ はそれぞれ文頭と文末を表す特殊記号である．

文字列 $c_1^i = c_1 c_2 \cdots c_i$ に対して，単語境界を示す列 $d_1^i = d_1 d_2 \cdots d_i$ を考える．$d_k = 1$ は文字 $c_k$ の直前に単語境界があることを示し，$d_k = 0$ は直前に単語境界がないことを示すとする．単語境界が $d_1^i$ である文字列 $c_1^i$ が発生する確率 $p(c_1^i, d_1^i)$，すなわち，

$$\bar{d_k} = \begin{cases} \varepsilon\text{（空文字列）} & d_k = 0 \\ \langle \mathrm{d} \rangle & d_k = 1 \end{cases} \quad (1)$$

としたとき，$c_1^i$ に明示的に単語境界を入れた文字列

$$\bar{d_1} c_1 \bar{d_2} c_2 \cdots \bar{d_i} c_i \quad (2)$$

の発生確率を trigram でモデル化し以下のように定義する．ただし，文頭と文末での特殊性を考慮し，長さ $m$ の文に対しては，その 0 番目の文字位置，$m+1$ 番目の文字位置に文頭記号 ⟨s⟩ と文末記号 ⟨/s⟩ を加える☆．ここで，文末位置 1 および文末記号の直前には単語境界はないとしてモデル化すると，文字位置 1 と

☆ これを導入することにより，すべての可能な（単語境界を入れた）文字列の発生確率の和 $\sum p(\bar{d_1} c_1 \bar{d_2} c_2 \cdots \langle/\mathrm{s}\rangle | \langle \mathrm{s} \rangle)$ が 1 となることが保証される．

<s> も し も し <d> 、 <d> 通 訳 電 話 国 際 会 議 事 務 局 <d> で す <d> か <d> ？ </s>
<s> は い <d> 、 <d> そ う <d> で す <d> 。 </s>
<s> 会 議 <d> に <d> 申 込 み <d> た い <d> の <d> で す <d> が <d> 。 </s>

**図 1** 学習データの例
Fig. 1 Example of training data.

文字位置 2 における発生確率は次式で定義できる．

$$p(c_1, 0) = p(c_1|\langle \mathrm{s} \rangle) \tag{3}$$
$$p(c_1, 1) = 0 \tag{4}$$
$$p(c_1^2, 00) = p(c_1, 0)p(c_2|\langle \mathrm{s} \rangle c_1) \tag{5}$$
$$p(c_1^2, 01) = p(c_1, 0)p(\langle \mathrm{d} \rangle|\langle \mathrm{s} \rangle c_1)p(c_2|c_1\langle \mathrm{d} \rangle) \tag{6}$$
$$p(c_1^2, 10) = 0 \tag{7}$$
$$p(c_1^2, 11) = 0 \tag{8}$$

また，文字位置 $i \geq 3$ の場合は次式で定義する．

$$p(c_1^i, d_1^{i-2}00) = p(c_1^{i-1}, d_1^{i-2}0) \times p(c_i|c_{i-2}c_{i-1}) \tag{9}$$
$$p(c_1^i, d_1^{i-2}01) = p(c_1^{i-1}, d_1^{i-2}0)p(\langle \mathrm{d} \rangle|c_{i-2}c_{i-1}) \times p(c_i|c_{i-1}\langle \mathrm{d} \rangle) \tag{10}$$
$$p(c_1^i, d_1^{i-2}10) = p(c_1^{i-1}, d_1^{i-2}1) \times p(c_i|\langle \mathrm{d} \rangle c_{i-1}) \tag{11}$$
$$p(c_1^i, d_1^{i-2}11) = p(c_1^{i-1}, d_1^{i-2}1)p(\langle \mathrm{d} \rangle|\langle \mathrm{d} \rangle c_{i-1}) \times p(c_i|c_{i-1}\langle \mathrm{d} \rangle) \tag{12}$$

以上の発生確率の定義を用いて，文 $s = c_1^m$ の発生確率を最大とする単語境界列 $d_1^m$ を求めることが本論文の目的となる．

### 2.2 単語分割の解探索

文末記号の直前には単語境界はないとしてモデル化したことから，入力文 $s = c_1^m$ に対する最適な単語分割 $d_1^m$ は，

$$\underset{\delta \in \{0,1\}^m}{\operatorname{argmax}}\ p(c_1^{m+1}, \delta 0) \tag{13}$$

として求めることができる．これを求めるには，動的計画法の一種であるビタビ・アルゴリズム（Viterbi algorithm）を用いることができる（**図 2** 参照）．図中の文字位置 $i$ の状態 1 と状態 0 は，それぞれ，文字 $c_i$ の前が単語境界となる $(d_i = 1)$ か否か $(d_i = 0)$ を表す．このとき，文字位置 $i$ では，

(1) 確率 $p(c_1^i, \delta 0)$ を最大とする $\delta (\in \{0,1\}^{i-1})$ とそのときの確率値 $P_0(i)$，
(2) 確率 $p(c_1^i, \delta 1)$ を最大とする $\delta (\in \{0,1\}^{i-1})$ とそのときの確率値 $P_1(i)$

を求めればよい．(1) の $\delta 0$ を $\delta_0(i)$，(2) の $\delta 1$ を $\delta_1(i)$ で表すと，$P_0(i), \delta_0(i), P_1(i), \delta_1(i)$ は以下のようにして求めることができる．

$$P_0(1) = p(c_1, 0) \tag{14}$$
$$\delta_0(1) = 0 \tag{15}$$

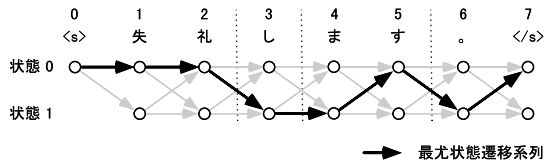



**図 2** ビタビ・アルゴリズムを用いた文の分割
Fig. 2 Segmentation of a sentence using the Viterbi algorithm.

$$P_1(1) = p(c_1, 1) \tag{16}$$
$$\delta_1(1) = 1 \tag{17}$$
$$P_0(2) = P_0(1)p(c_2|\langle \mathrm{s} \rangle c_1) \tag{18}$$
$$\delta_0(2) = 00 \tag{19}$$
$$P_1(2) = P_0(1)p(\langle \mathrm{d} \rangle|\langle \mathrm{s} \rangle c_1)p(c_2|c_1\langle \mathrm{d} \rangle) \tag{20}$$
$$\delta_1(2) = 01 \tag{21}$$

また，文字位置 $i \geq 3$ の場合は次式で定義する☆．

$$P_0(i) = \max(A_i,\ B_i) \tag{22}$$
$$\delta_0(i) = \begin{cases} \delta_0(i-1)0 & A_i \geq B_i \\ \delta_1(i-1)0 & A_i < B_i \end{cases} \tag{23}$$
$$P_1(i) = \max(C_i,\ D_i) \tag{24}$$
$$\delta_1(i) = \begin{cases} \delta_0(i-1)1 & C_i \geq D_i \\ \delta_1(i-1)1 & C_i < D_i \end{cases} \tag{25}$$
$$A_i = P_0(i-1)p(c_i|c_{i-2}c_{i-1})$$
$$B_i = P_1(i-1)p(c_i|\langle \mathrm{d} \rangle c_{i-1})$$
$$C_i = P_0(i-1)p(\langle \mathrm{d} \rangle|c_{i-2}c_{i-1})p(c_i|c_{i-1}\langle \mathrm{d} \rangle)$$
$$D_i = P_1(i-1)p(\langle \mathrm{d} \rangle|\langle \mathrm{d} \rangle c_{i-1})p(c_i|c_{i-1}\langle \mathrm{d} \rangle)$$

求められた $d_1^m$（図 2 中の最尤状態遷移系列）において，$d_k = 1$ である文字 $c_k$ の前で単語分割を行う．図 2 において単語境界を点線で示す．文字 trigram モデルを言語モデルとして用いた場合，以上の単語分割法により，入力文に対して最適な単語分割を求めることができる．

### 2.3 文字 $n$-gram モデルの関連研究との比較

文字 $n$-gram モデルを用いた関連研究としては，単語（あるいは未知単語のみ）を文字列によりモデル化する手法[11)~13)]や，文（単語列）をモデル化する手法[6)]等が提案されている．永井ら[11)]は文字に「単語の

☆ $\delta_0(i)$ と $\delta_1(i)$ において，$A_i = B_i$ および $C_i = D_i$ となる場合は，最長一致法等のヒューリスティクスを考慮して $\delta_0(i-1)$ を用いるよう定義した．

先頭文字であるか否か」と「単語の末尾文字であるか否か」の区切り情報を付与したものを予測単位としており，山本ら[6]は文字に「直後が単語境界であるか否か」の区切り情報を付与した拡張文字を予測単位としている☆.

本論文で提案する単語分割法は山本らの手法のように文をモデル化する手法であり，文字モデルのパラメータ数が少ない点に特徴がある．たとえば，文字unigramモデルを考えた場合，永井らの手法では異なり文字数 $N$ に対して，最大 $N \times 4$ のパラメータ数が存在し，山本らの手法は $N \times 2$ のパラメータ数が存在する．それに対して，我々の手法のパラメータ数は最大 $N+1$ でしかない（$+1$ は単語境界を表す $\langle \mathrm{d} \rangle$ による）．したがって，本手法は文字モデルを用いた手法の中でも特に小規模な学習データから統計的信頼性の高い確率値を推定しやすいという利点を持っている．

## 3. PPM* 言語モデルの適用

文字 $n$-gram モデルは，文字の生起を $(n-1)$ 重マルコフ過程により近似したモデルであり，文字の生起が過去の $(n-1)$ 文字に依存して決められる．ここで，単純に考えると，$n$ の値を大きくすればより精度の高いモデルが得られると考えられる．しかし，モデルの次数を大きくするにつれ，モデルのパラメータ数が指数的に増大し，必要な学習データの量を著しく増大させる結果となるため，学習データから統計的に信頼性の高いパラメータ値を推定することがますます難しくなる．

そこで，本章では，$n$-gram モデルの信頼性を損なうことなく，同時に精度を向上させるために，$n$ の値を可変にして，最適な長さの文脈を用いる可変長 $n$-gram モデル（variable-length $n$-gram model）を単語分割法の言語モデルとして用いる．

### 3.1 PPM* モデル

PPM*[7]は，データ圧縮アルゴリズム PPM[8),9)]を拡張したものであり，確率推定の条件部（文脈）として無限長文脈（無限の長さの文字列）を取り扱うことができる．PPM* モデルを自然言語のモデル化へ適用することで，モデルの次数の上限値という制約なしで，可変長 $n$-gram モデルを実現することができる．また，テキストデータ圧縮の場合には，テキスト中の1文字を読み込むごとに確率値が更新される動的なモデル（adaptive model）として用いられるが，本研究で用いる PPM* に基づく言語モデルは静的なモデルであり，学習データから獲得した確率値が，評価データによって更新されることはない．以下，PPM* モデルの確率推定法とデータ構造を，Cleary ら[7]の例を用いながら簡単に説明する．

#### 3.1.1 PPM* モデルの可変長文脈選択規則

PPM* モデルの基礎となる PPM モデルは，固定次数から次の文字を予測しようとするものであり，文字 $n$-gram モデルと同様に，単純に高次のモデルほど性能が良くなるというわけではない．実際，テキスト圧縮分野では，最高次数を 5（6-gram 以下の次数を文字予測に用いる）程度とした場合が，最も PPM モデルの性能が良いと報告されている[8),9),14),15)]．

本論文で用いる PPM* モデルは，固定次数のモデルから文字予測を行うという PPM の制限を取り払ったモデルであり，状況に応じて無限に文脈の長さを変化させることができる．PPM* モデルでは，状況に応じた最適な長さの文脈長を動的に決定するために，決定性文脈（deterministic context）という考えを導入している．決定性文脈とは，学習データにおいて，その文脈に後続する文字が一意に決められる文脈のことを指す．決定性文脈に対しては，未知文字が観測される回数は一様事前分布に基づく予測回数よりもはるかに低い（後続する文字が既知文字である確率が高い）ことが実験的に示されている[16)]．

ここで，単純に可変長モデルを実現することを考えたとき，以下の基準によって文脈の長さを変更すればよいことに注意しよう[17)]．

(1) 次数を大きくしたことによって，確率分布に大きな変化がみられる状況では，次数を大きくして次の文字を予測する．
(2) 次数を大きくしても，確率分布に大きな変化がみられない状況では，低い次数で十分である．

決定性文脈は，それ以上次数を大きくすることができても（より長い文脈が学習データ中に出現していても）学習データ中で次に観測される文字が1種類であることに変化はない．したがって，厳密には出現回数の違いから確率分布が変化する可能性はあるものの，決定性であるという簡単な情報から，それより次数を増やす必要はないと考えることが可能となる．

この決定性文脈の定義に基づき，PPM* モデルでは，次の規則に基づき文字予測の文脈を選択する．

(1) もし決定性の文脈があれば，その中から最短の文脈を選択する．
(2) もし決定性文脈がない場合は，非決定性文脈の中から最長のものを選択する．

☆ 実際には，文献6)では，文字に品詞情報と文字区切り情報を加えた拡張文字を用いることで品詞付けまで行っている．

**表 1** 予測に用いることのできる文脈と各々の確率値
Table 1 Contexts and each prediction probabilities.

| 予測 | | | $c$ | $p$ | 予測 | | | $c$ | $p$ |
|---|---|---|---|---|---|---|---|---|---|
| | 次数 $k=3$ | | | | | 次数 $k=0$ | | | |
| bra | → | c | 1 | 1/2 | → | a | 5 | 5/16 |
| | → | *Esc* | 1 | 1/2 | → | b | 2 | 2/16 |
| | | | | | → | c | 1 | 1/16 |
| | 次数 $k=2$ | | | | → | d | 1 | 1/16 |
| ra | → | c | 1 | 1/2 | → | r | 2 | 2/16 |
| | → | *Esc* | 1 | 1/2 | → | *Esc* | 5 | 5/16 |
| | 次数 $k=1$ | | | | | 次数 $k=-1$ | | | |
| a | → | b | 2 | 2/7 | → | $A$ | 1 | $1/\|A\|$ |
| | → | c | 1 | 1/7 | | | | | |
| | → | d | 1 | 1/7 | | | | | |
| | → | *Esc* | 3 | 3/7 | | | | | |

以上の規則により，状況に応じて動的に文脈の長さを可変として，後続する文字の生起確率を推定する．

### 3.1.2 PPM* モデルによる確率推定

PPM* モデルによる文字の生起確率計算について説明する．文字予測の際に直前の $k$ 文字を用いるモデルを次数 $k$ の有限文脈モデルと呼ぶ．PPM* は異なる次数の文脈モデルを組み合わせて文字予測を行う．各々の次数のモデルで，学習データ中にすでに出現した長さ $k$ の部分文字列とそれに後続する文字の出現回数 $c$，および，出現回数に基づく予測確率 $p$ を得る．

以下では例として，学習データ abracadabra から獲得した確率値を用いて文字列 bbra に後続する文字の確率を推定することを考える．この状況で，後続する文字を予測することのできる（すなわち，学習データに出現している）各文脈を**表 1** に示す．ここで，$A$ は対象言語の文字すべてを含む文字集合である．

3.1.1 項の文脈選択規則により選択された文脈が次の文字を予測できる場合，文字の生起確率として予測確率 $p$ を用いる．もし，次数 $k$ の文脈では予測できない未観測文字であった場合，その文脈におけるエスケープ確率（表中の *Esc*）を用いて，予測に用いる文脈を 1 つ次数の低い文脈に変更する．エスケープ確率とは各次数の文脈において観測されていない文字に対して割り当てられている確率であり，1 つ次数の低いモデルへ遷移する確率となる．PPM* モデルでは，このエスケープ確率の導入によって，確率値のスムージングを行っている．次の文字を予測できる文脈となるまで，文脈長を順次短くし，予測可能な文脈での確率分布に基づいて文字の生起確率を求める．ここで，どのような文字の生起確率を求める場合にも必ず処理が終了するように，対象言語の文字集合 $A$ 中のすべての文字を等確率 $1/|A|$ で予測する最低次 $k=-1$ のモデルを用意しておく．

以上の処理によって，PPM* モデルでは，各次数の確率分布を組み合わせて，後続する文字の生起確率を推定する．したがって，PPM* モデルの性能は，各次数の確率分布の混合方法，すなわちエスケープ確率の決定方法に依存する．エスケープ確率の推定法としては，いくつかの方法が提案されている[9),16),18)]．本研究では，その中の方法 C（Method C）と呼ばれる手法を用いた．方法 C を用いた PPM を PPMC と呼び，PPM* では実験的に PPM*C が最も良い性能を示している．

方法 C はエスケープ確率として，文脈 $c_{i-k}^{i-1}$ に未知文字が後続する確率

$$p(U|c_{i-k}^{i-1})\ ;\quad U=\{c|N(c_{i-k}^{i-1}c)=0\} \tag{26}$$

を以下のように推定する．ここで，$N(\alpha)$ は文字列 $\alpha$ が学習データ中に出現した回数である．方法 C では，学習データにおける後続する文字種の多い文脈ほど，その文脈に未知文字が後続する確率が高いという仮定に基づいている．すなわち，$U$ に属する文字が文脈 $c_{i-k}^{i-1}$ に後続する回数を，学習データにおいて $c_{i-k}^{i-1}$ に後続する文字の異なり数 $r(c_{i-k}^{i-1})$ と見積もり，

$$\tilde{p}(U|c_{i-k}^{i-1})=\frac{r(c_{i-k}^{i-1})}{n+r(c_{i-k}^{i-1})} \tag{27}$$

と推定する．ここで，$n=\sum_c N(c_{i-k}^{i-1}c)$ である．また，文脈 $c_{i-k}^{i-1}$ に（$N(c_{i-k}^{i-1}c_i)>0$ となる）既知文字 $c_i$ が後続する確率 $p(c_i|c_{i-k}^{i-1})$ は，

$$\tilde{p}(c_i|c_{i-k}^{i-1})=\frac{N(c_{i-k}^{i})}{n+r(c_{i-k}^{i-1})} \tag{28}$$

と推定する．PPM*C モデルでは $p(c_i|c_{i-k}^{i-1})$ の推定値 $\hat{p}(c_i|c_{i-k}^{i-1})$ を，これらを用いて，

$$\hat{p}(c_i|c_{i-k}^{i-1})= \begin{cases} \tilde{p}(c_i|c_{i-k}^{i-1}) & N(c_{i-k}^{i})>0 \\ \tilde{p}(U|c_{i-k}^{i-1})\hat{p}(c_i|c_{i-k+1}^{i-1}) & N(c_{i-k}^{i})=0 \end{cases} \tag{29}$$

とする．

たとえば，文字列 bbra に後続する文字を予測しようとする場合，2 つの決定性文脈（bra, ra）が発見できるため，最短（次数 2）の ra から次の文字を予測することとなる．文字 c が文字列 bbra に続いて出現する生起確率を推定する場合，決定性文脈 ra から文字 c が確率 1/2 で予測できるため，この状況における文字 c の生起確率は 1/2 となる．

文字列 bbra に続いて文字 d が出現する確率を推定する場合，最短 ($k=2$) の決定性文脈 ra からは d を予測できない ($N(\mathrm{rad})=0$) ため，1 つ低い次数 $k=1$ の文脈 a を用いる．この際，低次モデルへの遷

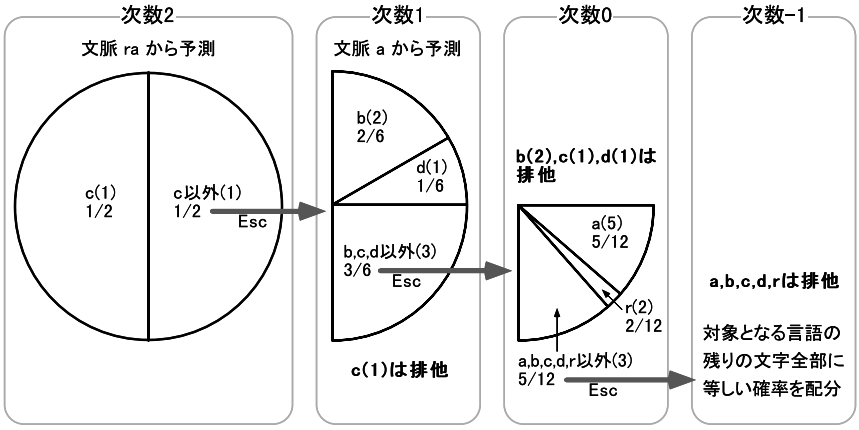



**図 3** PPM*C モデルの確率配分
Fig. 3 Probability distribution by the PPM*C model.

移確率として，文脈 ra におけるエスケープ確率 1/2 を用いる．文脈 a では，文字 d を確率 1/7 で予測できるので，文字 d の生起確率は，確率の積をとって $1/2 \times 1/7 = 1/14$ となる．

ここで，さらに文字 d の生起確率 1/14 をより正確な確率値に修正することができることに注意しよう．そもそも文字 c の生起確率を求めるのであれば，高次 $k = 2$ の文脈 ra から文字が予測されたはずであるので，文脈 a から文字 c を予測することはありえない．したがって，文脈 a における文字 c の予測確率を 0 へ修正する．この方法を排他（exclusion）と呼び，**図 3** に示すように，文脈 a における文字 d の予測確率を 1/6 へ修正する．このため，最終的に，文字 d の生起確率は 1/12 となる．

また，学習データ中に出現していない文字 t が，文字列 bbra に続いて出現する確率を推定する場合，次数 $k = 2$ から最低次 $k = -1$ まで遷移が繰り返されることになる．次数 $k = -1$ では，高次ですでに現れた文字の確率を排他する場合を除き，すべての文字を等確率で予測する．対象言語の文字集合を $A$ とすると，t は最低次 $k = -1$ において確率 $1/(|A| - 5)$ で予測される（図 3 参照）．したがって，文字 t の生起確率は次数 $-1$ に到達するまでの 3 度の遷移の確率との積をとって，$1/2 \times 3/6 \times 5/12 \times 1/(|A| - 5)$ となる．

### 3.1.3 PPM* モデルのトライ構造による表現

PPM* モデルを効率的に表現するデータ構造について簡単に説明する．無限長の文脈を実用的に用いるためには，記憶量と計算時間の問題を考慮する必要があり，確率推定に用いる文脈と出現回数（表 1 参照）

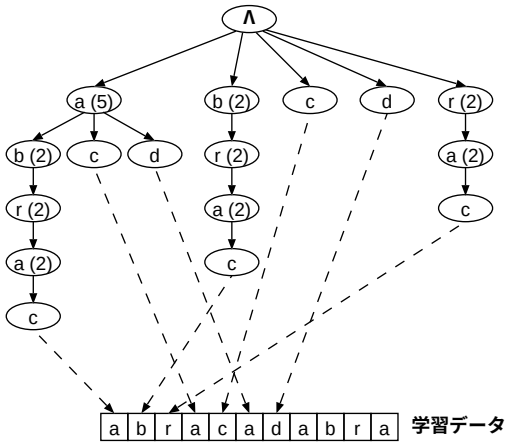



**図 4** 学習データ abracadabra に対する文脈トライ
Fig. 4 Context trie for the training data “abracadabra”.

を効率良く格納しておく必要がある．そこで，PPM* モデルでは，文脈とその出現回数をトライ構造を用いて格納する手法が提案されている[16]．これを文脈トライと呼ぶ．

例として，学習データ abracadabra に対する文脈トライを**図 4** に示す[7]．文脈トライの葉節点は，その葉節点の表す文脈が学習データ中に出現している文字位置へのポインタを持っている．このポインタを入力ポインタと呼ぶ．また，文脈トライ中の各節点には文脈の頻度が格納される．図中において，内点に格納される頻度を括弧内に示す．葉節点は学習データ中の出現位置を特定できるのであるから，頻度 1 である．葉節点の表す文脈のように一意的である（学習データ中に一度しか出現していない）場合，入力ポインタが指す学習データ中の出現部分を参照するだけで文脈に後

続する文字を調べることができる．

たとえば，図4中の左端部分では，a，ab，abr，abraはどれも一意的でなく，すべて入力文字列中に2回以上出現している．それに対して，abracは一意的である．したがって，トライ中の文脈abracに対応する葉節点cから入力文字列の先頭aへ入力ポインタを指す．これにより，文脈を長くする場合は，入力ポインタを動かすことのみが必要となる．

学習データから構築された文脈トライ☆を参照することで，前述したPPM*C モデルによる文字の生起確率の推定を行うことができる．図4には示していないが，当然，トライの各節点は子節点の数（方法Cにおけるエスケープ回数）を情報として持っている．たとえば，学習データ abracadabra から構築した文脈トライ（図4参照）を用いて，評価データ中の文字列bbraの次に文字cが出現する確率を求める場合，まず文字列bbraもしくはその接尾文字列（bra，ra，a）の中から最短の決定性文脈を探す．文字列が学習データ中に出現しているか否かはトライの根から走査し，文字列に相当する節点があるか否かで判定できる．上の4つの文字列の場合，文字列bra，ra，aは図4の文脈トライで受理できるが，文字列bbraは受理できない（次数1の節点bは子節点bを持たない）．文脈トライが文字列を受理できる場合，最後の文字に相当する節点がいくつの子節点を持つかを調べることで決定性文脈であるか否かを判定できる．3.1.2 項の確率推定の例で説明したように，この場合3つの文脈の中から最短の決定性文脈ra（図4中右端の次数2の節点）が発見できる．すなわち，文脈raに相当する節点の子節点は（文字cに相当する）1つのみであるから決定性文脈と判定できる．後続する文字cは葉節点に相当するため，文字cの頻度は1となり，方法Cに基づき文字cの生起確率を $1/(1+1) = 1/2$ と推定する．

また，決定性文脈が1つも発見できなかった場合は，3.1.1 項の文脈選択規則にあるように決定性でない文脈のうち最長のものから，PPM*C アルゴリズムを用いる．たとえば，文字列aaに後続する文字bの生起確率を求める場合，図4の文脈トライは文字列aaを受理できず，受理可能な文脈a（図4中左端の次数1の節点）は3つの子節点を持つため，この状況で一意の文字予測を行うことのできる文脈は学習データabracadabraにおいて観測されていないことが分かる．したがって，非決定性のうち最長の文脈aから後続する文字を予測する．この場合，次数1の節点aの3つの子節点に格納されている頻度の総和 $(2+1+1=4)$ をとることで，文字bの確率を $2/(4+3) = 2/7$ と推定する．

### 3.2 ビームサーチ法による単語分割の解探索

前章で提案した文字trigramモデルに基づく単語分割法よりも高い精度の単語分割を行うことを目的として，文字 $n$-gram モデルに代わり，PPM* モデルを文字モデルとして用いる．単語境界位置の付与された学習データ（図1参照）を用いて，日本語文字や単語境界等の特殊記号を予測単位とする PPM* モデルを作成する．その結果，作成されるトライを参照することで得られる文字の生起確率を用いて，単語分割を行う．

ここで，前章の文字trigramモデルを用いた場合では，どのような単語分割候補の確率を求める場合でも，文字位置 $i$ での生起確率に文字 $c_{i-2}$ 以前の文字列は影響を与えなかった．ゆえに，前章で提案したように，各文字位置 $i$ の直前と文字位置 $i-1$ の直前に単語境界があるか否かを条件とする 4つの確率を計算する単語分割法で，入力文に対する単語分割の最適解を容易に求めることができた．それに対して，本節では，PPM* モデルを言語モデルとして用いて最も高い確率値をとる単語分割候補を求める．前述したように，PPM* モデルは状況に応じて動的に条件部の文脈長が変化するモデルである．したがって，PPM* モデルでは，文字trigramモデルとは異なり，文字 $c_{i-2}$ 以前の文字列が文字 $c_i$ の生起に影響を与える可能性がある．

以上のように，PPM* モデルでは確率の条件部の長さが動的に変化するため，文字trigramモデルを用いた場合のように少ない計算量で最適解を求める単語分割法を定式化することが難しい．もちろん，全探索を行えば，PPM* モデルを用いた場合でも最適解を求めることはできる．単語分割は，1文中での処理であるため，無限長文脈を用いるからといって，条件部の先頭が，文頭記号より前の文字となることはない．また，入力文の先頭文字の直前（言い換えれば文頭記号の直後）と，文末記号の直前（最終文字の直後）は単語境界ではないとしていることから，入力文 $s = c_1^m$ に対して，各文字位置の前が単語境界であるか否かの条件分岐により，$2^{m-1}$ の単語分割候補の確率を比較すればよい（**図5** 参照）．しかし，一般には，多くの文は20文字以上から構成され，30文字以上で構成される文も珍しくない．文全体を処理する本手法では，

☆ トライの構築に関する詳細は文献7)において説明されている．また，実際には，高速化や記憶容量の節約のために，パトリシアトライのように，分岐のない節点を圧縮することや，各文脈から次数を1つ低くした文脈を指す接尾ポインタを追加する[7),15)]．

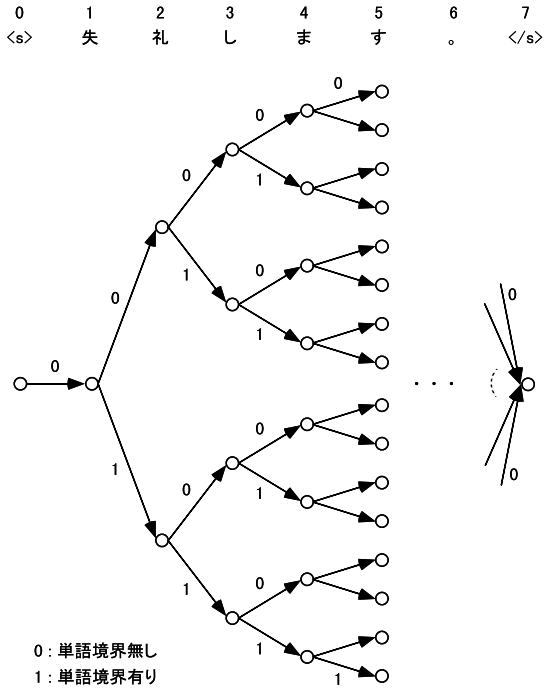



**図 5** 単語分割の解空間
Fig. 5 Solution space of word segmentation.

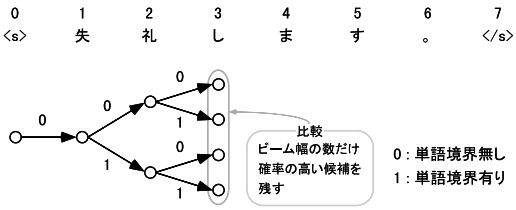



**図 6** ビームサーチ法の適用
Fig. 6 Example of the beam search.

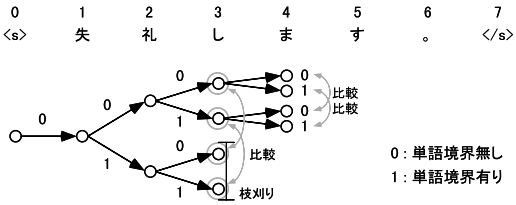



**図 7** ビームサーチ法による枝刈りの例
Fig. 7 Example of pruning by the beam search.

これらの文に対して，少なくとも $2^{19}$（30 文字の場合は $2^{29}$）以上の確率を計算し，それらを比較することとなる．このような長さの文となると，さらに文の長さが 1 文字増えるたびに指数的に単語分割候補が増加することは重大な問題である．極端な例ではあるが，実際のコーパスには 200 文字から成る文が含まれていた．以上のような長い文が入力される場合も含めて，全探索により解探索を行うことは，記憶容量や計算量の点から実時間での実行が困難となる．

以上の問題を解決するため，本論文ではビームサーチ法（beam search）を用い，尤度の高いパスだけを残し，尤度の低いパスの枝刈りを行うこととした（**図 6** 参照）．各文字位置までの単語分割候補のうち，確率の高い候補をビーム幅の数だけ残し，それらの候補のみを次の文字位置での解探索に用いる．単語分割候補の確率推定は，単語境界の有無を仮定した文字列から，PPM* モデルの表現であるトライを参照して，まず最短の決定性文脈（ない場合は，最長の文脈）を探し，PPM*C アルゴリズムに基づいて文字の生起確率を求める．

ここで，本論文で提案する単語分割手法は，入力文のすべての文字間に対して，単語境界があるか否かに相当する 2 値のどちらかを与える処理を行っていることを考えよう．単純に各文字位置までの文字列（単語分割候補）のうち，確率の高い上位解のみを後続する文字位置での探索に用いる場合（図 6 参照），ある文字位置 $i$ で，直前に単語境界があるか否かのどちらかの解のみが上位解となり，枝刈りによって，一方の可能性を消去してしまうことが考えられる．もし本来正解である方の解（$d_i = 1$ または $d_i = 0$）をすべて枝刈りしてしまった場合，可変長モデルであるために，過去の履歴での誤りが後の確率の推定に大きく影響を与えてしまう．以上の問題を回避するために，本論文では，前向き探索において，各文字位置の直前に単語境界があるか否かのどちらの可能性も残すために，2 つの仮定に対してそれぞれ別の枠で比較して枝刈りを行うこととする．すなわち，各文字位置の確率推定において，直前に単語境界がある場合とない場合について別個に 2 つのビーム幅を存在させる．このとき，各々のビーム幅を 1 とすると，その探索空間は図 2 に示している文字 trigram モデルを用いた場合と同じとなる☆（**図 7** 参照）．

入力文 $s = c_1^m$ の文末まで以上の処理を行い，文 $s$ に対する $\mathrm{argmax}_{\delta \in \{0,1\}^m} p(c_1^{m+1}, \delta 0)$（文末記号の前に単語境界がない状態に到達する最尤解）を求めることで，文 $s$ に対する単語分割の解を求めることがで

☆ 各々のビーム幅を 1 としても，文字 trigram モデルの場合は最尤解（確率最大の状態遷移）が得られることが保証されるが，PPM* モデルの場合は近似的な解しか求まらない（確率最大の状態遷移とは限らない）ことに注意する必要がある．しかし，それにもかかわらず，4.1 節での実験結果を見る限り，PPM* モデルによるビーム幅 1 の単語分割法の方が $n$-gram モデルによるビーム幅 1 の単語分割法より精度が高い．

**表 2** 学習データと評価データのサイズ
Table 2 Size of closed and open data.

| | 学習データ | 評価データ |
|---|---|---|
| 文数 | 10,000 | 2,697 |
| 単語数 | 141,658 | 31,724 |
| 文字数 | 251,699 | 58,522 |

きる．すなわち，得られた解 $d_1^m$ において，$d_k = 1$ となる文字 $c_k$ の前で単語分割を行えばよい．したがって，文字 trigram モデルを用いた場合と異なる点は，各々の文字位置までの単語分割候補がつねに 2 つであったのに対し，PPM* モデルではビーム幅の数だけ残すという点のみである．

## 4. 評価実験

以上で提案した単語分割法を評価するために，ADD（ATR Dialogue Database）コーパスを用いた評価実験を行った．クローズドテストには学習データを，オープンテストには学習データ以外の未知の評価データを用いた．それぞれのデータの文数，単語数，文字数を**表 2** に示す．

### 4.1 再現率と適合率による評価

再現率（*recall*）と適合率（*precision*）により単語分割の性能を評価する[12]．ここで，Std をコーパス中の単語数，Sys を本手法で分割された単語数，M を照合した単語数とすると，再現率と適合率は，recall = M/Std，precision = M/Sys で表される．

**表 3** に，PPM* モデルによる単語分割法と，文字 $n$-gram モデルによる単語分割法を適用した場合の単語分割精度を示す．本実験では，文字 $n$-gram モデルの確率はバックオフ・スムージング[19]付きの確率値を推定した．文字 $n$-gram モデル により単語分割を行う場合は，各文字位置で $2^{n-1}$ の分割候補を計算することで，ビタビ・アルゴリズムにより最適解を求めることができる．しかし，本実験では，高次 $n$-gram モデル ($n > 3$) と PPM* モデルによる単語分割では，各文字位置における 2 つの別枠のビーム幅を各々1 としたビームサーチ法により高速な解探索を行った．したがって，すべてのモデルにおいて，文字 trigram モデルと同じ探索空間とした場合の精度を求めている．

実験結果より，文字 $n$-gram モデルを用いる場合，文字 4-gram モデルの精度が最も高く，それ以上次数を大きくすることはかえって精度が低下していることが分かる．文字 $n$-gram モデルによる単語分割の精度も決して低いものではないが，PPM* モデルにより単語分割を行った結果の方が，はるかに精度が良い．単語分割の精度は，オープンデータでさえかなり高精度となっている．文字 trigram モデル以外は枝刈りによる近似解であったが，全体的に良い解が得られており，本論文の解探索法が有効であることが分かる．

**表 3** 単語分割法の精度
Table 3 Segmentation model performance.

| 言語モデル | クローズドテスト | | オープンテスト | |
|---|---|---|---|---|
| | 再現率 | 適合率 | 再現率 | 適合率 |
| 3-gram | 97.62% | 98.33% | 96.78% | 97.47% |
| 4-gram | 99.12% | 99.41% | 97.55% | 98.09% |
| 5-gram | 99.11% | 99.46% | 97.02% | 97.76% |
| 6-gram | 98.79% | 99.27% | 96.30% | 97.42% |
| PPM* | 99.69% | 99.79% | 97.67% | 98.27% |

本実験に用いた言語データは会話すなわち話し言葉であり，断片的で不完全な表現や省略が多いと考えられるが，本手法はそのようなデータを対象とした場合でも，高い精度で単語分割を行うことに成功している．また，今回用いた ADD コーパスの規模は小さいものであったにもかかわらず，文字 $n$-gram モデルおよび PPM* モデルともに，かなりの精度を達成している．これは，パラメータ数の少ない文字モデルの頑健性を示しており，未知語モデルとしても十分機能していると考えられる．文字モデルを言語モデルとして用いた場合，小さな学習データからでも，比較的簡単に信頼性のある確率値を獲得しやすいという点が，本手法の利点の 1 つといえる．

本実験では，未知語処理の機能を含む単語ベースの従来法を実現しなかったため，従来法との単語分割精度の直接比較は行わなかった．本論文の手法では未知語の概念そのものがないため，従来法よりも未知語解析力については優れていると考えられる．さらに，本手法の利点として計算量の問題がある．従来法では単語単位の解析を行うために，入力文の各文字位置に対して既知単語数分の探索（辞書登録語との照合）を行う必要がある．それに対して，本論文の手法は各文字位置で基本的に 2 つの可能性（境界が生じるか否か）に関する探索を行えばよい．たとえ従来法の解探索において枝刈りを行うにしても，解探索にかかる計算量ははるかに本手法の方が小さい．もちろん，PPM* モデルを用いた場合は文脈同定処理が必要となるため，かならずしも上記の計算量のみで高速とはいえないが，その問題については 4.3 節で後述するようにオートマトン変換によって解決できる．

### 4.2 単語分割の解探索に関する考察

本実験では，PPM* モデルを用いた場合の単語分割の解探索において，各文字位置に至る（単語境界があるか否かで区分けせずに比較した）上位解に関して続けて探索する方法（図 6 参照）に関しても単語分

割実験を行った．表3の結果は，各文字位置でビーム幅 1×2 の解を残して枝刈りを行った（図7参照）ので，条件を統一するために各文字位置でのビーム幅を2として解探索を行ったところ，単語分割精度は 0.5〜1%低下した．実験結果の誤り文を見比べたところ，別枠扱いをせずに単一のビーム幅を適用した場合は，ある文字位置で単語境界がないとする候補ばかりが上位解として続けて探索される傾向にあった．ゆえに，正解では分かち書きすべき文字間での単語境界を消失した状態で解探索を行っている場面が多かった．逆に，別個に2つのビーム幅を用いた場合は，分かち書きしようとする傾向が強く，正解では分かち書きしない文字間で分かち書きする場面があった．したがって，最長一致法で解が求まるような場合には前者の方法の方が正解となりやすいと考えることができる．しかし，実際には，そのような状況は少なく，逆に明らかに分かち書きすべき文字間で誤った解を求めてしまう場面が単語分割精度へ深刻な影響を及ぼしていた．

別枠扱いで 2 つのビーム幅を用いる場合には正解が得られるが，単一のビーム幅を適用した場合は誤った解となってしまう場合でも，後者のビーム幅を広くして探索空間を広げれば正解を求めることができるようにはなる．正解が解空間に存在する以上，その逆の場合も同様のことがいえる．しかし，本実験の結果により，少なくとも，小さなビーム幅で高速に単語分割を行う場合には，別枠で 2 つのビーム幅を用意した方が高精度であることを示すことができた．したがって，本論文で提案したように PPM$^*$ モデルおよび高次 $n$-gram モデルを用いた場合の探索では，各文字位置で直前に単語境界があるか否かの可能性をつねに残した状態で次の文字位置での探索に進む方法が有効であると結論できる．

### 4.3 確率推定における計算量増加への対応

本実験においては，PPM$^*$ モデルを用いた場合，文字 $n$-gram モデルを用いた場合と比較して計算時間が増加した．次数が無限に増える可能性がある以上，計算時間が増加することに関しては，ある程度やむをえない面があると思われる．

ここで，解空間が広大になったことのみの理由で，計算量が増大したのではないことには注意する必要がある．解探索にかかる計算量であれば，実際に本実験で行ったようにビーム幅を小さくすることで，計算量を軽減することができる．計算時間が増加したもう1つの原因は確率推定時の計算量である．PPM$^*$ モデルでは，可変長モデルの確率を求めるためにトライを用いて予測に用いる文脈を同定している．実際に確率推定を行う回数を考えると，この文脈同定処理はかなりの計算量となることが考えられる．可変長マルコフモデルの研究においては，構築された予測接尾木をオートマトンに変換することで，履歴の同定処理を不要としている[17]．$n$-gram モデルにおいては，オートマトンの状態は固定長 $(n-1)$ の文字列でラベル付けされるが，可変長モデルでは任意の長さの文字列を状態のラベルとして用いればよい．ここで，PPM$^*$ モデルの表現である文脈トライの各節点は，1つの文脈に相当するので，1つの状態と考えることができる．PPM$^*$ モデルのオートマトン変換を定式化できれば，少なくとも，確率推定時の計算量の増加を抑制することができる．確率推定時の計算量を減らすための改良を行うことは今後の課題であり，計算時間の比較に関する実験は，改めて行う予定である．

### 4.4 言語モデルの評価

提案した単語分割法のうち，PPM$^*$ モデルを言語モデルとして用いた方法が，文字 $n$-gram モデルを用いた場合よりも，高精度であることを実験において示すことができた．ここでは，さらに，純粋に言語モデルとしての評価を行うことを考える．言語モデルの性能尺度であるクロス・エントロピー $H$ は以下の式で定義される．

$$H(M,T) = -\frac{\sum_{i=1}^{n} \log p_M(s_i)}{\sum_{i=1}^{n} |s_i|} \qquad (30)$$

ここで，$M$ は言語モデル，$s_i$ は評価データ $T$ 中の $i$ 番目の文である．クロス・エントロピーの計算における分母の $|s_i|$ は文 $s_i$ を構成する文字の数となる．また，言語モデル $M$ による文 $s_i$ の生起確率 $p_M(s_i)$ は，$p(c_1c_2\cdots\langle/\mathrm{s}\rangle|\langle\mathrm{s}\rangle)$ とする．したがって，$|s_i|$ は文区切りとして文末記号までを含めた文字数となる．

言語モデルとしての比較実験においても，学習データと評価データともに，(分かち書きのスペースに相当する）単語境界記号を挿入した「分かち書き」日本語文とした．したがって，クロス・エントロピーは単語境界（単語間のスペース）まで考慮した値となる．ADD コーパスでの実験結果を**表4**に示す．また，**図8**に，クロス・エントロピー計算時の PPM$^*$ モデルの確率推定において，文字予測の開始次数（最短の決定性文脈もしくは最長の非決定性文脈）の統計を示す．

表4から，文字 $n$-gram モデルでは，$n=5$ の場合が最も1文字あたりのクロス・エントロピーが小さく，さらに次数を大きくすることは，かえってモデルの予測力を低下させていることが分かる．PPM$^*$ モデルは，文字 5-gram モデルよりもクロス・エントロピーを小さくすることに成功しており，予測力という

**表 4** 各言語モデルにおけるクロス・エントロピー
Table 4 Cross entropy by each language model.

| 言語モデル | $H$ |
|---|---|
| 文字 3-gram モデル | 2.3978 |
| 文字 4-gram モデル | 2.1205 |
| 文字 5-gram モデル | 2.0585 |
| 文字 6-gram モデル | 2.0774 |
| PPM* モデル | 1.9904 |

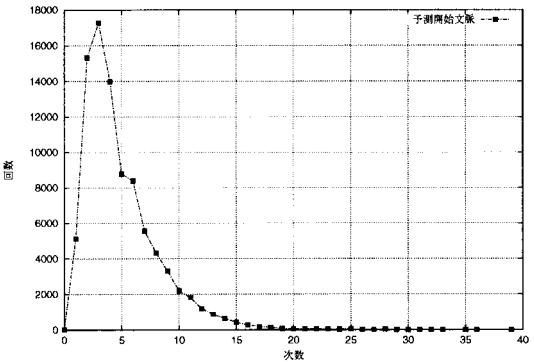



**図 8** PPM* モデルの確率推定で最初に選択した次数の統計
Fig. 8 Histogram of chosen orders in the PPM* model.

点で最も良い性能を持っている☆.

本実験の結果により，固定次数のマルコフモデルの予測力の限界を示すとともに，PPM* モデルにより，状況に応じて動的に文脈長を変化させることで，より精度の高い言語モデルを構築できることを示した．図 8 に示す次数の統計は，最終的に文字の確率を推定した次数ではないが，状況に応じて文脈長が変わり，様々な長さの文字列を予測に用いることを試みていることが分かる．たとえば，20 以上の長さの文脈を用いていることもあることから，定型的な表現等では，長い文字列から（ある程度決まりきった）次の文字を予測していると考えられる．2.3 節で述べたような文字モデルによる関連研究の多くは bigram モデルや trigram モデルを用いているが，本実験により，本論文で用いた PPM* モデルは文字 $n$-gram モデルよりも高い予測力を持つ言語モデルであることを示した．

## 5. おわりに

本論文では，日本語のような単語間で分かち書きをしない言語のための新しい単語分割法を提案した．入力文に対して最適な単語分割を見つけるために，本手法は確率的な文字言語モデルを用いる．ADD（ATR Dialogue Database）コーパスを用いた評価実験で，文字 $n$-gram モデルを用いた場合と，PPM* モデルを用いた場合の単語分割精度の比較を行い，PPM* モデルによる単語分割法の方が優れていることを示した．

本論文では，PPM*C モデルを用いたが，PPM には様々な改良モデルが提案されている[15]．今後は，本論文で用いた PPM* モデル等の PPM に基づくモデル（PPMD モデル，PPMZ モデル等）の自然言語のモデル化への適用とその比較評価を行い，その改良や応用に関する研究を行う予定である．特に，PPM* モデルの表現である文脈トライからオートマトンへの変換に関して定式化を行うことを考えている．

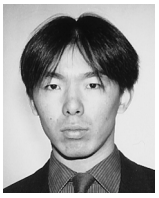
☆ 学習データと評価データともに「ベタ書き」文として単語境界（単語間のスペース）を考慮しない場合のクロス・エントロピーに関しても比較実験を行ったが，やはり PPM* モデルが最も小さい（1 文字あたり 2.9203 ビット）という結果が得られた．

## 参考文献

1) Church, K.W.: A stochastic parts program and noun phrase parser for unrestricted text, *Proc. 2nd Conference on Applied Natural Language Processing*, pp.136–143 (1988).
2) Cutting, D., Kupiec, J., Pedersen, J. and Sibun, P.: A practical part-of-speech tagger, *Proc. 3rd Conference on Applied Natural Language Processing*, pp.133–140 (1992).
3) Charniak, E., Hendrickson, C., Jacobson, N., and Perkowitz, M.: Equations for part-of-speech tagging, *Proc. 11th National Conference on Artifical Intelligence*, pp.784–789 (1993).
4) 永田昌明：形態素解析，岩波講座「言語の科学」第 3 巻「単語と辞書」，pp.53–92 (1997).
5) 北 研二：確率的言語モデル，東京大学出版会 (1999).
6) 山本幹雄，増山正和：品詞・区切り情報を含む拡張文字の連鎖確率を用いた日本語形態素解析，言語処理学会第 3 回年次大会発表論文集，pp.421–424 (1997).
7) Cleary, J.G. and Teahan, W.J.: Unbounded length contexts for PPM, *Computer Journal*, Vol.40, No.2, pp.67–75 (1997).
8) Cleary, J.G. and Witten, I.H.: Data compression using adaptive coding and partial string matching, *IEEE Trans. Communications*, Vol.32, No.4, pp.396–402 (1984).
9) Moffat, A.: Implementing the PPM data compression scheme, *IEEE Trans. Communications*, Vol.38, No.11, pp.1917–1921 (1990).
10) Teahan, W.J. and Cleary, J.G.: The entropy of English using PPM based models, *Proc. Data Compression Conference*, pp.53–62, IEEE Society Press (1996).
11) 永井秀利，日高 達：日本語における単語の造語

モデルとその評価，情報処理学会論文誌，Vol.34, No.9, pp.1944–1955 (1993).
12) Nagata, M.: A stochastic Japanese morphological analyzer using a forward-DP backward-$A^*$ N-best search algorithm, *Proc. 15th International Conference on Computational Linguistics*, pp.201–207 (1994).
13) 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, pp.75–103 (1998).
14) Bell, T.C., Cleary, J.G. and Witten, I.H.: *Text compression*, Prentice Hall, NJ (1990).
15) Bunton, S.: Semantically motivated improvements for PPM variants, *Computer Journal*, Vol.40, No.2, pp.76–93 (1997).
16) Teahan, W.J.: Modeling English text, PhD Thesis, University of Waikato, New Zealand (1997).
17) Ron, D., Singer, Y. and Tishby, N.: The power of amnesia: Learning probabilistic automata with variable memory length, *Machine Learning*, Vol.25, pp.117–150 (1996).
18) Witten, I.H. and Bell, T.C.: The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression, *IEEE Trans. Information theory*, Vol.37, No.4, pp.1085–1094 (1991).
19) Katz, S.M.: Estimation of probabilities from sparse data for the language model component of speech recognizer, *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol.ASSP-35, No.3, pp.400–401 (1987).

(平成 11 年 3 月 18 日受付)
(平成 11 年 12 月 2 日採録)



**小田　裕樹**（正会員）

1975 年生．1997 年徳島大学工学部知能情報工学科卒業．1999 年同大学大学院博士前期課程修了．同年，NTT ソフトウェア（株）入社．在学中，確率・統計的自然言語処理の研究に従事．現在，自然言語処理，情報検索等の研究開発支援に従事．言語処理学会会員．

**北　　研二**（正会員）

1957 年生．1981 年早稲田大学理工学部数学科卒業．1983 年から 1992 年まで沖電気工業（株）勤務．この間，1987 年から 1992 年まで ATR 自動翻訳電話研究所に出向．1992 年 9 月から徳島大学工学部勤務．現在，同助教授．工学博士．自然言語処理，情報検索等の研究に従事．言語処理学会，電子情報通信学会，日本音響学会，日本言語学会，計量国語学会，ACL 各会員．