TR-IT-0011

バイグラムを用いた日本語形態素解析における各種探索法の比較検討

三木 清一† 田代 敏久 竹沢 寿幸
Kiyokazu MIKI† Toshihisa TASHIRO Toshiyuki TAKEZAWA

1993. 8. 31

内容梗概

形態素解析における統計的手法の一つである、バイグラムを用いた形態素解析においては、解析するという問題は木の探索問題に帰着できる。今回、バイグラムを用いた形態素解析を、best-first サーチ、beam サーチ、ordered サーチのそれぞれで実現し、比較検討した。実験では、比較的小規模な辞書及びバイグラムを用い、その上で、どのような探索手法が適当であるかを実際に実行してみることで比較検討した。各探索手法を検討した結果、本モデル及び本実験の規模のもとでは正解率及び効率の観点から見て、ビームサーチが適当であると考えられる。

ATR 音声翻訳通信研究所
ATR Interpreting Telecommunications Research Laboratories
†京都大学工学部情報工学教室

目次

1　はじめに

形態素解析は自然言語処理において重要な処理のひとつである。従来、事前に与えた知識（一般にはルール）をもとに形態素解析を実現するというアプローチが主流であった。このような手法において問題となるのは、例えば例外となる知識（事実）が存在するなど、全ての場合で矛盾がないような完全なルールを人手で記述するのが困難であるということである。

これに対し、近年、大規模なコーパスが利用可能となった、また、計算機のメモリ空間が増大し、計算能力が向上したという状況のもと、形態素解析を行なうのに多くの例題から統計的な知識を自動的に抽出し、利用するアプローチが注目されてきている。

今回、このようなアプローチに基づくものであるバイグラムを用いた形態素解析について、実験及び考察を行なった。

以下、本報告書では、まず2節においてバイグラムを用いた形態素解析の手法の概要について述べ、形態素解析という問題が、本手法では木の探索問題に帰着できることを述べる。次に3節において一般的にどのような探索手法が存在し、それぞれどのような得失があるかを述べる。4節では、比較的小規模な辞書及びバイグラムを用いて実際に実験を行なった結果を示し、それをもとに考察を行なう。最後に、5節で今回の成果及び今後の課題について述べる。

2　バイグラムを用いた形態素解析

本実験で用いるバイグラムとは、あるトークンからあるトークンへの遷移について、その2つのトークンと（条件付）遷移確率の3つ組を記述したものである。このようなバイグラムを大規模コーパスから自動的に得ることは基本的には容易である。バイグラムを用いて形態素解析を行なうということを、（日本語）文字列及びバイグラムが与えられた時、与えられた文字列を生成する、最大尤度のトークン列を得ることと等価であるとみなす。図1にバイグラムを用いた形態素解析の例を示す。

前述の例のように尤度を求めた場合、まず、長さについての正規化の問題がある。つまり、トークン数が少ないトークン列ほど、全体の尤度が高くなるという問題である。しかし経験的に、形態素解析ではトークン数が少ない場合というのは、一般によい場合が多いため、今回正規化は行なわない。また、確率の積として尤度を求めると、アンダフローの問題が生じる可能性がある。これは、確率の対数をとってその和を尤度とすることで解決できる。

形態素解析をこのように定式化することにより、形態素解析の問題は、全ての可能なトークン列の中から最尤のトークン列を探す探索問題に帰着できる。以下、この可能なトークン列の全体のことをラティスと呼ぶ。ラティスは木で表現できる。ラティスの例を図2に示す。

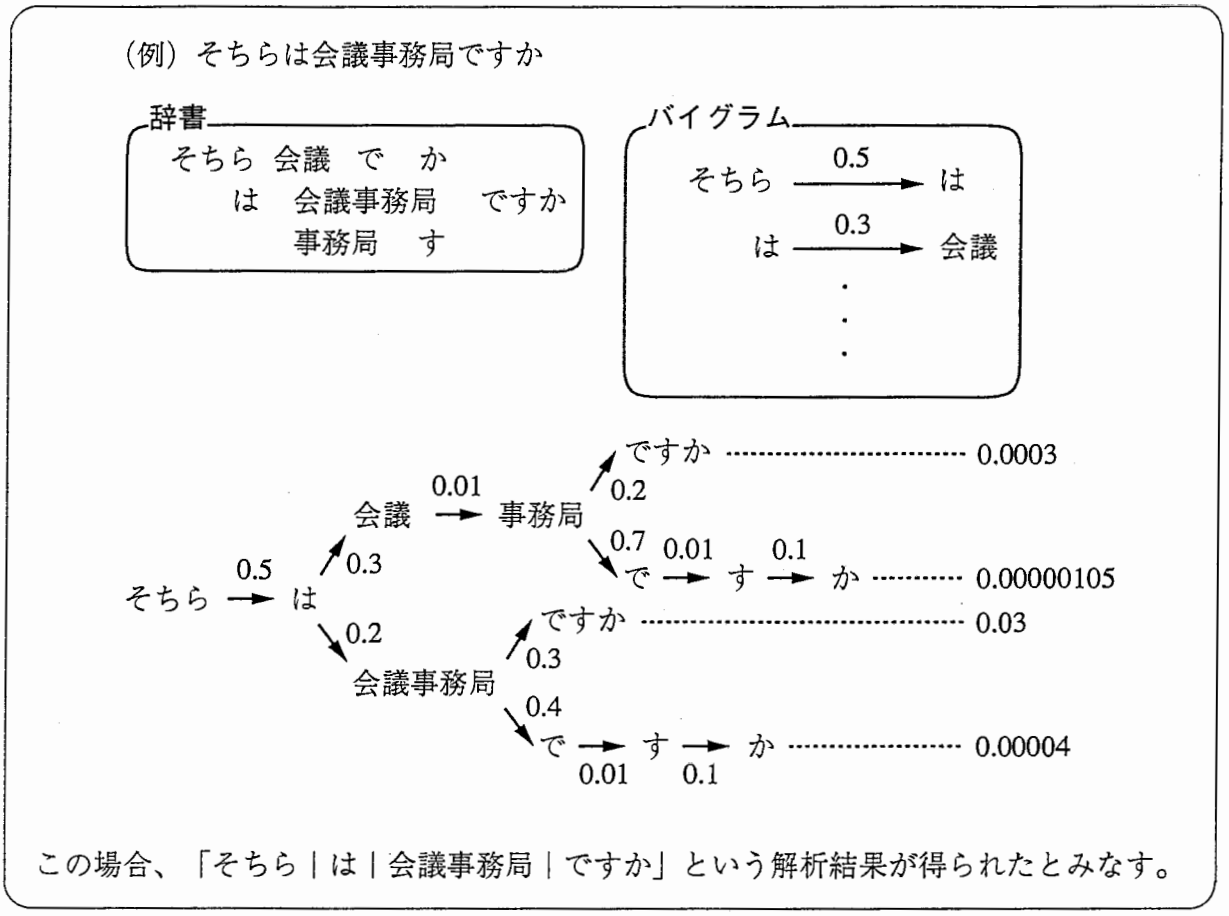

図 1: バイグラムを用いた形態素解析

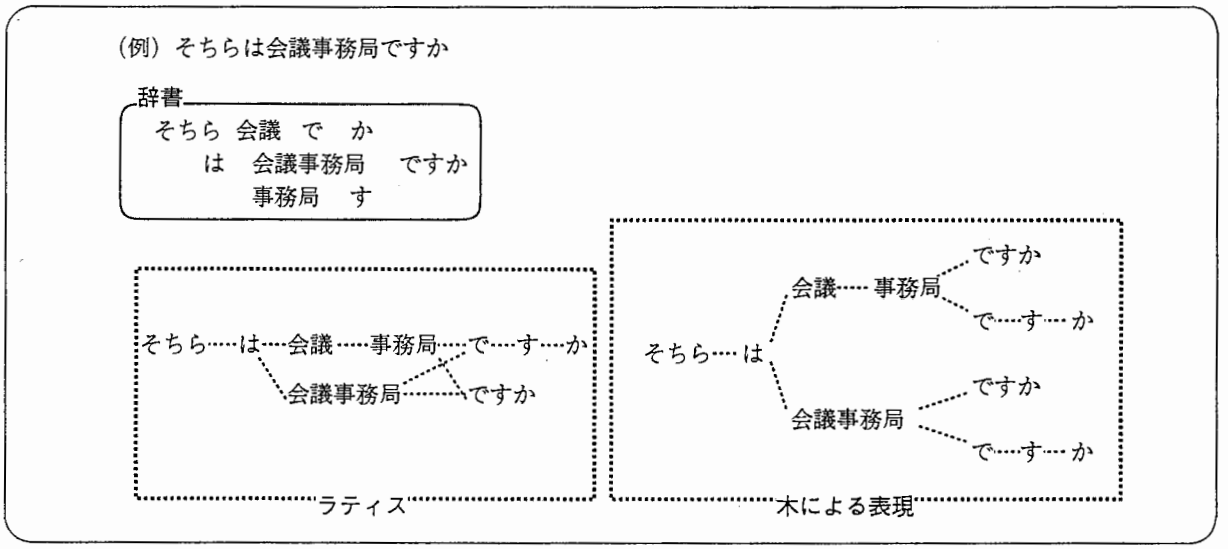

図 2: ラティスとそれの木による表現

3 探索手法

前節で述べたように、（バイグラムをもとにした）尤度を導入したことで、形態素解析をラティス（木）の探索問題に帰着できる。本節では、一般的にどのような木の探索手法があり、それぞれどのような得失があるかを述べる。本節で述べるのは、best-first サーチ、beam サーチ、ordered サーチ（A^* サーチ）である。

3.1 best-first サーチ

best-first サーチは縦形探索の一種で、あるノード（ノード A とする）から次のノードを選ぶ時に、ノード A の子ノードの中で最もスコアの高いノードを選ぶものである。図3の木を例にして探索時の動作を示す。

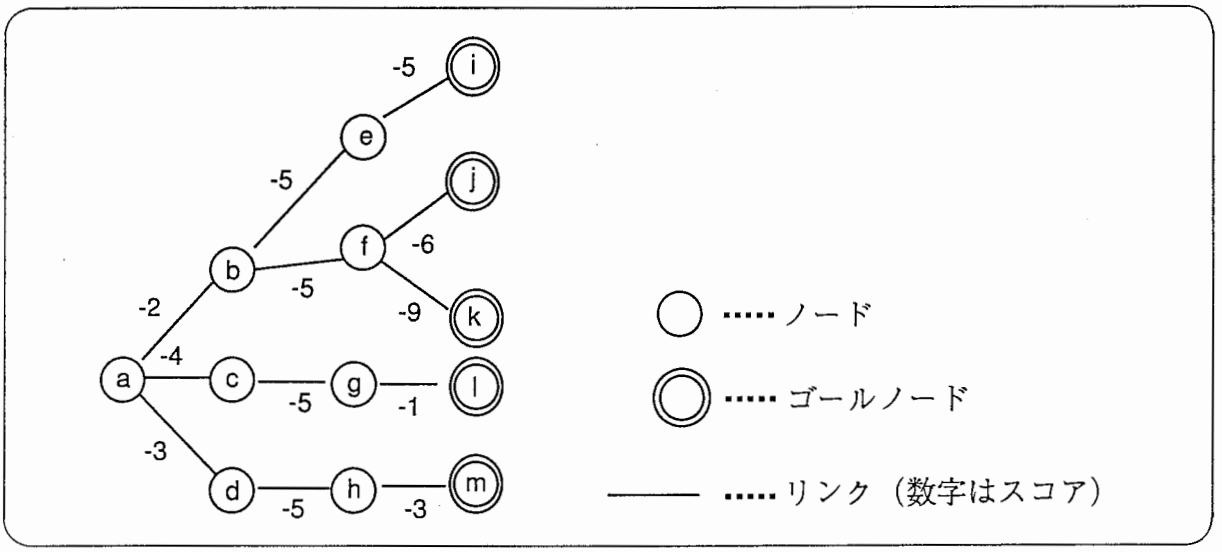

図 3: best-first サーチ

ノード a から探索を始めるとする。まず、ノード a の子ノード b,c,d を展開する。この展開の操作は、一般に実装時にはスタック（リスト）に入れる操作となるため、以下本報告書ではノードの「展開」と「プッシュ」という言葉をほぼ同義で用いることとする。展開された（あるいはプッシュされた）子ノードの中から、最もスコアの高いノード b を選ぶ（前述の「プッシュ」と同じ理由で、「選択」のことを「ポップ」とも呼ぶ）。同様に、ノード f（ここでは e も同じスコアであるが f を選んだとする）、ノード j が選ばれ、解がひとつ得られる。この解は「$abfj$」なる系列で、そのスコアは -13 である。この解が最尤解である保証は全くない。例えば、図にあるように、「$acgl$」なる系列はスコア -10 を持ち先程の解よりも高いスコアを持つが、選ばれない。

一般的にこの手法では、解の長さに比例した時間で解を得ることができ、展開ノード数も少ない。しかし最尤解は保証されず、初期段階で誤った場合に訂正する手段がない。

3.2 beam サーチ

beam サーチは横形探索の一種で、各レベル（深さ）において一定数（ビーム幅という）のノードしか残さないやり方である。図4の木を例にして探索時の動作を示す。

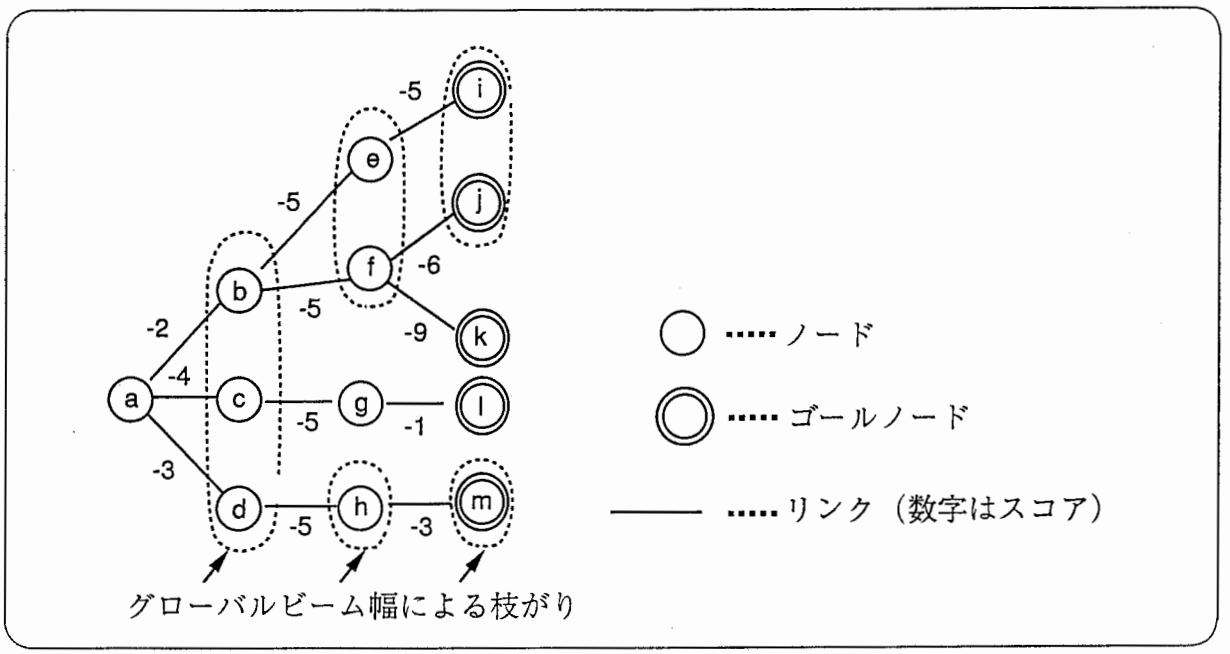

図 4: beam サーチ

ノード a から探索を始めるとする。まず、ノード a の子ノードを展開するのであるが、この時点（すなわち子ノードを展開する際）で枝がりを行なう場合がある。このときに残すノードの個数をローカルビーム幅と呼ぶ。この例では、ローカルビーム幅を 3 とする。従って、a の子ノード (b,c,d) は全て展開される。beam サーチは基本的に横形探索であるので、同じ深さ（レベル）のノードは同時に展開される。今、a と同じレベルのノードは他にないので、子ノードの展開はひとまず終る。再びここで（すなわち同レベルのノードからの子ノードが全て展開された後）、現在展開されているノード b,c,d について、枝がりを行なう場合がある。このときに残すノードの個数をグローバルビーム幅と呼ぶ。この例では、グローバルビーム幅を 3 とする。従って、現在展開されているノード b,c,d は全て残される。次に、e,f,g,h が展開され、グローバルビーム幅 3 により、（累積）スコアの高い方から、e,f,h が残される。以下同様の操作で、解候補「$abei$（スコア -12）」、「$abfj$（スコア -13）」、「$adhm$（スコア -11）」が同時に得られ、結果、「$adhm$」を解とする。この解が最尤解である保証はない。例えば、図にあるように、「$acgl$」なる系列はスコア -10 を持ち先程の解よりも高いスコアを持つが、選ばれない。

一般的にこの手法では、ビーム幅及び解の長さに比例した時間で解が得られ、ビーム幅及び解の長さに比例した展開ノード数になる。最尤解は保証されない。また、基本的に横形探索であるため、問題の性質によらず、解を求めるのに常に一定の時間（ビーム幅と解の長さに比例した値）がかかる。

3.3 ordered サーチ

ordered サーチでは展開すべきノードそれぞれに評価値がついており、その中で最も高い評価値を持つノードを展開する。図5の木を例にして探索時の動作を示す。

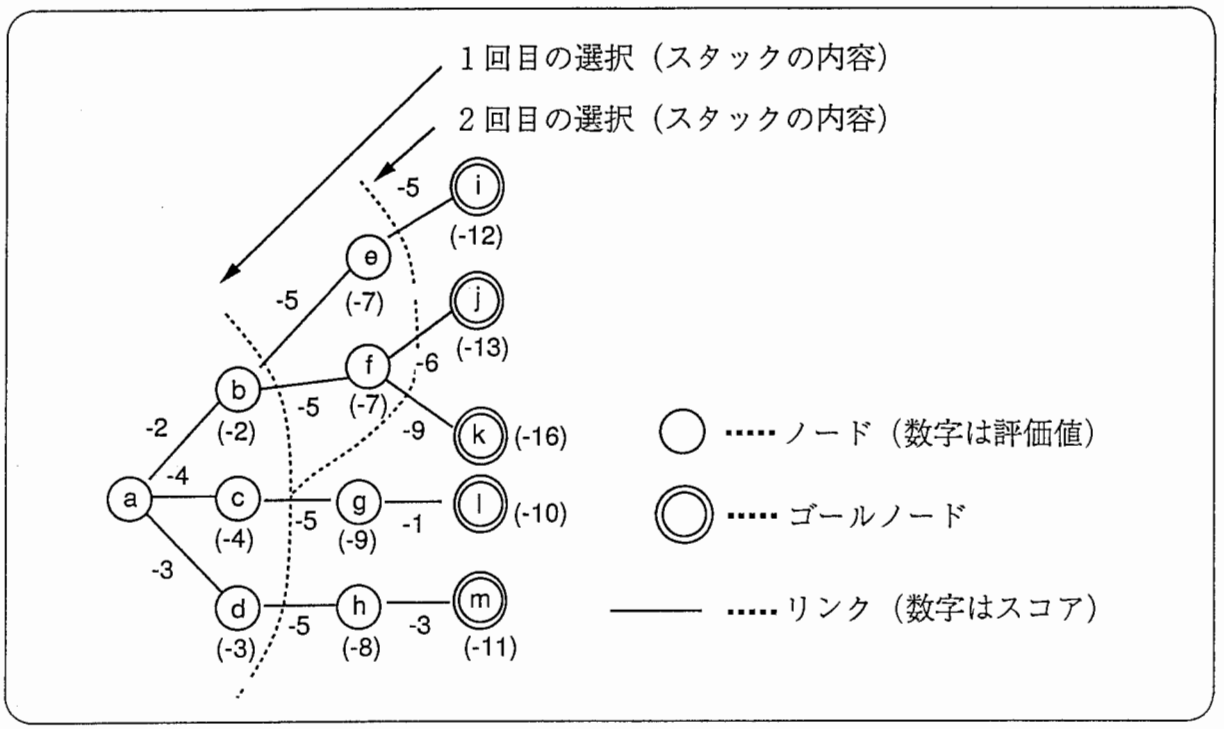

図 5: ordered サーチ

ノード a から探索を始めるとする。まず、ノード a の子ノードを展開する。次に現在展開されているノード b,c,d について、それぞれの評価値を調べ、最も大きい評価値を持つノードを選ぶ。この例では評価値はそのノードの持つ累積スコアとしてある。上に従い、ここではノード b が選ばれる。ノード b の子ノードを展開する。次に現在展開されているノード e,f,c,d について、それぞれの評価値を調べ、最も大きい評価値を持つノードを選ぶ。ここではノード d が選ばれる。以下同様の操作で、解として「$acgl$（スコア -10）」が得られる。これは最尤解となっている。

ordered サーチでは評価値を求める際にヒューリスティックを用いることができる。その中で、ヒューリスティックが一定の条件を満たすものを A^* サーチと呼ぶ。A^* サーチでは最尤解が保証される。ヒューリスティックを用いない場合（上の例）にも最尤解は保証されるが、一般的に盲目的な横形探索に近くなる。従って、最悪の場合、全探索と同程度の効率になる可能性がある。最もよい場合には best-first サーチと同程度の効率になりうる。ヒューリスティックが適当な場合には大抵の場合で縦形探索に近くなり、少ない時間で最尤解を得ることができる。

4 実験結果

本節では、前節で示したような探索手法を実際にバイグラムを用いた形態素解析に適用した結果を示す。

4.1 実験条件

本実験では見出し語とその品詞名を辞書項目とする辞書を使用する。また、バイグラムとして、見出し語間の遷移確率を記述したもの（以下、単語バイグラムと呼ぶ）及び品詞間の遷移確率を記述したもの（以下、品詞バイグラムと呼ぶ）を用いる。バイグラム中には特別なエントリとして「禁則」を導入する。これは絶対的に遷移確率が0であるような組である。これは人間が事前に与える。このことにより、本実験は純粋な統計的アプローチに基づくとはいえなくなるが、効率を考え導入した。スコア計算法を次に示す。見出し語 W 、品詞 C なるトークン A の次に見出し語 W 、品詞 C なるトークン B が続くとしたとき、そのスコアは、

$$\text{score} = a \times \ln(prob(W \rightarrow W)) + b \times \ln(prob(C \rightarrow C))$$

となる。ただし $prob(W \rightarrow W)$ は単語バイグラムに記述されている W から W への遷移確率とする（$prob(C \rightarrow C)$ も同様）。ただし、バイグラム中に記述がない遷移の場合、$\frac{1}{(\text{バイグラムのエントリ数})}$ で遷移確率を近似する。また、$a+b=1$ で $a,b \geq 0$ とする。

今回、2つの異なるバイグラムのもとで実験を行なった（それぞれ実験1、実験2と呼ぶ）。実験1は、open 及び closed の両方で行なった。実験2は、open でのみ行なった。実験1、実験2ともに正解は人間が与えることとした。表1及び表2に実験条件を示す。

表 1: 実験条件（実験1）

辞書項目数	単語バイグラム	品詞バイグラム	学習文数	テキスト数 (closed)	テキスト数 (open)
1641	1572(禁則 0)	309(禁則 2)	800	41	37

表 2: 実験条件（実験2）

辞書項目数	単語バイグラム	品詞バイグラム	学習文数	テキスト数 (closed)	テキスト数 (open)
9721	52763(禁則 0)	486(禁則 0)	20192	0	100

本実験ではスコアを求めるのに、パラメタ a,b に自由度がある。従って、まずパラメタを定めるための実験を行なった。適当なパラメタを求めることは、（バイグラムの）モデルを最適化することと考えられる。従ってこの実験では、最尤解が保証される ordered サーチ

を用い、パラメタを変化させて実験1の実験条件の実験を行なうことで適当な正解率及び効率を与えるパラメタを求めることとする。効率については、探索時にスタックにプッシュした回数及びスタックからポップした回数をもとに判断することとする。3節で述べたように、プッシュした回数は展開したノードの個数に等しい。スコア計算は展開時に行なわれるため、プッシュした回数は実質的に要した計算量に等しい。また、ポップした回数は選ばれたノードの個数に等しい。すなわち、ポップした回数が多い時は無駄に選ばれたノードが多いということを示す。この結果を表3、表4に示す。この結果から、パラメタは正解率にはあまり影響を与えず（図6）、効率の面では a を大きくするに従い指数関数的に遅くなる（図7）。この実験では、$a = 0.5$ を境にして正解率が変わっている場合がある（図6）ため、以下の実験ではパラメタとして $a = b = 0.5$ を用いることとする。

表 3: closed テキスト

a	b	正解数	正解数 (5位内)	プッシュ ノード	ポップ ノード
0	1	29/41(70.7%)	38/41(92.7%)	5503	2328
0.125	0.875	29/41(70.7%)	38/41(92.7%)	5202	2108
0.25	0.75	29/41(70.7%)	38/41(92.7%)	6995	2763
0.375	0.625	29/41(70.7%)	38/41(92.7%)	9717	3594
0.5	0.5	29/41(70.7%)	38/41(92.7%)	17020	6433
0.625	0.375	29/41(70.7%)	38/41(92.7%)	21050	7816
0.75	0.25	29/41(70.7%)	38/41(92.7%)	40382	14730
0.875	0.125	29/41(70.7%)	38/41(92.7%)	59316	22508
1	0	8/41(19.5%)	13/41(31.7%)	223395	77086

表 4: open テキスト

a	b	正解数	正解数 (5位内)	プッシュ ノード	ポップ ノード
0	1	28/37(75.7%)	34/37(91.9%)	8636	3276
0.125	0.875	28/37(75.7%)	34/37(91.9%)	8355	3097
0.25	0.75	28/37(75.7%)	34/37(91.9%)	10556	4124
0.375	0.625	28/37(75.7%)	34/37(91.9%)	13474	5304
0.5	0.5	29/37(78.4%)	34/37(91.9%)	26196	10388
0.625	0.375	29/37(78.4%)	34/37(91.9%)	35998	14552
0.75	0.25	29/37(78.4%)	33/37(89.2%)	57778	22749
0.875	0.125	29/37(78.4%)	33/37(89.2%)	63984	24978
1	0	5/37(13.5%)	16/37(43.2%)	201538	64516

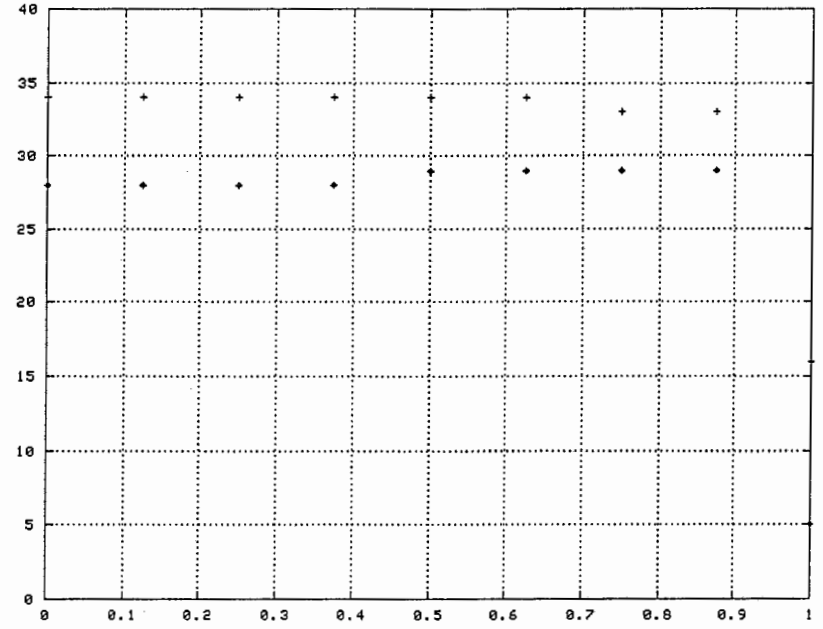

図 6: パラメタ a に対する 1 位正解数及び 5 位内正解数（open テキスト）

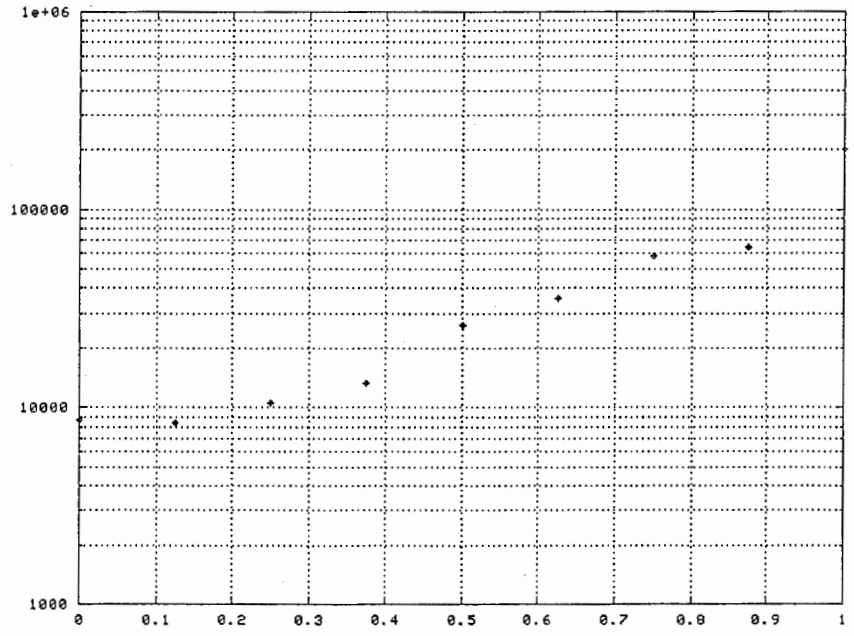

図 7: パラメタ a に対するプッシュノード数（open テキスト）

4.2 実験結果及び考察

表5、表6及び表7に実験結果を示す。ただし、beamサーチのビーム幅とはグローバルビーム幅である。また、今回、ローカルビーム幅は設定しなかった。つまりローカルビーム幅による枝がりは行なわなかった。

表 5: closed テキスト（実験 1）

探索法	正解数	正解数 (5 位内)	単語切り正解	単語切り正解 (5 位内)	プッシュノード	ポップノード
best-first	17/41(41%)	21/41(51%)	31/41(76%)	35/41(85%)	1145	637
beam（ビーム幅 10）	29/41(71%)	37/41(90%)	37/41(90%)	39/41(95%)	4963	1966
beam（ビーム幅 20）	29/41(71%)	38/41(93%)	37/41(90%)	39/41(95%)	8028	3081
beam（ビーム幅 30）	29/41(71%)	37/41(90%)	36/41(88%)	38/41(93%)	10840	4021
ordered	29/41(71%)	38/43(93%)	37/41(90%)	40/41(98%)	17020	6433

表 6: open テキスト（実験 1）

探索法	正解数	正解数 (5 位内)	単語切り正解	単語切り正解 (5 位内)	プッシュノード	ポップノード
best-first	13/37(35%)	16/37(43%)	18/37(49%)	21/37(57%)	1149	587
beam（ビーム幅 10）	28/37(76%)	32/37(86%)	33/37(89%)	34/37(92%)	5328	1992
beam（ビーム幅 20）	29/37(78%)	33/37(89%)	34/37(92%)	35/37(95%)	9078	3176
beam（ビーム幅 30）	27/37(73%)	31/37(84%)	32/37(86%)	33/37(89%)	12131	4204
ordered	29/37(78%)	34/37(92%)	34/37(92%)	36/37(97%)	26196	10388

表 7: open テキスト（実験 2）

探索法	正解数	正解数 (5 位内)	単語切り正解	単語切り正解 (5 位内)	プッシュノード	ポップノード
best-first	33/100(33%)	34/100(34%)	61/100(61%)	61/100(61%)	3393	1695
beam（ビーム幅 10）	77/100(77%)	87/100(87%)	92/100(92%)	95/100(95%)	18920	6874
beam（ビーム幅 20）	76/100(77%)	87/100(87%)	92/100(92%)	96/100(96%)	34289	12027
beam（ビーム幅 30）	76/100(77%)	87/100(87%)	91/100(91%)	96/100(96%)	47559	16505
ordered	80/100(80%)	89/100(89%)	97/100(97%)	97/100(97%)	461277	170567

以下、実験結果を概観し、それをもとに考察を行なう。best-first サーチは速いが正解率が低かった。beam サーチと ordered サーチはほぼ同じ正解率を出しているが、beam サー

チの方がかなり速かった。特にbeamサーチの中では、ビーム幅20のbeamサーチが適当であった。ビーム幅が大きくなった時に結果が悪くなっているのは本実験では、beamサーチにおいてローカルビーム幅を用いてないことが理由として考えられる。今回、orderedサーチはヒューリスティックを用いておらず、3節で示したように、全探索に近い動作を行なったため結果のように遅かったと考えられる。従ってorderedサーチの効率に関してはヒューリスティックの導入により改善される余地がある。

以上のことから、本実験での探索空間であったラティスは各枝のスコアにあまり偏りがなかったと考えられる。このような場合、best-firstサーチ或は（ヒューリスティックのない）orderedサーチはかなり不利である。また、本実験では、orderedサーチとbeamサーチの正解率がほぼ一致したということから、beamサーチにおいて最尤解があまり枝がりされなかったといえる。従って、結論としてbeamサーチを用いるのが最善であるといえる。

ここで、今回用いたモデルであるバイグラムについても考察する。まず、本実験において、形態素解析結果の誤り方には次の3つの場合が存在する。

1. 見出し語（単語）の区切り方及び品詞の付け方を誤る場合

2. 見出し語の区切り方は正しいが、品詞の付け方を誤る場合

3. 品詞の付け方は正しいが、見出し語の区切り方を誤る場合

実験において誤った解析結果の具体例をここでいくつか示す。

上記の1番の誤りでは、「･･･８月２２日から２５日･･･」という部分を解析した結果として、本来「･･･から（格助詞）２５（数詞）･･･」を得るべきところを、「･･･から（格助詞）２（数詞）５日（普通名詞）･･･」を得たというものがある。これは、バイグラムを調べた結果、単語バイグラム中に「から　２　-5.1428」というエントリが存在したためと分かった。このように、完全に例題にのみに頼り、何ら一般化（例えば、「から」の後には「一般的に数詞」が続くことが多いなどという知識）を（手動であれ自動であれ）行なわなければ、バイグラムのモデルを用いる限りではこのような知識の「誤った」適用は回避できない。

上記の2番の誤りは非常に多く、「私」という見出し語を、「（代名詞）」（以下、品詞は括弧にくくることで示す）と解析すべきところを「（普通名詞）」と誤ったり、「お」という見出し語を、「（接頭語）」と解析すべきところを「（普通名詞）」と誤ったりした。これは、今回、見出し語のバイグラムと、品詞のバイグラムを独立に扱ったために生じたと考えられる。この問題を回避するためには、見出し語と品詞を組み合わせたものをトークンとするバイグラムを用いることが考えられる。しかし、こうすることにより、見出し語と品詞を独立に学習して用いることが不可能になる。また、別の方法として、各見出し語の品詞生起確率(各見出し語がどういう確率である品詞となるか)を求め、それとバイグラムを組み合わせて用いることが考えられる。この方法では各言語モデルは独立に学習することができる。

また同じく2番の誤りで、今回5位候補まで求めたのであるが、それぞれの解候補自体には正解はないが（見出し語の区切り方は全て正しい）、5位内の候補それぞれを適当につなぎ合わせることで正解が得られるという場合がいくつかあった。この場合、候補数を増やすことにより、正解が候補中にでてくると考えられる。また、候補数を増やさなくても、得られた候補をまた探索空間として、更に上位の文法的知識を適用することで正解が得られると考えられる(形態素解析の出力をラティスとすればよい)。

上記の3番の誤りの例は今回存在しなかった。また、存在したとしても、本質的に1番の誤りと変わらないと考えられる。

結論として、今回用いた、見出し語と品詞を独立に扱うバイグラムのモデルは、不十分であるといえる。そこで、各見出し語の品詞生起確率を加えたモデルで再度実験してみた。結果を表8に示す。

実験はビームサーチ (ビーム幅10) についてのみ行なった。各モデルのパラメータは、1：1：1に重み付けした。完全な正解 (単語切り、品詞割当ともに正解) 率は向上したが、単語切りのみの正解率は逆に低下した。これは、単語切りに有効な情報である単語バイグラムの重みが相対的に低下したためと思われる。

表 8: open テキスト（実験 2） 品詞生起確率使用

品詞生起確率	正解数	正解数 (5 位内)	単語切り正解	単語切り正解 (5 位内)	プッシュ ノード	ポップ ノード
無し	77/100(77%)	87/100(87%)	92/100(92%)	95/100(95%)	18920	6874
有り	78/100(78%)	90/100(90%)	90/100(90%)	92/100(92%)	19011	6964

5 まとめ

今回、形態素解析における統計的手法のひとつであるバイグラムを用いた形態素解析を、best-first サーチ、beam サーチ、ordered サーチのそれぞれで実現し、比較検討した。実験では、比較的小規模な辞書及びバイグラムを用い、その上で、どのような探索手法が適当であるかを実際に実行してみることで比較検討した。各探索手法を検討した結果、本モデル及び本実験の規模のもとでは正解率及び効率の観点から見て、ビームサーチが適当であったといえる。また、本実験で用いたバイグラムのモデルについても、解析結果を調べることにより考察を行なった。結果、今回のモデルは、かなり近似度が大きく、精度を上げるためにはモデルの改善が必要であると考えられる。また、データベース中のデータにはかなりノイズが多く (「。」を普通名詞としたり、「会議」を固有名詞とするなどの例が見られた)、純粋に機械的に作成したバイグラムデータの信頼性が低いという問題もある。

今後の課題として、探索手法について、モデルが適切であれば、常に最尤解が保証される ordered サーチ（A^* サーチ）が有効であると考えられる。つまり、モデルがよく、最尤解が、（人間が見て）正解に最も近いという場合である。この時、適切なヒューリスティックを用いることによって探索効率をかなり高めることができると考えられる。また、この場合、beam サーチも最尤解をあまり枝がりしないようであれば用いることができると考えられる。ヒューリスティックは一般に問題の性質やモデルに大きく依存し、人間の経験に基づき導入されることが多い。従って、モデルが変化した時にヒューリスティックを新たに考える必要があるなどの問題がある。ヒューリスティックについても統計的なアプローチにより自動的に生成することができないかどうか検討する必要がある。

また、Viterbi サーチ（dynamic programming）を今回述べたような探索手法と統合して用いることも考えられるが、その手法を考案する必要がある。

今後のアプローチとしては、まず適切なモデルを見つけ、その後適当なヒューリスティックを求めて ordered サーチを用いるのが適当であると考える。

A　プログラムの説明

以下、今回作成したプログラムを使用する際に必要となると思われる事項について概説する。

A.1　データフォーマット

本プログラムは、辞書、単語バイグラム及び品詞バイグラムをもとにして、漢字かな混じり文字列を形態素解析する。また、解析結果の正解を記述したファイルを与えることにより、本プログラムの解析結果の正解率を求めることもできる。以下、各データのフォーマットを説明する。

1. 辞書

   次のフォーマットに従うファイルを実行ファイルと同じディレクトリに`lex.gra`という名前で作成する。このファイルが存在しないことは許されない。

   フォーマット

   （品詞名）　（見出し語）
   ・１行１項目
   ・#で始まる行はコメントとみなされる

   記述例を下に示す。
   （例）「本動詞　あがる」

2. 単語バイグラム

   次のフォーマットに従うファイルを実行ファイルと同じディレクトリに
   word-bigram-tableという名前で作成する。このファイルが存在しないことは許されず、単語バイグラムを用いない時でも上述のファイル名の空のファイルを作っておく必要がある。

   フォーマット

   （辞書の見出し語）　（辞書の見出し語）　遷移確率の自然対数
   ・１行１項目
   ・#で始まる行はコメントとみなされる

   記述例を下に示す。
   （例）「まず　原稿　-4.6052」

   また、禁則を示すファイルを次のフォーマットに従って、実行ファイルと同じディレクトリにword-kinsokuという名前で作成する。このファイルについても、必要ない時でも、空のファイルが必要である。

   フォーマット

   （辞書の見出し語）　（辞書の見出し語）
   ・１行１項目
   ・#で始まる行はコメントとみなされる

   記述例を下に示す。
   （例）「あし　あし」

3. 品詞バイグラム

次のフォーマットに従うファイルを実行ファイルと同じディレクトリに
hinshi-bigram-table という名前で作成する。このファイルが存在しないことは許されず、品詞バイグラムを用いない時でも上述のファイル名の空のファイルを作っておく必要がある。

フォーマット

（辞書の品詞名） （辞書の品詞名） 遷移確率の自然対数
・１行１項目
・# で始まる行はコメントとみなされる

記述例を下に示す。
（例）「連体詞 代名詞 -7.5033」

また、禁則を示すファイルを次のフォーマットに従って、実行ファイルと同じディレクトリに
hinshi-kinsoku という名前で作成する。このファイルについても、必要ない時でも、空のファイルが必要である。

フォーマット

（辞書の品詞名） （辞書の品詞名）
・１行１項目
・# で始まる行はコメントとみなされる

記述例を下に示す。
（例）「格助詞 格助詞」

4. 品詞生起確率

次のフォーマットに従うファイルを実行ファイルと同じディレクトリに
hinshi-hinshi-seiki-kakuritsu という名前で作成する。このファイルが存在しないことは許されず、品詞バイグラムを用いない時でも上述のファイル名の空のファイルを作っておく必要がある。

フォーマット

（辞書の見だし名） （辞書の品詞名） 生起確率の自然対数
・１行１項目
・# で始まる行はコメントとみなされる

記述例を下に示す。
（例）「私 代名詞 0.0000」

5. 日本語文字列

１行１文で、空白を入れずに記述する。ファイル名は任意である。

6. 正解ファイル

１行１文で、次のフォーマットに従う必要がある。ファイル名は任意である。

フォーマット

問題文を見出し語で区切った結果 | 問題文を品詞で区切った結果
・区切りは、1文字半角スペースでなければならない
・見出し語での正解と品詞での正解の間の区切りは、
半角 | でなければならない

記述例を下に示す。
(例)「そう です 。 | 副詞 助動詞 記号」

A.2 プログラムの起動方法

どの探索法を用いるプログラムでも、

実行ファイル名 問題文ファイル名 正解ファイル名 [-A]

で起動する。問題文ファイル名及び正解ファイル名は省略できない。-A なるオプションを付けると、解析結果を表示する。付けない場合、実行プログラムの解析結果の正解率のみ表示する。

プログラム起動後、対話形式でいくつかのパラメタの入力をうながされるので、適当な値を入力すること。