TR-IT-0205

# ルールバイグラムを用いた音声認識結果の再順序づけ

## An Exprimental Study on Reordering Speech Recognition Candidates Using CFG Rule N-gram

小暮 悟† 竹澤 寿幸
Satoru KOGURE† Toshiyuki TAKEZAWA

1997. 2

内容梗概

50会話の正解木を用いてルールバイグラムを構築し、ルールバイグラムを用いた言語スコアと音響スコアによって求められる総合スコアで音声認識結果を再順序づけした。

この再順序づけの実験により、認識率の向上が確認できた。つまり、ルールバイグラムの学習の有効性が示された。

これを音声認識自体に組み込むことで、さらに認識率が向上することが期待できる。

ATR音声翻訳通信研究所
ATR Interpreting Telecommunications Research Laboratories
†豊橋技術科学大学 情報工学課程
†Department of Information and Computer Sciences
Toyohashi University of Technology

目次

# 1　はじめに

音声認識において構文知識を用いると、認識の候補を文法にそった文だけに絞り込むことが可能となる。また、文法は日本語の一般的な構文規則を記述したものなので汎用性を持つ。しかし、文法による制約のみでは音声認識の曖昧性を解消できないことも多い。

これに対して、計算機の基本性能の向上、つまりは演算速度の向上により、対話コーパスから得られる統計的な連接情報を利用することが考えられる。

この統計的な連接情報、つまり N-gram モデルには、単語 N-gram(word N-gram)、品詞 N-gram、 preterminal N-gram など、さまざまな種類がある。ここでは、ルールに注目した。

つまり、 rule N-gram を、音声認識に用いて、認識率をあげようということである。必然的に、一文中において単語数や品詞数よりもルール数の方が多い。よって、それだけ制約をかけるのだから、認識率は向上するものと予想される。

そこで、ルールバイグラムの有効性を調べるため、音声認識の結果を、総合スコアによって再順序づけしてみた。総合スコアは、ルールバイグラムを用いて計算される言語スコアと、音声認識によって得られる音響スコアから求める。

以下、2章では、今回用いた統計的言語モデルの説明などの概要を示し、3章では、実験の方法を述べ、4章で実験結果とその考察を示す。最後に5章では、まとめと今後の課題を述べることにする。

## 2 概要

### 2.1 統計的モデル

#### 2.1.1 N グラムモデル

あるトークン列 $\{T_1, T_2, \cdots, T_{N-1}, T_N\}$ について、直前のトークンが $T_{i-1}$ である時、次のトークンが $T_i$ である確率 $P(T_i|T_{i-1})$ を バイグラムと呼ぶ。また、トークン $T_i$ の出現確率 $P(T_i)$ をユニグラムという。

一般に 確率 $P(T_i|T_{i-1}T_{i-2}\cdots T_{i-n+1})$ を N-gram と呼ぶ。

今回の研究では、トークンとしてルールを選んだので、それぞれ ルールバイグラム ( rule bigram )、ルールユニグラム ( rule unigram ) と呼ぶ。

また、今回の実験では、どこまでのルールを使うかで2種類の rule N-gram を構築している。一つは、すべてのルール列を使用する word rule N-gram で、もう一つは preterminal までのルール列を使用する preterminal rule N-gram である (図2.1、図2.2)。

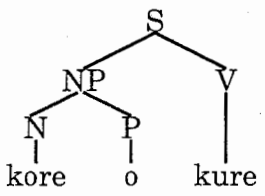


図 2.1: 構文木の例

```
word rule list
  (N<==>kore) (P<==>o) (NP<==>(N P)) (V<==>kure) (S<==>(NP V))

preterminal rule list
  (NP<==>(N P)) (S<==>(NP V))

word list
  kore o kure

preterminal list
  N P V
```

図 2.2: word rule bigram と preterminal rule bigram

#### 2.1.2 N グラムモデルの問題点

例えば、 学習用コーパスを用いてルールバイグラムを構築したとする。 そしてこのルールバイグラムを、 新しい文の言語スコアを求めることに使用するとして、その新しい文のあるルール列が学習用コーパスに一度も現れない場合にその値が 0、つまり $P(R_i|R_{i-1}) = 0$ 、となってしまいその文の言語スコア (その文の確からしさ、確率) が計算できない。

この問題点は、学習コーパスを大きくとることで改善されるかも知れないが、統計的モデルを用いる時の一つの大きな問題点である。

今回は、この問題点を解消するために、スムージングと呼ばれる手法を用いた。スムージングは、例えばルールバイグラムを以下のように定義し直すことである。

$$P'(R_i|R_{i-1}) = \lambda_0 + \lambda_1 P(R_i) + \lambda_2 P(R_i|R_{i-1})$$

$$\lambda_0 + \lambda_1 + \lambda_2 = 1$$

こうすることで、$P(R_i|R_{i-1}) = 0$ であっても、ルールバイグラムを計算できる。

この $\lambda$ を求めるのには、削除補間法と呼ばれる方法を用いた。

### 2.2 ルールバイグラムの生成

ルールの適用順序、すなわち、構文解析法としては、音声認識に組み込むことを考えて、ボトムアップLR構文解析法を用いた。

### 2.3 言語スコアの導出

学習コーパスからルールバイグラム $P(R_i|R_j)$ が計算されているとして、ある認識結果の構文木の言語スコアは、構文木から求まるルール列を $\{R_1, R_2, \cdots, R_{N_r-1}, R_{N_r}\}$ とすると、その言語スコアは、

$$S_L = P(R_1| < BEGIN >) \cdot P(R_2|R_1) \cdots P(R_{N_r}|R_{N_r-1}) \cdot P(< END > |R_{N_r})$$

となる。ここで、$< BEGIN >$ はルール開始記号、$< END >$ はルール終了記号を示す。また、$N_r$ は文に含まれるルールの個数を示す。

実際の言語スコアは、音響スコアとの対応を考えて、

$$\begin{aligned} S_L &= \log_{1.0001} P'(R_1| < BEGIN >) \\ &+ \sum_{i=2}^{N_r-1} \log_{1.0001} P'(R_i|R_i - 1) \\ &+ \log_{1.0001} P'(< END > |R_{N_r}) \end{aligned}$$

とした。

### 2.4 音声認識結果の再順序づけ

まず、総合スコアであるが、

$$S_T = S_r + (\omega + \frac{\alpha}{N_r}) S_L$$

とした。

ここで、$N_r$ はルール数である。

### 2.4.1 ルール数による正規化

$\alpha = 0$ とすると、

$$S_T = S_r + \omega S_L$$

となる。すなわち、言語スコアをルール数で正規化せずに、ただ重みを かけて計算することになる。

また、$\omega = 0$ とすると、

$$S_T = S_r + \frac{\alpha}{N_r} S_L$$

となる。すなわち、言語スコアをルール数で正規化して、重みを かけて計算することになる。

### 2.4.2 再順序づけ

求まった総合スコア順に音声認識の候補をソートすることで、再順序づけができる。詳しい例については次章で述べる。

## 3 実験

実験の流れを以下に示す。

1. ルール列の算出
2. ルールバイグラムの訓練(学習)
3. 再順序づけ
4. 認識率の計算

### 3.1 ルール列の算出

学習用コーパスの構文木からルール列を求める。

学習用コーパスとしては、Open なデータとして「50 会話の正解木」を使用した。また、比較対象として、Closed なデータ「評価用 12 会話の正解木」を使用した。

表 3.1にそれぞれのコーパスについて示す。

表 3.1: 学習用コーパス

| 学習用コーパス | 総文数 | 述べルール数 | | ルールの種類 | | 文法の名前 | 文法規則 | |
|---|---|---|---|---|---|---|---|---|
| | | W | P | W | P | | W | P |
| 50 会話の正解木 | 1960 | 54888 | 34214 | 3009 | 1382 | cfg-big-a | 4193 | 2214 |
| 評価用 12 会話の正解木 | 396 | 12188 | 7682 | 1543 | 833 | cfg-al-2-1 | 2818 | 1808 |

注：W は word、P は preterminal を示す。

```
INPUT: s ou d e s u k a  -->  Success [1]
<sent>
|--<cl>
   |--<cl1>
      |--<vp-sfp>
         |--<vaux-sfp>
            |--<vaux-optt-syusi>
            |  |--<verb-cop-syusi>
            |     |--<adv-desu>
            |     |  |--sou
            |     |--<aux-cop-desu-syusi>
            |        |--<auxstem-desu>
            |        |  |--de
            |        |--<vinfl-spe-su>
            |           |--su
            |--<aux-sfp>
               |--ka
```

図 3.1: 学習用コーパスの正解木の例

学習用コーパスの構文木の例を図 3.1に示す。

この構文木は、見た目には分かりやすいが、いざ、コンピュータによる処理をさせようとすると困難である。よって、まず、図3.2のような木に変換してから ルール列を求める。

図3.3にその変換例を示す。この解析には、ボトムアップLR構文解析を用いている。そして、ルールバイグラムを生成する時に使用するルール列は、図3.3、図3.4となる。

```
(<sent>
   (<cl>
      (<cl1>
         (<vp-sfp>
            (<vaux-sfp>
               (<vaux-optt-syusi>
                  (<verb-cop-syusi>
                     (<adv-desu> sou )
                     (<aux-cop-desu-syusi>
                        (<auxstem-desu> de )
                        (<vinfl-spe-su> su ))))
               (<aux-sfp> ka ))))))
```

図 3.2: lisp 的な木への変換

```
(<adv-desu><==>sou)
(<auxstem-desu><==>de)
(<vinfl-spe-su><==>su)
(<aux-cop-desu-syusi><==>(<auxstem-desu><vinfl-spe-su>))
(<verb-cop-syusi><==>(<adv-desu><aux-cop-desu-syusi>))
(<vaux-optt-syusi><==>(<verb-cop-syusi>))
(<aux-sfp><==>ka)
(<vaux-sfp><==>(<vaux-optt-syusi><aux-sfp>))
(<vp-sfp><==>(<vaux-sfp>))
(<cl1><==>(<vp-sfp>))
(<cl><==>(<cl1>))
(<sent><==>(<cl>))
```

図 3.3: ルール列への変換／ word までのルール列

```
(<aux-cop-desu-syusi><==>(<auxstem-desu><vinfl-spe-su>))
(<verb-cop-syusi><==>(<adv<==>(<adv-desu><aux-cop-desu-syusi>))
(<vaux-optt-syusi><==>(<ver b-cop-syusi>))
(<vaux-sfp><==>(<vaux-optt-syusi><aux-sfp>))
(<vp-sfp><==>(<vaux-sfp>))
(<cl1><==>(<vp-sfp>))
(<cl><==>(<cl1>))
(<sent><==>(<cl>))
```

図 3.4: ルール列への変換／ preterminal までのルール列

## 3.2 rule N-gram の計算

rule N-gram はトークンがルールであるという点を除けば、単語 N-gram や preterminal N-gram と同じである。したがって、その計算法は同じとなる。

すなわち、ルールユニグラムは、ルール $R_i$ の出現頻度 $c(R_i)$、ルールの述べ出現数を $N_r$ とすると、

$$P(R_i) = \frac{c(R_i)}{N_r}$$

で求められる。

また、ルールバイグラムは、ルール $R_j$ の次にルール $R_i$ が出現する頻度 $c(R_i|R_j)$ とすれば

$$P(R_i|R_j) = \begin{cases} \frac{c(R_i|R_j)}{c(R_j)} & \cdots \quad c(R_j) \neq 0 \\ 0 & \cdots \quad c(R_j) = 0 \end{cases}$$

となる。

## 3.3 再順序づけ

再順序づけを行なう音声認識結果は、表 3.2の通りである。また、その音声認識のビーム幅は、500000, 750000, 1000000, 1500000 の 4 種類である。

表 3.2: 音声認識結果

| 会話記号 (ID) | 話者名 | 文数 |
|---|---|---|
| TAC23034 | FMAKA | 36 |
| TAS12009 | FYUYO | 24 |
| TAS22001 | FHITA | 24 |
| TAS33002 | FYOAS | 50 |
| TCC22074 | FHITA | 28 |

音声認識結果の再順序づけは、総合スコア $S_T$ によっておこなうことにする。総合スコア $S_T$ は、音響スコア $S_r$ と言語スコア $S_L$ から求まる。

$$S_T = S_r + (\omega + \frac{\alpha}{N_r})S_L$$

### 3.3.1 言語スコアの計算

音声認識結果の構文木は、図 3.2のような形をしている。一つの構文木から一つのルール列を導出して、そのルール列 $\{R_1, R_2, \cdots, R_{N_r-1}, R_{N_r}\}$ から、その構文木の言語スコア $S_L$ が計算される。

$$
\begin{aligned}
S_L &= \log_{1.0001} P'(R_1| < BEGIN >) \\
&+ \sum_{i=2}^{N_r} \log_{1.0001} P'(R_i|R_i - 1) \\
&+ \log_{1.0001} P'(< END > |R_{N_r})
\end{aligned}
$$

#### 3.3.2 総合スコアの計算

総合スコアは、音響スコアと言語スコアの2つのスコアから計算できる。

しかし、一つの認識候補について 音響スコアは一つであるが、言語スコアは複数存在する。よって、ここでは、複数の言語スコアの中から、最大の言語スコアを採用して、総合スコアを計算する。

#### 3.3.3 再順序づけ

ある入力文に対して複数の認識候補があり、 それぞれについて、 総合スコアを計算する。それから、 総合スコアの大きい順に認識候補を降順にソートすることで音声認識結果の再順序づけが可能となる。

#### 3.3.4 音声認識結果の format

図3.5に音声認識結果の format を示す。

#### 3.3.5 再順序づけ後の format

図3.6に再順序づけ後の音声認識結果の format を示す。

### 3.4 認識率の計算

再順序づけされた認識結果から、認識率が求まる。今回は、単語の再現率を示す、単語認識率と、文の正解率を示す 文認識率を求めた。

また、正解が認識候補の1位になった時の 1位認識率、正解が認識候補の5位までに入った時の 5位認識率、正解が認識候補の20位までに入った時の 20位認識率を求めた。これにより、6位以下20位以上の認識候補が5位以内に入った、などの議論ができるようになる。

```
1 Answer: 正解の音素列
SS   Sentence Start

Recognition terminater ** frame ** msecCPU ** secElaps ** PU
;; Partial Lis
PS
(1) 音声認識候補1 score= 音響スコア ** 総合スコア **
[1:1]
(pause_unit
構文木
)
[1:2]
(pause_unit
構文木
)
:
:
(2) 音声認識候補2 score= 音響スコア ** 総合スコア **
[2:1]
(pause_unit
構文木
)
:
PE

2 Answer:
:
:
```

図 3.5: 再順序づけ前の音声認識結果

```
1 Answer: 正解の音素列
SS   Sentence Start

Recognition terminater ** frame ** msecCPU ** secElaps ** PU
;; Partial Lis
PS
(1) 音声認識候補1 score= 音響スコア ** 総合スコア **
[1:1] preterminal|word の 列
Rule_Score = 言語スコア
[1:2] preterminal|word の 列
Rule_Score = 言語スコア
:
:
(2) 音声認識候補2 score= 音響スコア ** 総合スコア **
[2:1] preterminal|word の 列
Rule_Score = 言語スコア
:
PE

2 Answer:
:
:
```

図 3.6: 再順序づけされた音声認識結果

# 4　実験結果、および考察

## 4.1　スムージングの係数

スムージングの係数 $\lambda$ は、計算の結果、表のようになった。

表 4.1: スムージングの係数

| 学習コーパス | ルールリスト | $\lambda_0$ | $\lambda_1$ | $\lambda_2$ |
|---|---|---|---|---|
| Open | Word | 0.089398 | 0.049056 | 0.861546 |
| Open | Preterminal | 0.061468 | 0.068433 | 0.870099 |
| Closed | Word | 0.145308 | 0.027423 | 0.827269 |
| Closed | Preterminal | 0.150619 | 0.003317 | 0.846064 |

これにより、 $P'(R_i|R_j)$ を求めた。

## 4.2　ルール数で正規化することの有用性

総合スコア $S_T$ の計算時において、

1. $\alpha = 0$ として 最適な $\omega$ を使用する。

2. $\omega = 0$ として 最適な $\alpha$ を使用する。

の 2 種類の方法がある。

今回は、5 つのタスク＆話者から 1 つ (会話番号： TAS22001、話者： FHITA) を選び、ビーム幅 1000000 の音声認識結果について、正規化の有用性を調べた。(図 4.1～図 4.8)

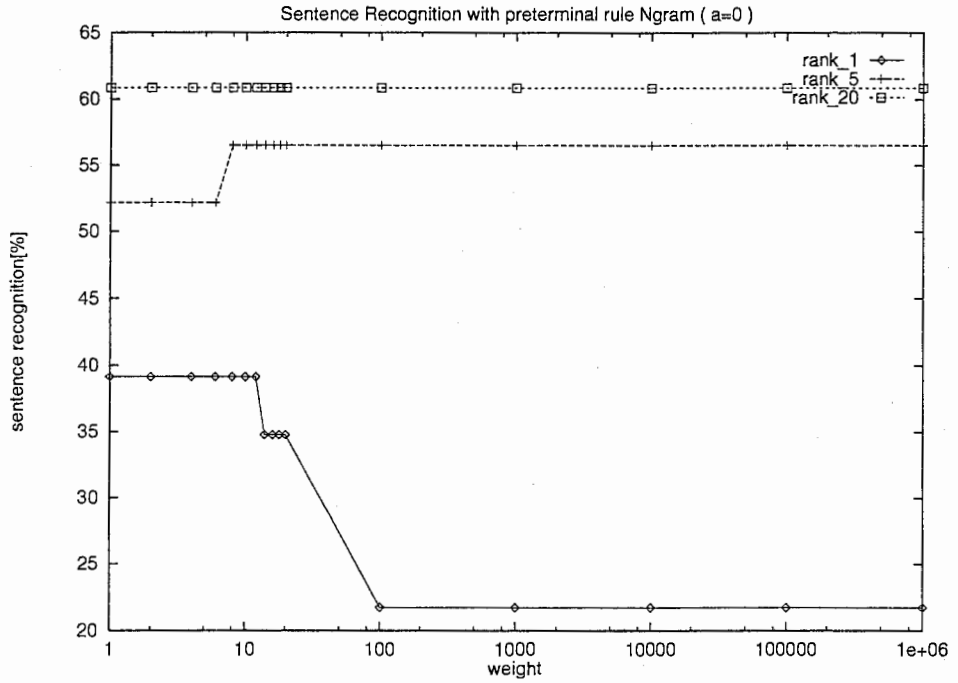


図 4.1: 正規化しない時の文認識率／preterminal rule N-gram／log scale

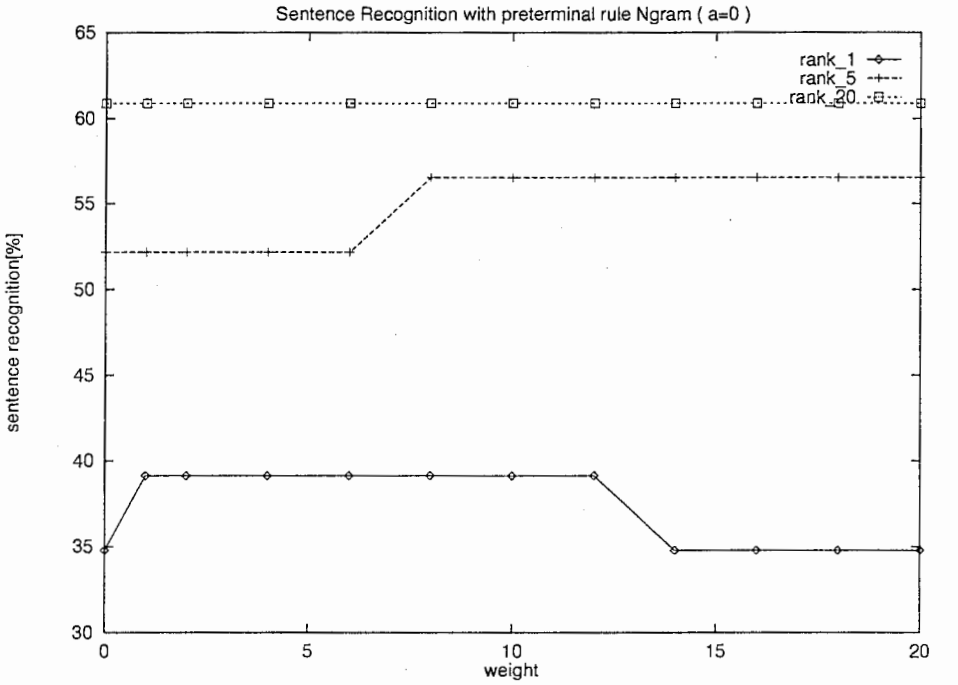


図 4.2: 正規化しない時の文認識率／preterminal rule N-gram／normal scale

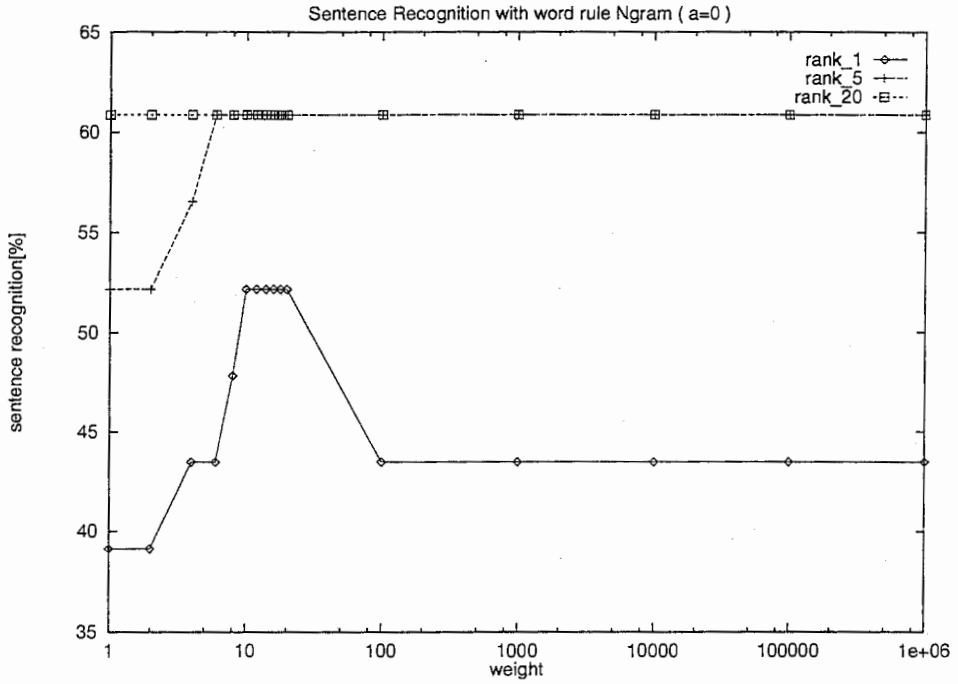


図 4.3: 正規化しない時の文認識率／ word rule N-gram ／ log scale

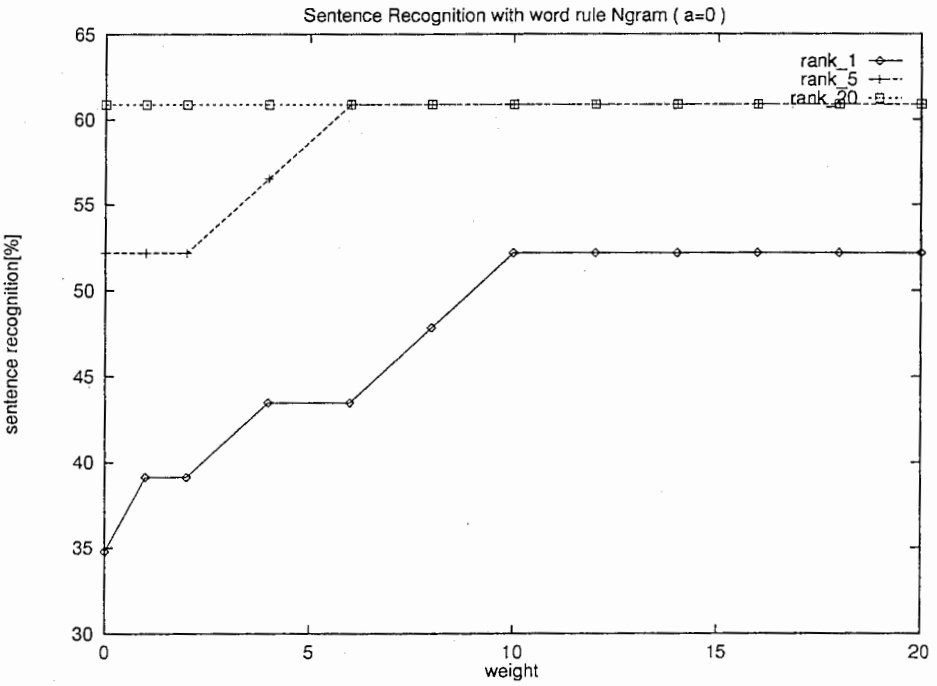


図 4.4: 正規化しない時の文認識率／ word rule N-gram ／ normal scale

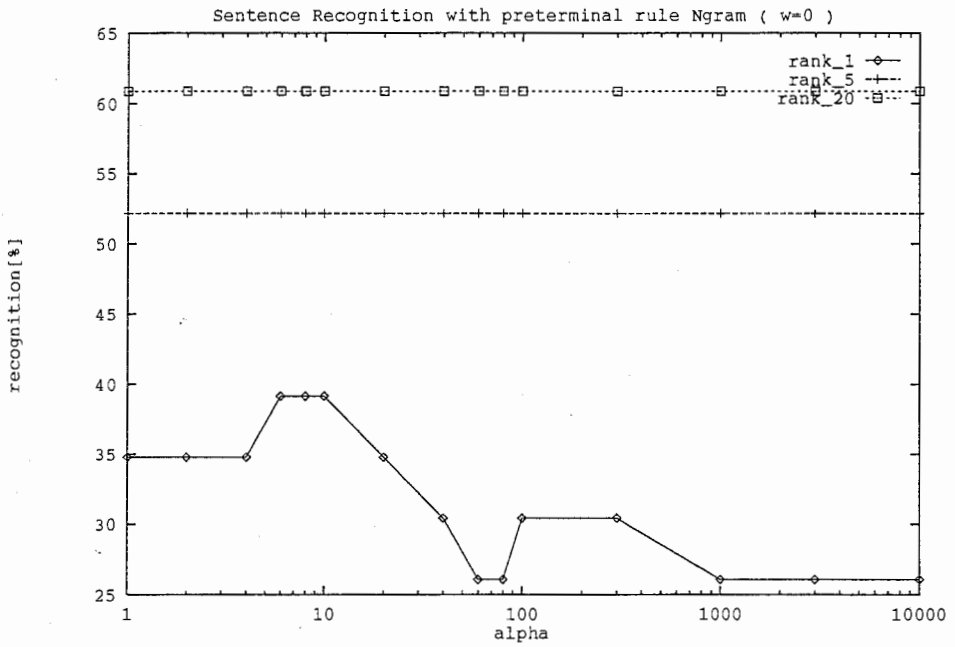


図 4.5: 正規化した時の文認識率／preterminal rule N-gram／log scale

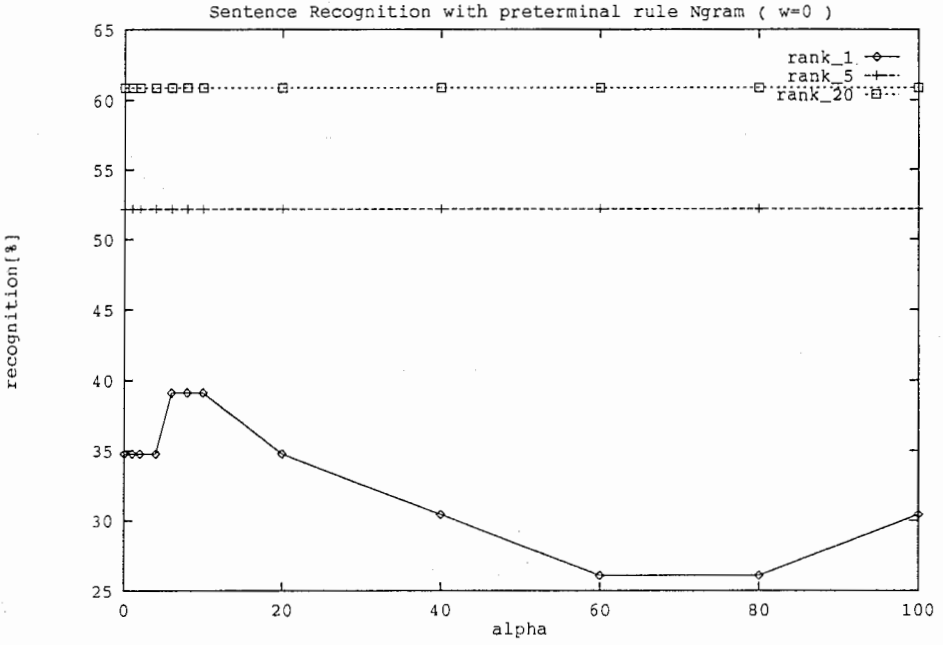


図 4.6: 正規化した時の文認識率／preterminal rule N-gram／normal scale

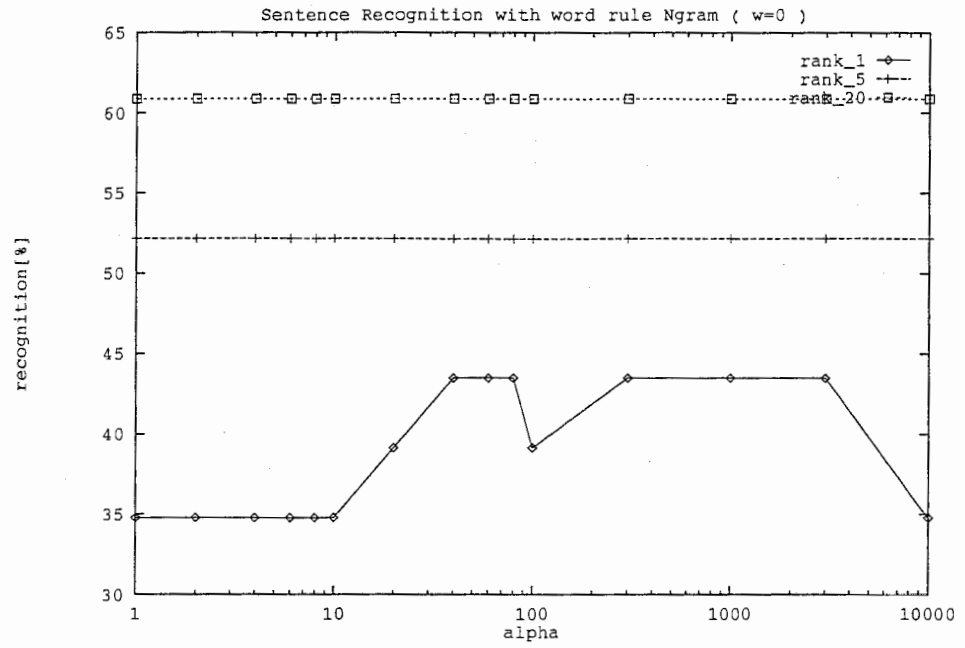


図 4.7: 正規化した時の文認識率／ word rule N-gram ／ log scale

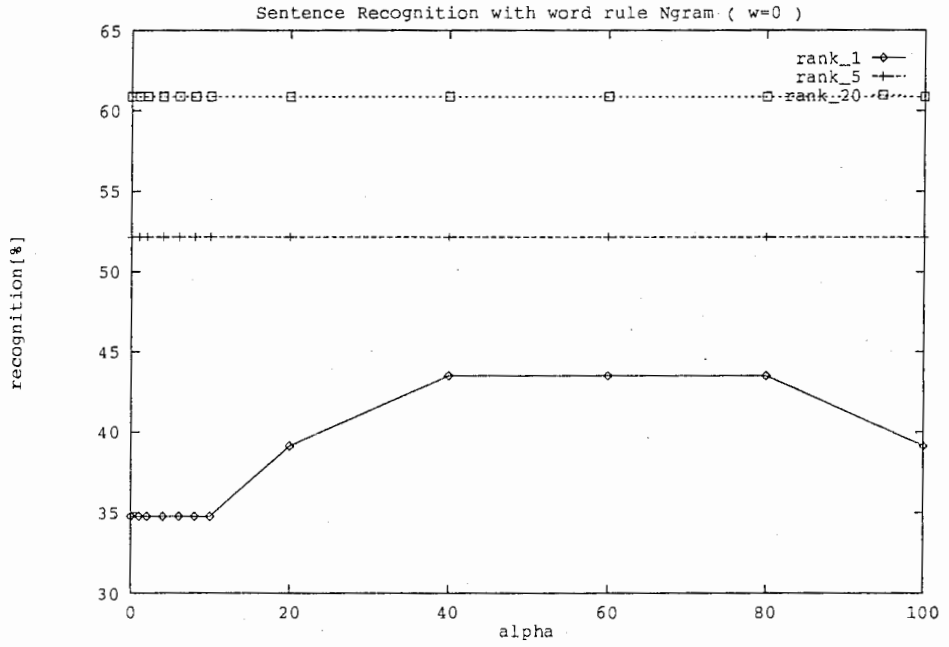


図 4.8: 正規化した時の文認識率／ word rule N-gram ／ normal scale

このグラフから、まず、正規化をする、しないに関わらず、グラフにピークがあり、ルールバイグラムが有用であることが分かる。認識率の向上はほぼ同程度と言える。認識率向上の度合が同程度であれば計算量の少ない方がいいので、正規化をしないことにした。

また、word rule N-gram と、preterminal rule N-gram の両グラフのピーク位置を考慮して、正規化をせずに($\alpha = 0$)、$\omega = 10$ として、5つのタスクで再順序づけの実験を行なうことにした。

### 4.3 再順序づけ実験結果

$\omega = 10$、$\alpha = 0$ の条件(ルール数による正規化なし、言語スコアの重み10)で再順序づけの実験を行なった。

ページ数の関係上すべてのグラフを載せることはできないので、5つの中の1つ(会話番号：TAS33002、話者：FYOAS)についてのグラフ(図4.9～図4.16)と、全タスクの平均のグラフ(図4.17～図4.24)を示す。

この結果より、ルールリストとして word までとった場合と、preterminal までとった場合で、単語認識率と文認識率に差が見られることが分かる。つまり、制約のきつい word まで見た rule N-gram の方が認識率が高くなっている。

しかしながら、preterminalまでのルールNグラムモデルでも、十分効果があるといえる。

次に、Open データと Closed データの比較であるが、これについては顕著な差は出ていない。

参考データとして、Open データの Closed データに対するカバレージと、Open データと closed データのパープレキシティを示す。

また、再順序づけの例を、図4.25に示す。

表 4.2: カバレージ

| 種類 | カバレージ [%] |
|---|---|
| 文法 | 98.8 |
| 累積カバー率 (word rule bigram) | 89.4 |
| 種類カバー率 (word rule bigram) | 68.6 |
| 累積カバー率 (preterminal rule bigram) | 92.2 |
| 種類カバー率 (preterminal rule bigram) | 73.7 |

表 4.3: パープレキシティ

| 種類 | パープレキシティ [%] | 累積ルール数／総文数 |
|---|---|---|
| CLOSED (preterminal rule bigram) | 3.49 | 19.4( 7682/396) |
| CLOSED (word rule bigram) | 3.49 | 30.8(12188/396) |
| CPEN (preterminal rule bigram) | 9.33 | 28.0(54888/1960) |
| OPEN (word rule bigram) | 7.44 | 17.5(34214/1960) |

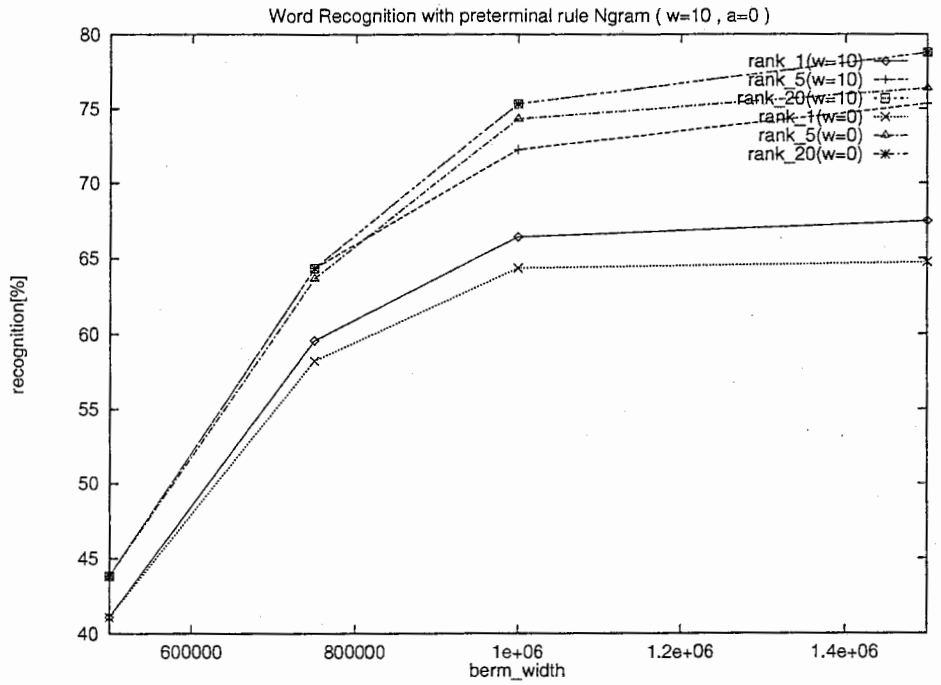


図 4.9: 単語認識率 (Open rule) ／ preterminal rule N-gram ／ TAS33002.FYOAS

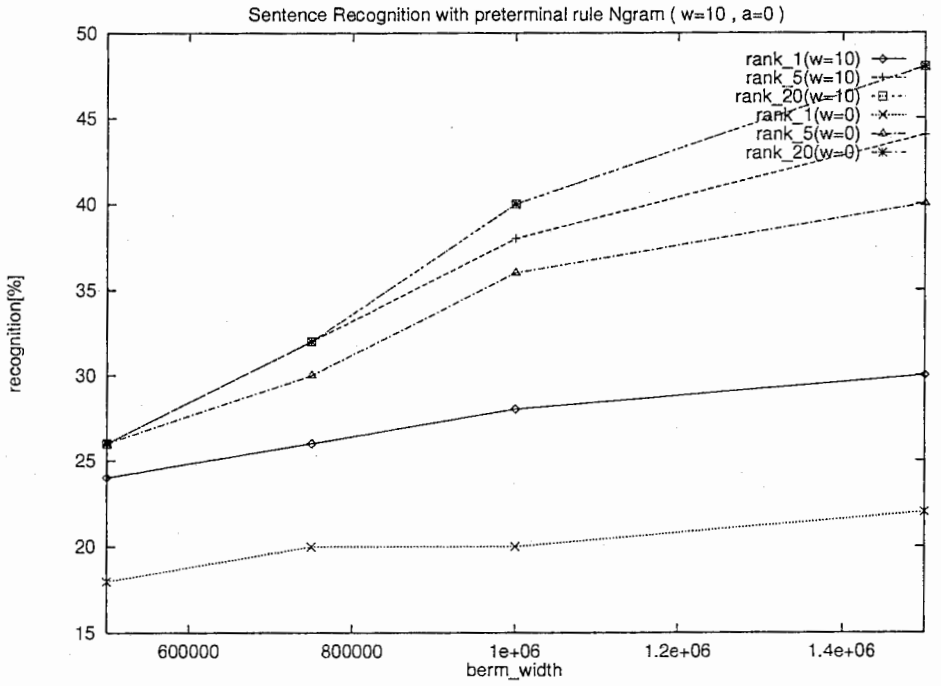


図 4.10: 文認識率 (Open rule) ／ preterminal rule N-gram ／ TAS33002.FYOAS

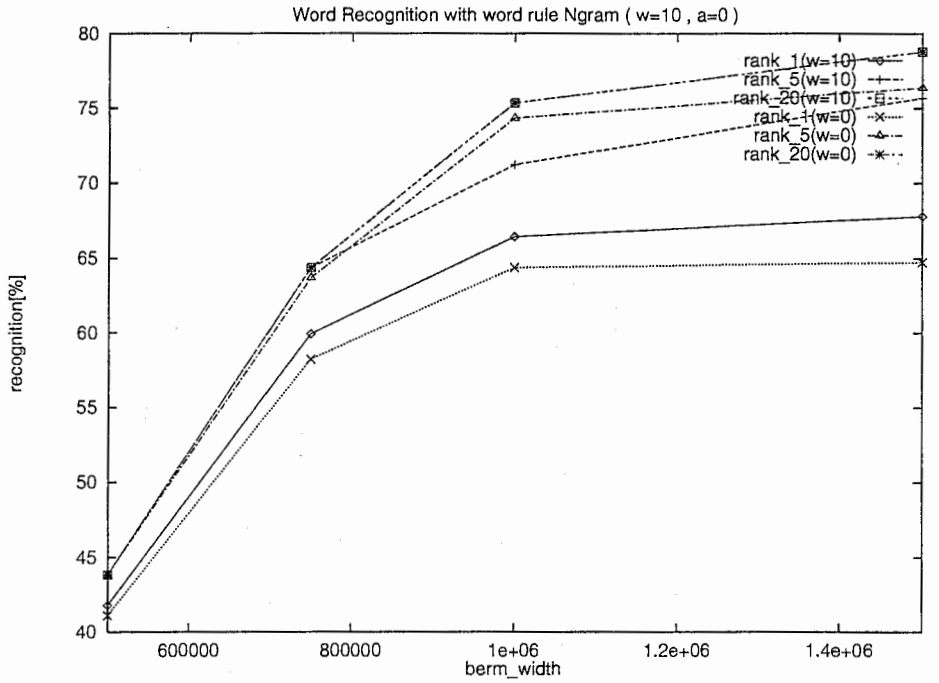


図 4.11: 単語認識率 (Open rule) ／ word rule N-gram ／ TAS33002.FYOAS

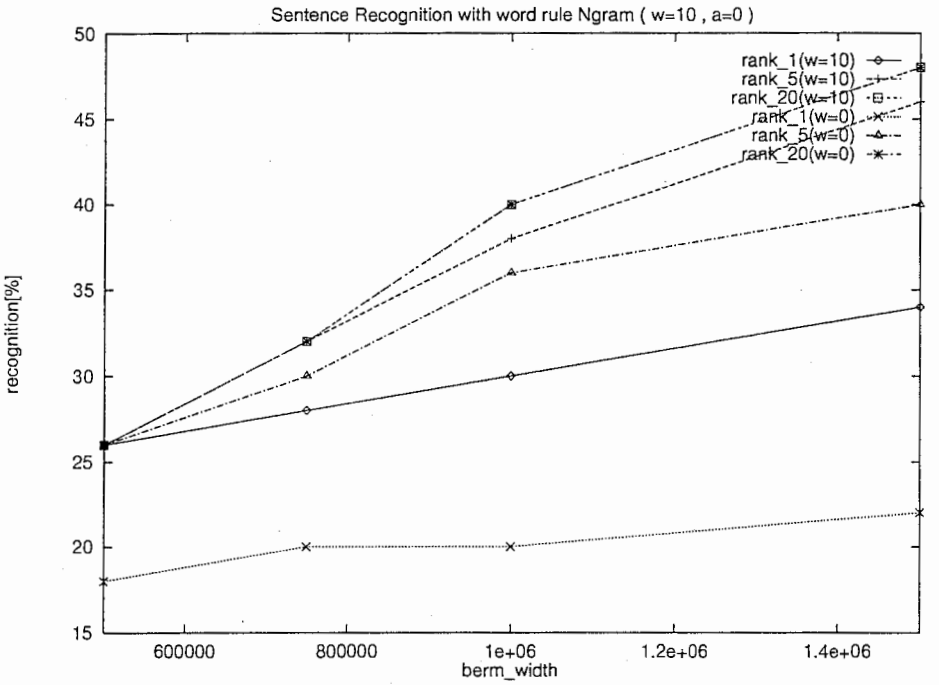


図 4.12: 文認識率 (Open rule) ／ word rule N-gram ／ TAS33002.FYOAS

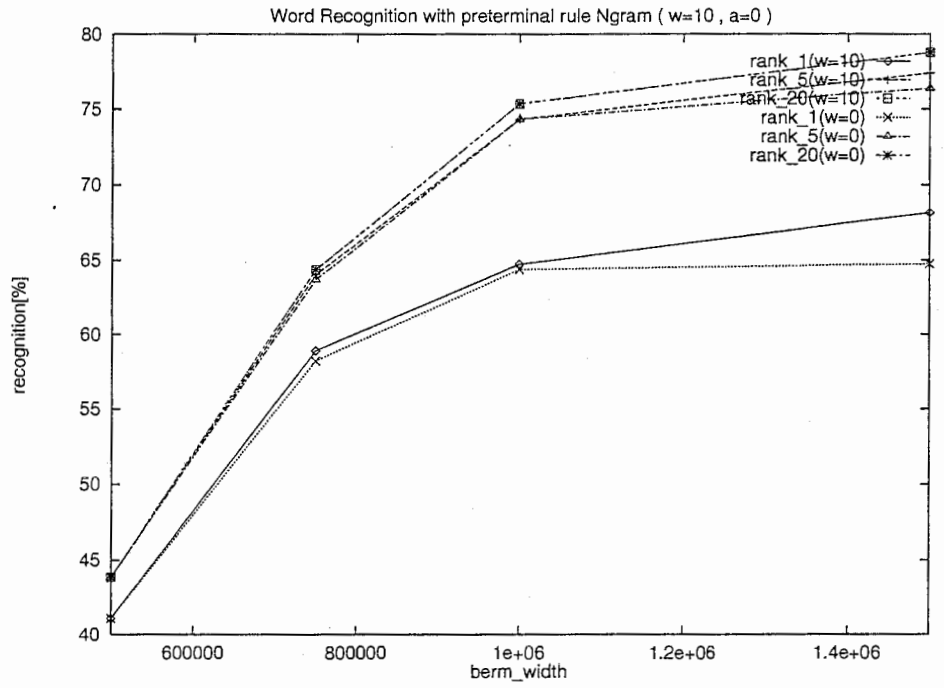


図 4.13: 単語認識率 (Closed rule) ／ preterminal rule N-gram ／ TAS33002.FYOAS

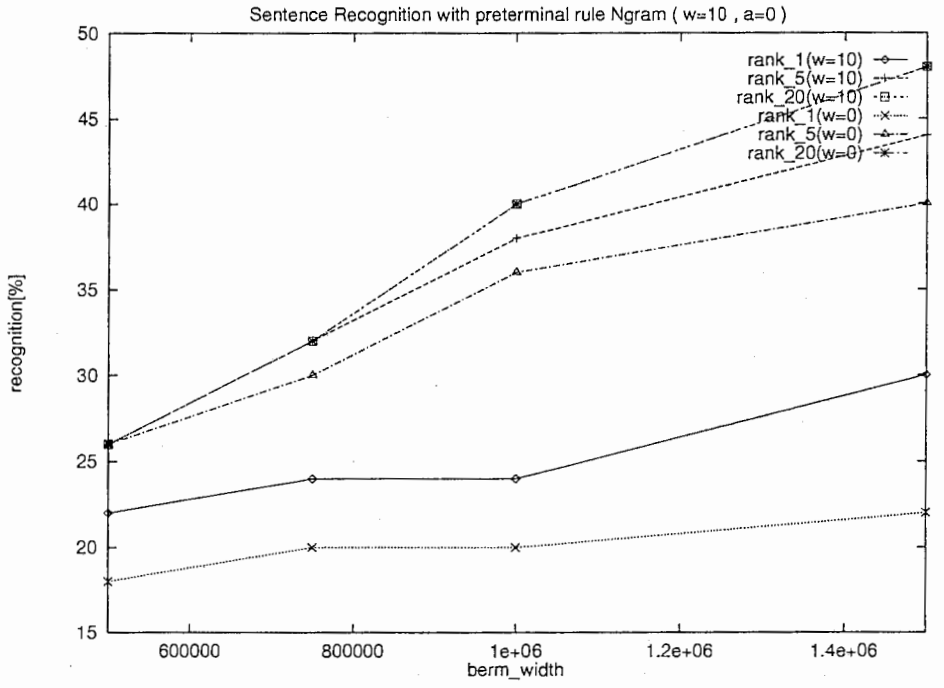


図 4.14: 文認識率 (Closed rule) ／ preterminal rule N-gram ／ TAS33002.FYOAS

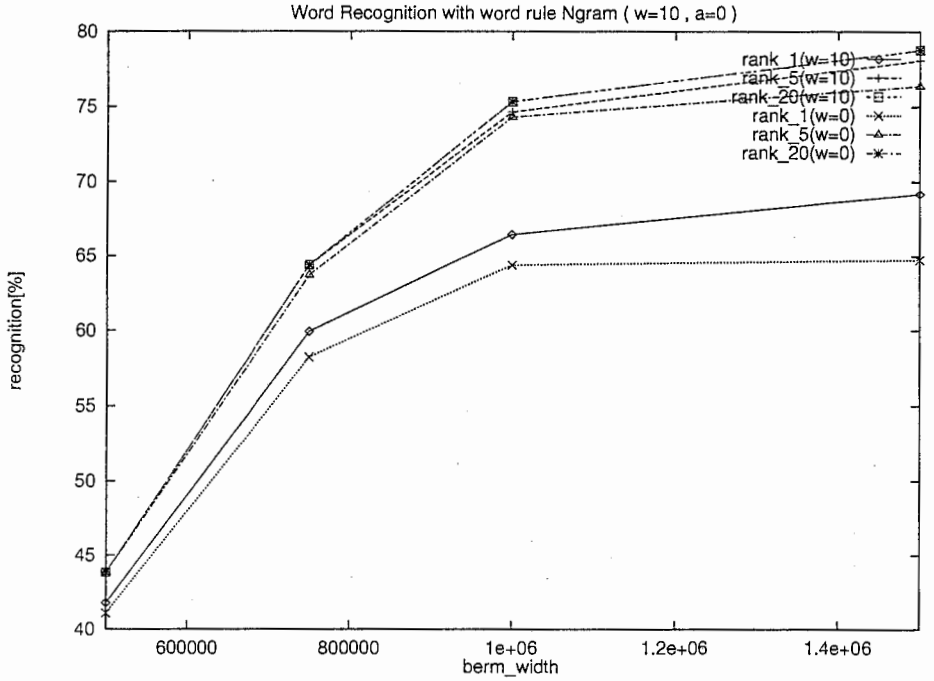


図 4.15: 単語認識率 (Closed rule) ／ word rule N-gram ／ TAS33002.FYOAS

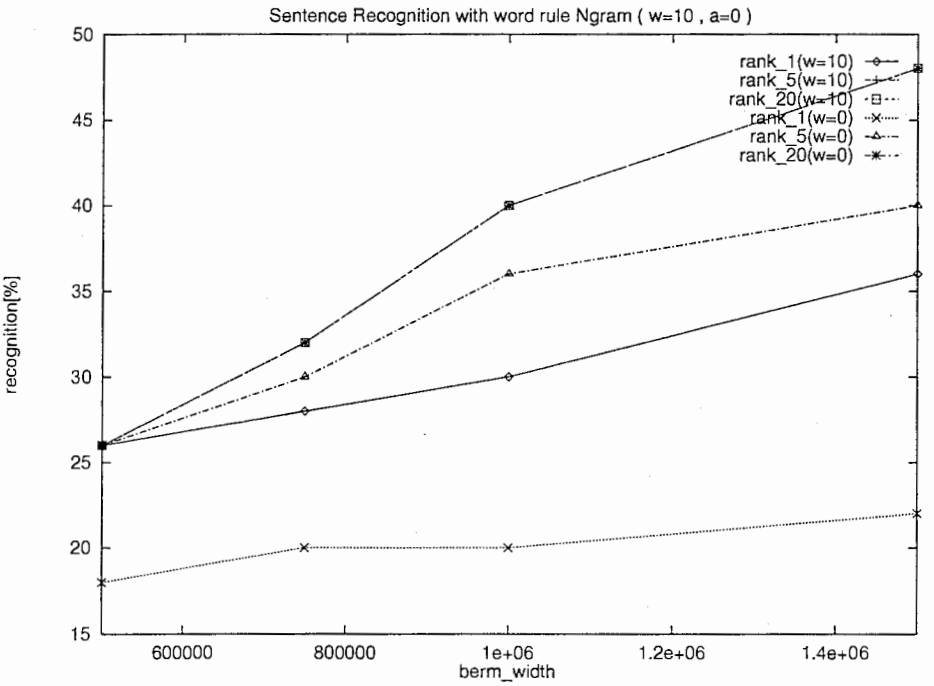


図 4.16: 文認識率 (Closed rule) ／ word rule N-gram ／ TAS33002.FYOAS

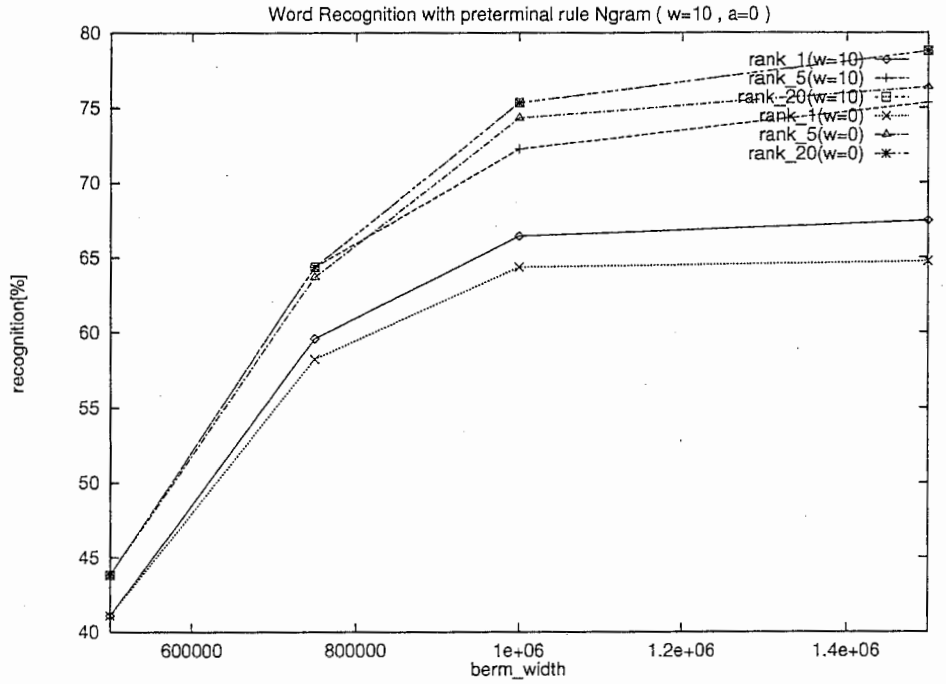


図 4.17: 単語認識率 (Open rule) ／ preterminal rule N-gram ／ ALL.AVERAGE

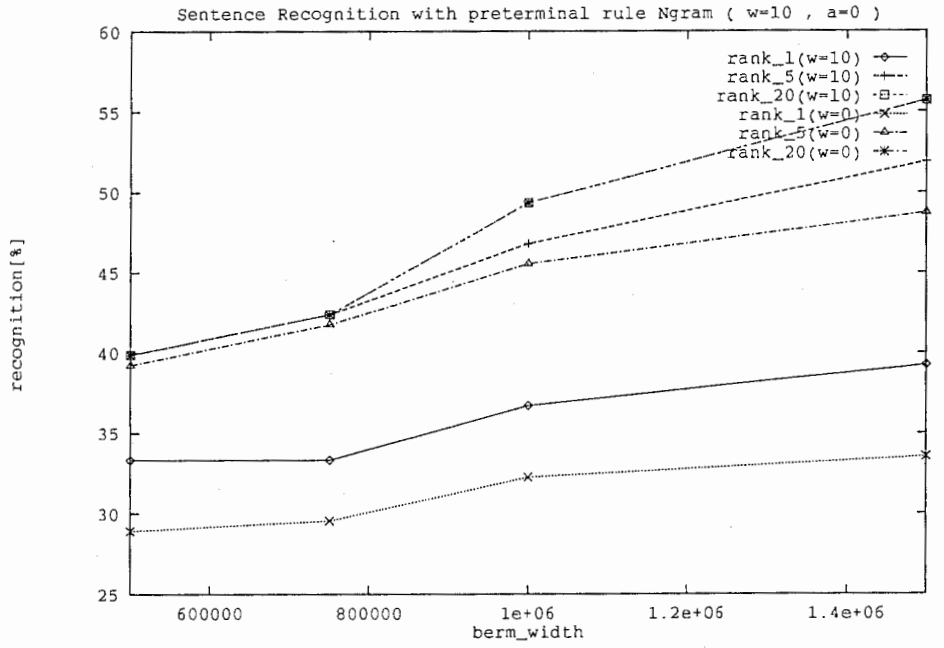


図 4.18: 文認識率 (Open rule) ／ preterminal rule N-gram ／ ALL.AVERAGE

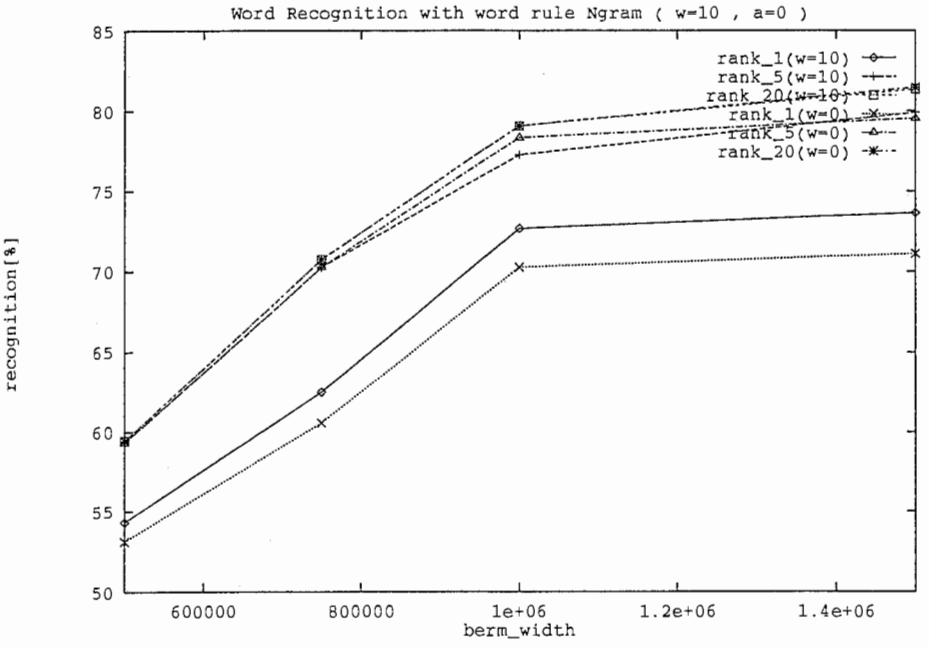


図 4.19: 単語認識率 (Open rule) ／ word rule N-gram ／ ALL.AVERAGE

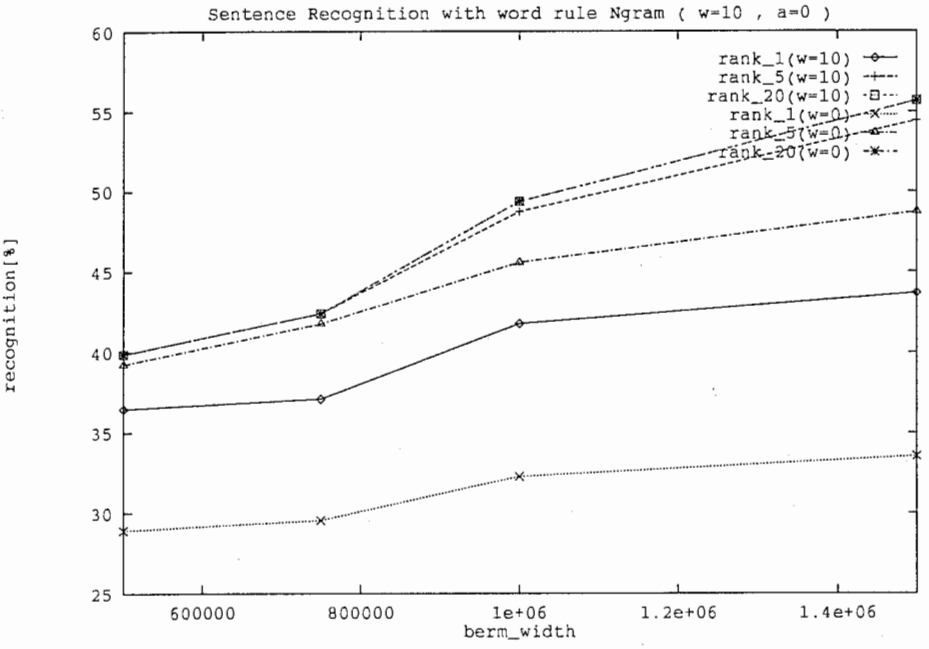


図 4.20: 文認識率 (Open rule) ／ word rule N-gram ／ ALL.AVERAGE

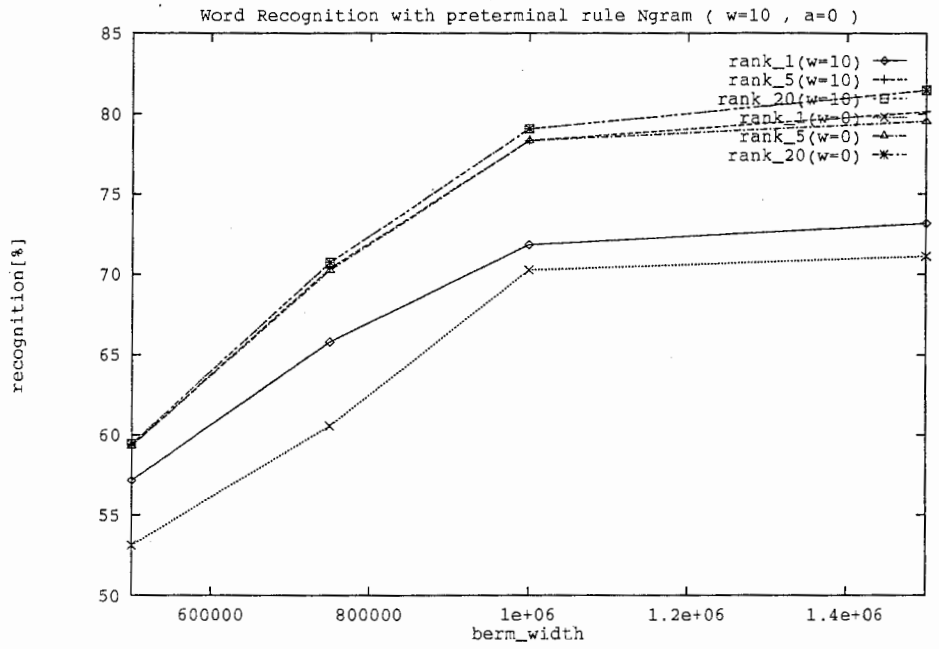


図 4.21: 単語認識率 (Closed rule) ／ preterminal rule N-gram ／ ALL.AVERAGE

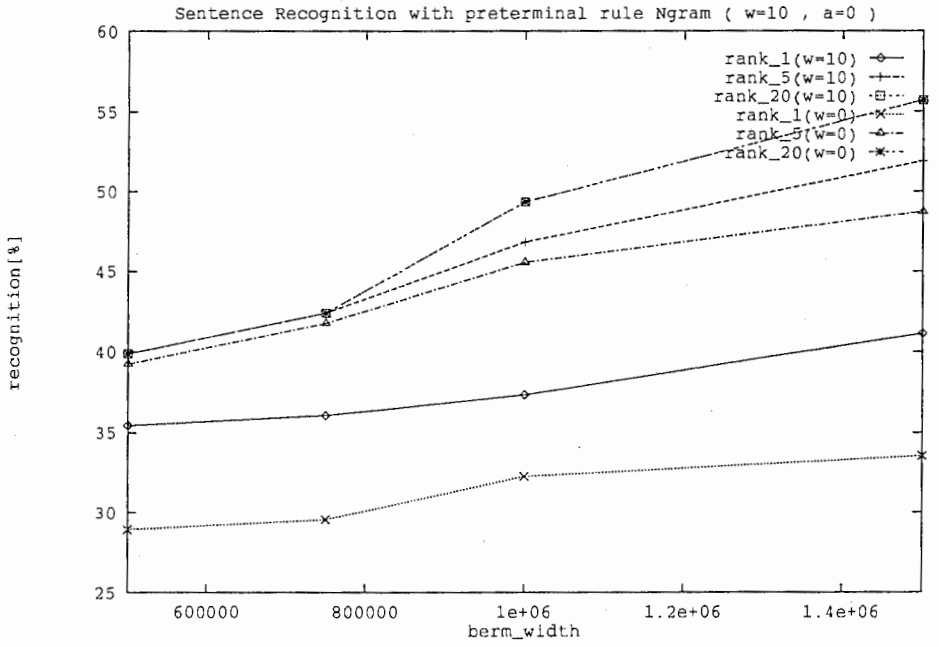


図 4.22: 文認識率 (Closed rule) ／ preterminal rule N-gram ／ ALL.AVERAGE

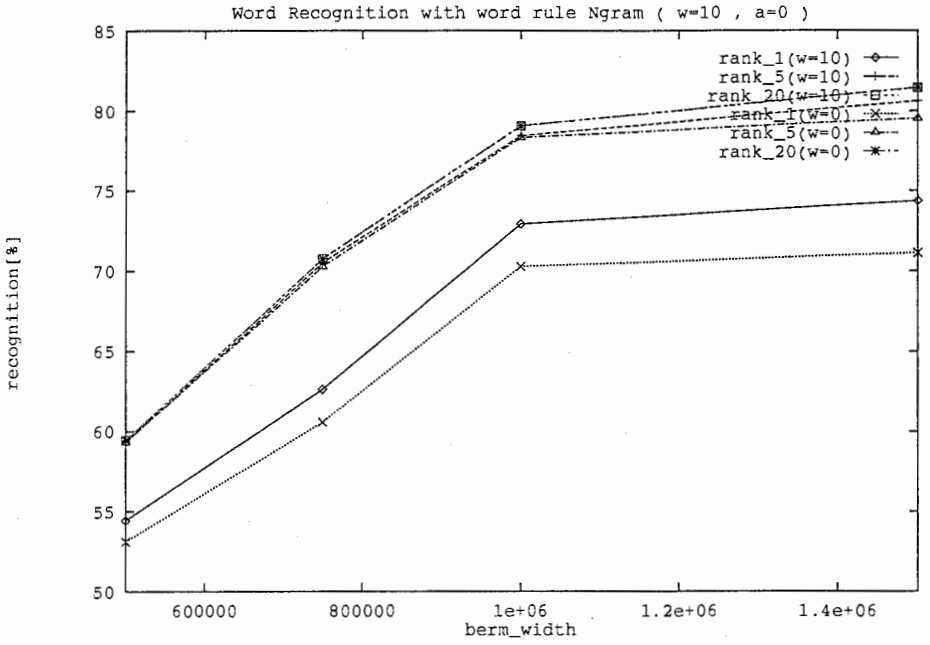


図 4.23: 単語認識率 (Closed rule) ／ word rule N-gram ／ ALL.AVERAGE

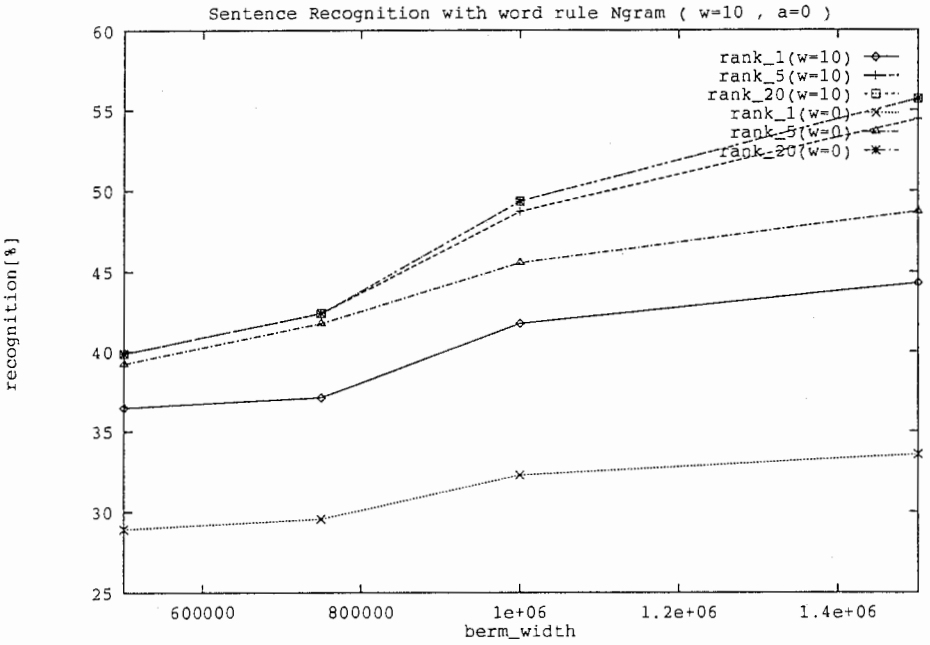


図 4.24: 文認識率 (Closed rule) ／ word rule N-gram ／ ALL.AVERAGE

$\omega = 0, \alpha = 0$ で再順序づけした時

(1) s,u,z,u,k,i,n,o,k,o,t,o,o,sh,i,m,a,s,u score= 65129640 41.219 65129640 41.219
[1:1] 鈴木／の／こと／を／し／ま／す
Rule_Score 0
(2) s,u,z,u,k,i,n,a,o,k,o,t,o,m,o,o,sh,i,m,a,s,u score= 64936346 41.097 64936346 41.096
[2:1] 鈴木／直子／と／申／し／ま／す
Rule_Score 0

$\omega = 10,\ \alpha = 0$ で再順序づけした時

(1) s,u,z,u,k,i,n,a,o,k,o,t,o,m,o,o,sh,i,m,a,s,u score= 64936346 41.097 63897398 40.439
[1:1] 鈴木／直子／と／申／し／ま／す
Rule_Score -1038947
(2) s,u,z,u,k,i,n,o,k,o,t,o,o,sh,i,m,a,s,u score= 65129640 41.219 63255679 40.033
[2:1] 鈴木／の／こと／を／し／ま／す
Rule_Score -2066907

図 4.25: 再順序づけの例

## 5 まとめ

今回の実験で、ルールバイグラムの有用性は示せた。

また、wordまでのrule listによって構築されるrule N gramとpreterminalまでのrule listによって構築されるrule N gramに若干の認識率の差が現れ、wordまでという制約のきつい方が認識率が高くなるという結果を得た。

今後の課題としては、まず、rule N gramを音声認識に組み込んで、認識率が少なくとも今回の実験の認識率ぐらいにまでは向上することを確認する必要がある。

また、今回の実験におけるrule N gramではなくpreterminal N gramを用いた実験を行ない、rule N gramとpreterminal N gramをどのように比較していくかも考えていく必要がある。

さらに、rule N gramとpreterminal N gramは言語モデルとしては独立なので、音声認識への適用の箇所が異なり、併用が可能である。よって、併用した場合の実験も行なっていく必要がある。

## 参考文献


[1] 竹澤寿幸, 森元逞: “部分木に基づく構文規則と前終端記号バイグラムを併用する対話音声認識手法”, 電子情報通信学会, VOL.J79-D-II NO.12 DECEMBER, (1996.12)

[2] 竹澤寿幸, 森元逞: “構文規則と前終端記号バイグラムを併用する対話音声認識手法の高速化と高性能化”, 電子情報通信学会, 信学技報 NLC96-53 SP96-84, (1996.12)

[3] 池田徹志, 竹澤寿幸: “音声認識結果の再順序づけにおける言語モデルの比較検討”, ATR テクニカルレポート, TR-IT-0126 (1995.8)

[4] 古谷成章, 竹澤寿幸: “ポーズ単位に基づく音声言語統合処理における音声認識候補の言語解析実験”, ATR テクニカルレポート, TR-IT-0153 (1996.2)

[5] 田代敏久: “音声言語処理のための構文解析ツールキットユーザーズマニュアル”, ATR テクニカルレポート, TR-IT-0142 (1995.12)

[6] 堤真理子, 周旻, 甲斐充彦, 中川聖一: “対話音声認識の言語制約としての文脈自由文法と統合的モデルの比較”, 日本音響学会, 日本音響学会講演論文集 p.175, (1996.3)

## A　今回作成したツール、実験結果

/home/as22/xskogure　以下のディレクトリを説明します。
詳細は各ディレクトリの README ファイルを参照して下さい。

```
Database/
    Training/   50 会話の正解木
        Tree/       正解木のデータ
        Data/       Tree/ 以下のファイルから木構造のみを切り出したデータ
        Ruledata/   Data/ 以下のファイルから作成されたルール列
        Work/       バイグラム学習のワークディレクトリ
    Test/       評価用 12 会話の正解木
        Tree/       正解木のデータ
        Data/       Tree/ 以下のファイルから木構造のみを切り出したデータ
        Ruledata/   Data/ 以下のファイルから作成されたルール列
        Work/       バイグラム学習のワークディレクトリ
    Reordering/
        Open/     50 会話の正解木のルールバイグラムを使用
            Work/         再順序づけ実験のワークディレクトリ
            TESTAL-SD/    音声認識結果、再順序づけ実験結果
        Closed/   評価用 12 会話の正解木のルールバイグラムを使用
            Work/         再順序づけ実験のワークディレクトリ
            TESTAL-SD/    音声認識結果、再順序づけ実験結果
Tools/
    Perl/       実験のため作成した perl プログラム
    C/          実験のため作成した C プログラム (ソース)
    HP-UX/      実験のため作成した C プログラム、HP 用実行ファイル
    SunOS/      実験のため作成した C プログラム、Sun 用実行ファイル
```