TR-I-0131

HMM音韻認識における音韻連鎖情報の利用

北川 英一郎†, 伊藤 克亘††
花沢 利行, 川端 豪, 鹿野 清宏

(ATR自動翻訳電話研究所)
ATR Interpreting Telephony Labs.

1990.1

内容梗概

音声認識のための言語モデルの検討を行った。ここでは特に、語彙の仮定をせず日本語音節が持つ連鎖統計情報のみを用いる言語モデル、すなわち音節連鎖のn-gram (n = 0, 1, 2, 3, 4)モデルを検討した。

また言語モデルの評価として、HMM音韻モデルとの組合せによる連続音声中の音韻の認識実験を行った。

本報告は、学外実習生 北川 英一郎(早稲田大学)と、伊藤 克亘(東京工業大学)が行った実習の報告書である。

---

†早稲田大学修士課程,　　††東京工業大学修士課程

実習報告

HMM-LR音声認識システムと音節トライグラムを用いた音韻タイプライタの検討

1989年　8月31日

早稲田大学修士1年

白井研究室　北川英一郎

1.はじめに

LRパーザーに日本語音節の文法を用いることにより、日本語の発音としてはあり得ない音韻の組み合せを排除して効率よく音韻認識を進めることができる。しかし、発音としてあり得ても、日本語の語彙としては存在しないか、または通常は使われない組み合せは数多く存在する。そのような意味のない組合せをさらに除外するためには、音節のトライグラムを用いて日本語の局所的な確率構造を有効に利用するのが良いと期待できる。このトライグラムの有効性を確認･評価するために実験を行う。

比較のため、日本語音節の連鎖を表現する文法を用いたHMM-LR法により、文節に対して音韻認識を行う。次に日本語の音節を単位としたトライグラムによりHMMの認識スコアを修正し、結果を比較検討する。

2.システム構成

実験に用いたシステムの構成を図1に示す。

図1.(a)はトライグラムを用いないHMM-LRシステムである。まず始めに日本語音節の文法規則を作成し、それを予めLRテーブルに変換しておく。HMM-LRパーザーが音韻を予測し、その音韻のHMMモデルと入力音声データーが照合される。照合の結果その確率を示すスコアが返される。予測された音韻系列のスコアにより、ビームサーチを行いながら文節の音韻を認識する。

図2.(b)はトライグラムを用いてスコアを修正するHMM-LRシステムである。HMM-LRパーザーが予測した音韻にトライグラムモデルを対応させてトライグラムによるスコアを求める。このスコアに適当な重みをかけてHMMモデルのスコアと足し合わせてスコアを修正して、このスコアをもとにビームーサーチにより文節に対して音韻認識をする。

3.音声資料

HMMモデルの学習データ及び評価データを表1に、トライグラム作成用データを表1及び表2に示す。

これらの試料を12kHzでサンプリングし窓長21.3msec、周期3msecのハミング窓で切り出し12次のLPC分析を行う。分析により(1)スペクトル(WLR)、(2)ケプストラムの差分、(3)パワーの3種の特徴量を抽出し、セパレートベクトル量子化を行う。量子点数は、順に256、256、64とする。

4.HMM音韻モデル

4.1HMM音韻モデルの学習

スペクトル(WLR)、ケプストラムの差分、パワーの3種の特徴量を用いてセパレートベクトル量子化を行い、その量子化コードにより学習する。

4.2HMM音韻認識

継続時間長を制御して、トレリスアルゴリズムにより認識する。LRパーザーにより予測された音韻系列をもとに音韻モデルを連結し、ビームーサーチにより枝刈りしながら入力音声ベクトル量子化(VQ)コードとの照合を行う。音韻モデルを文節にまで組み上げ最終的にスコアが1位となる文節を決定する。

5.トライグラム

表2に示したトライグラム用のデータベースより、予め日本語音節のトライグラムの確率を計算しておく。

予測LRパーザーの予測する音韻系列を音節の系列に変換し、日本語音節を単位としたトライグラムを用いて、音節の系列が生成される確率値を計算する。その確率値の対数をとってスコアにし、ウエイトをかけてHMMのスコアとの和をとり、それを新たなスコアとする。

P_H　HMMモデルによる確率
SY_i　音節の系列
P_i　トライグラムの確率
P_T　トライグラムによる文節の確率
S　HMMモデルとトライグラムモデルの双方を評価したトータルのスコア

$$P_i = P(SY_i | SY_{i-1}, SY_{i-2}) \quad , \quad SY0 = SY\text{-}1 = Q(\text{無音}) \qquad (1 \leqq i \leqq n)$$
$$P_T = P1 \cdot P2 \cdots \cdot Pn$$
$$S = (1.0 - w)\log P_H + w\log P_T \qquad (w = 0.12)$$

6.日本語音節文法

LRパーザーに与える日本語音節文法を図2に示す。

音節の文法を与えることにより、HMMの音韻認識結果をそのまま音節に変換し、日本語の表記にすることができる。ここでは撥音を記号/=/で表している。

7.文節音声の音韻認識実験

7.1　実験条件

トライグラムを用いるときと用いないときのどちらも同じ条件で実験を行う。VQデータを間引きしてフレーム周期を9msecとし、計算時間の短縮のためにビームサーチによる枝刈りを行う。HMMのスコアに対して閾値を設け、その値以下の確率のときも枝刈りする。枝刈りの条件を表3に示す。

7.2　実験結果

文節No1から文節No60まで実験を行った。音韻認識の結果は、文字列のDPマッチングにより正解音韻と比較して、正解(Correct),脱落(Delete),挿入(Insert),置換(Substitute)の4種の基準の評価を行った。その結果を表4に示す。

上記の結果を集計して、以下の計算式により、2通りの音韻認識率を計算した。

1つめは単純な正解率(Percent Correct)で、、発声した文節の音韻中のどれだけが正解音韻として検出されたかを表す。

2つめは認識結果の正確さを考慮した認識率(PercentAccuracy)である。

Correct　認識結果の正解音韻数
Subs　置換された音韻数
Dels　脱落した音韻数
Ins　挿入された音韻数
CorrectLength　文節の正しい音韻数

PercentCorrect = Correct/CorrectLength

PercentAccuracy = (Correct-Subs-Dels-Ins)/CorrectLength

計算した音韻認識率を表5に示す。

表5を見るとPercentCorrectで4.80%,Percent Accuracyで9.64%認識率が向上しており、HMMのスコアをトライグラムで修正することが非常に有効であるとわかる。次にさらに詳しく検討を進めていく。

ビームサーチにより、どれだけの文節が枝刈りされたかを表6に示す。

表の中でLocalは1つの接点から伸びる枝の数による枝刈りを、Grobalは全体の枝の数(ビーム幅)による枝刈りを表す(条件は表3を参照)。

これを見ると、トライグラムを用いることによって、枝刈りされる文節の数は半分近くに減っている。トライグラムの効果はパーザーにAcceptされる文節の音韻認識率を向上させるだけでなく、本来認識される文節がビームサーチによる枝刈りにより落ちてしまうことも防いでいると言える。しかも、トライグラムを用いたときは、ほとんどがLocalで枝刈りされており、実験条件を変えてLocalの枝数をさらに増やせは、Acceptされる文節も増えて音韻認識率もさらに向上するものと期待される。

トライグラムによる効果を文節ごとに比較したのが表7である。表中で＋は認識率の増加を、－は減少を表している。

これを見るとPercent CorrectとPercent Accuracyの両方もしく片方が増えている文節は26個、両方もしくは片方が減っている文節は4個、変化のない文節は24個、片方が増えて片方が減っている文節が6個となっている。すなわち全60文節中の40%以上がトライグラムにより向上しており、完全に悪影響を受けている文節はわずか7%である。このことからもトライグラムの効果がわかる。

文節認識の具体例の1部を図3に示す。

No1,No8は改善例、No37は余分な文字が付加された例、No5は一部改善されて一部脱落した例である。

最後に音節パープレキシティを計算して表8に示す。

ここでは、トライグラムの確率は音節が単位となっているため、音節のパープレキシティを求めた。音韻のパープレキシティは、音節文法が複雑で状態確率が求められなかったため計算できなかった。音韻のパープレキシティでないため、LR文法で音韻の連鎖として音節を定義した効果は含まれていない。

音節パープレキシティはトライグラムを用いた方が圧倒的に少なく、ある音韻の次に来る音韻をトライグラムが良く限定していることがわかる。

音節パープレキシティは、以下の式で計算した。

p(i,j,k)	音節*SY*i,*SY*j,*SY*kと連鎖する状態の確率
p(i|j,k)	音節*SY*i,*SY*jと連鎖した状態のときに*SY*kが続く確率
H	1音節当たりのエントロピー
F	パープレキシティ

$$H = -\sum_{i}\sum_{j}\sum_{k} p(i,j,k)\log p(i|j,k)$$

$$F = 2^{H}$$

8.まとめ

HMM-LR音韻認識システムのスコアをトライグラムを用いて修正することにより、高い音韻認識率が得られることが確認された。60文節中の438音韻において、Percent Correctで85.85%、Percent Accuracyで81.05%の音韻認識率が得られた。また、音節のパープレキシティはトライグラムを用いない値111から5.65に減少させることができた。トライグラムにより、音節の連鎖を良く限定していることが確認された。今後はトライグラム作成用のデータベースの音節分布に対する考察が必要であろう。

9.謝辞

一か月にわたって指導してくださった川端さん、花沢さん、一緒にテニスの試合をしてくださった鹿野さん、研究の機会を与えてくださった樽松社長、他ATRの皆さんに心から感謝します。HMMのことは、ATRに来るまでは何も知らなかった私も多少ながらHMMのことについてわかるようになりました。ATRに来て本当に良かったと思います。

10.参考文献

北研二、川端豪、斎藤博昭
HMM音韻認識と拡張LR構文解析法を用いた連続音声認識
ATR Technical Report

花沢利行、中村哲、川端豪
ベクトル量子化アルゴリズムのHMM文節・認識による評価

花沢利行、北研二、鹿野清宏
HMM音韻モデルの文認識による評価
日本音響学会講演論文集3－6－6

L.R.Rabiner,B.H.Juang
An Introduction to Hidden Markov Models
IEEE ASSP MAGAZINE JANUARY 1986

S.E.Levinson,L.R.Rabiner,M.M.Sondhi
An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition

中川聖一
確率モデルによる音声認識
電子情報通信学会

表1. 音声試料

話者	男性1名(MAU)
HMM学習データ	重要語5240単語と バランス216単語中の音韻
トライグラム 作成データ	表2に示したシラブルデータ
評価データ	国際会議の問合せを想定した会話文中の 279文節

表2 シラブルデータベース

出典	音節数
甘えの構造	45,434
キーボード会話	58,869
ニューズウィーク	38,272
朝日新聞声の欄	41,160
電話会話	170,156
合計	353,891

表3 ビームサーチの条件

条件	トライグラムを 用いないとき	トライグラムを 用いるとき
ビーム幅	170	170
枝分かれ数	5	5

表4　実験の集計結果(文節No1から文節No60まで)

評価	トライグラムなし	トライグラムあり
正解	355	376
脱落	21	27
挿入	41	21
置換	62	35

表5　音韻認識率(文節No1から文節No 60まで)

音韻認識率	トライグラムなし	トライグラムあり
Percent Correct	81.05	85.85
Percent Accuracy	71.69	81.05

表6　ビームサーチにおいて枝刈りされた文節の個数と割合
(文節No1から文節No60まで)

枝刈り場所	トライグラムなし	トライグラムあり
Local	20	15
Grobal	11	3
計	31	18

表7トライグラムにより音韻認識率の変化した文節数(＋は増加、-は減少、0は変化なし)

Percent Correct	Percent Accuracy	認識率の変化した文節数
＋	＋	21
＋	0	0
0	＋	5
0	0	24
－	＋	4
＋	－	2
－	0	0
0	－	2
－	－	2

表8　音節パープレキシティ

	音節パープレキシティ
トライグラムなし	111
トライグラムあり	5.65

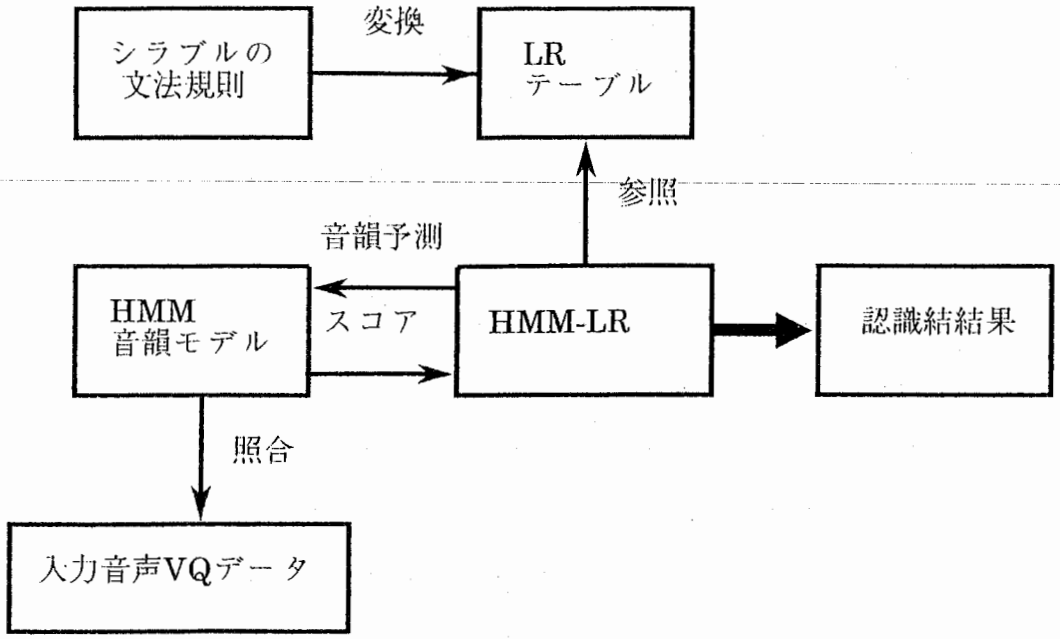

図1.(a)HMM-LRシステム(トライグラムなし)

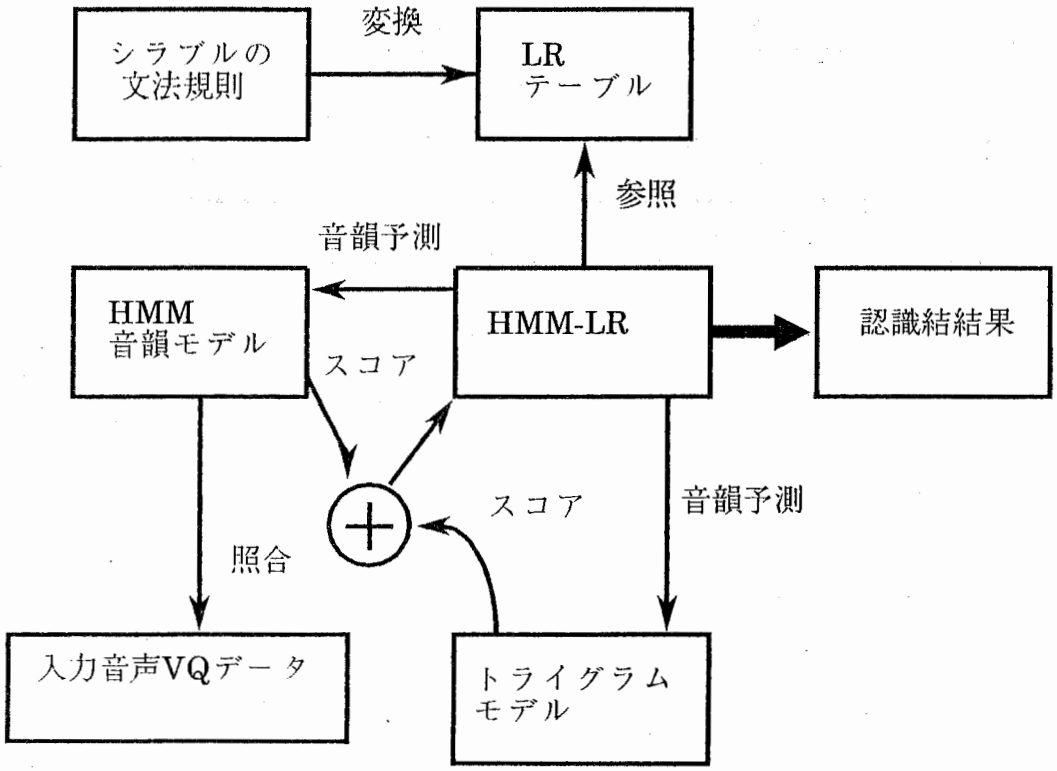

図1.(b)HMM-LRシステム(トライグラムを含む)

図－2　日本語文節文法

Oct 23 19:25 1989 /ifg1/G-TH/sp51.gra Page 1

```
;;
;; Generate Japanese syllablic contexts
;; Aug. 8.1989 by E.Kitagawa
;;
(<start> <--> (<_start>))
(<_start> <--> (q1 <sequence> q2))
(<sequence> <--> (<syllable_f>))
(<sequence> <--> (<syllable_f> =))
(<sequence> <--> (<sequence> <syllable>))
(<syllable> <--> (<syllable_f>))
(<syllable> <--> (<syllable_f> =))
(<syllable> <--> (<syllable_n>))
(<syllable> <--> (<syllable_n> =))
(<syllable_f> <--> (w a))
(<syllable_f> <--> (<vowel>))
(<syllable_f> <--> (b <vowel>))
(<syllable_f> <--> (g <vowel>))
(<syllable_f> <--> (m <vowel>))
(<syllable_f> <--> (n <vowel>))
(<syllable_f> <--> (r <vowel>))
(<syllable_f> <--> (z <vowel>))
(<syllable_f> <--> (d <a_e_o>))
(<syllable_f> <--> (t <a_e_o>))
(<syllable_f> <--> (s <a_e_o>))
(<syllable_f> <--> (h <a_e_o>))
(<syllable_f> <--> (p <a_e_o>))
(<syllable_f> <--> (k <a_e_o>))
(<syllable_f> <--> (y <a_u_o>))
(<syllable_f> <--> (by <a_u_o>))
(<syllable_f> <--> (gy <a_u_o>))
(<syllable_f> <--> (hy <a_u_o>))
(<syllable_f> <--> (my <a_u_o>))
(<syllable_f> <--> (ny <a_u_o>))
(<syllable_f> <--> (ry <a_u_o>))
(<syllable_f> <--> (zy <a_u_o>))
(<syllable_f> <--> (sy <a_u_o>))
(<syllable_f> <--> (py <a_u_o>))
(<syllable_f> <--> (cy <a_u_o>))
(<syllable_f> <--> (ky <a_u_o>))
(<syllable_f> <--> (sh i2))
(<syllable_f> <--> (h i2))
(<syllable_f> <--> (ch i2))
(<syllable_f> <--> (p i2))
(<syllable_f> <--> (k i2))
(<syllable_f> <--> (s u2))
(<syllable_f> <--> (h u2))
(<syllable_f> <--> (ts u2))
(<syllable_f> <--> (p u2))
(<syllable_f> <--> (k u2))
(<syllable_f> <--> (sh ii))
(<syllable_f> <--> (h ii))
(<syllable_f> <--> (ch ii))
(<syllable_f> <--> (p ii))
(<syllable_f> <--> (k ii))
(<syllable_f> <--> (s uu))
(<syllable_f> <--> (h uu))
(<syllable_f> <--> (ts uu))
(<syllable_f> <--> (p uu))
(<syllable_f> <--> (k uu))
(<syllable_n> <--> (q t <a_e_o>))
(<syllable_n> <--> (q s <a_e_o>))
(<syllable_n> <--> (q p <a_e_o>))
(<syllable_n> <--> (q k <a_e_o>))
(<syllable_n> <--> (q sy <a_u_o>))
(<syllable_n> <--> (q py <a_u_o>))
```

```
(<syllable_n> <--> (q cy <a_u_o>))
(<syllable_n> <--> (q ky <a_u_o>))
(<syllable_n> <--> (q sh i2))
(<syllable_n> <--> (q ch i2))
(<syllable_n> <--> (q p i2))
(<syllable_n> <--> (q k i2))
(<syllable_n> <--> (q s u2))
(<syllable_n> <--> (q ts u2))
(<syllable_n> <--> (q p u2))
(<syllable_n> <--> (q k u2))
(<syllable_n> <--> (q sh ii))
(<syllable_n> <--> (q ch ii))
(<syllable_n> <--> (q p ii))
(<syllable_n> <--> (q k ii))
(<syllable_n> <--> (q s uu))
(<syllable_n> <--> (q ts uu))
(<syllable_n> <--> (q p uu))
(<syllable_n> <--> (q k uu))
(<vowel> <--> (i))
(<vowel> <--> (ii))
(<vowel> <--> (e))
(<vowel> <--> (ee))
(<vowel> <--> (ei))
(<vowel> <--> (<a_u_o>))
(<a_u_o> <--> (a))
(<a_u_o> <--> (aa))
(<a_u_o> <--> (u))
(<a_u_o> <--> (uu))
(<a_u_o> <--> (o))
(<a_u_o> <--> (oo))
(<a_u_o> <--> (ou))
(<a_e_o> <--> (a))
(<a_e_o> <--> (aa))
(<a_e_o> <--> (e))
(<a_e_o> <--> (ee))
(<a_e_o> <--> (ei))
(<a_e_o> <--> (o))
(<a_e_o> <--> (oo))
(<a_e_o> <--> (ou))
```

図3　文節認識例

No1　正解音韻　　d-a-i-i-Q-k-a-i
　　認識音韻(トライグラムなし)　d-a-r-i-i-i-Q-k-u-a-i
　　認識音韻(トライグラムあり)　d-a-ii-Q-k-a-i

No5　正解音韻　　k-a-i-g-i-N-i
　　認識音韻(トライグラムなし)　k-u-a-i-i-u-i-i-N-u
　　認識音韻(トライグラムあり)　k-a-ii-N

No8　正解音韻　　g-o-k-i-b-ou-s-a-r-e-r-u
　　認識音韻(トライグラムなし)　g-o-k-i-b-ou-s-a-r-u-r-u
　　認識音韻(トライグラムあり)　g-o-k-i-b-ou-s-a-r-e-r-u

No37　正解音韻　　s-a-N-k-a-o
　　認識音韻(トライグラムなし)　s-a-N-k-a-o
　　認識音韻(トライグラムあり)　s-a-N-k-a-oo-k-u

付録　全文節認識結果

文節番号	正解数の差	正解－挿入の差	トライグラムなし				トライグラムあり			
			正解	脱落	挿入	置換	正解	脱落	挿入	置換
1	−2	1	8	0	3	0	6	1	0	1
2	0	0	6	0	1	0	6	0	1	0
3	0	2	4	0	2	1	4	0	0	1
4	0	0	7	0	0	0	7	0	0	0
5	3	1	5	1	4	1	2	3	0	2
6	−1	−1	6	0	0	1	7	0	0	0
7	0	1	6	0	3	1	6	0	2	1
8	1	1	11	0	0	1	12	0	0	0
9	0	0	6	0	0	0	6	0	0	0
10	1	-2	6	0	0	0	5	0	1	1
11	0	0	7	0	0	1	7	0	0	1
12	1	−1	3	2	0	1	4	0	2	2
13	0	−1	3	0	0	1	3	0	1	1
14	1	1	5	0	0	1	6	0	0	0
15	1	1	4	0	0	1	5	0	0	0
16	0	0	6	0	0	0	6	0	0	0
17	1	2	4	0	1	1	5	0	0	0
18	0	4	8	0	4	0	8	0	0	0
19	0	0	7	0	0	0	7	0	0	0
20	0	0	2	2	0	1	2	2	0	1
21	4	5	6	1	1	5	10	1	0	1
22	1	1	8	0	0	1	9	0	0	0
23	0	0	7	0	0	0	7	0	0	0

文節番号	正解数の差	正解－挿入の差	トライグラムなし				トライグラムあり			
			正解	脱落	挿入	置換	正解	脱落	挿入	置換
24	0	0	3	0	0	0	3	0	0	0
25	0	1	6	0	0	2	6	1	1	1
26	0	0	6	1	0	1	6	1	0	1
27	2	2	4	0	1	2	6	0	1	0
28	0	1	5	0	1	0	5	0	0	0
29	0	0	7	0	0	0	7	0	0	0
30	0	0	2	2	0	1	2	2	0	1
31	−1	4	7	1	5	4	6	3	0	3
32	1	1	7	0	0	1	8	0	0	0
33	0	0	6	0	2	0	6	0	2	0
34	1	4	4	0	3	1	5	0	0	0
35	0	0	7	0	0	0	7	0	0	0
36	1	1	5	1	0	2	6	1	0	1
37	−1	−3	6	0	0	0	5	0	2	1
38	3	3	10	1	0	6	13	2	0	2
39	0	0	6	2	0	0	6	2	0	0
40	0	0	4	0	0	0	4	0	0	0
41	0	0	4	0	0	0	4	0	0	0
42	0	0	6	2	1	1	6	2	1	1
43	1	1	5	0	0	1	6	0	0	0
44	2	1	8	1	0	5	10	2	1	2
45	0	0	6	0	1	0	6	0	1	0
46	1	2	6	0	1	1	7	0	0	0

文節番号	正解数の差	正解－挿入の差	トライグラムなし				トライグラムあり			
			正解	脱落	挿入	置換	正解	脱落	挿入	置換
47	0	0	7	0	0	0	7	0	0	0
48	−1	1	3	2	0	2	2	2	0	3
49	1	1	8	1	0	0	9	0	0	0
50	2	2	5	0	0	3	7	0	0	1
51	0	−1	6	0	0	0	6	0	1	0
52	1	1	7	0	0	1	8	0	0	0
53	0	0	5	0	1	0	5	0	1	0
54	−1	0	6	0	2	1	5	1	1	1
55	0	0	5	0	0	0	5	0	0	0
56	0	0	9	1	0	0	9	1	0	0
57	0	0	6	0	0	0	6	0	0	0
58	1	1	5	0	1	5	6	0	1	4
59	2	3	5	0	1	2	7	0	0	0
60	1	2	13	0	2	2	14	0	1	1

実習報告

HMM音韻認識における音節連鎖統計情報の利用

東京工業大学修士2年
田中研究室
伊藤克亘

1989/12/22

1 はじめに

LR パーザに日本語音節の文法を用いることで音韻タイプライターを構成すると、日本語の発音としてはありえない音韻の組み合わせ (例えば、子音は連続しない、など) を排除することはできるが、それだけでは数多くの候補が生成されてしまう。

しかし、それらの候補のなかには日本語の発音としてはありえるが通常は用いられないような音韻の組み合わせを持つものは数多く存在すると考えられ、それらを除外することで、効果的な候補の絞り込みが可能になるだろう。そのように「日本語らしくない」音韻の組み合わせを除外するためには、日本語の音節連鎖に関する局所的な確率構造を用いることが有効であると期待できる。[2]

そのような確率構造を表現するモデルとして音節のトライグラムが考えられる。しかしトライグラムは組合せの数が非常に多いため実際に存在する三つ組をカバーするのは、かなり大量の訓練データを用いても難しい。

例

シラブル・データベース
5 つのうち最大のデータベース
39919 文節に出現する組合せ 11456 (全体の 0.9%)
ほかの 4 つのデータベースに対して 80% 程度しかカバーしない。
81053 文節に出現する組合せ 23809 (全体の 1.9%)

そこで、三つ組の出現頻度をスムージングする必要がある。ここでは、削除補間法 (deleted interpolation method) [4] を用いて、三つ組、二つ組、一つ組の出現頻度と、音韻が等確率に出現した場合の情報を用いてスムージングを行い、日本語の音韻トライグラムの性質と、それを音声認識に用いる場合の有効な利用法について検討を行う。

2 削除補間法について

確率モデルを他のモデルを用いて補間するためには、モデルの推定用のデータと、評価用のデータが必要である。よいモデルを得るためには限られたデータのうち、推定用にも、評価用にもできるだけ多くのデータを利用したい。そこで、データを分割して、交互に推定と評価を行い、得られた結果を平均してモデルの推定と補間を行うのが削除補間法である。したがって、この方法では推定にも評価にも全サンプルを用いることができる。

ここでは、直前の $N-1$ 個のシラブルから次のシラブルを予測する言語モデル P^N を次の式で置き換える。

$$P^N = \sum_{i=0}^{N} q_i f_i$$

f_i: i-gram モデルの確率値 (f_0 はシラブルが均等に出現すると仮定した確率)

ここで、

$$\sum_{i=0}^{N} q_i = 1$$

これは、f_i の推定精度が悪い場合には、f_{i-1} によって、f_{i-1} の推定精度も悪い場合には f_{i-2} によってというように近似する方法である。

q_i は次のように求める。

$$\hat{q}_i = \frac{1}{N_s} \sum_{j=1}^{N_s} c_i^j$$

ここで

$$c_i^j = \frac{q_i f_i^j}{\sum_{k=0}^{N} q_k f_k^i}$$

f_i^j : j 番目のサンプルを除いたデータから得た i-gram モデルの確率値
N_s : 訓練データのシラブル数

このようにして繰り返していくと生成確率 (3.1.1参照) について最適な q_i を決定することができる。この算出は EM アルゴリズムになっている。[3]

3 音節のトライグラム

3.1 言語モデルの評価方法

言語モデルの評価基準として、以下に述べる方法を用いた。

3.1.1 生成確率

次の式であたえられる P_{gen} が生成確率である。これは、文節中の一つの音節が、その言語モデルによって生成 (=予測) される確率の平均の値である

$$P_{gen} = \frac{1}{N_s} \sum_{i=1}^{N_s} p_i$$

N_s : 文節の音節数 (文節の最後の無音を 1 つとして数える)
p_i : 文節の i 番目の音節に対する言語モデルの確率値

3.1.2 test-set perplexity

test-set perplexity F_{test} は次の式で定義される。

$$F_{test} = 2^{H_{test}}$$

ただし、

$$H_{test} = -\frac{1}{N_p} \sum_{i:\text{テストデータ中の全音韻}} \sum_{s:i\text{ から分岐可能な音韻}} p(s) \log p(s)$$

H_{test}: テストデータを正しく解析するときに遭遇する分岐点 (音韻単位) での平均エントロピー
$p(s)$: s に対する言語モデルの確率
N_p : テストデータ中の音韻数

この量は、テストに使用する文を日本語音節文法で構文解析するときに実際に遭遇する分岐点での平均分岐数を数えあげたものである。

3.1.3 coverage

テストデータに出現する全パタンのうち、カバーできたパタンの割合を示す。

3.1.4 音節パープレキシティ

音節パープレキシティ F_m は次の式で定義される。

$$F_m = 2^{H_m}$$

ただし、

$$H_m = -\sum_{i:\text{全てのパタン}} p(i) \log f(i)$$

p(i): 訓練データ中にパタン i が出現する確率
f(i): パタン i に対する n-gram モデルの確率値
H_m: 1 音節当たりのエントロピー

3.2 トライグラムの削除補間法による補間

前節で述べた削除補間法によって、トライグラムの補間を行なった。そのときの条件を表1に示す。

表 1: 収束条件など

収束条件	$\forall i \mid q_i - \hat{q}_i \mid < 1.0 \times 10^{-8}$
訓練データの文節数	81053
音節数	353891
シンボルの種類	111

この条件で表2の係数が得られた。

表 2: 削除補間法によって求めた係数

q_0	0.00133
q_1	0.01599
q_2	0.04524
q_3	0.93744

このようにして得られたモデルの性質について調べた結果を以下で述べる。

3.2.1 トライグラムのタスク依存性

訓練データの母集団の違いによって、トライグラムなどの性質がどのように異なるのかを調べるために、データベース毎にトライグラムなどを作成した。それぞれのデータベースについてのデータを表3に示す。次に、互いに異なるデータベースをテストデータとしてそれぞれのトライグラムなどの生成確率と、coverage を求めた。その結果を表4に示す。

表 3: シラブルデータベース

	出典	音節数
jpn	甘えの構造	45434
key	キーボード会話	58869
new	ニューズウィーク	38272
ppr	朝日新聞声の欄	41160
tel	電話会話	353891

この表から、ユニグラム (音節の出現確率) はこれらのデータベース程度の規模であれば、母集団には関係なくかなり精度のよいものが得られると考えられる。

しかし、バイグラム・トライグラムに関しては coverage がテストデータによってかなりばらつくことからこれらのデータベース程度の規模では、統計量を求めるのにはデータ

表 4: 言語モデルのタスク依存関係

収束条件: 収束回数 10 回

シンボル数 : 111

	jpn	key	new	ppr	tel
jpn		-1.58 -2.21(74) -1.69(95) -1.50(100)	-1.47 -2.00(78) -1.59(96) -1.47(100)	-1.47 -2.00(78) -1.60(96) -1.45(100)	-1.57 -2.19(75) -1.68(96) -1.45(100)
key	-1.52 -2.00(79) -1.62(97) -1.44(100)		-1.67 -2.29(73) -1.72(95) -1.47(100)	-1.59 -2.14(76) -1.66(96) -1.46(100)	-1.10 -1.26(93) -1.37(99) -1.42(100)
new	-1.30 -1.67(85) -1.47(98) -1.43(100)	-1.57 -2.31(71) -1.66(96) -1.50(100)		-1.43 -1.96(79) -1.57(97) -1.45(100)	-1.60 -2.37(71) -1.71(95) -1.46(100)
ppr	-1.26 -1.57(88) -1.43(99) -1.43(100)	-1.33 -1.72(84) -1.50(98) -1.49(100)	-1.38 -1.83(82) -1.53(98) -1.46(100)		-1.39 -1.91(81) -1.56(97) -1.45(100)
tel	-1.52 -1.82(84) -1.60(98) -1.46(100)	-1.05 -1.13(96) -1.36(99) -1.48(100)	-1.71 -2.12(78) -1.71(96) -1.49(100)	-1.63 -1.99(81) -1.66(97) -1.47(100)	

表内の各ブロックは上から

補間したモデル
トライグラムモデル
バイグラムモデル
ユニグラムモデル

の順に並んでいる。

ただし、

カバーできないパタンの確率は 1.0×10^{-5} に置き換えている
確率値は常用対数で示されている
() 内は coverage である
縦が訓練用、横が評価用である

が不足していると考えられる。また、このデータベースのうち tel と key はかなり似ているタスクなのだが、表を見るとこの 2 つのデータベースの組み合わせは、他の組み合わせに比べて、バイグラム・トライグラムどちらの場合も非常に高い生成確率と coverage を示していることから、出現するパタンの種類もその出現確率もタスクによって変化する可能性も考えられる。

削除補間法で補間したモデルは、すべての組み合わせでトライグラムより生成確率が大きくなっており、生成確率についてはかなり精度がよくなっていることがわかる。

3.2.2 トライグラムと訓練データ量の関係

次に、訓練用のデータの量とモデルの精度について調べてみる。もっともデータ数の多い tel を分割して訓練に用い、3.2.1 の実験で tel に対して平均的な結果が得られた jpn をテストに用いて実験を行なった。その結果を表5に示す。

表 5: データ量と統計モデルの精度の関係

収束条件: 収束回数 10 回
シンボル数 : 111

	1/5	2/5	3/5	4/5	全体
補間したモデル	-1.55	-1.54	-1.54	-1.53	-1.52
トライグラムモデル	-2.20(73)	-2.01(79)	-1.93(81)	-1.87(83)	-1.82(84)
バイグラムモデル	-1.68(95)	-1.63(97)	-1.62(97)	-1.61(97)	-1.60(98)
ユニグラムモデル	-1.46(100)	-1.46(100)	-1.46(100)	-1.46(100)	-1.46(100)
q_0	0.004	0.003	0.002	0.001	0.001
q_1	0.037	0.022	0.016	0.013	0.010
q_2	0.117	0.079	0.064	0.057	0.051
q_3	0.842	0.896	0.918	0.929	0.938
出現する三つ組のパタン数	5852	7960	9367	10595	11456
出現する二つ組のパタン数	1843	2210	2388	2536	2626
出現するシンボルの数	92	94	95	96	96

ただし、

カバーできないパタンの確率は 1.0×10^{-5} に置き換えている
また、各モデルの欄には生成確率が常用対数で示されている

この結果から、ユニグラムについては訓練用のデータが 7984 文節 (1/5 のとき) でデータが十分であることがわかる。また、バイグラムについても jpn というタスクについては 39919 文節でかなり十分な量に近付いてきていると考えられる。しかし、トライグラムについては、まだまだ不十分であり、39919 文節のときとテキストデータベース全体である

81053 文節のときとを比べても訓練データの数が倍になるのと同時に三つ組のパタン数も倍になっているので、81053 文節 (全データベース使用時) でも不十分だと考えられる。

トライグラムによる生成確率がかなり大きく変化するのに比べて、補間したモデルによる生成確率は少しずつ大きくなっているもののかなり安定した値になっている。また、どの確率もトライグラムよりかなり大きな値となっている。このことから、削除補間法による補間は生成確率についてはデータが不十分なほど効果的であるといえるであろう。

4 文節音声の音韻認識実験 (その 1)

4.1 実験条件

実験に用いたシステム [5] の構成を図 1 に示す。図 1(a) はトライグラムなどの言語モデルを用いない HMM-LR システムである。パーザが日本語音節の文法規則に従った音韻を予測し、その音韻の HMM モデルと入力音声が照合される。照合の結果、その確率を示すスコアが返される。予測された音韻系列のスコアにより、ビームサーチを行いながら文節の音韻を認識する。

図 1(b) はトライグラムを用いて候補の絞り込みを行なう HMM-LR システムである。認識されている音韻系列のスコアに、その音韻系列に対する言語モデルの確率を求め、適当な重みをかけて HMM によるスコアと足し合わせてその音韻系列に対するスコアとしている。そのスコアをもとにビームサーチを行いながら文節の音韻を認識する。

トライグラム (出現頻度 1 以下の三つ組については、一定値に置き換えている) と、削除補間法により補間を行なったモデルで、言語モデル以外の条件は全く同じにして、文節音声の音韻認識実験を行なった。その条件を表 6 に示す。

4.2 実験結果

279 文節に対して実験を行なった。音韻認識の結果得られた音韻列は、正解音韻と文字列の DP マッチングにより比較して、正解、脱落、挿入、置換の基準で評価を行なった。

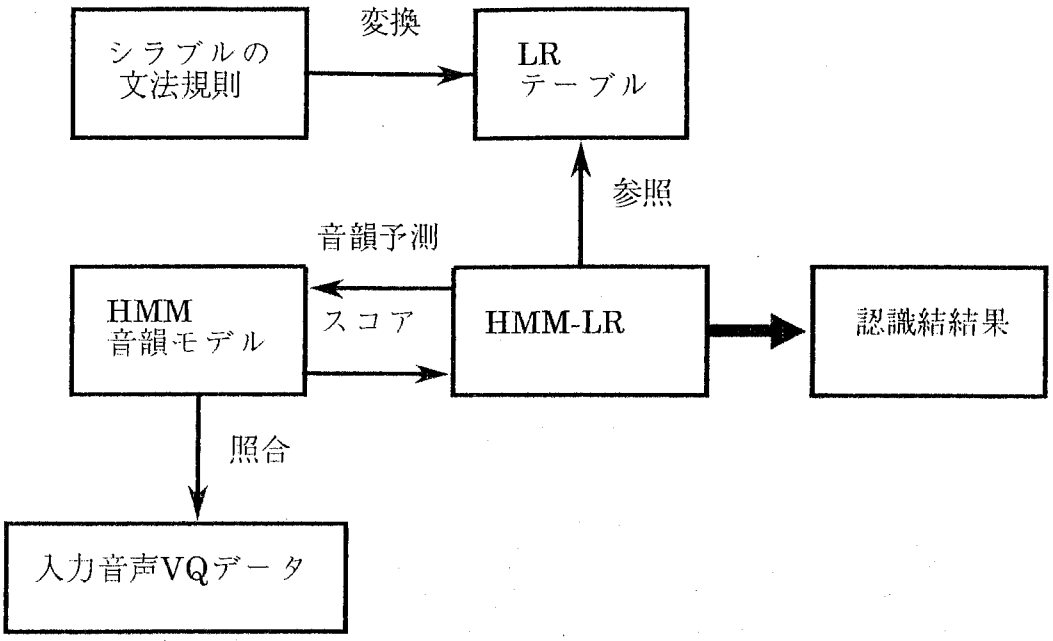

図1.(a)HMM-LRシステム(トライグラムなし)

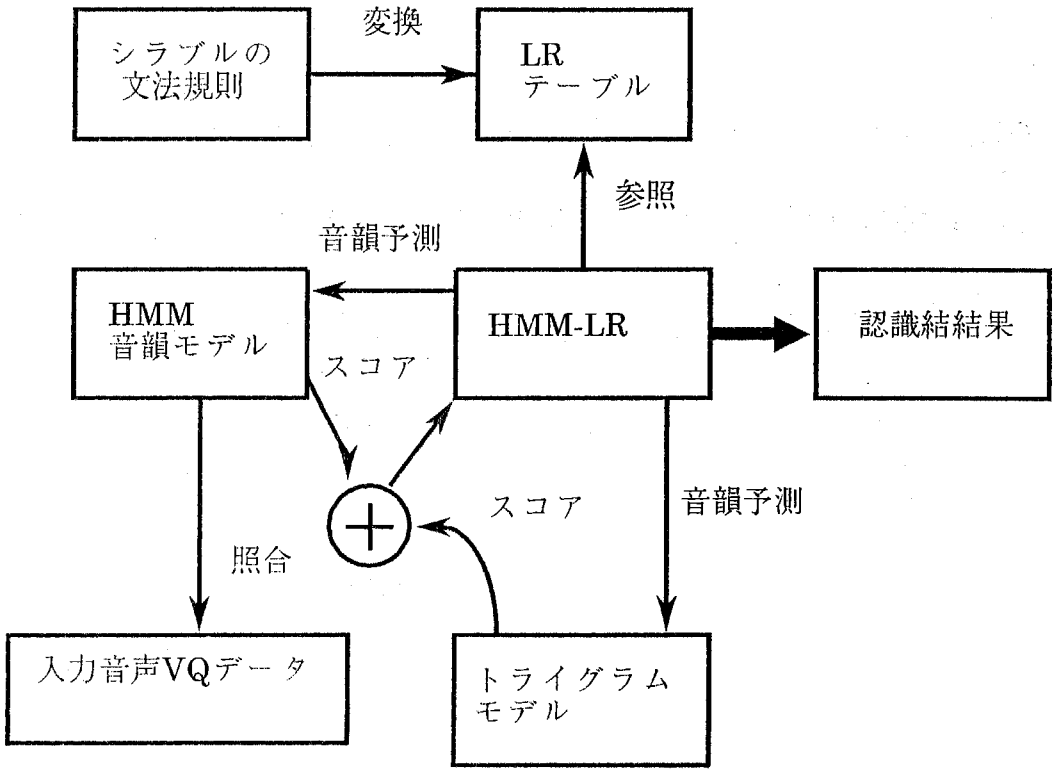

図1.(b)HMM-LRシステム(トライグラムを含む)

表 6: 音韻認識実験の条件

音声資料

話者	男性 1 名 (MAU)
HMM 学習データ	重要語 5240 単語とバランス 216 単語中の音韻
評価データ	国際会議の問合せを想定した会話文中の 279 文節

これらの資料を 12 kHz でサンプリングし窓長 21.3 msec 、周期 3 msec のハミング窓で切り出し 12 次の LPC 分析を行なう。分析により、

1. スペクトル (WLR)
2. ケプストラムの差分
3. パワー

の 3 種の特徴量を抽出しセパレートベクトル化を行なう。量子点数は順に、256, 256, 64 とする。
HMM 音韻認識は継続時間長を制御して、トレリスアルゴリズムにより行なう。

枝刈り場所	トライグラムなし	トライグラムあり
Local	18	18
Grobal	170	170

スコア S は次の計算式で求めている。

$$S = 0.88 \log P_{\text{音韻モデル}} + 0.12 \log P_{\text{言語モデル}}$$

この結果を集計して、以下の計算式により、2通りの音韻認識率を求めた。1 つめは、単純な正解率 ($P_{Correct}$) で、発声した文節の音韻中のどれだけが正解音韻として認識されたかを表す。2 つめは、認識結果の正確さを考慮した認識率 ($P_{Accuracy}$) である。

$$P_{Correct} = \frac{N_{Correct}}{N_p}$$

$$P_{Accuracy} = \frac{N_{Correct} - N_{Subs} - N_{Dels} - N_{Ins}}{N_p}$$

ただし、

N_p: 文節の正しい音韻数
$N_{Correct}$: 認識結果の正解音韻数
N_{Subs}: 置換された音韻数
N_{Dels}: 脱落した音韻数
N_{Ins}: 挿入された音韻数

計算した音韻認識率などの結果を表7に示す。表から、補間を行なったモデルを用いた方が全体では認識率が悪くなっていることがわかる。

表 7: 認識結果 (その 1)

	トライグラムモデル	補間したモデル
test	3.92	4.22
生成確率	-0.96	-0.93
coverage	98	
$P_{Accuracy}$	87.4	84.4
$P_{Correct}$	90.7	89.2
枝刈りされた正解の文節数	61	79
文節認識率 (1 位)	57.0	47.0
文節認識率 (5 位まで)	70.3	62.4

まず、実験に用いた文節に対する 2 つのモデルのタスク分岐数と生成確率を見てみる。生成確率は補間を行なった方が大きくなっていて、よくなっているといえるが、タスク分岐数については逆に悪くなっている。また、実験に用いた文節に対するトライグラムの coverage は 98 % となっている。前節の結果から、テキストデータベースのデータ量はトライグラムモデルを構成するのに十分とはいえないので、この数字は、テキストデータベース全体と実験に用いた文節は非常に似ていることを表していると考えられる。一般に補間を行なうと、本来のモデルで大きな確率値をもつものは小さな値になり、小さな値の

表 8: 比較的大きな値に補間されるパタンの例

<de re ba>	<ga tsu u>
<re ba> の確率 = 0.0885	< tsu u > の確率 = 0.0898
<ba> の確率 = 0.00393	<u> の確率 = 0.0460
補間後の確率 = 0.00408	補間後の確率 = 0.00481

ものが大きな値になるので、このように訓練用のデータと実験用のデータが非常に似ているときには、全体的にみると、モデルの精度が悪くなってしまったようにふるまってしまうのであろう。

以下では、さらに詳細に検討を加え、モデルの補間法の問題点を考察する。

まず、補間を行なったために、文節を正しく認識するようになったものについて、原因を考えてみる。

このような文節は全体で 4 文節あり、そのうち 1 つについては原因が特定できなかったが、残り 3 つについては全て < syo te i > というテキストデータベース中に出現しなかったパタンが含まれており、このパタンに対してトライグラムでは 0 だった確率が 0.0138 と比較的大きな確率に補間されたために、言語モデルの確率値がより適当な値となり他の候補よりよいスコアになったと考えられる。

次に、補間が悪影響を及ぼし文節を正しく認識しなくなったものについて、その原因を考えてみる。

このような文節は全体で 33 文節ある。そのうち、11 文節はトライグラムでは枝刈りされなかったものが途中で枝刈りされてしまったもので、残りは最終的な順位が低くなってしまったものなのである。これらのうち 1 例については、原因が特定できなかったが、その他の文節は枝刈りされたものもそうでないものもモデルの精度が悪化したために、トライグラムのときには枝刈りされていたような「日本語らしくない」候補が枝刈りされなくなったり、低い確率値をもっていたような候補のスコアがよくなってしまったために、正解の候補が枝刈りされたり 1 位でなくなったりしたと考えられる。

ここで、どのような補間が悪影響を及ぼしたかを考えるため、悪影響を及ぼしたようなパタンのいくつかについて見てみる。

表 8に比較的大きな値に補間されたパタンを示す。この <de re ba> と <ga tsu u> の 2 つのパタンは、ユニグラムとバイグラムの確率値はほぼ同じ大きさであり、補間されたモデルの確率もほぼ等しいのだが、実際には <de re ba> は存在しそうだが <ga tsu u> はあまり存在しないように思える。このように<A B C> という三つ組を考えたときに<A B> <B C> の組み合わせはそれぞれよく現れるが、<A B C> という組み合わせはほとんど存在しないというケースはかなりあると考えられ、画一的に補間をする限りトライグラムモデルの精度を悪化させる可能性はまぬがれないと思われる。

5 トライグラムの改良

前節の実験の結果などを参考に、トライグラムの改良を行なった。

5.1 長母音を考慮したモデル

実験に用いたトライグラムと同じ訓練データから得られたバイグラム確率値のよいものを表9に示す。

表9: バイグラムの確率の良いパタン

パタン	確率	出現回数
rya ku	1.00000	5
pyo u	1.00000	104
pya ku	1.00000	5
nyo zi	1.00000	1
myo u	1.00000	15
hyu u	1.00000	1
gyu u	1.00000	2
byo u	1.00000	44
bya ku	1.00000	15
chu u	0.99310	288
nyu u	0.99242	131
hya ku	0.99078	215
kyu u	0.99034	410
pyu u	0.98649	146
gyo u	0.97842	136
ryu u	0.97222	35
byu u	0.95833	23
hyo u	0.95122	117
za i	0.84622	974
zyo u	0.83661	809

この表の太字のところが、長母音となる母音の連鎖を含む二つ組である。このように、長母音は、非常に連鎖の可能性が高いため、長母音を母音として区別することにより、パタン数をそれほど増やさないで、よりモデルの精度を上げることができると考えられる。

また、認識に用いられている音韻モデルでは、長母音と母音は区別されているので、認識に利用するときには、効果的であると考えられる。

表 10: 出現頻度の高い二つ組

パタン	出現頻度
de su	3753
ma su	3370
ha i	3021
ka i	2330
te i	2229
e e	2198
shi te	2187
a no	2169
shi ta	2126
i ma	1888
i ta	1877
N de	1868
ko to	1859
ri ma	1722
u ka	1600
Q te	1575
sho u	1569
yo u	1557
ko u	1526
ma shi	1524

5.2 コンテキストを考慮して補間を行なうモデル

これまでの補間では削除補間法を用いて画一的にスムージングの係数を決定していたが、表 10 に示したような出現頻度の高い二つ組の次に来る音節に関するトライグラムについては十分なデータがあると考えられ精度も高いと考えられるだろうし、逆にほとんど出現しない二つ組の次に来る音節に関するトライグラムについては、非常に精度が悪いと考えられる。したがって、直前の二つ組 (以下、これをコンテキストと呼ぶ) ごとに重み係数を決定した方が精度の良いモデルになると考えられる。

ここでは、計算の都合で削除補間法ではなく、訓練データを 2 つに分割して片方を用いてトライグラムなどの推定を行ない、別の片方で (各コンテキストごとに) 係数を決定した。(Held-out 法) 係数を決定するのには次の式を用いている。[1]

$$\hat{q}_i = \sum_{s:\text{全シラブル}} \frac{N_{train}(s)p_i}{N_{context}}$$

$N_{train}(s)$: 評価データ中の係数を決定するコンテキストでのシラブル s の出現回数

$N_{context}$：評価データ中での係数を決定するコンテキストの出現回数

ただし、

$$p_i = \frac{q_i f_i}{q_1 f_1 + q_2 f_2 + q_3 f_3}$$

f_i：訓練データ中での i-gram の確率値

このようにして繰り返していくとコンテキスト毎に生成確率について最適な q_i を決定することができる。

このようにして決定した係数の具体的な例を表11に示す。このように、コンテキストごとにかなり異なった値になることがわかる。

表 11: コンテキストを考慮して求めた係数

コンテキスト	< de su >	< da i >	< ki so >
出現回数	1862	467	7
q_1	0.0014	0.0704	0.4765
q_2	0.0030	0.0000	0.0000
q_3	0.9957	0.9296	0.5235

5.3 4 gram

データ数さえ十分であれば、コンテキストが長ければ長いほどモデルの精度はよくなるはずである。そこで、4 gram モデルについても調べてみる。

6 文節音声の音韻認識実験 (その 2)

前節で改良を行なった言語モデルのうちいくつかを用いて、文節音声の音韻認識実験を行なった。条件は、4 節と同じである。(5.2 で示したモデルはいくつかの音韻について実験を行なった結果、余り効果が見られなかったので実験を行なわなかった。)

6.1 実験結果

それぞれのモデルに対する実験結果を表12に示す。

表 12: 認識実験結果 (その 2)

	1	2	3	4	5	6
test-set perplexity	4.01	4.34	3.37	3.83	3.53	13.6
生成確率	-1.05	-0.99	-1.01	-0.88	-0.91	-2.05
coverage	96		94			
$P_{Accuracy}$	88.9	85.4	89.9	86.3	88.8	72.4
$P_{Correct}$	91.2	88.6	91.9	90.0	91.9	80.6
枝刈りされた正解の文節数	48	64	45	73	52	140
文節認識率 (1 位)	63.1	52.7	67.4	53.4	62.0	20.4
文節認識率 (5 位まで)	77.4	67.7	78.5	64.5	72.4	35.5

1. 長母音を考慮したモデルのトライグラム
2. 長母音を考慮したモデルのトライグラムを補間したもの
3. 4-gram モデル
4. 4-gram モデルを補間したもの
5. 4-gram モデルをトライグラムモデルだけで補間したもの
6. ゼログラムモデル (言語モデルを用いない)

このうち、いくつかの結果について考察を行なう。

6.1.1 長母音を考慮したモデル

長母音を考慮することによって、文節を正しく認識するようになったものについて、その原因を考えてみる。

その原因については、主に次の 2 つがあげられる。

1. 長母音を考慮していない言語モデルでは、音韻認識部で、別の音韻列として認識されたものでも、等しい確率を与えていた。

   例

   o-o-s-a-i と oo-s-a-i はともに、o-o-sa-i として扱われていた。

   そのため、長母音を含むとみなせる音韻列が認識の途中に良いスコアで現れると、字面としては同一の候補も (認識のスコアは異なるが、言語モデルのスコアが等しいために) 枝刈りされずに残ることが多く、その他の候補が枝刈りされることが多かったと考えられる。

2. 長母音を考慮することで、言語モデルが結果的により長いコンテキストを反映することになり、より精度の高い予測を行なうようになった。

   例

   > 文節番号 7 (tourokuo) は、これまで、tourokuou と誤っていた。この候補の語尾の部分は <ku o u> というパタンで、このパタンに対しては 0.00746 の確率が与えられていた。ところが、長母音を考慮することによって語尾の部分が、<ro ku ou> というパタンで、予測されることになりこの確率が 0 であるために、tourokuou と tourokuo という 2 つの音韻列についての言語モデルの確率は次のようになり、tourokuo が 1 位になるようになった。

次に、悪影響を及ぼした原因について考えてみる。実際に長母音を考慮することによって正解でなくなった文節は 4 つしかなく、ほとんど悪影響はないと考えられるが、その 4 つが正解でなくなった原因を調べてみると次の 2 つに分けられる。

1. シンボルの種類が増えたので実質的には訓練用のデータが減ったので、言語モデルの信頼性が低下してしまった。このような例は全体で 3 例あった。

   例

   > 文節番号 75 (cyoukounomino) は、< kou no mi > に対する確率が 0 となる。その他のパタンは全てこれまでの確率値より大きくなっているが、文節全体としては確率の常用対数で、-1.09 から -1.78 へと小さくなってしまった。

2. 長母音になり得る母音の連鎖を全て長母音としてしまったため、長母音とならない (長母音になり得る) 母音の連鎖に対する確率値が小さくなってしまった。このような例は 1 つあった。

   例

   > 文節番号 142 (youkade) は、これまで yo-u-ka-de として認識されたものが正解となっていたが、新しいモデルでは訓練データ中の yo-u は全て you としているため、yo-u を含んだパタンに対する確率は 0 となってしまい文節全体としての確率は常用対数で -1.31 から -3.42 へと小さくなってしまった。(しかし、you-ka-de に対する確率は常用対数で、-1.31 から -1.15 へと大きくなっている。)

6.1.2 4 gram モデル

4-gram モデルでも認識率は向上しており、効果をあげた原因としては前述した 2、悪影響を及ぼした原因 (ほとんどないが) は 1 があげられる。

7 まとめ

7.1 補足

ここまでの実験結果では、テキストデータベースと実験に用いた文節が似ているために削除補間法の効果はほとんど現れていなかった。また、これまでの実験ではトライグラムの確率が 0 になるものは 1.0×10^{-5} で flooring していたが、その値が適当である保証はなにもなかった。そこで、ここでは、test-set perplexity と認識率との関係を調べるためにも、flooring する確率を補間したモデルと補間しないモデルでテスト用の文節にたいする生成確率が等しくなるようにして実験を行なうことにする。

そこで、テキストデータベースの中でも特に実験に用いた文節と似ている key と tel を訓練データから除いて作ったトライグラムと、それを補間したモデルの両方で 4 節と同様の実験を行なった。トライグラムなどの条件を表13に示す。

表 13: トライグラムなどのデータ

収束条件	$\forall i \mid q_i - \hat{q}_i \mid < 1.0 \times 10^{-8}$	
訓練データの文節数	29197	
音節数	114807	
シンボルの種類	259	
係数	q_0	0.00234
	q_1	0.04287
	q_2	0.14528
	q_3	0.80951

実験結果を表14に示す。削除補間法を用いたもののほうが認識率がよくなっており、test-set perplexity も小さくなっているため、test-set perplexity が認識率と関係する可能性を示唆していると考えられる。また、削除補間法による補間もトライグラムの精度がよくない場合には効果があることも示されているといえるだろう。

表 14: 認識実験結果 (その 3)

	トライグラムモデル	補間したモデル
test	6.09	5.32
生成確率	-1.34	-1.35
coverage	86	
$P_{Accuracy}$	79.5	80.6
$P_{Correct}$	85.0	85.4
枝刈りされた正解の文節数	94	87
文節認識率 (1 位)	39.4	40.1
文節認識率 (5 位まで)	57.3	57.3

7.2 より精度のよい音韻連鎖に関する言語モデルにむけて

さらに精度のよいモデルをつくるための考察を行う。

7.2.1 その他の補間の方法

ここでは、補間の方法として画一的に削除補間法を用いる方法と、triclass モデルを用いたものを試してみたが、その他にコンテキストの出現頻度に応じて補間の係数を変化させる補間法も提案されている。(バックオフスムージング法 [1])

しかし、triclass モデルのところでも述べたようにこれらの補間法は全て生成確率に対する最適な重み係数を決定するようになっているので、認識に用いるときに効果的であるかどうかはわからないだろう。(test-set perplexity を考慮にいれる必要がある。)

7.2.2 その他のモデル

トライグラム、4-gram などでは、かなりのデータ量がないと十分なモデルが構成できない。そういった場合、例えば 2 音節からなる単語などでよく出現するもの (です、ます、など) は場合によっては、トライグラムで予測するよりも、バイグラムで予測した方が精度が上がるということも考えられる。逆に 3,4 音節以上の音節の連鎖パタンでもよく出現するものが存在すれば、そのパタンについては、なるべく長いパタンを使って予測した方がよいだろう。ここでは、長母音について実質的にコンテキストを長くすることで、モデルの精度が向上したことを示したが、上で述べたことを考慮するとコンテキストを固定せずにパタンの出現頻度に応じてコンテキストを変化させることのできるモデル (tree モデル) をためしてみる価値はあるだろう。

7.2.3 (音韻連鎖に関する)文節間の情報の利用

主な言語モデルによる認識実験に用いた文節の平均分岐数と生成確率を計算した例を表に示す。

このようにどの言語モデルでも文節の先頭の分岐数が非常に大きくなっている。これは、例に示した文節であれば先頭の to (もしくは tou) の確率をトライグラムであれば <# # to(tou)> (# は無音もしくは文節の区切りをあらわす)、4 gram であれば <# # # to(tou)> というパタンとしていて、実際には、バイグラムの <# to(tou)> というパタンに対する確率値で代用しているためである。HMM-LR のように左から右へと処理を進める方式では、処理の最初にたくさんの候補が生成されるとそれだけ正解の候補が枝刈りされる可能性も増すうえ、このような表現方法ではコンテキストを長くしてもその効果を十分に生かせなくなってしまう。そこで、たとえば例に示した文節の直前の文節が <so no> であれば、先頭の to (tou) に対するパタンはトライグラムなら<no # to(tou)>、4 gram なら<so no #to(tou)> のようにすれば少しでも精度が高くなると考えられる。

8 謝辞

二カ月半にわたって指導して下さった花沢さん、川端さん、非常に有益な助言を下さった鹿野さん、研究の機会を与えて下さった榑松社長、その他 ATR の皆さんに心から感謝致します。短い間でしたが、非常に多くのことを学ぶことができたと思います。これからも、音声認識側の視点を意識した自然言語処理の研究を進めていきたいと思います。

参考文献

[1] 中川聖一. 確率モデルによる音声認識. 電子情報通信学会, 1988.

[2] 北川英一郎. HMM-LR 音声認識システムと音節トライグラムを用いた音韻タイプライタの検討. ATR 実習報告, 8 1989.

[3] P. Brown. Speech recognition by statistical methods. 11 1985.

[4] F. Jelinek and R. Mercer. Interpolated estimation of markov source parameters from sparse data. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*, pages 381–397, North-Holland, 1980.

[5] K. Kita, T. Kawabata, and H. Saito. Hmm continuous speech recognition using predictive lr parsing. In *ICASSP*, page 703, 1989.