TR-IT-0154

音声合成のための音声データ自動ラベリング手法

西村 公志, ニック キャンベル

1996.2

ABSTRACT

本報告では、音声合成のための音声データ自動ラベリング手法として、HMM(Hidden Markov Model) と DTW(Dynamic Time Warpping) のアライニング手法の有効性の比較を行うと共に HMM と DTW を用いた多段階処理手法を検討した。HMM と DTW の有効性の比較は各手法の結果とハンドラベルとの比較で行なった。この結果、HMM の方がハンドラベルに対して絶対誤差が小さいことが分かり、一文全体のような大きな単位では、HMM が DTW より有効であることが分かった。さらに、HMM により大局的なセグメンテーションを行なった後、DTW により、局所的なセグメンテーションを行なうという多段階処理手法を検討した。多段階処理手法を行なった結果、HMM の時より、ハンドラベルとの絶対誤差は小さくなった。

目次

1 はじめに

音声認識・合成システムなどの音声情報処理システムの研究開発では音素の位置情報を示す音素ラベルデータは重要な役割を果たしている。特に音声合成では、合成用音素片データの作成のために音声データの自動ラベリング手法は不可欠である。また、既知の単語列の処理のためには、音韻境界、およびテキストから音韻列を予測することが必要である。そのため、以前から自動ラベリング手法として、DTW(Dynamic Time Warpping) や HMM(Hidden Markov Model) を用いた手法が提案されてきた。そして、自動ラベリングによって作成された音素ラベルデータを利用し、音声合成が行なわれる。

しかし、このような自動ラベリングによって作成された音素ラベルデータを使用して作成された合成音声は、人間が聞いた時に不自然な部分が残ってしまい、不満が残る品質となってしまう。その理由としては、音素ラベルデータ中の音素の位置情報が音声データの持つその音素本来の位置とのずれによるところが大きい場合がある。そこでそれら音素ラベルデータの品質を向上させるために、音素ラベルデータをラベリングの専門家 (ラベラー) の手によって修正を加える。自動ラベリングの手法が開発される以前は、ラベラーがすべて手作業で音素データラベルを作成していた。その当時に比べ、ラベラーによる修正は、ラベラーの負担が軽くなったとはいえ、まだまだラベラーにかかる負担は大きいものである。また、扱う音声データが大量になってくると、人間によるラベリングにはとてつもない時間がかかることになってしまう。そのために、ラベリング作業で人間に頼る部分をできる限り少なくする必要がある。

今回、音韻境界の予測のために用いる手法は次の 2 つとする。

- HMM Tool Kit(Entropic HTK) を用いたアラインメント手法
- Dynamic Time Warpping(DTW) を用いたアラインメント手法

両手法について音声認識の分野では、以前は DTW がよく使われていが、最近では HMM の方がよく使われるようになってきた。合成では、認識とは違いトレーニングデータとテストデータが同じクローズなデータに対して処理を行なう。つまり、認識の場合は音韻の並びは予め与えられていないものに対してラベリングを行なうが、合成の場合は音韻の並びは予め与えられたものとして、アラインするだけである。この部分が認識とはもっとも違う点である。しかし、合成のアラインに最も必要なことは、合成した音声がスムーズに聞こえることであり、これは、音韻境界のスペクトルひずみをすべての音素について最小化することである。今回は、HMM と認識ではあまり使われなくなった DTW の両アライニング手法を用い、その有効性を比較する。

2 HMM(Hidden Markov Model) を用いたアライニング

今回、HMM 法を用いたアライニングには、HTK(HMM Tool Kit, Entropic) を使用した。

HMM 法は、音素内での音韻特徴の変化を数個の状態の遷移で表し、各状態での複数の音韻標準パタンの生起確率と状態間の遷移確率を与えて、入力音声をこのモデルに当てはめて認識する方法である。この方法では、確率モデルを与えるための学習処理が複雑になるが、認識時の処理量が比較的少なくてすむ特徴がある。

HMM での音韻アラインメントの流れを図 1 に示す。この図の詳説は、次の小節で示す。

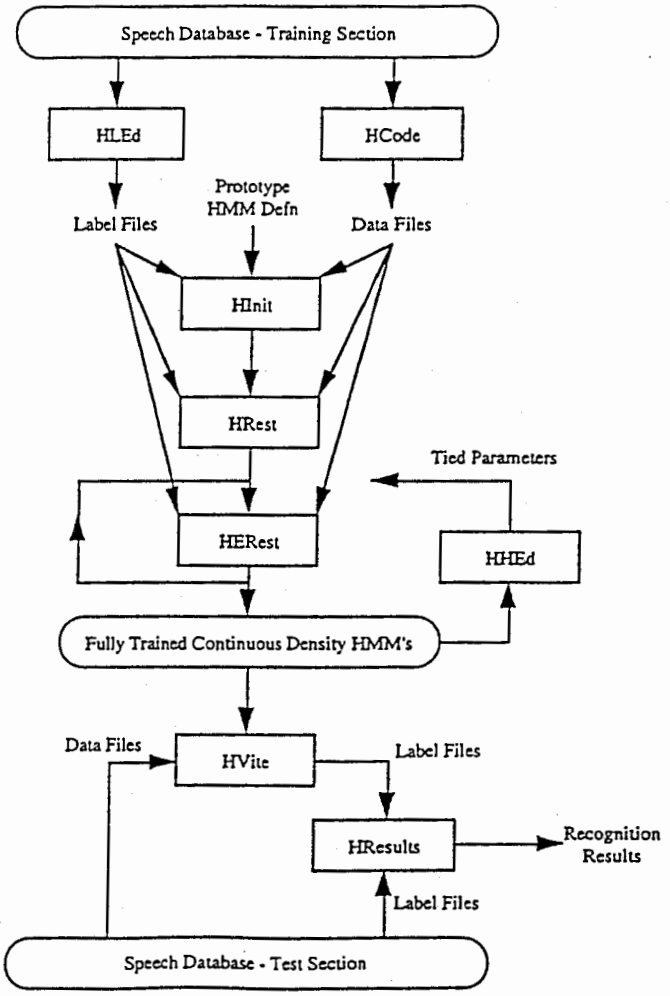

図 1: HMM による音声認識の概念図

2.1 アライニング手順

今回、話者 MHN に対して HMM を用いて行なったアライニングまでの手順を以下に示す。

音声データにラベリングを行なうまでの手順として、図 1 中のコマンド HLEd と HInit で行なわれる作業は行なわず、既存の音素パタンのデータを使用した。また、 HEResults による認識結果を出力するまではいかず、 HVite によるラベルファイルの出力までの作業を行なった。

1. 既存の不特定話者の各音素パタン (モノフォンモデル) を初期値として与える。
2. 音声データをケプストラムに変換する。 (以降ケプストラムをデータとして扱う)

   (HCode による変換、サンプリング周期は 5msec とし、エネルギー、Δ を考慮して 12 次のメルケプストラム係数 (MFCC) を作成する)

   HCode -m -e -d -f 5.0 -F ESPS -O ESPS 音声データファイル MFCC ファイル
3. 話者の持つ各音素の特徴を学習する。

   (HRest によって、学習を行なう。)

   HRest -T 1 -m 1 -v 0.1 -G ESPS -L ラベルファイルのディレクトリ -X 拡張子 -l 音素名 -o 音素の特徴ファイル -S 学習に使う MFCC ファイル 音素の特徴の出力ファイル
4. 各音素間のつながりを考慮し、各音素の特徴を再学習する。

   (HERest によって学習する。この処理は、 2 回繰り返す)

   HERest -T 1 -m 1 -d 各音素の特徴ファイルのあるディレクトリ名 -e 音素の特徴の学習後のファイルの出力先ディレクトリ -G ESPS -L ラベルファイルのディレクトリ -X 拡張子 -S 学習に使う MFCC ファイル 学習する音素のリストファイル
5. 学習した各音素の特徴を用いて、アライニングを行なう。このアライニングには、 Viterbi のアルゴリズムが用いられている。

   (HVite によるアライニング)

   HVite -p 1.0 -s 0.0 -v 0.1 -d HMMs1 -P ESPS 学習した音素のリストファイル ラベルの並びファイル MFCC ファイル

2.2 使用した話者データ

HMM のアライニングに使用した話者データは、 MHN のスピーチデータ 503 文とした。ただし発声内容は ATR の Bset と同様ものである。

HMM の学習には、話者 MHN のスピーチデータから 100 文 (MHN001.sd から MHN100.sd) を使用し、学習に使用した文を含めて 503 文についてアライニングを行なった。

3 DTW(Dynamic Time Warpping) を用いたアライニング

音声は、おなじ人が同じ単語を発声してもその継続時間はその都度変わり、しかも非線形に伸縮する。そのため、同じ発声データ同士を非線形に伸縮して時間正規化を行ない非線形マッピングをとる必要がある。DTW とはこの非線形マッピングを動的計画法のアルゴリズムで行なう方法である。

また、同じ発声データあれば、DTW を用いてアライニングを行なうことができる。

以下に DTW の基本的アルゴリズムを示す。[1]

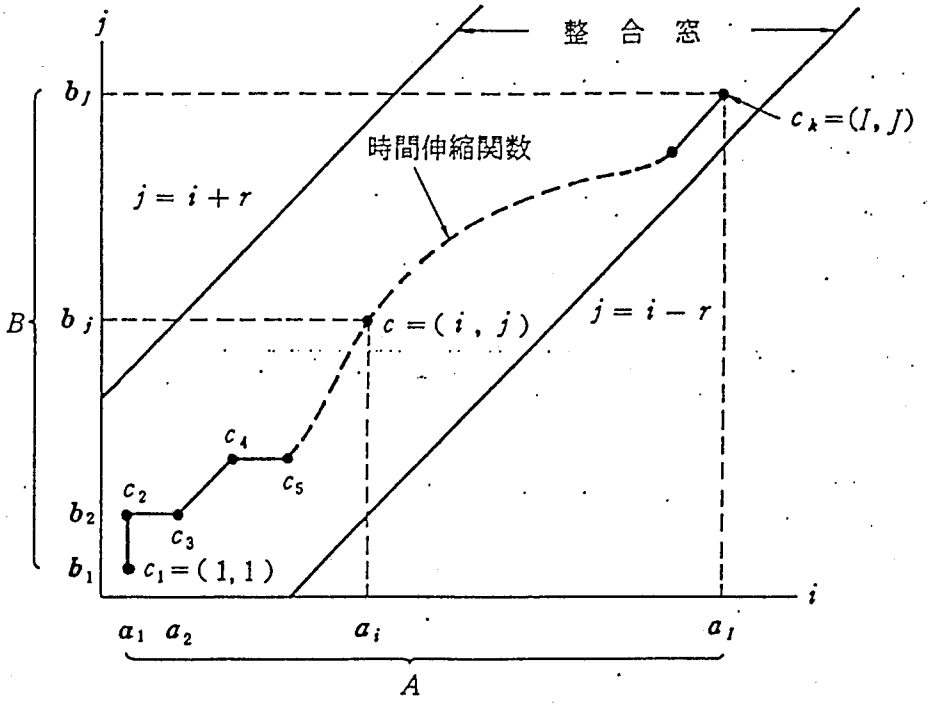

図 2: 時間伸縮関数と整合窓

まず、対応づける 2 つの時系列を $A = a_1, a_2, ..., a_i, ..., a_I$ と $B = b_1, b_2, ..., b_j, ..., b_J$ で表す。このとき、図 2[1] を考えると、A, B 両パタンの時間の対応、つまり時間伸縮関数は、この平面上の格子点 $c = (i, j)$ の系列 F で表現することができる。

$$F = c_1, c_2, ..., c_k, ...c_K, \quad c_k = (i_k, j_k) \tag{1}$$

2 つの特徴ベクトル a_i と b_j とのスペクトル距離を $d(c) = d(i, j)$ で表すと、F に沿った距離の総和は、

$$H(F) = \sum_{k=1}^{K} d(c_k) \cdot w_k / \sum_{k=1}^{K} w_k \tag{2}$$

で表され、この値が小さいほど A と B の対応づけが良いことを示す。ここで、w_k は、F に関連した正の重み係数である。例えば、$w_k = (i_k - i_{k-1}) + (j_k - j_{k-1})(i_0 = j_0 = 0)$ とすると、

$\sum_{k=1}^{K} w_k = I + J$ となり、式2は次のように簡単化される。

$$H(F) = \frac{1}{I+J} \sum_{k=1}^{K} d(c_k) w_k \tag{3}$$

この式を次のような制限のもとで、F に関して最小化することを考える。

(i) 単調性と連続性から

$$o \leq i_k - i_{k-1} \leq 1, \text{ および } 0 \leq j_k - j_{k-1} \leq 1 \tag{4}$$

(ii) 境界条件から

$$i_1 = j_1 = 1, \ i_K = I, k_k = J \tag{5}$$

(iii) 極端な伸縮を防ぐために r を定数として

$$|i_k - j_k| \leq r \tag{6}$$

これを整合窓の条件と呼ぶ。
部分点列 $c_1, c_2, ..., c_k (c_k = (i,j))$ に対する部分和を考えると、

$$\begin{aligned} g(c_k) &= g(i,j) = \min_{c_1,...,c_{k-1}} \left[\sum_{l=1}^{k} d(c_l) w_l \right] \\ &= \min_{c_1,...,c_{k-1}} \left[\sum_{l=1}^{k-1} d(c_l) w_l + d(c_k) w_k \right] \\ &= \min_{c_{k-1}} \left[\min_{c_1,...,c_{k-2}} \left\{ \sum_{l=1}^{k-1} d(c_l) w_l \right\} + d(c_k) w_k \right] \\ &= \min_{c_{k-1}} \left[g(c_{k-1}) + d(c_k) w_k \right] \end{aligned} \tag{7}$$

これは、動的計画法 (Dynamic Programming) の定式化になっている。

上で示した、F に関する (i) ～ (iii) の制限と、w_k の定式化を用いてこの式を書きかえると次式となる。

$$g(i,j) = \min \begin{bmatrix} g(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \\ g(i-1,j) + d(i,j) \end{bmatrix} \tag{8}$$

したがって、$g(1,1) = 2d(1,1)$,$j = 1$ として、整合窓の範囲内で i を変えながら上式を計算し、次に j を増加させて $j = J$ となるまで同様の計算を繰り返せば、最後に $g(I,J)/(I+J)$ として、A, B の 2 つの時系列間の時間正規化後の距離が求まる。

3.1 アライニング手順

DTW は同じ発音データ同士を非線形に伸縮させマッピングをとる。そのため、基準となる音声データ (基準音声) が必要である。

以下DTW によるアライニング手順を示す。

1. 基準音声とマッチングする音声 (対象音声) をケプストラムに変換する。この変換は、HMM に用いたものと全く同じである。

2. ケプストラムを元に、音声データ同士をDTWでマッピングをとる。

   この時、マッピング結果と同時に音声データ同士の類似性を示す値もスカラー量で出力される。

   (c言語の実行ファイルdtwによって、DTWを行う。)

3. マッピング結果から、基準音声から、対象音声にラベルを写す。

   (c言語の実行ファイルdtw2xlbによって、対象データをラベリングする。ラベルの出力形式はESPS形式)

3.2 使用した話者データ

今回DTWの評価に使用したデータは、ATRのBsetに含まれる話者のデータを用いた。基準となる話者(基準話者)と、テストされる話者(テスト話者)を表1に示す。

表1: 使用した話者とデータ数

基準話者	テスト話者	使用する文章の数
MHT	MHO	249/503文
MHT	FKN	273/503文
FKS	FTK	288/503文
FYM	MYI	228/503文

基準話者と、テスト話者の各文章のラベルファイルが完全に一致していないものについてはDTWを行なわなかった。この理由は、同じ発声内容であっても話者によって、ポーズがあったりなかったりのような、小さな単位での違いがあるからで、この違いがラベリング結果に影響しないように、ラベルが完全に一致しているもののみにラベリングを行なった。そのため、すべての話者503文の音声データを使用しなかった。

4 HMMとDTWの比較結果

HMMと、DTWの比較を行なうためにそれぞれのハンドラベルとの絶対値誤差を表2に示す。ただしこの結果からは、単純にどちらの方が性能が良いとはいえない。これは、基準として扱ったハンドラベルが絶対的なラベルとはいえないからで、ハンドラベルは人間がラベル付けを行なっているので、その日の体調や環境によって若干の誤差を含んでしまうからである。しかし、絶対的な基準がない以上ハンドラベルが現段階でもっとも良いアライニング結果ということとし、HMMとDTW評価の基準とした。また、各音素のハンドラベルとの誤差を示したボックスグラフを付録Bに示す。

表 2: HMM と DTW の絶対値誤差

使用手法	データ	絶対値誤差 (msec)	サンプル数
HMM	MHN	17.0533	26649
DTW	MHT から MHO へ	39.8325	12114
DTW	MHT から FKN へ	39.1613	15342
DTW	FYM から MYI へ	31.3668	14377
DTW	FKS から FTK へ	37.3235	13257

表とボックスグラフから、一文全体に関しては HMM の方が DTW よりもハンドラベルとの絶対誤差が小さくかつ各音素のラベルのばらつきが小さいのでラベリングには適していると思われる。

5 多段階処理手法

音声合成で、自然な合成音声を得ようとすれば、音声単位の接続点において音響特徴が滑らかに変化していることが望ましい。このような観点から見ると、アライニングされた音声データ同士が、音素境界において共通した音響特徴を持っていることが必要であり、マッチングによって類似の特徴を示す箇所を求めることが有効であると考えられる。

検討した多段階処理手法は大局的セグメンテーションに HMM を用い、局所的セグメンテーションに DTW を用いることとした。

5.1 アライニング手順

大まかなアライニングは、HMM によるアライニング方法で行なうので具体的方法は、省略する。

DTW を用いた局所的セグメンテーションの手順を次に示す。

1. テストデータの全文章をすべて 2 音素に分割する。
2. 分割した 2 音素について、同じ組合せのもの同士を集める。
3. 上記の処理によってできた共通する 2 音素を含むすべての音声データについて以下の処理を繰り返す。
   (i) 基準の音声データ (A) を決める。
   (ii) A との DTW を A 以外のすべての音声データ (B) についておこなう。
   (iii) マッチング結果から、B の 2 音素の境界ラベルを A に写す。
   (iv) B から A に写された境界と、A の持つ境界との時間差を計算する。
   (v) A との時間差の平均をとり、その分 A の境界をずらす。

5.2 使用した話者データ

使用した音声データは2節で使用した話者データと同じデータとした。さらに、大局的セグメンテーションは、2節や3節によって行なわれているものとし、2節で行なった自動ラベリングの結果を用いて、局所的セグメンテーションを行なった。

5.3 多段階処理の結果

今回、時間的な問題で多段階処理手法の局所的セグメンテーションを全2音素のペアに適用することができなかった。途中までの行なった結果とハンドラベルとの絶対誤差を表3に示す。結果から、全体をHMMによってセグメンテーションした結果より、さらにDTWにより局所的セグメンテーションを行なった方が、ハンドラベルとの絶対誤差が小さくなった。

表3: 多段階処理の途中結果

データの分類	絶対誤差	サンプル数
HMM+DTWの全体	17.1045	26649
HMMのみの部分	17.8354	21418
HMM+DTWの部分	14.1115	5231

図3にHMMで大局的セグメンテーションを行なったあとに局所的にDTWを行なった結果を示す。

```
#
25.9137 121 ##
25.9437  121  a
26.0237  121  r
26.110471 140 a   ; distance=0.705890 ; label gap= +0.016770
26.170439 140 y   ; distance=0.730776 ; label gap= -0.013260
26.2137  121  u
26.2537  121  r
26.307516 140 u   ; distance=0.703633 ; label gap= -0.026180
26.401304 140 g   ; distance=0.733849 ; label gap= +0.007600
26.4937  121  e
26.5337  121  N
26.5937  121  j
26.654271 140 i   ; distance=1.000000 ; label gap= -0.019429
26.8137  121  ts
26.8637  121  u
```

図3: HMMの後に2音素DTWを行なった結果(/arayurugeNjitsu/)

6 まとめ

今回の大きな目的として、合成時にスムーズな音を出すような音素の境界を決定することがある。つまり、合成において最も必要な音素の境界は、同じ音素の境界ならばできるだけ同じ一で境界を決定していることである。

今回の実験から、HMM の後に DTW を行なう多段階処理手法を用いることで、HMM よりハンドラベルとの絶対誤差が小さくなることが分かった。さらに、DTW を使うことによって、同じ 2 音素の組合せであっても、他とは類似性の低い音声データを見つけることができ、これらのデータについては合成時に利用しないなどの対策が可能である。

参考文献

[1] 小池 恒彦, 筧 一彦, 他 共著: 音声情報工学,NTT 技術移転株式会社,1987

[2] HTK Hidden Markov Model Toolkit V1.5, Entropic Reserch Laboratories Inc

[3] ESPS Programms A-L, Entropic Reserch Laboratories Inc

[4] 匂坂 芳典, 伊藤 いずみ, 木田 浩子, 他: 音声データ・ラベリング・マニュアル,ATR 自動翻訳電話研究所,1993.3

A シェルスクリプト

A.1 局所的 DTW を行なうスクリプト

```
#!/usr/local/bin/tcsh  # tcsh の立ちあげ

/usr/local/ESPS/bin/echeckout
/usr/local/ESPS/bin/hcheckout

set asta = "*"
set ubar = "_"
set colon = ":"

set hm = /home/as52/xnishi/data1
set wk = $hm/work_$1
set t_d = $hm/tmp_dir
set dt  = /home/as52/xnishi/data/work2

[ -f $t_d/all.$1.3phn ] || $hm/3phone..test $1 # 各文章データを便宜的に 3 音素づつに分割
[ -f $t_d/all.$1.3phn ] || cat $t_d/$1/* > $t_d/all.$1.3phn
                                        # 分割された文章データをひとまとめにする
echo "end cat process (make 3phn type data)"

[ -f v-c_$1_smpl.2phn ] || $hm/v-c_2phnsmpl.test $t_d/all.$1.3phn v-c_$1.2phn > v-c_$1_smpl.2phn  # 分
割したデータから、母音 - 子音の組合せをとり出す
```

```
echo "make v-c_$1.2phn and v-c_$1_smpl.2phn"

[ -f c-v_$1_smpl.2phn ] || $hm/c-v_2phnsmpl.test $t_d/all.$1.3phn c-v_$1.2phn > c-v_$1_smpl.2phn # 分
割したデータから、子音-母音の組合せをとり出す
echo "make c-v_$1.2phn and c-v_$1_smpl.2phn"

[ -f v-v_$1_smpl.2phn ] || $hm/v-v_2phnsmpl.test $t_d/all.$1.3phn v-v_$1.2phn > v-v_$1_smpl.2phn # 分
割したデータから、母音-母音の組合せをとり出す
echo "make v-v_$1.2phn and v-v_$1_smpl.2phn"

[ -f smpl.2phn.$1 ] || cat v-v_$1_smpl.2phn v-c_$1_smpl.2phn c-v_$1_smpl.2phn > smpl.2phn.$1 # 各
組合せを一つにまとめる
echo "end cat process (to make lanking_file)"

[ -f 2phnlank.$1 ] || awk '{if($6!=NULL && $7==NULL)printf"%s.%s\n", $1,$2}' smpl.2phn.$1 | sort | ur
同じ2音素の数を数えてソートする
echo "end make lanking_file"

if( !(-d $hm/$1/2phnlbl/) ) then
  mkdir $hm/$1/2phnlbl
  mkdir $hm/$1/2phnlbl/xlb
  mkdir $hm/$1/2phnlbl/lb
endif
 $hm/mk2phnlbl.test $1 $2 $3 # 各2音素のラベルデータをラベルデータから切り出す

if(!(-d $hm/$1/2phnsd/)) then
  mkdir $hm/$1/2phnsd
endif

#set phones = `cat $hm/2phnlank.$1`
set phones = `cat $hm/$4`
set sample = 50

while(1 < $phones[1] || $#phones) # 2個以上の2音素だけ計算する
  if($2 < $phones[1] && $phones[1] < $3) then
    set DIR = $phones[2]$phones[1]
    if($phones[1] > $sample) then
      $hm/rand_sample $phones[1] $sample > sample_dat_$DIR # ランダムに50個の数値を取り出す
    endif

    echo "start mk2phnsd.test $1 tmp set $phones[2]"
    [ -d $hm/$1/2phnsd/$DIR ] || $hm/mk2phnsd.test $1 tmp set $phones[2]
                                  # 音声データから、欲しい2音素の部分を切り出す
    echo "start 2phn_dtw.test2 $phones[2]"
    [ -d $dt/lb/$DIR ] || $hm/2phn_dtw.test2 $1 $DIR $phones[1]
                                # 切り出した2音素にDTWをかける

    echo "start diff_flame_2phn.test2 $phones[2]" # DTWの結果から新しくラベルを移動する量の計算と、
相対的な波形の類似性の計算、新しいラベルの出力
```

```
    set ONSO = `echo "$phones[2]:r"`
#    echo "$phones[2]:r"
    [ -f $dt/dist_$DIR.$1 ] || $hm/diff_flame_2phn.test2 $1 $DIR dist_$DIR.$1 $ONSO $phones[1]

    echo "end diff_flame_2phn.test2 $phones[2]"
    [ -f $hm/sample_dat_$DIR ] && mv $hm/sample_dat_$DIR $wk/sample/sample_dat_$DIR

    # 以降ファイルを消す手続き
    echo "start rm process $phones[2]"
     set TMP  = `awk '{printf"%s\n",$1'} $dt/dist_$DIR.$1`
    while($#TMP)
      set FILE = $TMP[1]:r
      echo "$FILE"
        rm -r $dt/map/$DIR/$FILE$ubar$asta $dt/cep/$DIR/$FILE$asta
        rm -r $hm/$1/2phnsd/$DIR/$FILE$asta $dt/lb/$DIR/$FILE$ubar$asta
        shift TMP
    end
    rm $wk/tmp
    echo "end rm process"
  endif
  shift phones
  shift phones
end

/usr/local/ESPS/bin/efree
/usr/local/ESPS/bin/hfree

#echo "end 2phn dtw_process"
```

A.1.1 2音素毎の局所的DTWの実行スクリプト

```
#!/usr/local/bin/tcsh
#example 2phn_dtw.test2 test a##581 581

setenv HCORERCE MFCC_D_E

set USE_ESPS_COMMON=off
#export USE_ESPS_COMMON

set esps_path = /usr/local/ESPS/bin
set htk_path  = /usr/local/HTK/HTK_V1.5/bin.sun4
set out       = /home/as52/xnishi/data
set hm        = /home/as52/xnishi/data1

set SPEAKER = $1
set SD_D    = $2
set DAT_NUM = $3

@ count = 0
```

```
#set wndw=50
set wndw=0

set ubar  = "_"
set asta  = "*"
set colon = ":"

[ -d $out/work2/cep/ ] || mkdir $out/work2/cep
[ -d $out/work2/map/ ] || mkdir $out/work2/map
[ -d $out/work2/lb/  ] || mkdir $out/work2/lb
[ -d $out/work2/cep/$SD_D/ ] || mkdir $out/work2/cep/$SD_D
[ -d $out/work2/map/$SD_D/ ] || mkdir $out/work2/map/$SD_D
[ -d $out/work2/lb/$SD_D/ ] || mkdir $out/work2/lb/$SD_D

set WAV = `ls $hm/$SPEAKER/2phnsd/$SD_D`

while($#WAV) # 全データに対して、ケプストラム作成
  set BNAME = `basename $WAV[1] .d`
#  echo "$BNAME"
  [ -f $out/work2/cep/$SD_D/$BNAME.cep ] || $htk_path/HCode -m -e -d -f 5.0 -F ESPS -O ESPS $hm/$SPEA
  shift WAV
end

echo "end make .cep file"

set WAV0 = `ls $hm/$SPEAKER/2phnsd/$SD_D`
while($#WAV0)
  set BNAME1 = `basename $WAV0[1] .d`
  if($DAT_NUM > 51) then
    set WAV_No = `cat $hm/sample_dat_$SD_D`
    set loop_limit = 51
  else
    set WAV_No = `ls $hm/$SPEAKER/2phnsd/$SD_D`
    set loop_limit = $DAT_NUM
  endif

  @ count = 1

  while($count < $loop_limit)
    if($DAT_NUM > 51) then
      set BNAME2 = `basename $hm/$SPEAKER/2phnsd/$SD_D/$asta.$WAV_No[1].$asta .d`
    else
      set BNAME2 = `basename $hm/$SPEAKER/2phnsd/$SD_D/$WAV_No[1] .d`
    endif

    if($BNAME1 != $BNAME2) then

# マッピングファイルの作成と、波形の類似度の計算
      [ -f $out/work2/map/$SD_D/$BNAME1$ubar$BNAME2.map ] || $hm/DTW/dtw $out/work2/cep/$SD_D/$BNAME1
```

```
# マッピング結果からラベルファイルを出力
      [ -f $out/work2/lb/$SD_D/$BNAME1$ubar$BNAME2.xlb ] || $hm/DTW/dtw2xlb $hm/$SPEAKER/2phnlbl/xlb/

      @ count = $count + 1
    else
      echo "same filename $BNAME1"
    endif
    shift WAV_No
  end
  echo "end about $BNAME1"
  shift WAV0
end
echo " end 2phn_dtw.sh loops are $count"
```

A.1.2 2音素に対するラベリングスクリプト

```
#!/usr/local/bin/tcsh
#example diff_flame_2phn.test2 test a.##581 dist_list a 581

set hm = /home/as52/xnishi/data1
set dt = /home/as52/xnishi/data
set wk = $dt/work2
set lbl = $hm/$1/2phnlbl

set no_lbl = 00
set exist  = 10
set last   = 11

set LBL_DIR   = $2
set dist_list = $3
set DAT_PHN   = $4
set DAT_NUM   = $5

set ubar  = "_"
set asta  = "*"
set colon = ":"

[ -d $wk/flame/ ] || mkdir $wk/flame

[ -d $wk/flame/$LBL_DIR/ ] || mkdir $wk/flame/$LBL_DIR

[ -f $wk/$dist_list ] && rm $wk/$dist_list

set DAT0   = `ls $lbl/xlb/$LBL_DIR`
while($#DAT0)
  set BNAME0 = `basename $DAT0[1] .xlb`
# 基準となるデータのラベルからそのラベルの時間を取り出す
  set REF_TIME = `gawk -f pickup_mid_time.2phn.awk $lbl/xlb/$LBL_DIR/$DAT0[1] $no_lbl $LBL_DIR 0.0 `
```

```
  if($DAT_NUM > 50) then
    set DAT1 = `cat $hm/sample_dat_$LBL_DIR`
    set loop_limit = 51
  else
    set DAT1 = `ls $lbl/xlb/$LBL_DIR`
    set loop_limit = $DAT_NUM
  endif

#  [ -f $wk/flame/$BNAME0 ] && rm $wk/flame/$BNAME0
  @ count = 1

#  echo "flag  Ref_flameNo flame difference distance time_sum time_diff flamesize_sum" > $wk/flame/$I

  while($count < $loop_limit)
    if($DAT_NUM > 50) then
      set BNAME1  = `basename $lbl/xlb/$LBL_DIR/$asta.$DAT1[1].$asta .xlb`
    else
      set BNAME1 = `basename $lbl/xlb/$LBL_DIR/$DAT1[1] .xlb`
    endif
    if($BNAME0 != $BNAME1) then
     # 基準ラベル以外のデータからのラベル時間のとりだし
      set TEST_TIME = `gawk -f pickup_mid_time.2phn.awk $lbl/xlb/$LBL_DIR/$BNAME1.xlb $no_lbl $LBL_DI

  #    echo "$DAT0[1] = $REF_TIME $BNAME1.xlb = $TEST_TIME"

     # 基準ラベルの時間がどこにマッピングされたかの計算
      set OLD_FLAME = `gawk -f calc_flame.awk $wk/map/$LBL_DIR/$BNAME0$ubar$BNAME1.map $REF_TIME $TES

     # 計算結果の出力
      echo "$OLD_FLAME[1] $OLD_FLAME[2] $OLD_FLAME[3] $OLD_FLAME[4] $OLD_FLAME[5] $OLD_FLAME[6] $OLD_

    @ count = $count + 1
    endif
    shift DAT1
  end
  echo "$DAT0[1] `gawk -f sum.awk '$wk/flame/$LBL_DIR/$BNAME0'`" >> $wk/$dist_list
  echo "$DAT0[1] `gawk -f sum.awk '$wk/flame/$LBL_DIR/$BNAME0'`"
  shift DAT0
end

echo "start label change process"
# ラベルの変更作業
set DAT = `ls $lbl/xlb/$LBL_DIR`
set AVE = `gawk -f caluc_ave.awk $wk/$dist_list $lbl/xlb/$LBL_DIR 0.0 0`

[ -f $wk/distance ] && rm $wk/distance

while($#DAT)
```

```
  set BNAME = `basename $DAT[1] .xlb`
  echo "$BNAME $AVE[1]" >> $wk/distance
  set MAX = `gawk -f find_max_dist.awk $wk/distance`
  shift AVE
  shift AVE
  shift DAT
end

set AVE = `gawk -f caluc_ave.awk $wk/$dist_list $lbl/xlb/$LBL_DIR 0.0 0`
set DAT = `ls $lbl/xlb/$LBL_DIR`
[ -d $wk/lblname ] || mkdir $wk/lblname

while($#DAT)
  set BNAME = `basename $DAT[1] .xlb`
  [ -f $wk/lblname/$BNAME ] && rm $wk/lblname/$BNAME
  gawk -f caluc_ave.awk $wk/$dist_list $wk/flame/$BNAME $AVE[1] 1 > $wk/lblname/$BNAME

  echo "$DAT[1]"

  set SOURCE_LBL = $DAT[1]:r:r
  [ -f $hm/work2/$SOURCE_LBL ] && cp $hm/work2/$SOURCE_LBL $hm/work2/tmp
  [ -f $hm/work2/$SOURCE_LBL ] && set INPUT = $hm/work2/tmp
#  [ -f $hm/work2/$SOURCE_LBL ] || set INPUT = $hm/$1/ESPSLB/$SOURCE_LBL.res
  [ -f $hm/work2/$SOURCE_LBL ] || set INPUT = $hm/$1/ESPSLB/$SOURCE_LBL.xlb
  gawk -f new_label_out.awk $INPUT $DAT_PHN $AVE[2] $AVE[1] $MAX $lbl/lb/$LBL_DIR/$BNAME.lb > $hm/wor

  shift AVE
  shift AVE
  shift DAT
end
rm $hm/work2/tmp
unset DAT0
unset DAT1
unset BNAME0
unset BNAME1
```

B　ボックスグラフ

各図の縦軸は、時間を表している。(単位:msec) この時間はハンドラベルを基準として、ハンドラベルの示す境界より遅いところを境界とした時は正、逆に早いところを境界とした時は負として、その差の統計をとったものである。箱の幅は、観測数の平方根に比例し、切れ込み部分は中央値の 95% 信頼区間を示す。

B.1　HMM によるアライニング

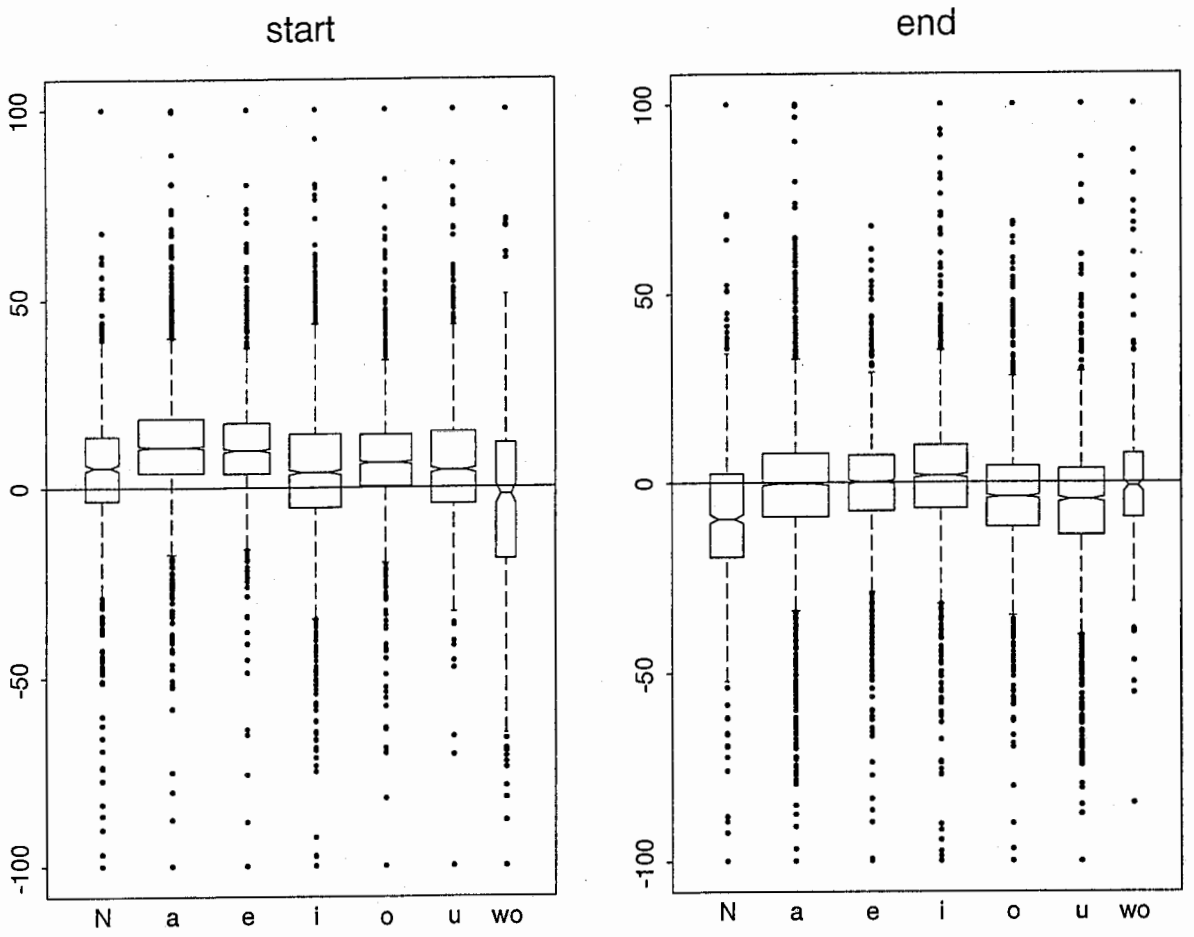

図 4: HMM によるアライニング結果とハンドラベリングの誤差 (母音)

この結果から HMM は、母音の始まりが全体的にハンドラベリングより遅く、その影響で、子音の終わりも全体的に遅くなっている。また、母音の終わりは遅くはないので、子音の始まりもあまり遅くなっていない。しかし、全体的に各子音はそれぞれ特徴のあるばらつきをしている。

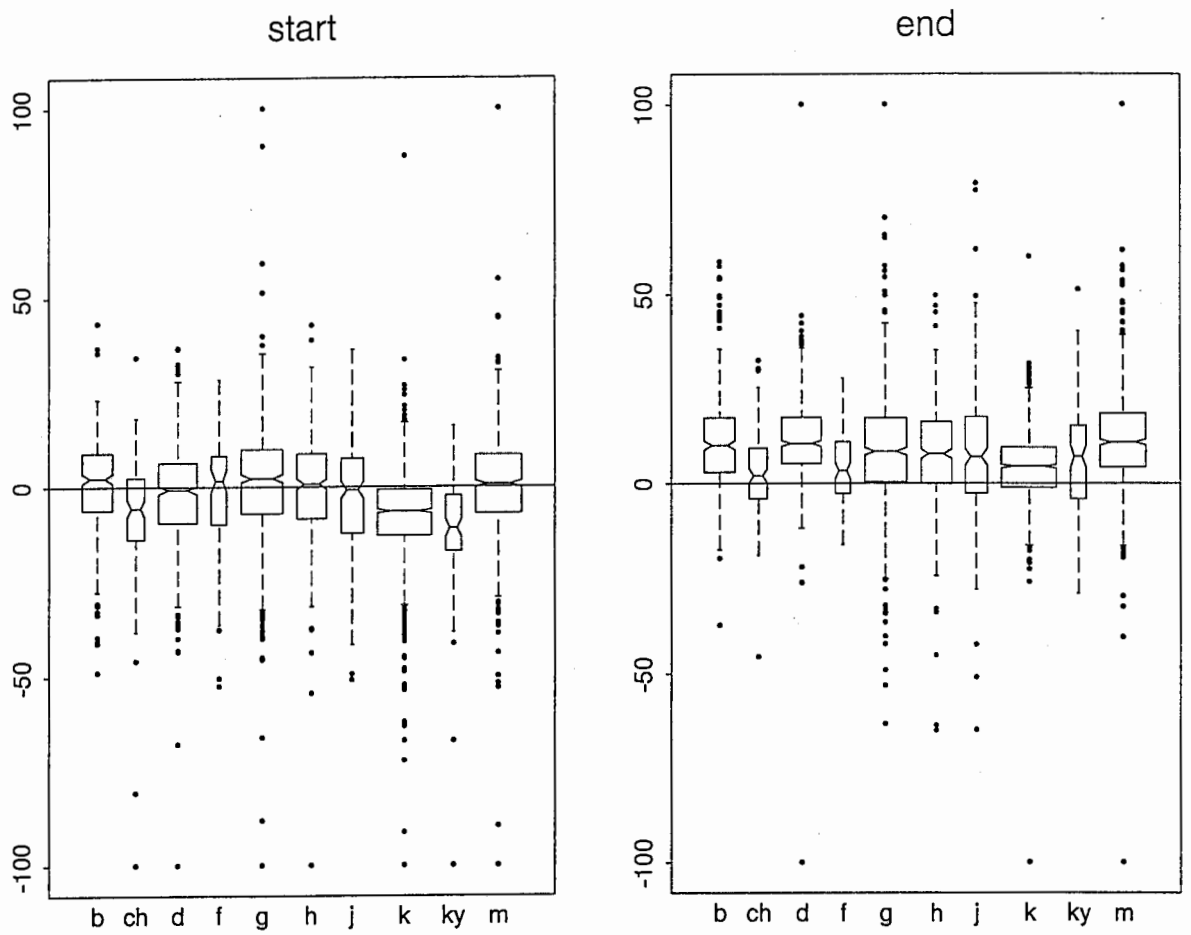

図 5: HMM によるアライニング結果とハンドラベリングの誤差 (子音 1)

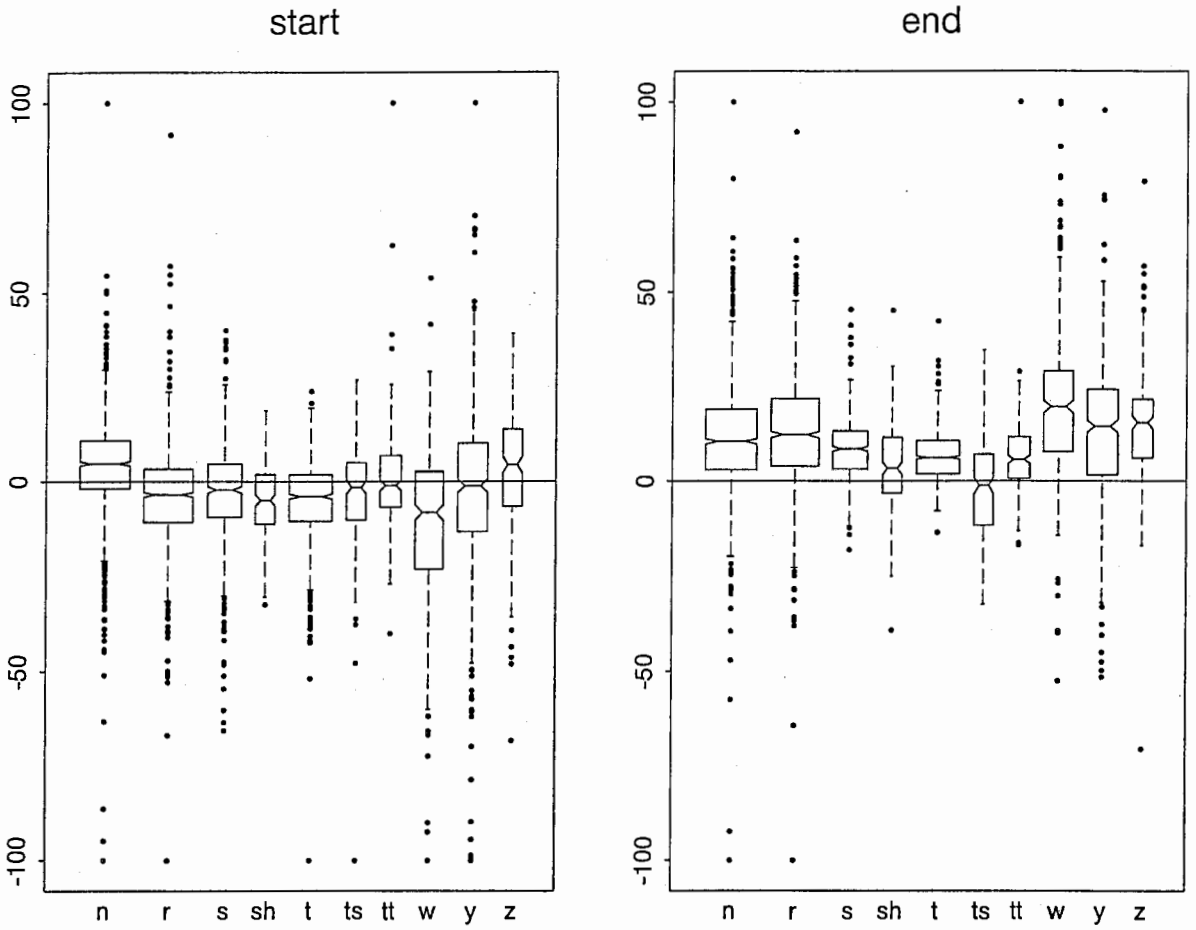

図 6: HMM によるアライニング結果とハンドラベリングの誤差 (子音 2)

B.2 DTW によるアライニング (男性話者 (MHT) から男性話者 (MHO))

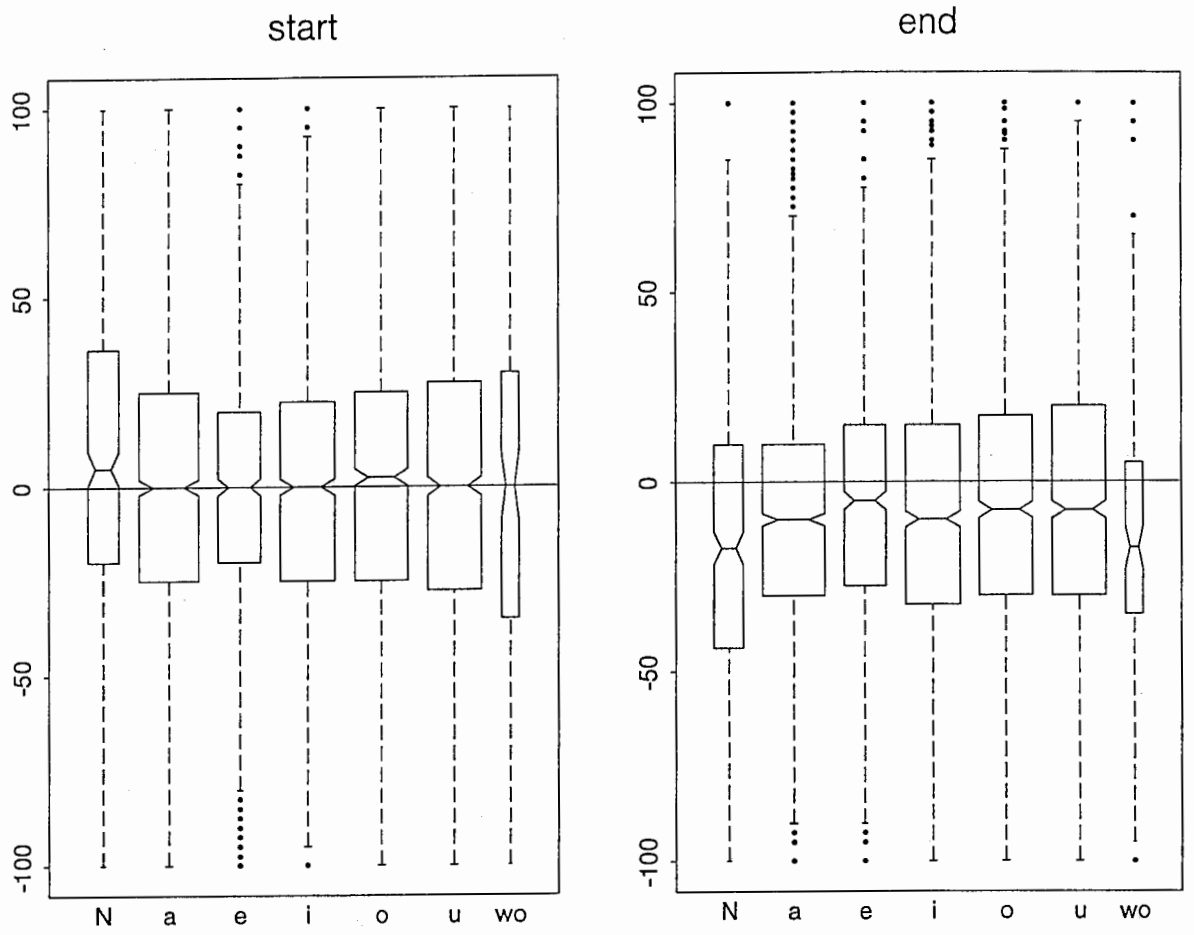

図 7: DTW によるアライニング結果とハンドラベリングの誤差 (母音)

この結果から、HMM に比べてハンドラベルとの差にばらつきがある。しかし、HMM のように子音によってばらつきの特徴が少ない。この DTW の傾向は、基準となる話者が変わった場合でも変化は少ない。

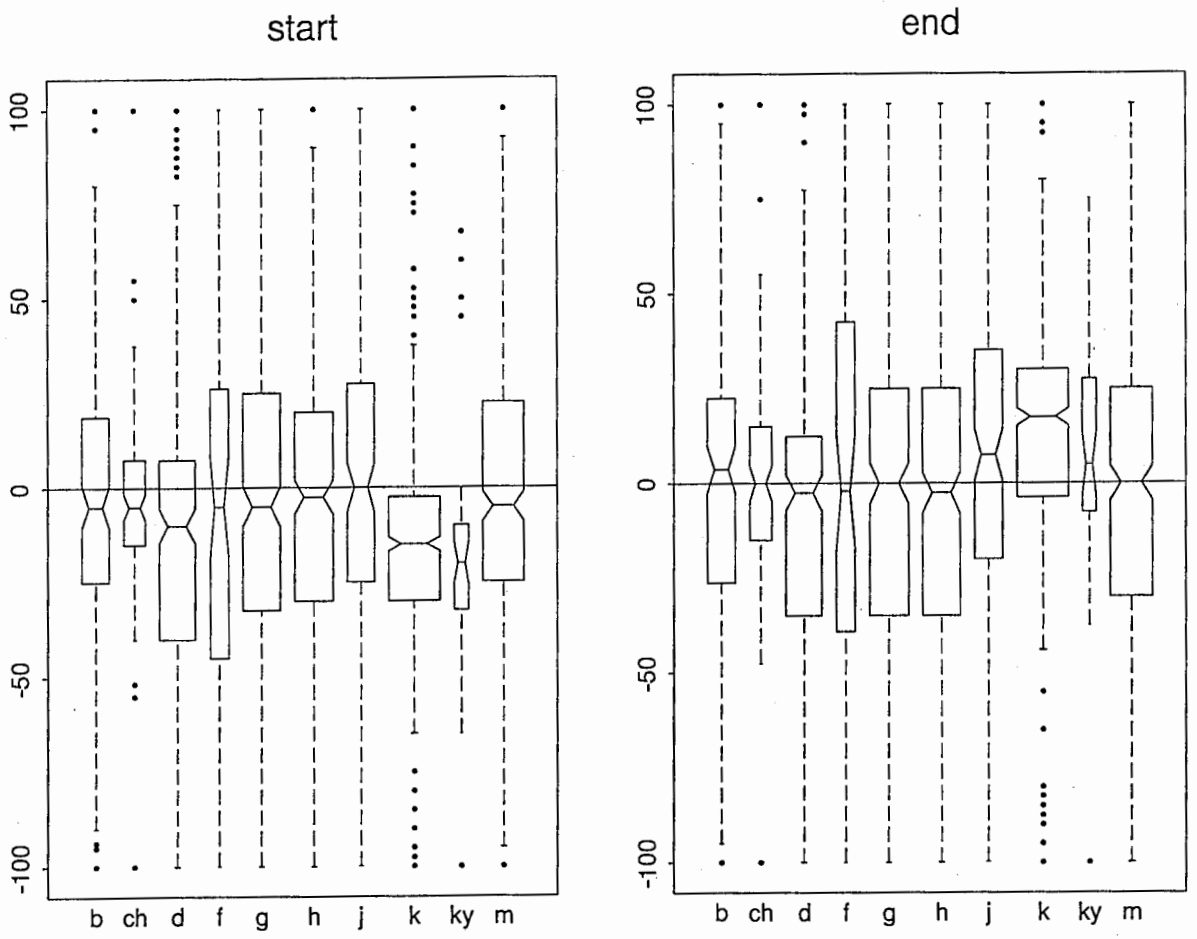

図 8: DTW によるアライニング結果とハンドラベリングの誤差 (子音 1)

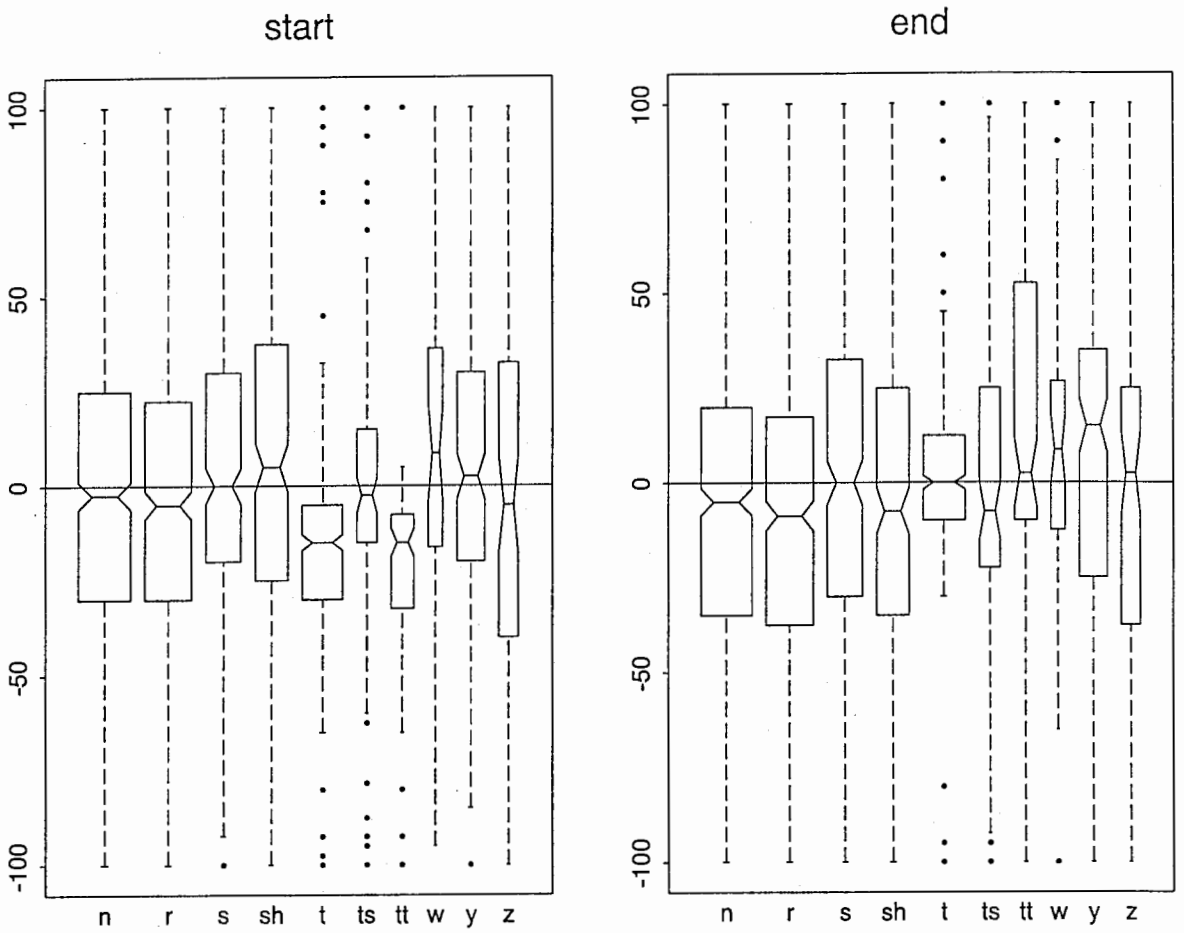

図 9: DTW によるアライニング結果とハンドラベリングの誤差 (子音 2)

B.3 DTW によるアライニング (男性話者 (MHT) から女性話者 (FKN))

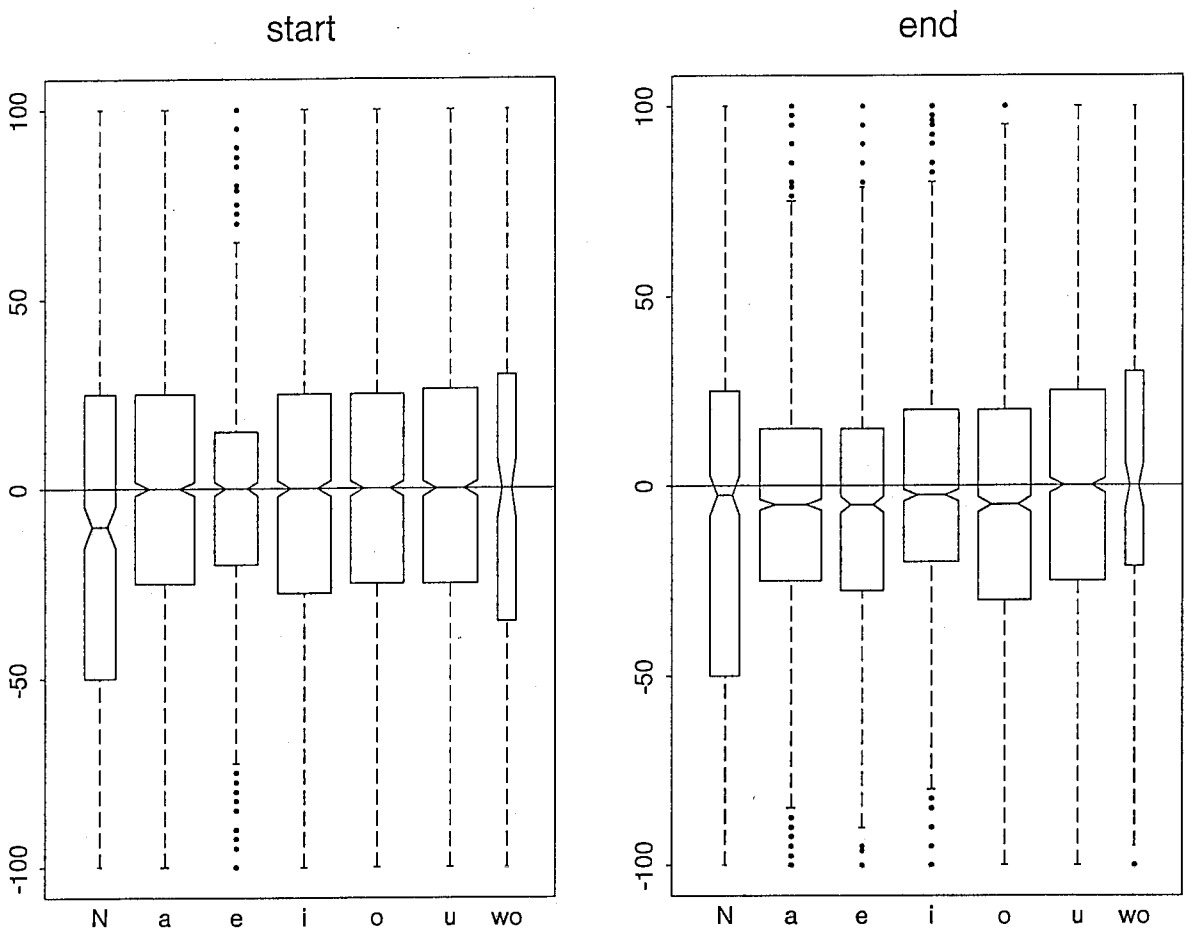

図 10: DTW によるアライニング結果とハンドラベリングの誤差 (母音)

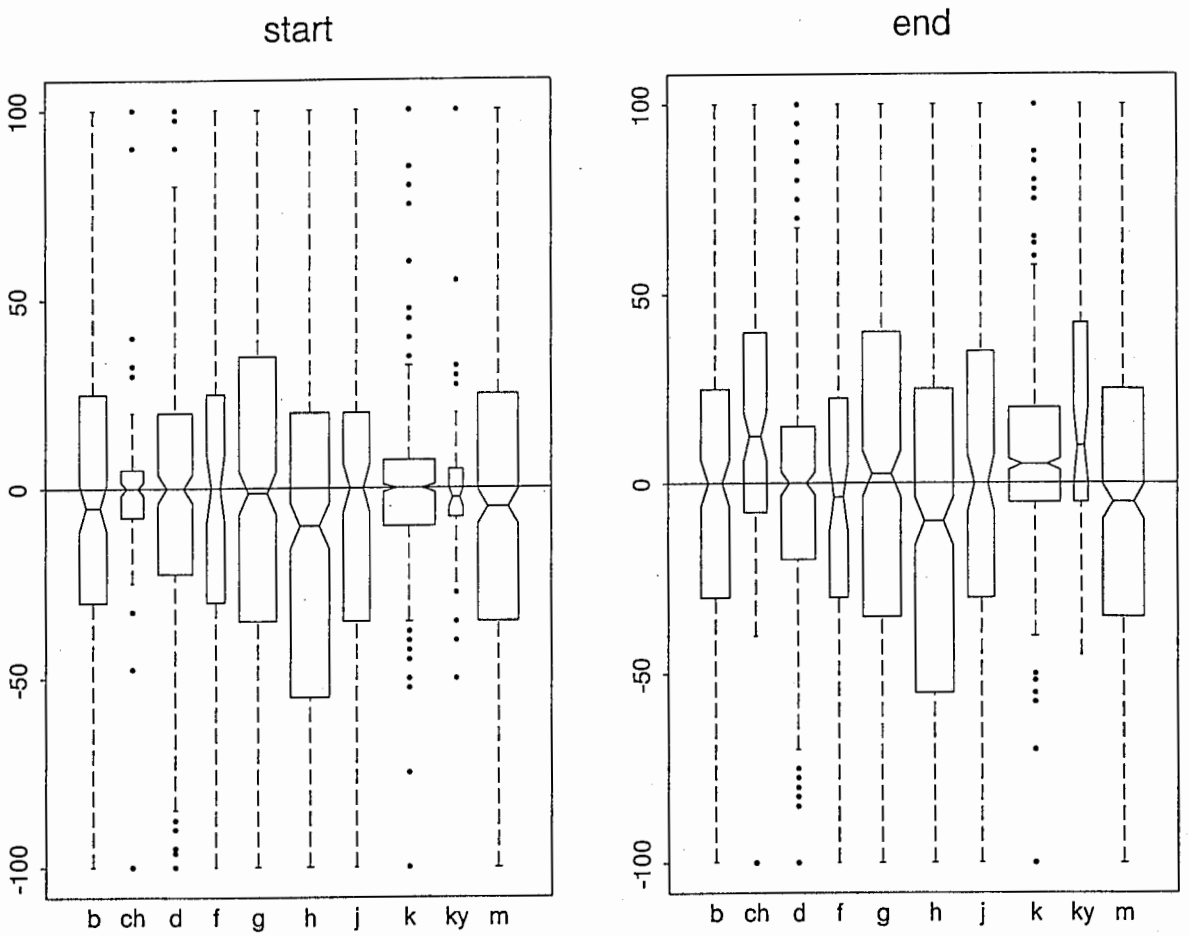

図 11: DTW によるアライニング結果とハンドラベリングの誤差 (子音 1)

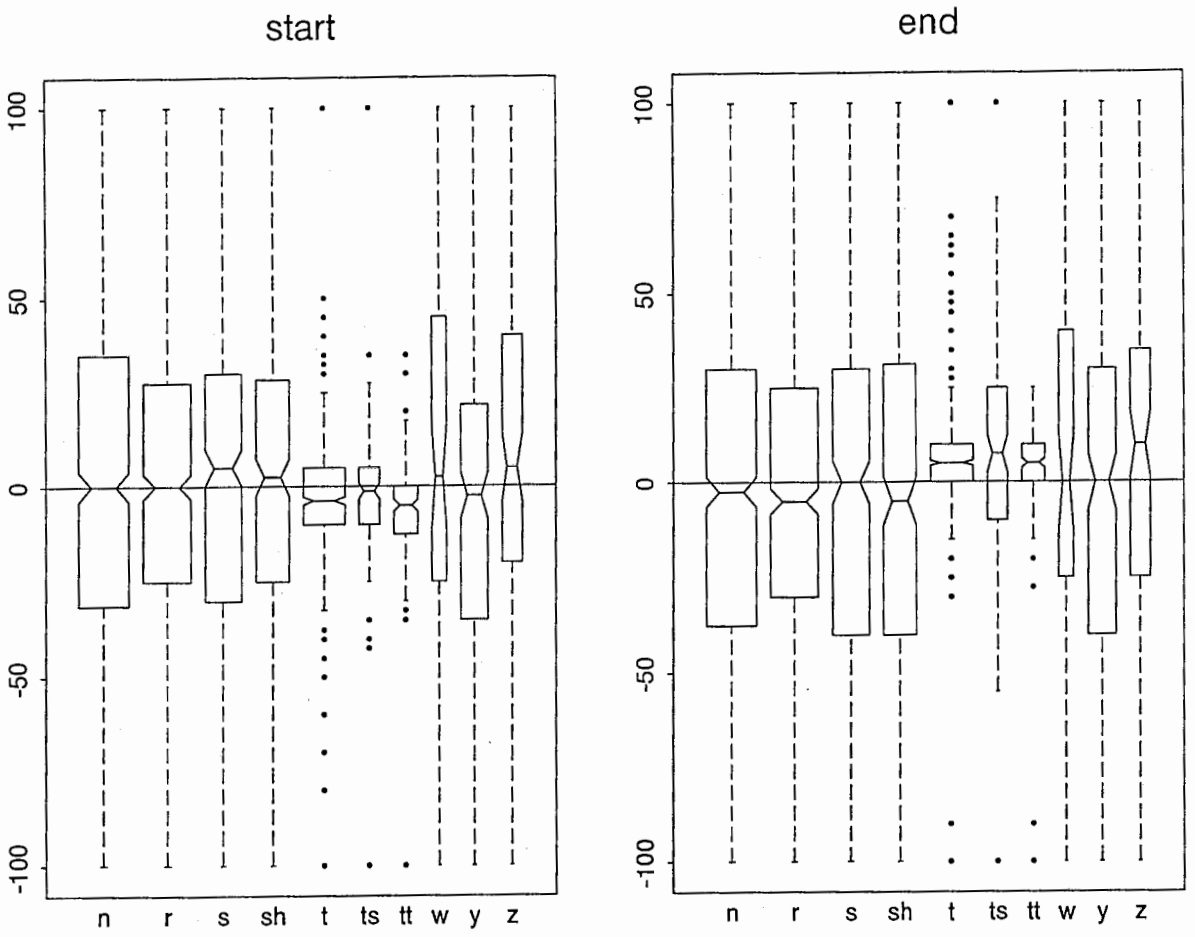

図 12: DTW によるアライニング結果とハンドラベリングの誤差 (子音 2)

B.4 DTW によるアライニング (女性話者 (FYM) から男性話者 (MYI))

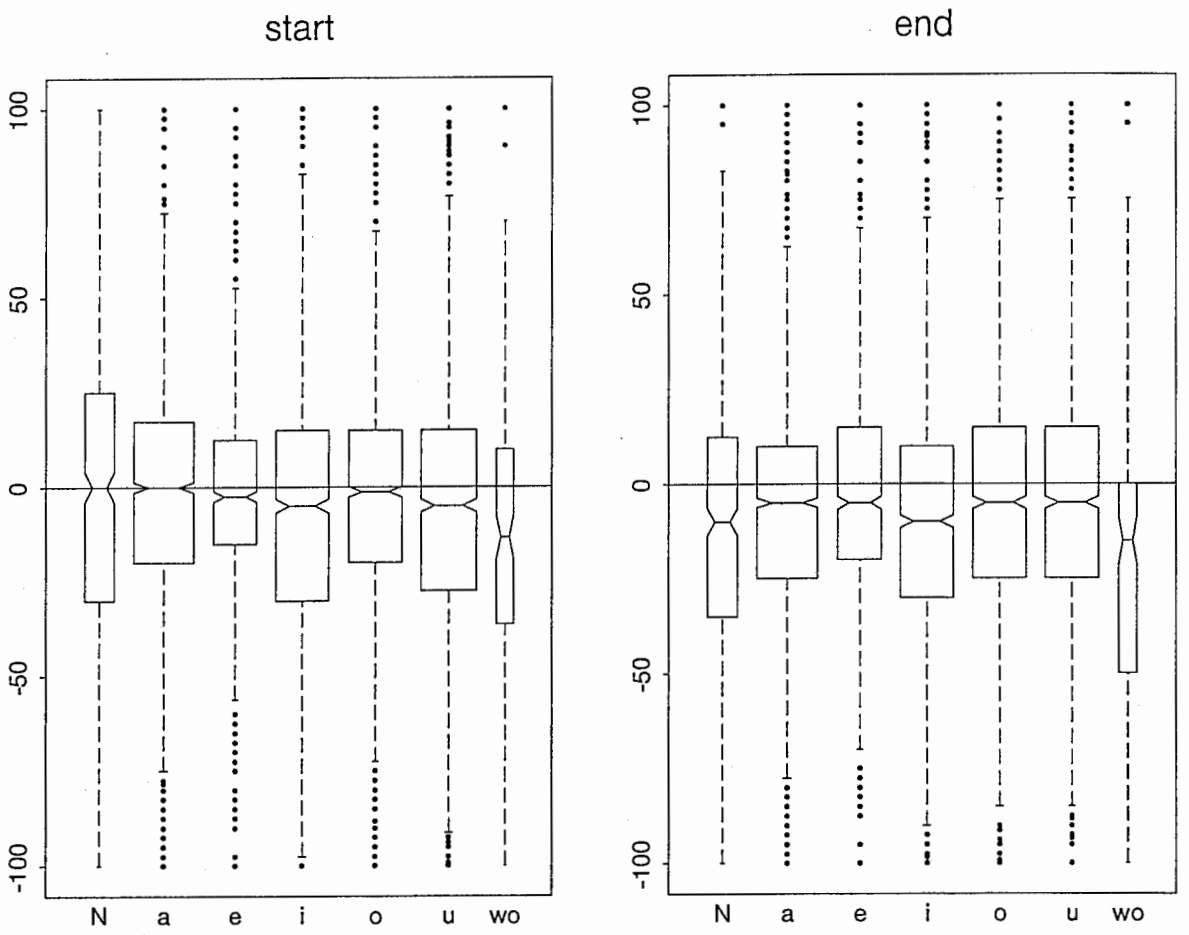

図 13: DTW によるアライニング結果とハンドラベリングの誤差 (母音)

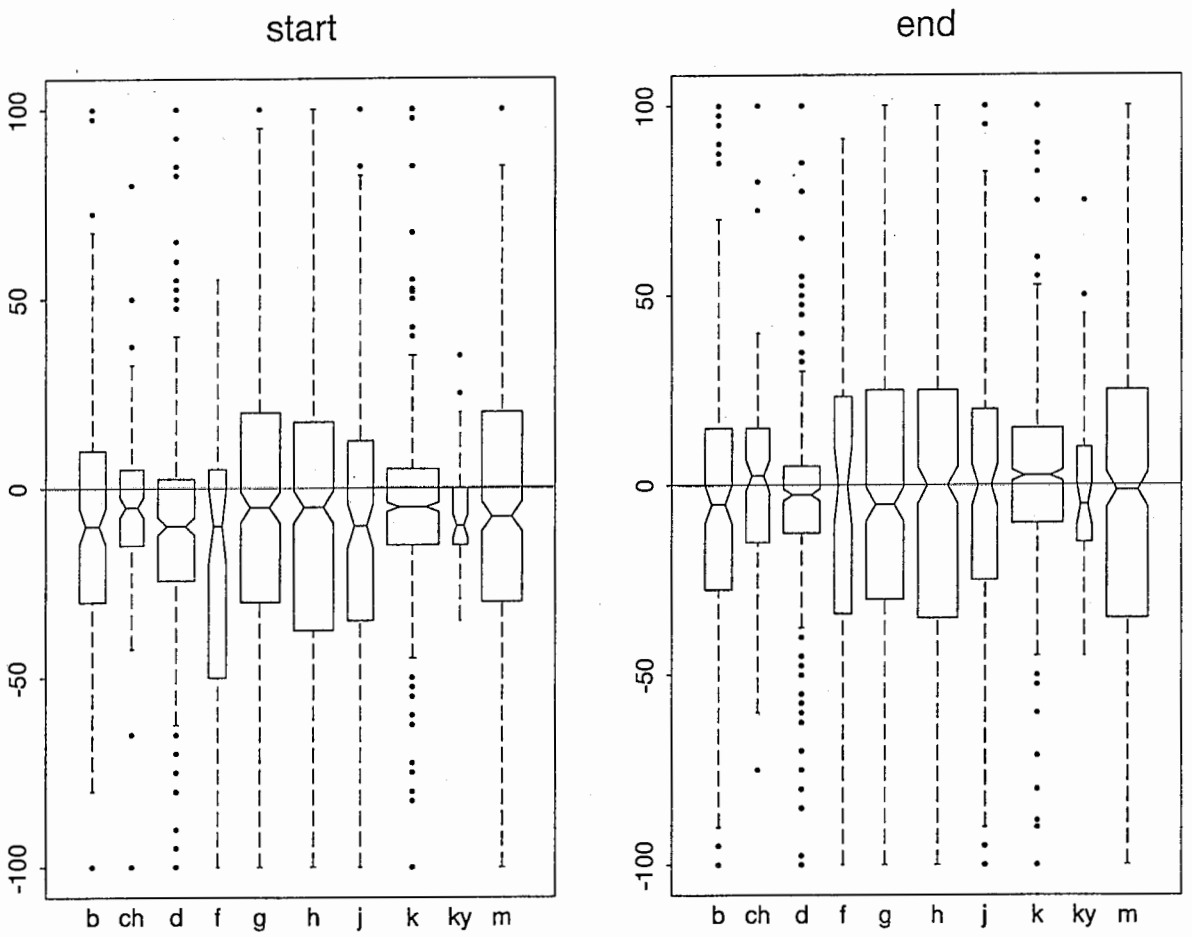

図 14: DTW によるアライニング結果とハンドラベリングの誤差 (子音 1)

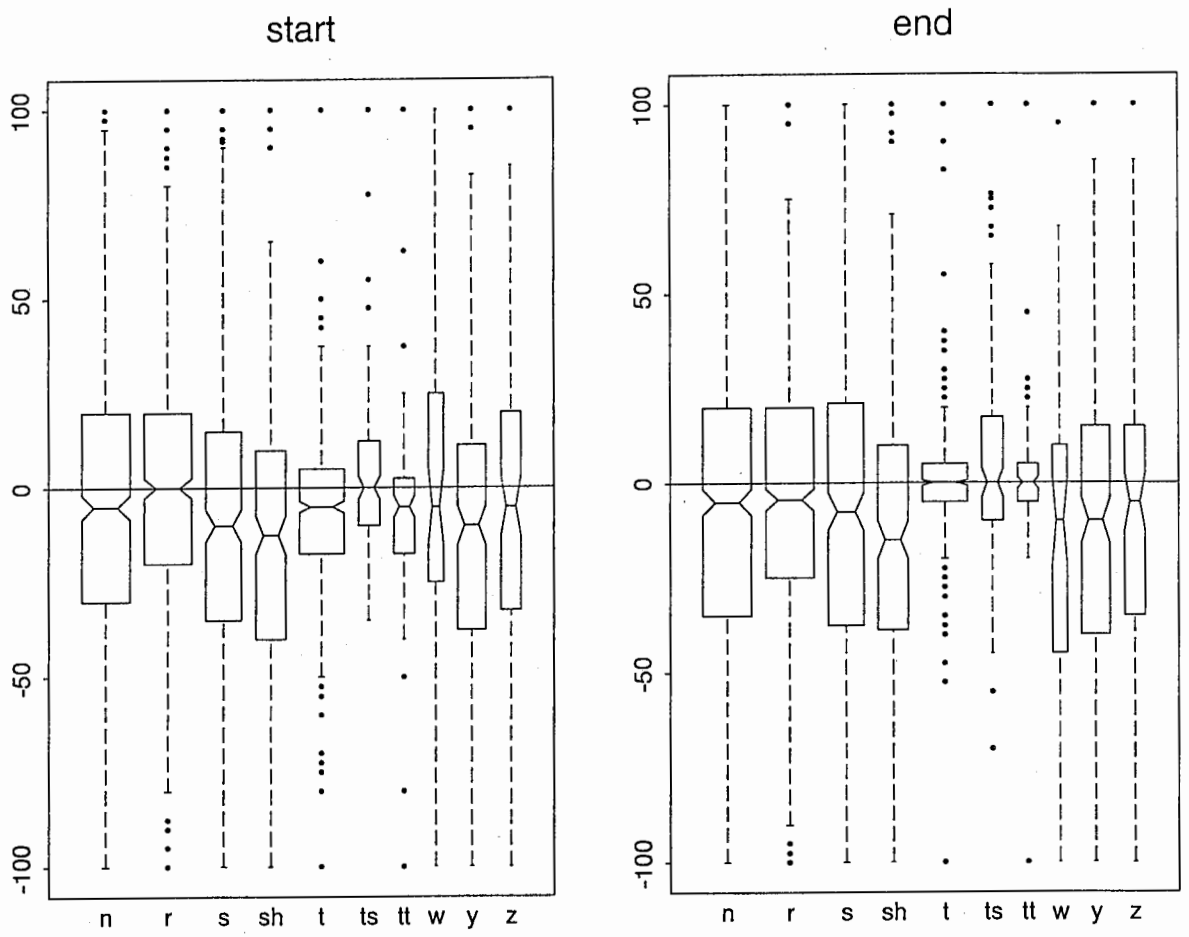

図 15: DTW によるアライニング結果とハンドラベリングの誤差 (子音 2)

B.5 DTW によるアライニング (女性話者 (FKS) から女性話者 (FTK))

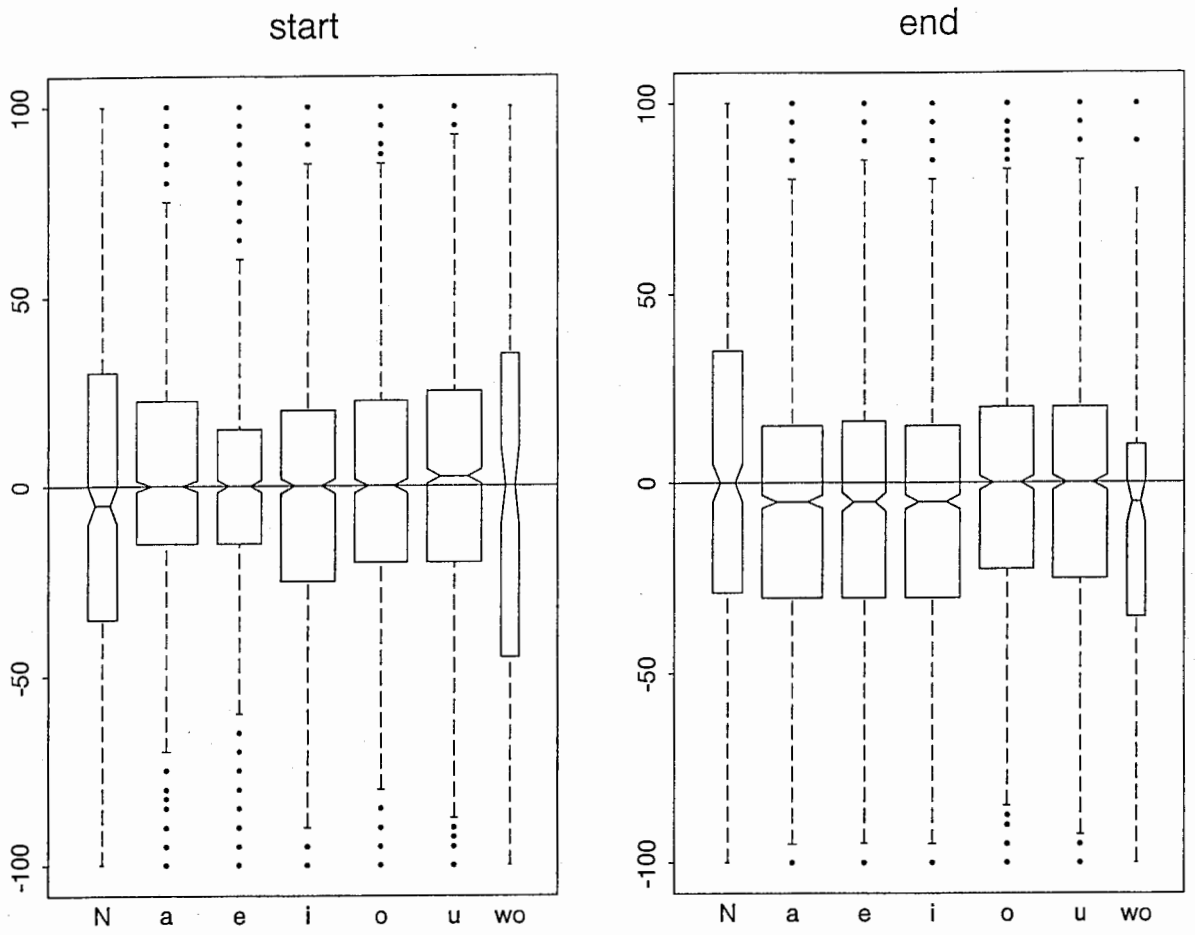

図 16: DTW によるアライニング結果とハンドラベリングの誤差 (母音)

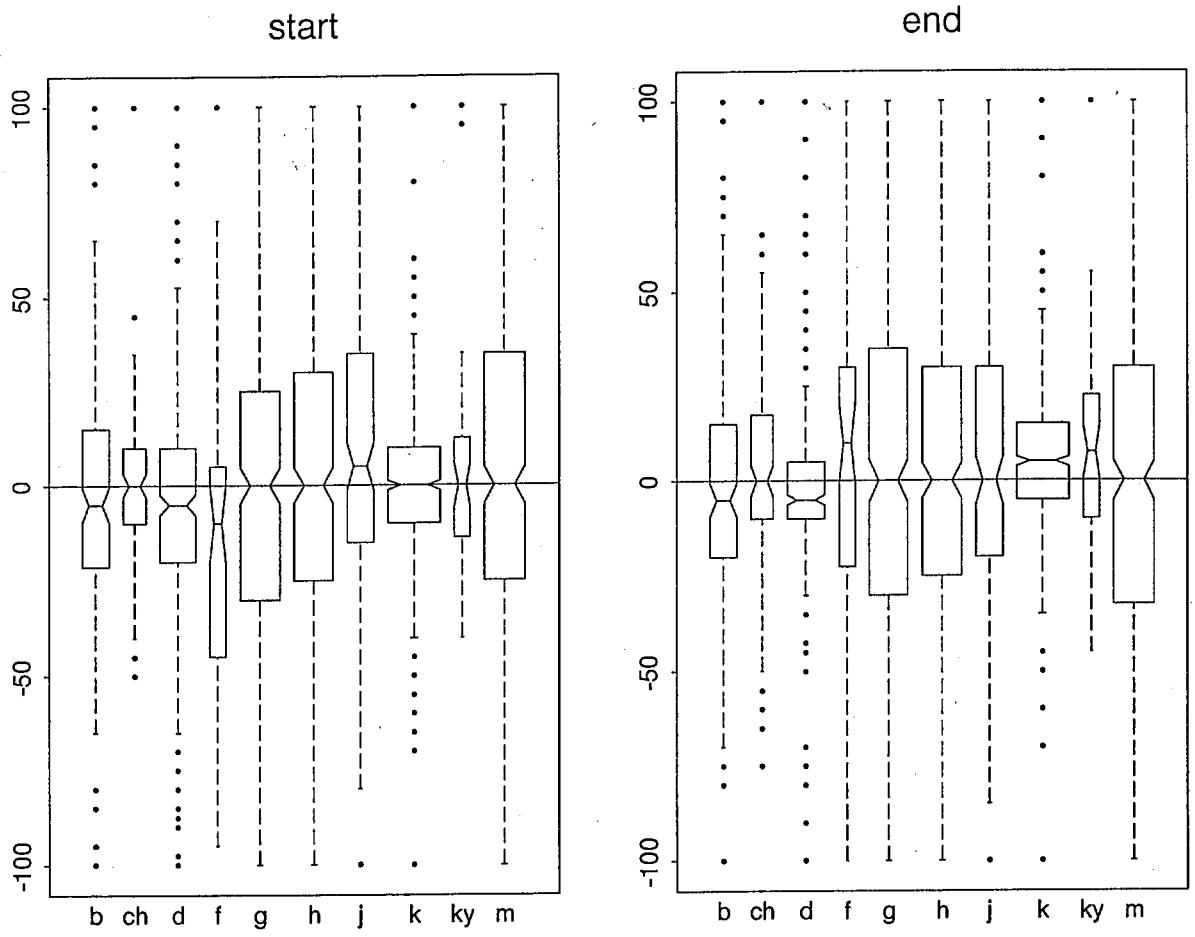

図 17: DTW によるアライニング結果とハンドラベリングの誤差 (子音 1)

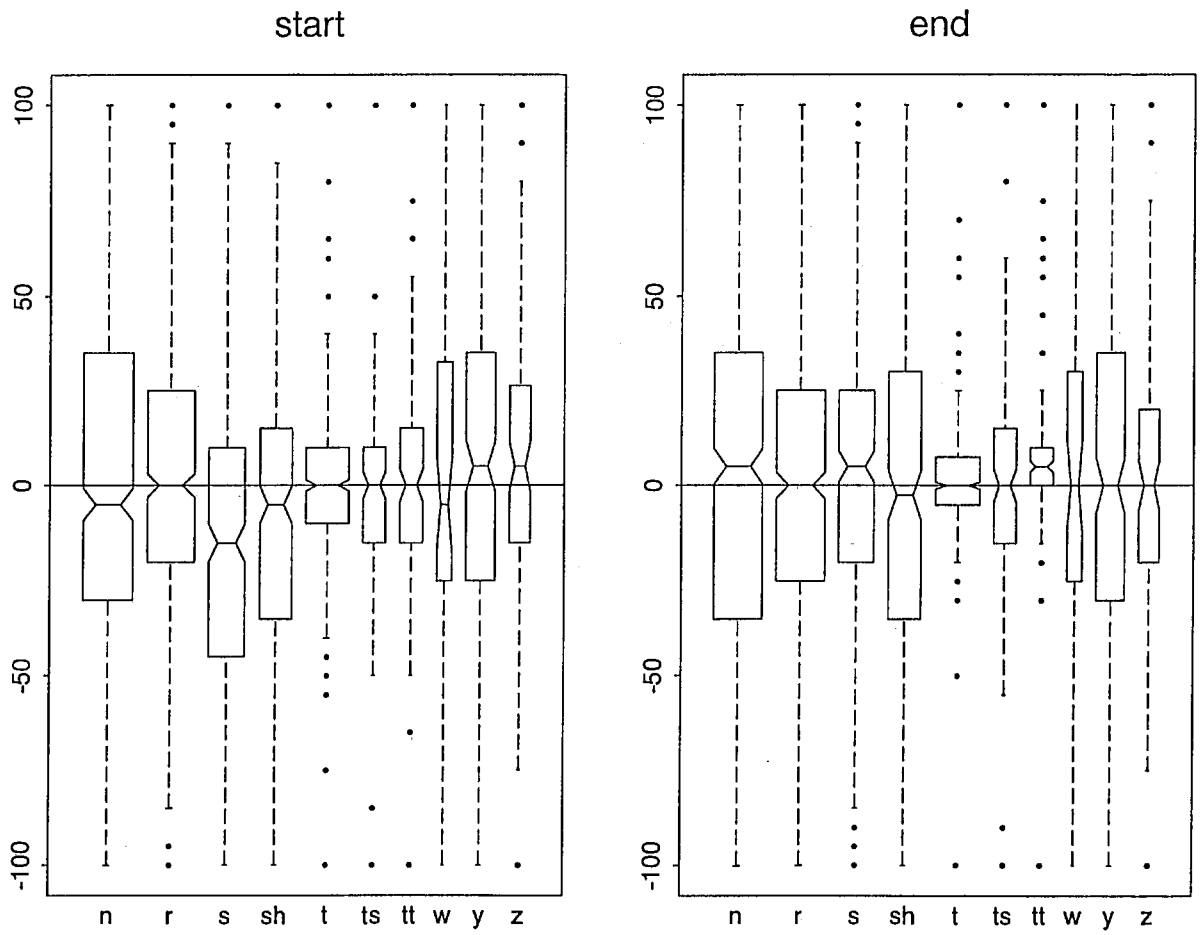

図 18: DTW によるアライニング結果とハンドラベリングの誤差 (子音 2)