CUI での音節の切り出しソフトの開発について

平成 13 年 2 月 14 日

情報電子工学科 竹野研究室
船橋 崇将

目 次

概要

昨年の研究において、UNIX 上で動作する日本語文書読み上げソフトについての改良が行われた。このソフトの仕組みは、1 音節ごとに用意されている音声データを連結して出力する方法を用いている。そのことから各音節のデータをユーザが自分で録音した音声のデータに置き換えることが可能である。しかし、録音した 1 音節ごとの音声データは音声の出だしの位置、音量、音程が整っていないため、このまま音声データを連結して出力しても、音節ごとに ばらつき が生じて聞き取りにくいものとなる。そのため音声データの編集を行なう必要が生じる。本稿では音声データを作成する方法と音声データの編集についての考察と問題について検討する。また、読み上げソフトは CUI と呼ばれる環境においての使用が考えられているので、音声データの編集も CUI 環境という条件をつけて行なうことにする。

1　はじめに

この研究室で開発されている文書読み上げソフト yomi は、ひらがなに変換された文書に 1 つ 1 つ音声データを割り当て、それを連結して音声を出力させるという方法を用いるもので、作られた目的は文書の誤字脱字を減らす、文書の中に変換ミスがないかを調べるといった文書の確認をすることに用いることを主なものとしている。このソフトは文書を音声で出力させるしくみであるため、画面を使う必要はない。そのため CUI と呼ばれる UNIX のような文字やテキストのみであらわされる環境において、いろいろな用途に使える可能性があり、その一つに視覚障害者への活用もできるのではないかと考えられている。

昨年の研究 [1] では、このソフトの改良が行なわれた。しかし、その改良は漢字からひらがなへの文書の変換に関するもので、音声データに関するものは行なわれておらず、今回はこのソフトの音声データの編集に関しての考察を行うことにした。

このソフトには、最初からあらかじめ音声データが用意されているが、ソフトの動作原理からして、ユーザが自分で録音した音声データで置き換えることが可能になっている。録音して作成した音声データはそのまま読み上げソフトに使うこともできるが、各データにはそれぞれ音声部分の出だしの位置や音量、音声の長さにばらつきが生じており、音声を流して聞いても聞き取り易いものとはいえない。その解決のため、作成した音声データを読み上げソフトに使えるように音声データの編集を行なう必要がある。音声データの編集は録音した音声データを整えることである。そのために音声データに入っている音声以外の不要な部分を取り除くことが最も重要である。各音節ごとのデータの音声部分の出だしをそろえるために、録音した音声データから音声部分を探しだす方法についても考える。

2　開発にあたって

2.1　日本語文書読み上げソフト

この読み上げソフトはフリーソフトとして公開されている。一般的にフリーソフトの利点は

1. 自由に配布ができる
2. 無料で入手できる
3. ソースが公開されているためユーザが改良できる

といったものが挙げられる。場合によっては制限を持つものもあるが、フリーである分多くの人に利用してもらえることでソフトの改良、開発に参加してもらうことができる。しかし、フリーであるためにソフトの使用には自分で責任を負わなければならない欠点を持つ。

この読み上げソフトはフリーとすることで、ユーザがソフトの改良や音声データの変更を自由に行なえる利点を持たせている。

2.1.1 読み上げのしくみ

このソフトの動作の流れを Fig.1 に示す。まず、漢字とかなの混ざった文章の漢字の部

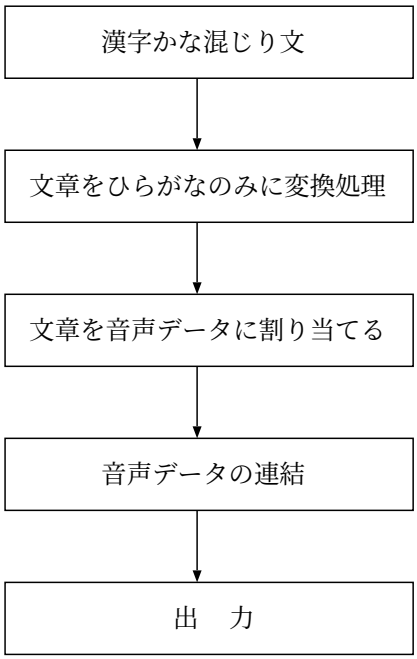

Fig. 1 全体の流れ

分をひらがなに変換する。次に、変換を行った結果ひらがなのみで構成された文章に 1 音節ごとに音声データを当てはめていく。そして、その音声データの出力を行うが、1 音節ごとに音声を出力すると、音声は途切れ途切れになって聞こえてしまうため、文章に当てた 1 音節ごとのデータを連結して出力することで、音声の途切れが生じないようにしている。

2.1.2 音声データ出力について

このソフトに用いられる音声は 1 音節ごとの音声データとして用意されている。このソフトで使われる音声データは以下の通りである。

1. 50 音データ (45 個)
2. 濁音・半濁音のデータ (24 個)
3. 「きゃ」「しょ」等の拗音を含むデータ (47 個)
4. アルファベット (A 〜 Z の 26 個)

5. 数字 (0 〜 9 、百 (ヒャク) 千 (セン) 等の単位、ハッピャク・サンゼン等の特殊な発音の数値 18 個)

6. 無音データ (1 個)

これらの音声データをひらがなに変換された文章に 1 文字ずつ割り当て連結させて出力する。その際、一度に連結する文章の長さは、句点があらわれるまでの長さである。

読み上げソフトにはあらかじめ編集された音声データ用意されている。これらの音声に不満が生じた場合や音声データが消えてしまった場合に対して、ユーザが音声データを作成し、そのデータに置き換えて使うことができるようになっている。

ここでユーザが音声データを作成する上で二つの問題が生じる。一つは録音方法に関して考える必要があるということである。音声データの録音時において、必要な音声データの数は前に説明したように 50 音や濁音、拗音等、その数は 100 音節以上である。そのため、それらを 1 音節ごとに録音していくとかなりの時間がかってしまう。録音作業を短時間で行なえるようにするための方法について考える必要がある。

もう一つは録音した音声データを読み上げソフトに登録する前に音声データの編集する必要があることである。この読み上げソフトの特徴は 1 音節ごとに用意されている音声データを連結して出力することである。その際、音声データはファイルとして置かれているものをそのまま連結に使用する。ただ録音しただけの音声データは各音節ごとに音声の出だしやデータの長さ、音量の大きさが整っていないため、このままの音声データを使って音声を連結させて出力すると、1 音節ごとに不規則に音声がでてきて聞き取りにくいものとなってしまう (Fig.2)。

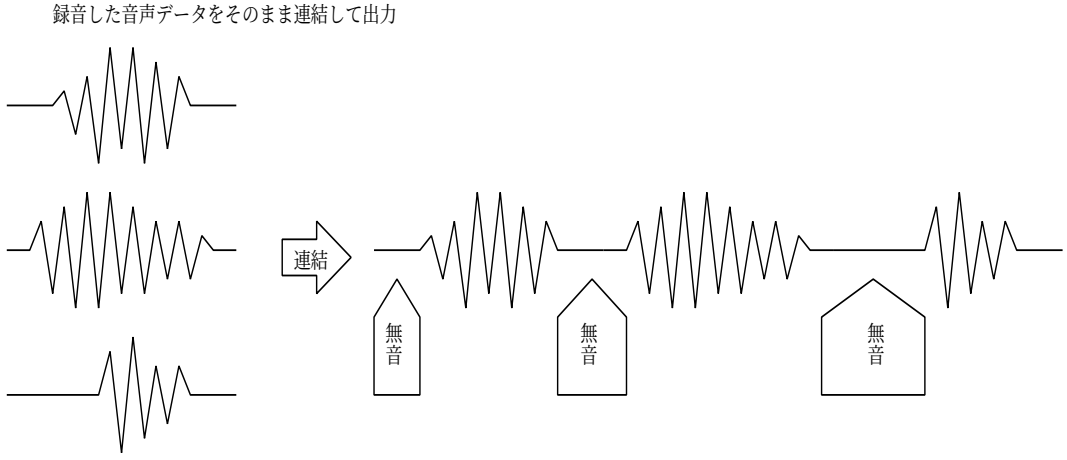

Fig. 2 録音したデータをそのまま使う

音声データの編集を行なって Fig.3 に示したように、音声データの不要部分を取り除くことで読み上げの音声を聞き取り易くなるようにする必要がある。音声データの編集に関して、録音した音声データから音声だけのデータを作りだすために、音声以外の不要部分を取り除く方法について考察する。

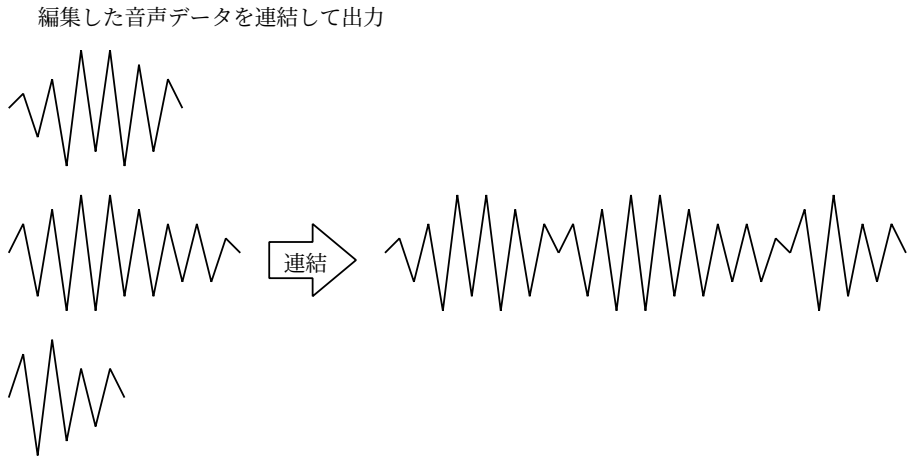

Fig. 3　音声データの無音部分を取り除く

3　音声データの録音に関して

音声データの録音方法に関して、1 音節ごとに音声を録音していくのでは時間がかかってしまう。その問題を解決する方法として 1 回の録音で複数の音節を吹き込んだ音声データを作成し、後でその音声データを 1 音節ごとに切り分けて個々の音声データを作成していくというものについて考えてみる。

1 度に複数の音節を録音すれば、録音作業に費す時間を減らすことができる。しかし、複数の音節の入った音声データに分ける作業を手動で行ってしまうと時間を費してしまうので、自動的に切り分けを行なわせる方法を条件として考えることとする。

複数の音節を録音したデータを波形であらわすと Fig.4 のように示すことができる。波形が大きくでている区間が各音節の音声の部分とみなせば、自動的に切り分ける方法は切り分けが出来そうである。この方法を用いての音声データ作成について検討してみる。

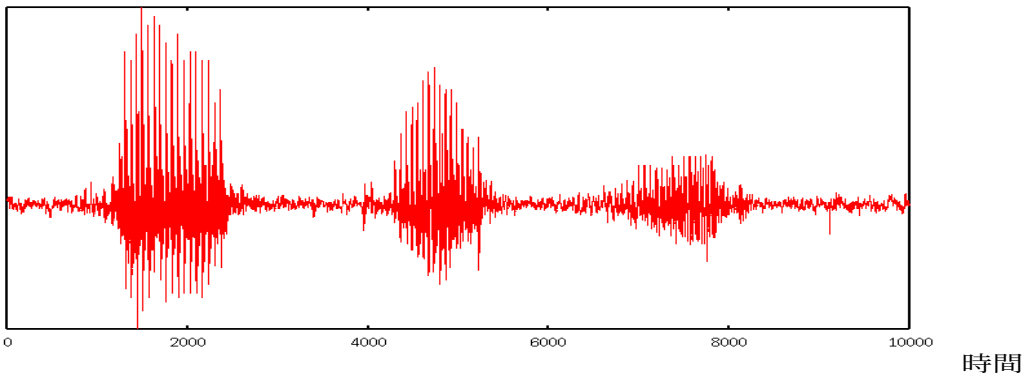

Fig. 4　音声波形複数のデータファイル 「がぎぐ」

1. 費す時間は録音だけなので短時間で音声データを作成できる
2. 小さな雑音波形は無視できるが大きな雑音だった場合はその部分も音節に含まれて切り分けられる可能性がある
3. 1 音節の音声を複数の波形区間であらわしている場合、この方法だと別々の音節のデータとみなして切り分けられてしまう
4. 小さな波形でも音声に必要である場合に対処できない

2. に対しては後で音声データの編集作業によって取り除くことができるが、 3. や 4. は別々のデータに分けられてしまうため、その音節のデータの録音をやりなおす必要性が生じてしまう。3. や 4. の条件を満たす音節は、子音を持つ音声データにかなり入っているため、うまく切り分けられる音声データは少ないようである。この方法では 1 音節ごとに録音する方法とたいして変わらないかもしれない。

4　音節の頭だしに関して

通常、録音した音声データを波形で見てみると Fig.5 に示されているように、音声は出だしの辺りには無く、ある程度の無音部分が存在してから音声部分があらわれている。この部分は読み上げ時において連結して出力させた際に音声を途切れ途切れにする不要な部分である。音声部分の後ろにも無音部分が存在している。これらの部分を取り除き Fig.6 のような音声データの出だしを音声部分からはじまるものにしたい。

4.1　頭だしの方法

音声データの編集に関して、GUI と呼ばれる MS-Windows のような、画面を見ながらマウスを使って作業する環境を考えるならば、音声データの編集作業を行える音声波形エディタは、市販されているものやフリーのものなど既に数多く存在している。それらのエディタは、主に音声データを編集画面に波形で表示して、画面を見ながら音声データの切り取り・コピー・貼り付け等の編集が行えるようなツールである。これらのツールを利用すれば、音声波形のあらわれている位置を容易に探しだすことができる。

しかし、 GUI 環境における編集方法は画面を見て作業するものであるため、視覚障害者には利用できるものではない。画面を利用しないで音声データの頭だし作業を行う方法を考えてみる。画面を見ないで行う方法は限られてしまい、切り分けを行う手段として自動的に行うか、手動で行なうかがまず挙げられる。

4.1.1　自動で頭だしの位置を知る

音声データは波形の単位時間ごとの音声信号の強さを示す値が記録されている。このことから Fig.7 に示したように、特定の範囲から各データの値を読みとっていき、その値を

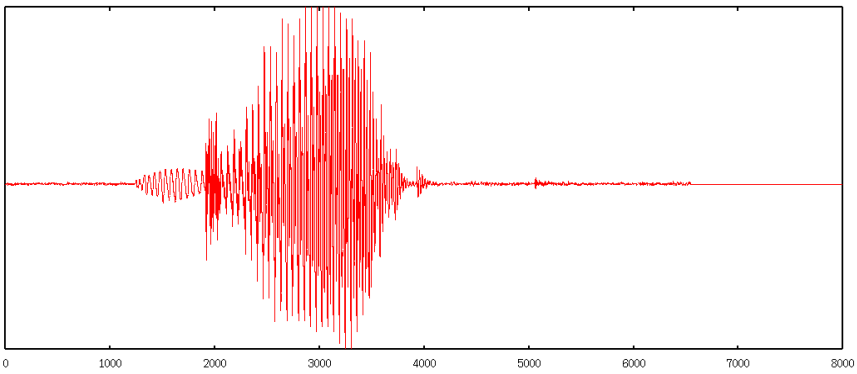

Fig. 5 データの始まりに無音部分 「ぐ」

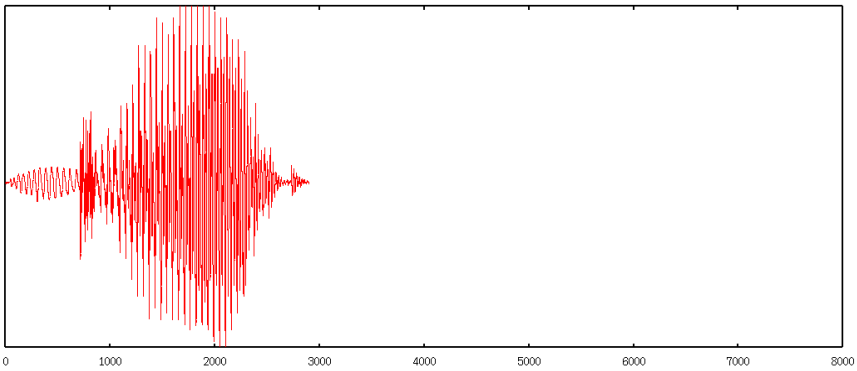

Fig. 6 データの始まりに音声部分 「ぐ」

もとにして導き出される関数と x 軸の交わるところを頭だしの位置とする方法ができるのではないか。この方法ができるならば音声データの頭だし作業は自動で行なえるので短時間で作業を行なうことができるだろう。

しかし各音節の音声データの波形を見てみると、Fig.8 に示されているように、この方法では音声データによっては関数で導きだした結果によっては無音の部分を含んでいたり、音声の必要な部分を削除して取り出してしまうおそれがあり、しかも、関数を求めるデータの範囲の基準が曖昧なため、失敗する可能性が高いとおもわれる。このことから頭だしの位置を自動的に探る方法は難しいと推定する。

4.1.2 音声を聞きながら頭だし

人が直接関わって音声データの頭だしを行なう方法として、音声データを出力させて音声を聞いていき、その結果で音声の出だしの位置を決めるという方法が考えられる。この方法は普通に音声データを出力させては意味がないので、音声データを出力させる位置を任意で指定するという方法をとることにする。そうすれば音声データの音声部分の頭の位

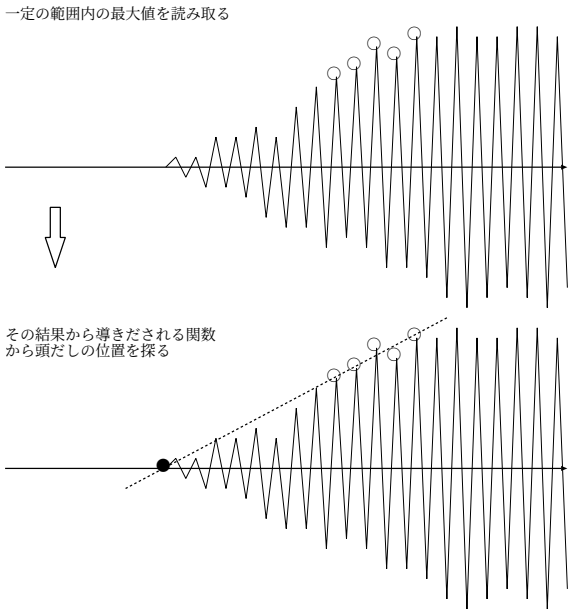

Fig. 7 特定範囲の値から関数を導き出す

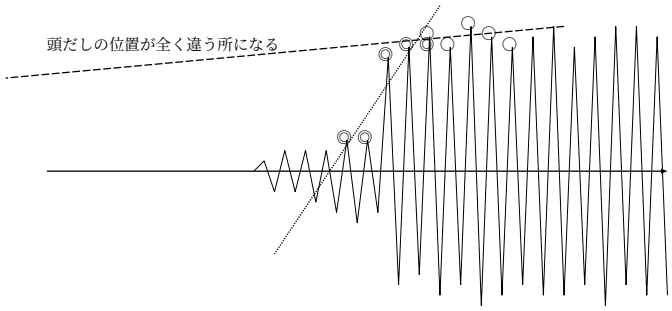

Fig. 8 頭だしの位置が全く違う

置を探り易くなるのではないかと考えられる。

音声データを聞いて頭だしを行う作業の流れを Fig.9 にあらわす。まず、音声データの出力開始の位置を指定して音声を流す。その結果、目的の音節の音声として聞こえないなら再び出力開始の位置を変えて聞いてみる。その音節の音声として聞こえた場合でも、まだ無音部分が含まれているおそれがある場合は、音声データの出力開始の位置を変えて音声を流してみるという作業を自分が納得するまで繰り返す。

この方法は音声を聞いて頭だしの値を決定するのはユーザ次第である。そのためユーザによっては音声の頭の位置にこだわりすぎて、音声の頭の位置を決定できずに何度も試行錯誤を繰り返してしまう場合や、前に流した音声の結果と聞き比べるために、またその位置を指定して音声を流すという二度手間をしてしまう場合など、1 音節のデータの切り出

しにかなりの時間をかけてしまうおそれがある。

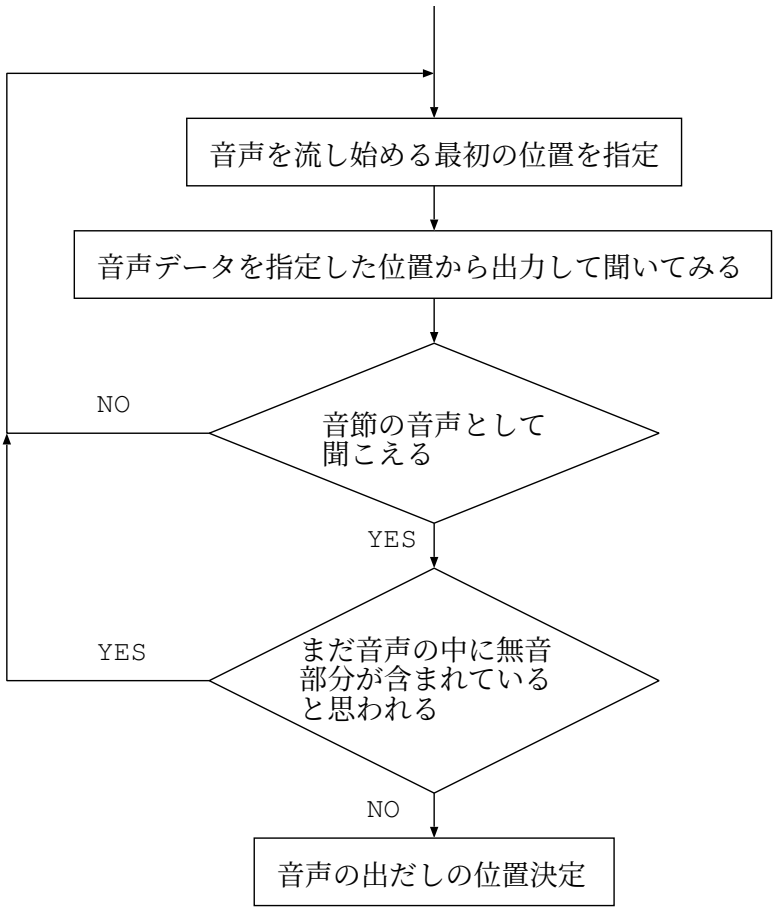

Fig. 9 音声を聞きながら出だしの位置探索

4.2 音節の頭だしにおけるデータ変換

前に述べたように、普通に音声を流しても音声部分と無音部分を聞き分けることは難しい。そこで、音声データの無音部分を聞き分ける方法として、音声をゆっくり流すことで無音部分を聞き分けることができるのではないかと考えた。

音声をゆっくり流す方法は、音声再生機能をもつソフトによるものではなく、音声データ量を増やして音声を流す時間を延ばすという方法で行うことで解決できる。

4.2.1 音声データに関して

音声データファイルの構造は、音の記録設定の情報 (ヘッダ情報) と音声が記録されているサウンドデータ (音声データ) 部分から成り立っている。

ヘッダ情報には主に以下のことが書かれている。

1. 音の録音形式 (AU 形式・WAV 形式)
2. サンプリング周波数 (1 秒あたりに記録されるデジタル信号の数)
3. 量子化ビット数 (1 つのデジタル信号値を何ビットであらわすか)
4. モノラル・ステレオの識別
5. 音声データ部分のファイルのサイズ
6. 符号化形式 (音のデータをデジタル化する形式)

これらの情報をもとにして音声データを加工・処理することができるようになっている。

音声の流れる速さをゆっくりにしたいなら、音声データの波形の形を伸ばすことで音声の流れる時間が延びて音声をゆっくり聞くことができる。

音声の流れる時間を延ばす方法は、音声データ量を増やすことで解決する。この方法を用いて出力した音声データは低い音程になって聞こえてしまうが、目的は音声データの音声の出だしの明確化であるためそれは問題としない。音声データ量を増やすには、音声データ変換ツールを使う方法が挙げられる。しかし、音声データ変換ツールは音の編集を目的としているものである。そのため、データ量を変えるだけという目的で使うのには向いていない。さらに変換ツールを持っていないユーザにはこの方法が使えないという欠点がある。

音声データ変換ツールを用いずに音声データ量を増やす方法についても考える必要がある。そして、その二つの方法を用いた音声データ量増加についての結果を比較し考察する。

4.2.2 音声データ変換ツールを利用する方法

Lance Norskog 氏によって作成された音声データ変換ツール SoX(Sound eXchange) は

1. 音声データ形式の変換
2. サンプリング周波数の変換
3. データサイズ (8bit $\leftrightarrow$ 16bit) の変換
4. モノラル・ステレオの変換

といった機能を備えている。音声データ量の増減に関してはサンプリング周波数の変換を利用することで解決する。

データ量を増やす過程を Fig.10 にグラフであらわしてみた。ここに、サンプリング値 8000Hz の音声データがあるとする。この音声データは 1/8000 ごとにデータが記録されている。ここで、音声データ変換ツールを使い、音声データのサンプリング周波数を 2 倍にする。すると、そのデータは 1/16000 秒ごとのデータに変換される。その際、変換前のデータにはないデータとデータの間の値は音声データ変換ツールが補ってくれる。ここで用いた音声データ変換ツールには、サンプリング周波数を変える方法が幾つか用意されていて、今回使用したのは最も変換処理が簡単に行われる方法を利用した。サンプリング周波数の変換でデータ量は増えた。しかし、これは 1 秒間に出力する音声データの数が増えただけで、実際に音声を流して聞いても音声は変換前と違いはない。そこで、このデータのヘッダ情報に記録されているサンプリング周波数の値を 8000Hz に書き換えと、この音声データは 1 秒間に 8000 個 のデータ量しか出力しなくなり、そのため音声の流れる時間は長くなる。

4.2.3 1 次補間を用いる方法

音声データ編集ツールを用いる方法では編集ツールによるサンプリング周波数変換を行い、変換した音声データをまたサンプリング周波数値を変えるという 2 度の手間を使わなければならない。そこで 1 次補間を用いて、1 度の作業で音声データ量を増やす方法について考察する。

補間とは、離散的なデータを連続的なものとみなした場合、データとデータの間にある値を導きだす方法であり、1 次補間はデータ同士を直線で結び、その直線上の値を導きだす補間方法である。1 次補間ならば求め方が簡単であるため、データ処理も速く短時間で済ませることができが、その結果音声の質が雑なものになってしまっているおそれがある。SoX を用いた音声データと比べて音声の質があまりに酷いものであった場合はこの方法を使うことができない。

Fig.11 に、周波数の違いによるデータの位置関係をあらわした。1 次補間によって求める値 $Y1$ の計算式は、このグラフをみてわかるように、

$$Y1 = X1 + \frac{s1 + h2 - t1}{h1}(X2 - X1)$$

であらわすことができる。

この式を用いて音声データ量を増やしていく流れを Fig.12 を用いて説明する。まず、増加後の音声データの容量を確保する。次に確保した容量が満たされるまで 1 次補間を行う必要があるかを探索し、必要ならば 1 次補間を用いて、データを出力していく。という作業を繰り返していく。

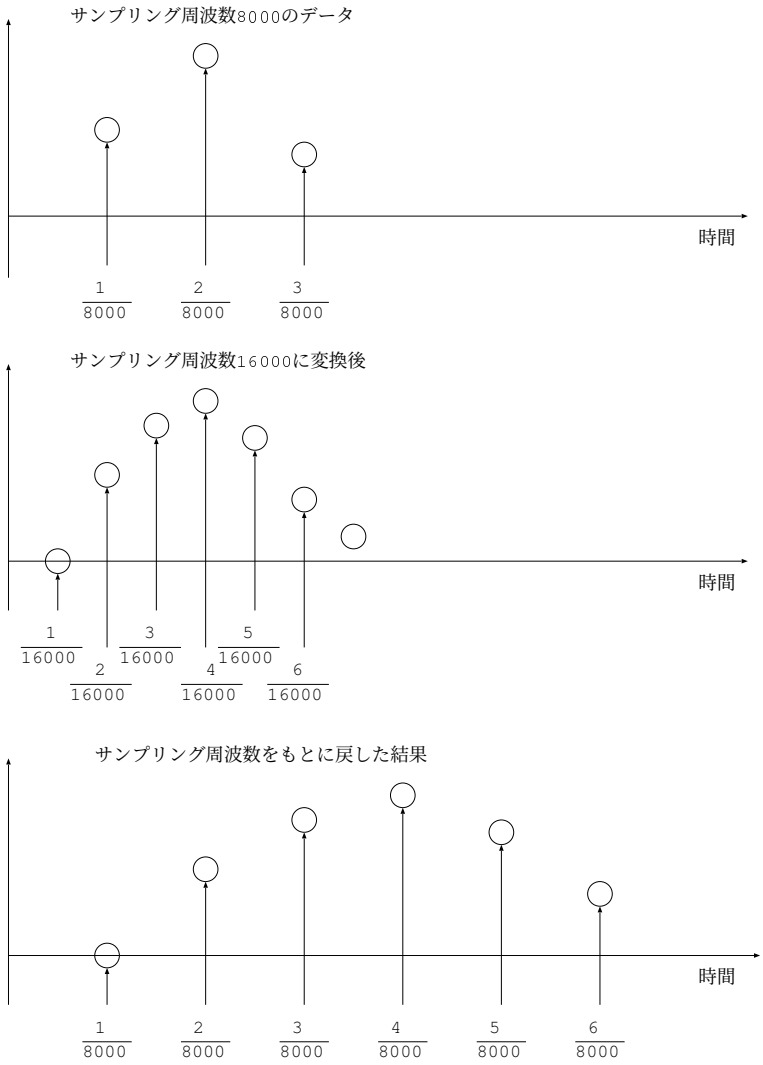

Fig. 10 サンプリング周波数の変換によるデータ量の変化

4.2.4 変換ツールと 1 次補間の比較

音声データ量を増やすふたつの方法を行った結果を比較する。ある音節の音声データ量を 1.2 倍にして二つの違いを調べてみた。

音声データ編集ツールを用いてデータ量を増やした結果を Fig.13 、1 次補間を用いてデータ量を増やした結果を Fig.14 にあらわす。この二つの違いを調べるために両方データの差をとり、その差を Fig.15 にあらわした。この結果を見てみると、二つの方法の差はとても小さくあらわれている。このことから音声の聞きながら頭出しを行う際において、音声をゆっくり聞くための音声データ量の増加方法はどちらを使っても違いが聞き取れない程度のものであることがわかった。これによって音声データ量を増やす方法は、1 次補間による方法で行えることがわかった。

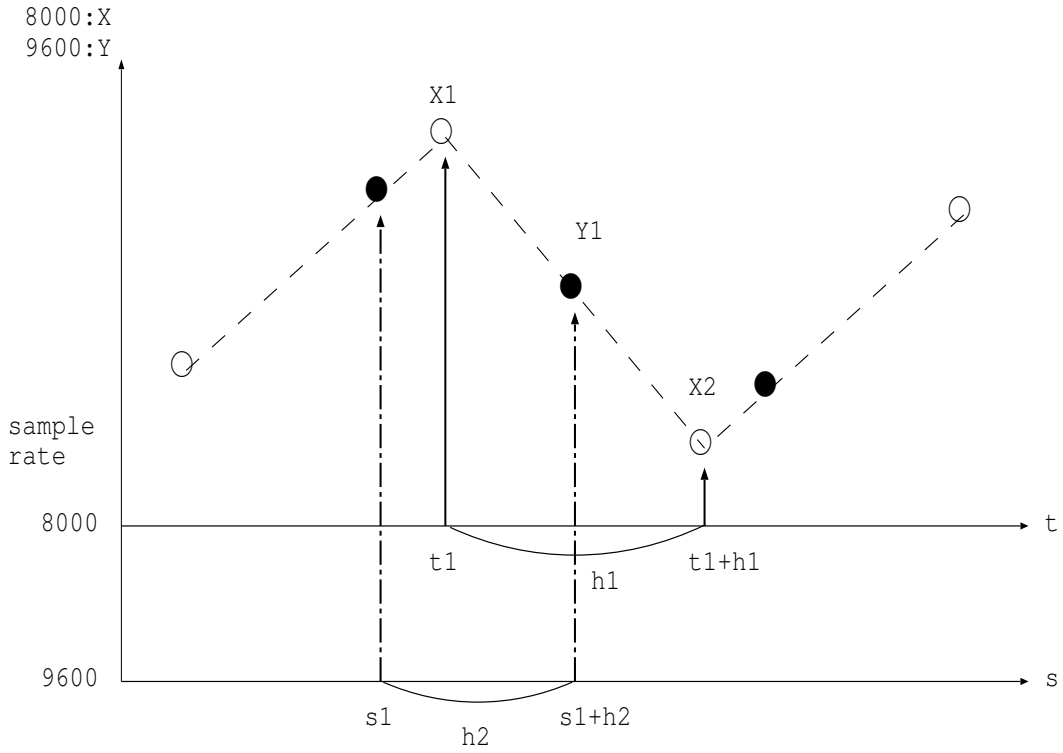

Fig. 11 周波数の違いによるデータの位置関係

5 音声を聞きながら頭だしを行う

音声データの出力開始の位置を指定して聞いていく方法について、何度も位置を指定するという作業では効率が悪いように感じられる。音声を聞いて判断する方法を、もっと簡単な作業で行えないか考えてみる。そして、これらの方法が実際に使えるか実験を行い、その結果について考察してみる。

5.1 無音部分を聞いていく

今まで考えていた方法は音声部分を聞いて音声の頭の位置を確認するものであった。この方法では頭の位置を決定した音声データに無音部分が本当に入っていないかを確認できなかった。そこで、音声の部分を聞いて頭の位置を探すのではなく、聞いていく部分を無音の部分にして頭だしを行う方法を考えてみる。

無音部分を聞いていく方法は Fig.16 に示したように、録音した音声データの最初の位置から特定の位置までを音声で出力して、その中に音が含まれていないかを確認して音が入っていなければ音声を流す範囲を広げて、また音声を出力して音の確認をしていくという方法である。この方法では、最初は聞いても無音だけしかない出力されないが、波形のあらわれている部分まで音を流す範囲が広がったとき、その位置で音がでてくる。そこを

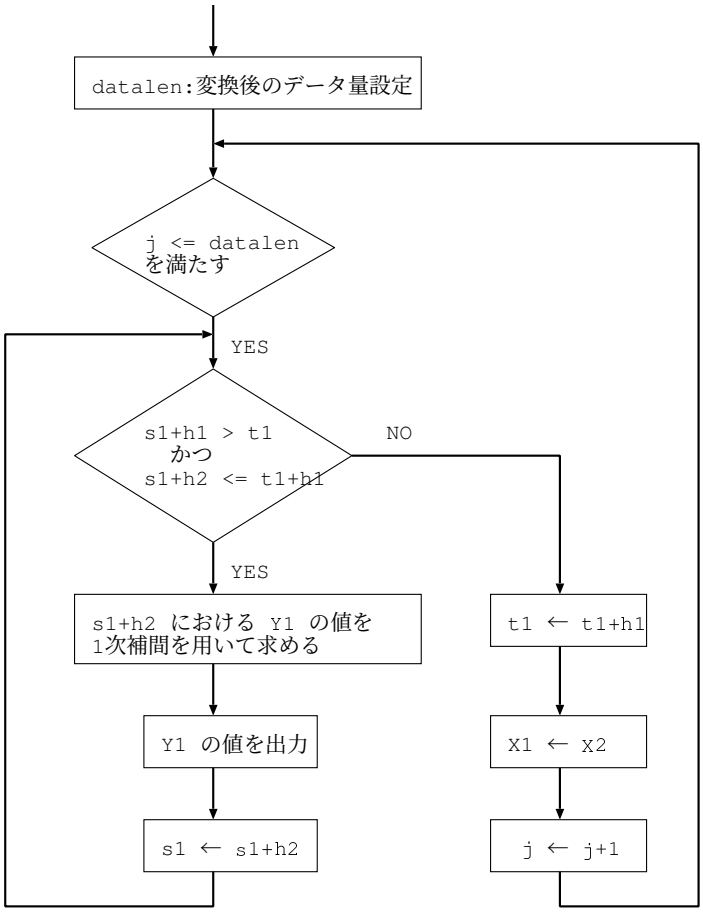

Fig. 12 1 次補間を行う流れ

音声の頭の位置と決めることができる。

無音部分を聞いていく過程を Fig.17 に示した。まず、音声データの最初の位置から指定された終了位置までを音声として出力し、出力結果に音が入っていたかの確認をする。音が入っていたのならば終了位置を出力して終る。入っていないならば終了位置を増加値の分だけ加え、音声データの最初から終了位置までを出力する。これらの作業を繰り返すという流れで行なう。

無音部分を聞いていく方法は、音声データを出力した際に音が入っているとした場合は出力されるデータの終りの位置でほんの一瞬だけ出力されるため、聞き逃してしまうおそれがある。今回の実験において、移動するデータの範囲は、1 回ごとに 10 データブロック (1 データブロック = 1/(サンプリング周波数) 秒分のデータ) とする。

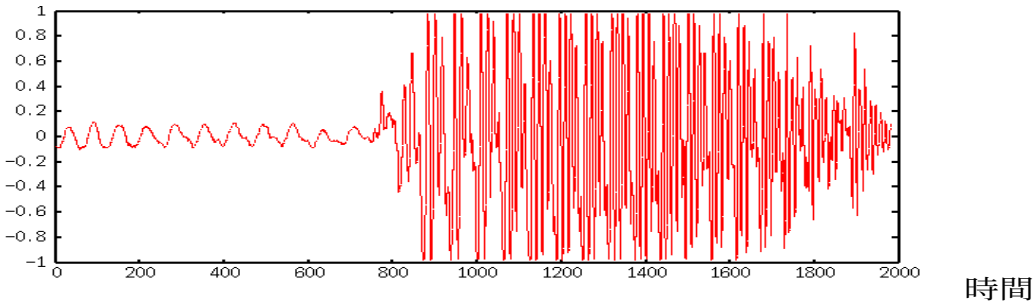

Fig. 13 音声データ編集ツールを用いてデータ量を増加させたもの

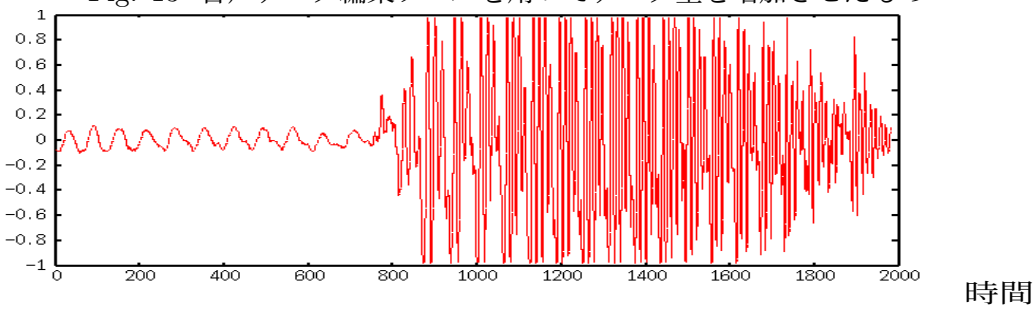

Fig. 14 1 次補間を用いてデータ量を増加させたもの

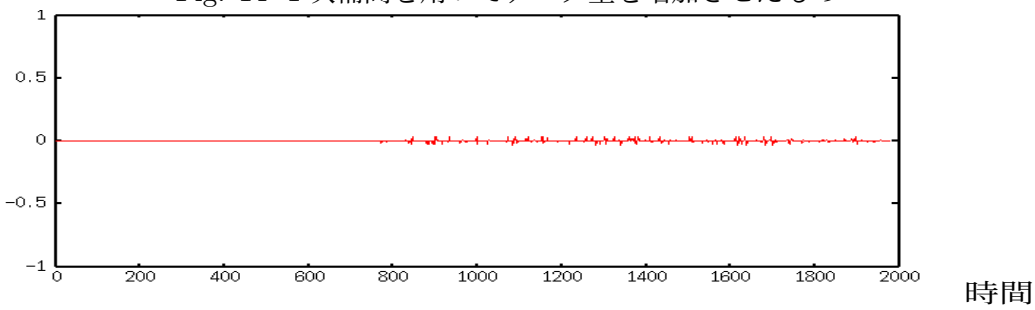

Fig. 15 上の二つのデータの差

5.2 音声部分を聞いていく

無音部分を聞いていく方法だけではその方法が本当に有効なのかがわからない。そのため、比較するために音声部分を聞いていく方法を考えてみる。無音部分を聞いていく方法は音声データの始めの位置から範囲を広げて聞いていく方法であった。音声部分を聞いていく方法は音声データの後ろから音声部分を聞いていき、音声に必要な部分を探っていくもので、Fig.18 にその方法を示した。音声データの後半に音声の出力開始位置を指定し、その位置から音声データの終りまで出力して聞いてみる、その出力結果が必要な音声として聞こえなければ、必要な音声が聞こえるまで出力開始位置をデータの前の方へ移動しながら聞いていくというものである。

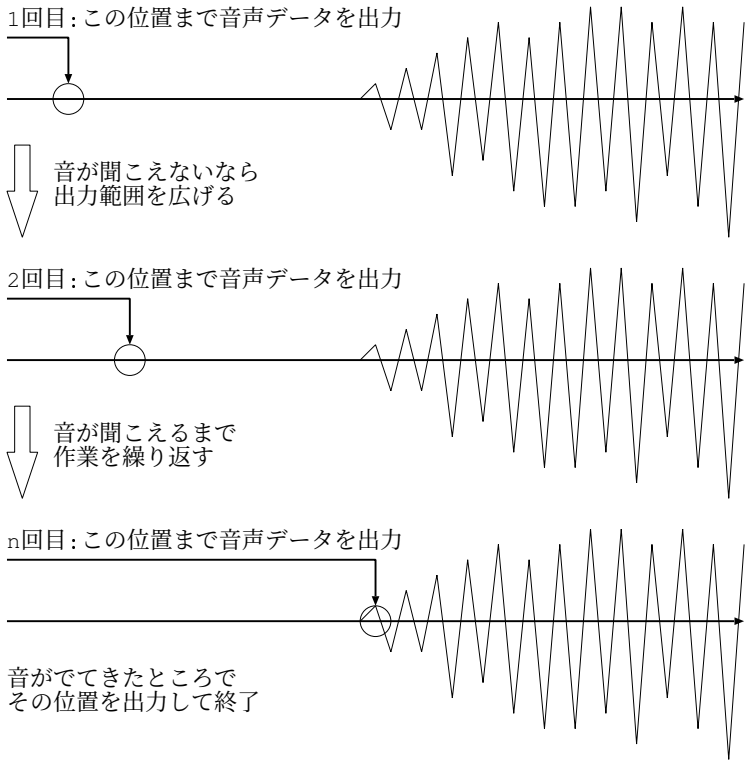

Fig. 16 無音部分を聞いていく過程

Fig.19 に音声部分を聞いていく過程を示す。まず、頭だしを行う音声データのファイルサイズを確認する。そして、最初の音声出力開始位置と移動する間隔を指定する。後は指定したデータの範囲を出力して、必要な音声が聞こえるまで聞いていく。今回、音声部分を聞いていく方法も移動するデータの範囲は、1 回ごとに 10 データブロックとする。そして、最初に指定する出力開始位置をデータサイズの 1/3 の位置とした。

音声部分による方法は音節の音声が聞こえた位置で決定するため、この方法は本当に必要な位置を探しだせるはずである。

5.3 選択肢による音声の頭だし作業

前に述べた二つの方法では、音声の頭の位置まで少しずつ近付いていくため、音声を繰り返し聞いていく作業は結構な回数になってしまう。そのため、1 音節あたりの作業回数を少なくしないと、全ての音声データの頭だしを短時間で行うことができなくなる。

頭だしの作業回数をもっと減らすための方法について考えてみる。頭だしの作業回数が多くなるのは、少しずつ位置をずらして聞いていくからであり、聞いていく範囲を少しずつではなく、かなり大きく広げれば作業回数を減らせるだろう。しかしそれでは音声に必要な部分を削除してしまうおそれがある。

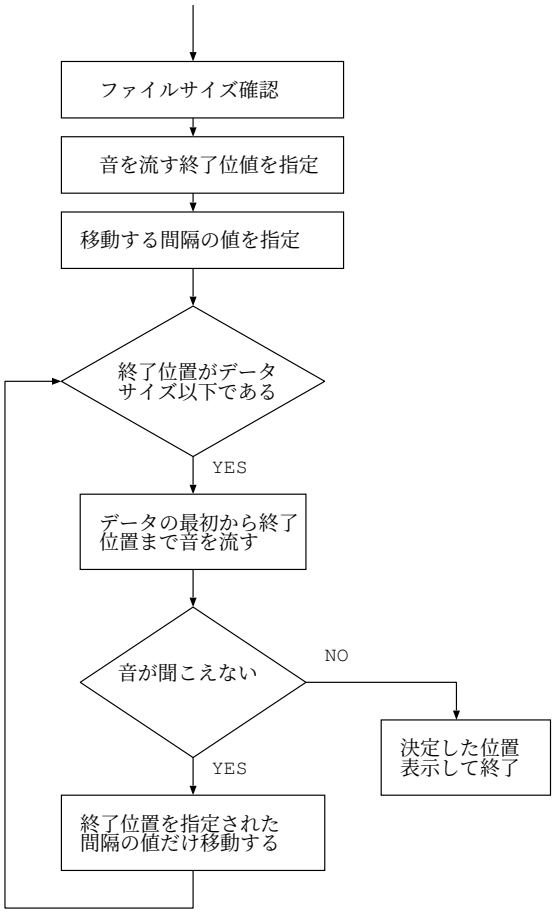

Fig. 17 無音部分のチャート

音声を流す位置を 1 度に複数用意して聞いていき、その中から選んでいけば作業回数を減らすことができるはずである。

選択方法による音声部分の頭だしについても、無音と有音による方法で考えてみる。今回、選択方法による音声の頭だし作業を実験する際において、1 回ごとの作業に用いる選択肢の数は 3 つとする。

5.3.1 選択肢で無音部分を聞いていく

選択方法による無音部分による頭だしの流れを Fig.20 に示した。まず、 A、B、C の音声の出力終了位置を指定した 3 つの選択肢を用意する。次に音声データの最初の位置から各選択肢の指定位置までの音声データを出力して音を聞いていくが、この場合の聞き分けの基準は、無音部分を聞いていく方法と同じで、音の有無の確認である。各選択肢で出力された音声データを聞いた結果、音が入っていない選択肢を選んでいく。

この方法による選択肢の過程を Fig.21 に示した。まず、音声データ全体のサイズを調

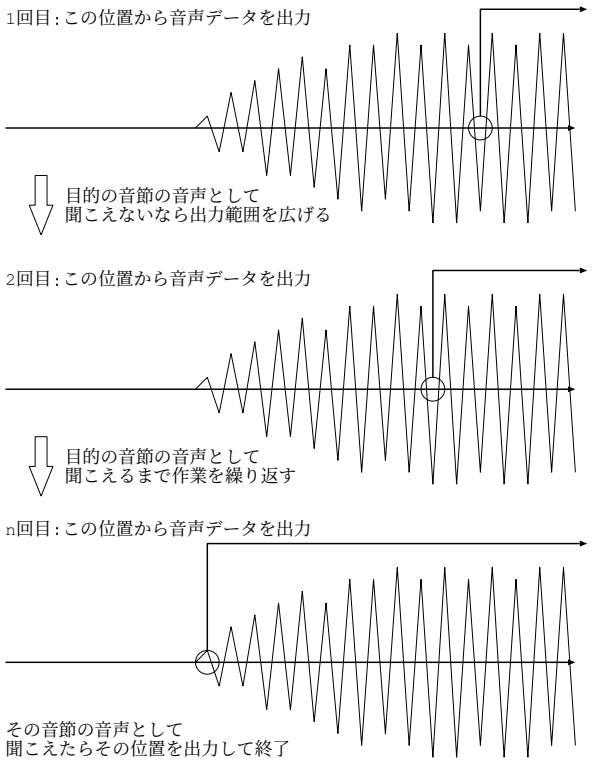

Fig. 18 音声部分を聞いていく過程

べて、そのサイズから 3 等分した値を各選択肢の位置と指定して順に音声データを出力して聞き比べる。その結果、無音で出力された選択肢を選ぶ。選んだ選択肢の値を 3 等分してその各値を新たな選択肢の値に割り当てる。D は 3 等分する際のデータのサイズを知るために必要なものである。このようにして選択作業を繰り返し行っていく。ここで注意するのは、複数の選択肢が無音の出力をした場合の選択であるが、目的は音声の部分つまり音声の出だしの位置であるから、選択する優先順位は音声側に近い C ＞ B ＞ A の順である。例えば、A と B が無音であった場合は 音声部分に近い B を選ぶ。最終的にどの音も聞き分けるのが難しくなったところを頭の位置と決定する。

5.3.2 選択肢で音声部分を聞いていく

選択肢で音声部分を聞いていく方法を Fig.22 にあらわした。まず、音声データの開始位置を指定した選択肢 A、B、C を用意する。次に選択肢の位置から音声データの終りまでを順に出力して聞いていく。選ぶ基準は作業をする音節の音声として聞こえたかである。

この方法の過程を Fig.23 にしめした。頭だしを行う音声データのサイズを確認した後にそのサイズを 3 等分した位置をその音声データの出力開始の位置の選択肢とする。後は各音節の音声として聞こえる選択肢を選んでいく。ここでも選択する際の優先順位は C

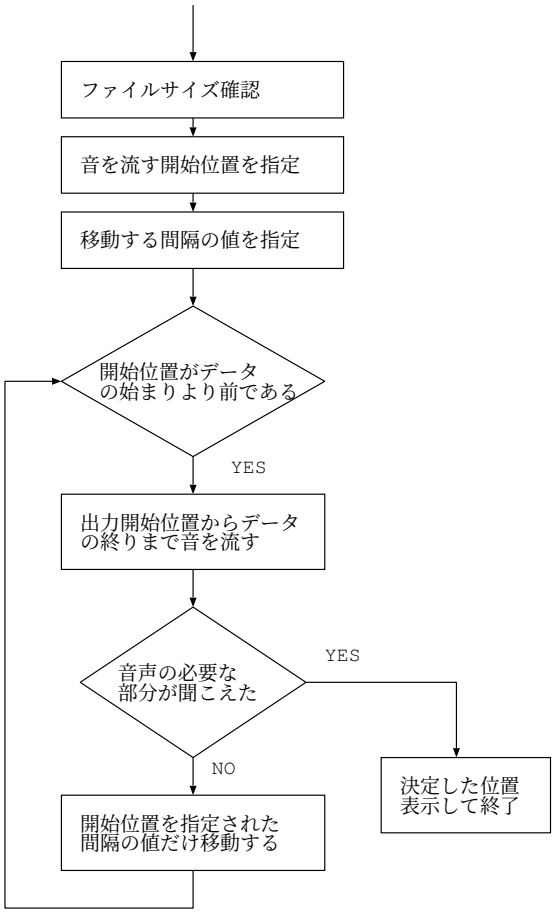

Fig. 19 音声部分のチャート

＞ B ＞ A とする。

最終的にどの音も聞き分けるのが難しくなったところで頭の位置を決定する。

5.4 頭だしの実験と結果

音声データの頭だしの 4 つの方法を調べるために、研究生の学生 5 人に協力してもらい、実際に音声の頭だしの作業を行ってみた。

いくつかの音声データに対して、先に述べた 4 つの方法を行ってもらい、音声の頭の位置と決定するまでに行った作業回数を記録して、その結果を比べてみてどう違うかについて検討する。そして、それぞれの位置と回数の平均値を求め、その差を調べてみる。今回、頭だしの作業に用いた音節の音声データはサンプリング周波数 8000Hz の au 形式の音声データで、調べる音節は 「お」「か」「ぴ」「べ」「ひゅ」 の 5 つとする。これらはまだ何の編集もしていないものである。

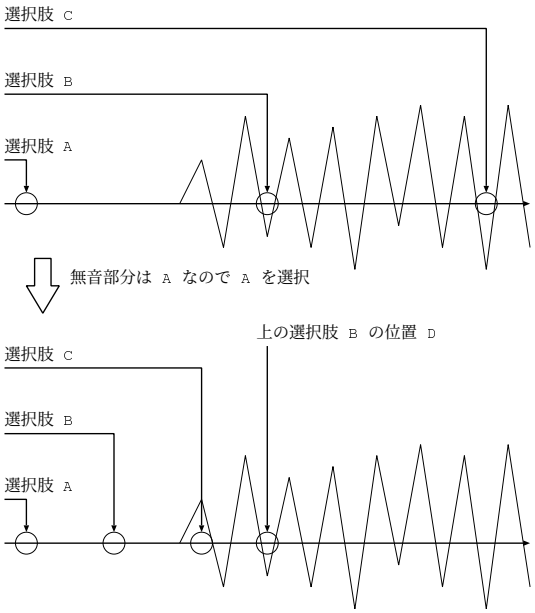

Fig. 20 無音部分を選択していく過程

実験の結果を Table 1 にあらわした。この表には各方法ごとによる音節の位置とその作業回数があらわされている。そして、この表の結果が音声データのどの位置なのかを見るために波形のグラフを Fig.24 にあらわした。

まず、音声データの位置について調べてみる。個人ごとの無音と選択無音の結果を比べてみると、位置の大きな差はあまり見られない。音声と選択音声についても同じことがいえる。しかし、無音・選択無音と音声・選択音声で比べると、音声の位置が大きく違っていることがわかる。この違いがあらわれる理由は Fig.25 に示すように、無音部分による方法は音が有るか無いかの確認であり、音は波形があらわれたところで出るので、波形の出始めで頭の位置を決定することになるからである。

音声部分による方法は作業を行う人が音節の音が聞こえるまで行うため、音声に必要な部分で決定しているためである。

無音と音声の頭だしの結果の違いの差の部分は雑音とみなすことができる。そのため無音で頭だしをした場合は、その音が雑音の音でも、そこで頭の位置が決定してしまうという欠点がある

音節ごとの結果をくらべてみると、選択方法による結果が各人によって大きく異なる値があらわれている。これは Fig.26 にあらわしたように選択した結果次第で、次の選択肢の値が大きく変わってしまうためである。

次に、位置の決定までの作業回数について調べてみる。まず、無音と無音選択を比べてみると、回数が明らかに違っていることがわかる。音声と選択音声で比較しても、その回数の差は大きく違っていることがわかる。大きく違ってしまうのは無音と音声の場合、音声データの出力範囲を少しずつ広げていくという方法であるため、音声を出力して聞き分

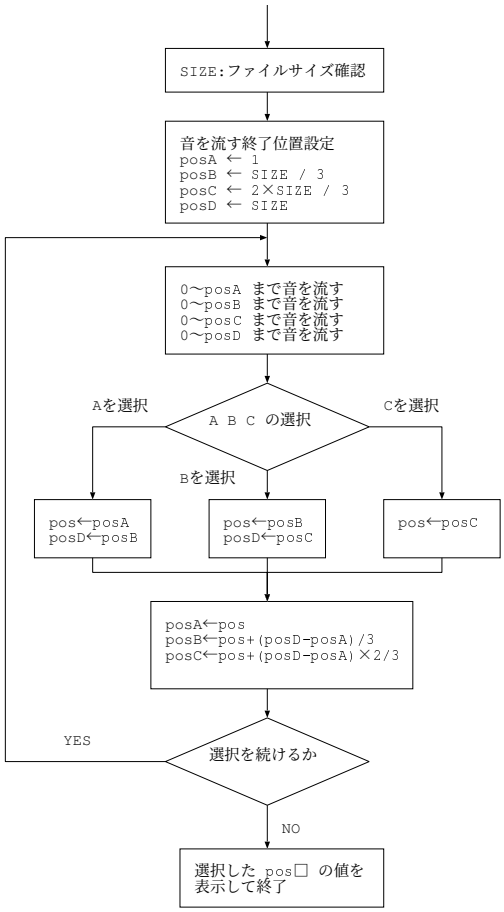

Fig. 21 無音部分を選択していくチャート

けるという作業の回数がどうしても増えてしまうからである。選択肢による方法は選択していく度に、出力する位置の範囲がデータサイズの 1/3 ずつになっていく。そのため、データサイズが 8000 個あった場合は 8 回で選択肢がどれも同じ値になってしまう。そのため選択肢による方法は、無音と音声よりも少ない回数で頭だしが行える。

無音と音声で比べた場合、どちらも音節によって回数が大きく異なっているのがわかる。音節によって非常に少ない回数で頭だし作業が行えるが、全体でみると作業回数が大きくなる音節が多いため、あまり効率がいいとはいえない。最後に各音節の被験者の平均を求めた結果について考察する。4 つの方法による各音節の位置の平均値と被験者が求めた値を比べてみると、選択方法に各人の結果との違いが大きくあらわれているのがわかる。選択次第でその差が大きくでてしまうため、この平均した結果はあまりあてにならない。しかし、回数については各人の結果と比べてみると、だいたいは平均値とかわらないことから、選択方式による方法ならば 1 音節あたりの頭だし作業を少ない回数で行えることがわかる。

実際に音声の頭だしの作業を行ってみた結果、それぞれの方法による利点と欠点を考え

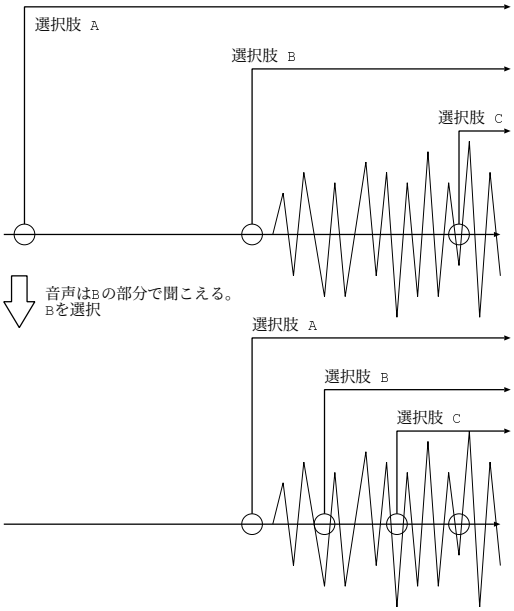

Fig. 22 音声部分を選択していく過程

てみる。

1. 音声部分を聞いていく方法は、出力された音声が必要な部分を含んでいるかを聞き分ける必要があるため、聞き分けにかなり集中する必要がある。しかも、作業回数も多いという欠点を持っているが、この方法が最も音声の頭を正確に見つけられる利点がある。

2. 無音部分を聞いていく方法は音がでてくるまで聞いていくという作業は、音があるかないかの確認であるため、頭だし作業があまり疲れない面を持っているが、作業回数が多く、実際の音声部分の位置とのずれが大きい可能性を持っている。

3. 選択方式により無音部分を聞いていく方法は、音の有無の確認を行うことと選択肢から選んでいくことだけなので、最も短時間で行える方法だとおもわれるが、決定した音声の頭の位置と実際の音声の頭の位置が、大きく違っている可能性も持っている。

4. 選択方式により音声部分を聞いていく方法は、音声の聞き分けと、選択肢によるものなので、少ない作業回数で音声の頭だしが行えるが、選択方式による方法は結果の値に大きな差があるため、正確に頭の位置を求めたい場合は、それほど有効であるとはいえない。

以上のことから、音声データの音声の頭だし作業における最も有効な方法は、頭の位置を正確に求めたいならば、音声部分を聞いていく方法であるが、これは時間を大量に費して

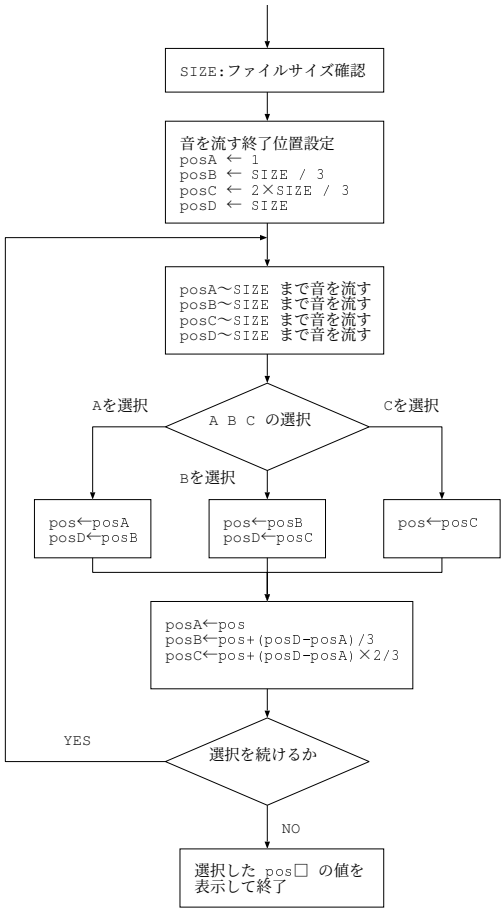

Fig. 23 音声部分を選択していくチャート

しまい、また集中力を要するため、1 度に多くの音節の頭だしを行うのは困難である。音声データの頭だし作業にあまり時間をかけたくないのであれば、選択方式の無音で聞き分けていく方法が最も有効である。

6 まとめ

今回、研究室で開発されている文書読み上げソフトの改良のために、音声データの編集について考察した。読み上げソフトの仕組みから考えて、音声データの編集作業の内容を、録音した音声データから音声以外の不要部分を取り除くというものにした。そのために、音声の頭の位置を知る方法について考察した。まず、音声データから音声部分の頭の位置の探索を自動的に行う方法について考えたが、この方法ではうまくいきそうにないため、手動による方法について調べてみた。頭だしの方法は CUI 環境で行うことを目的としているため、頭だしの方法は音を流して聞き比べるという方法で考えていくことにした。そして、それらの方法を実際に行い、その結果を比較・検討した。検討した結果、音

声の頭だし作業は作業を行う人の状況に合った方法で行うことにするのがよいというものになった。

しかし、音声の頭だし作業を人の手による作業で行う以上、全ての音節データの頭だし作業にはかなりの時間がかかってしまうという問題は解決されていない。自動的に音声データの頭だしが行われることが望ましいが、それについての追求は今後の課題として残される。

今回行なった音声データの編集は音声の出だしの探索であった。しかし、音声データの編集は他にも音声データの音程、音量の統一や音程を変えずに音声データの長さを調節することなど、まだ残された課題はある。

参考文献

[1] 佐藤 健美: “UNIX における日本語文書読み上げソフトの開発について”, 新潟工科大学卒業論文, (2000)

[2] 山崎信英 北川博雄: “特集 音声合成”, 月刊 C マガジン, 3 月号, pp24–58, (ソフトバンク社,1994)

頭だし結果									
音節		無音		選択無音		音声		選択音声	
		位置	回数	位置	回数	位置	回数	位置	回数
お	一人目	164	16	141	7	1196	0	1063	3
	二人目	154	15	180	4	1186	1	1993	2
	三人目	144	14	140	6	1196	0	1328	3
	四人目	174	17	136	3	1196	0	1461	3
	五人目	164	16	204	7	1156	4	1993	3
か	一人目	554	55	539	8	857	78	889	6
	二人目	214	21	1031	4	877	75	1071	5
	三人目	544	54	548	3	877	76	849	4
	四人目	554	55	910	3	897	74	728	5
	五人目	544	54	539	10	837	80	910	3
ぴ	一人目	94	9	41	4	252	77	116	3
	二人目	74	7	230	3	912	11	1703	2
	三人目	84	8	66	5	132	89	116	3
	四人目	64	6	41	4	182	84	40	4
	五人目	54	5	79	4	672	35	229	3
べ	一人目	84	8	75	10	204	96	175	5
	二人目	84	8	218	4	584	58	2919	3
	三人目	84	8	75	9	164	100	46	4
	四人目	84	8	218	4	84	108	3	5
	五人目	84	8	89	4	124	104	89	4
ひゅ	一人目	204	20	4	3	964	51	656	3
	二人目	124	12	4	3	914	56	3	1
	三人目	94	9	112	4	1174	30	1146	4
	四人目	94	9	2456	4	674	80	3	4
	五人目	154	15	475	5	844	63	1801	3
平均値	お	160	15.6	160	5.4	1186	1.0	1567	2.6
	か	482	47.8	713	5.6	869	76.6	889	4.6
	ぴ	74	7.0	91	4.8	430	59.2	440	3.0
	べ	84	8.0	135	6.2	232	93.2	646	4.2
	ひゅ	134	13.0	610	3.8	914	56.0	721	3.0

Table 1 頭だし方法の位置と回数の結果

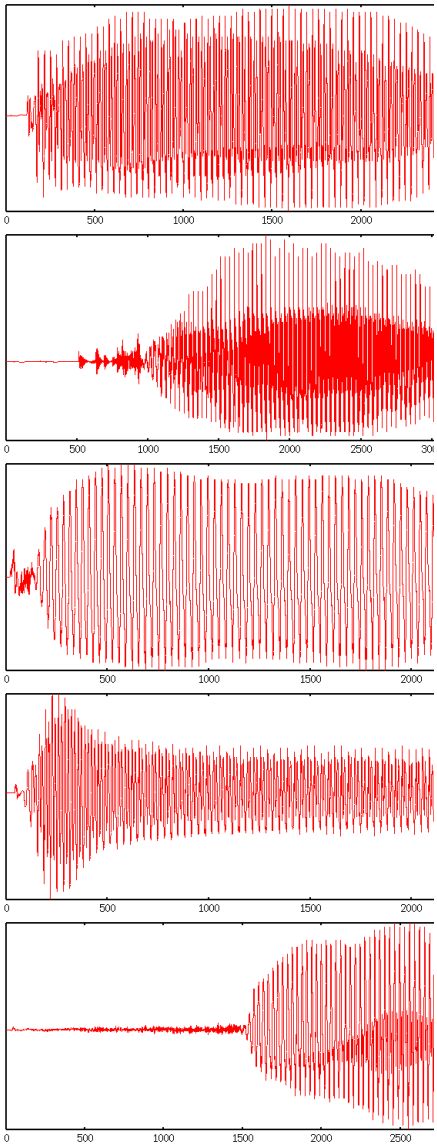

Fig. 24 上から順に「お」「か」「ぴ」「べ」「ひゅ」の音声波形

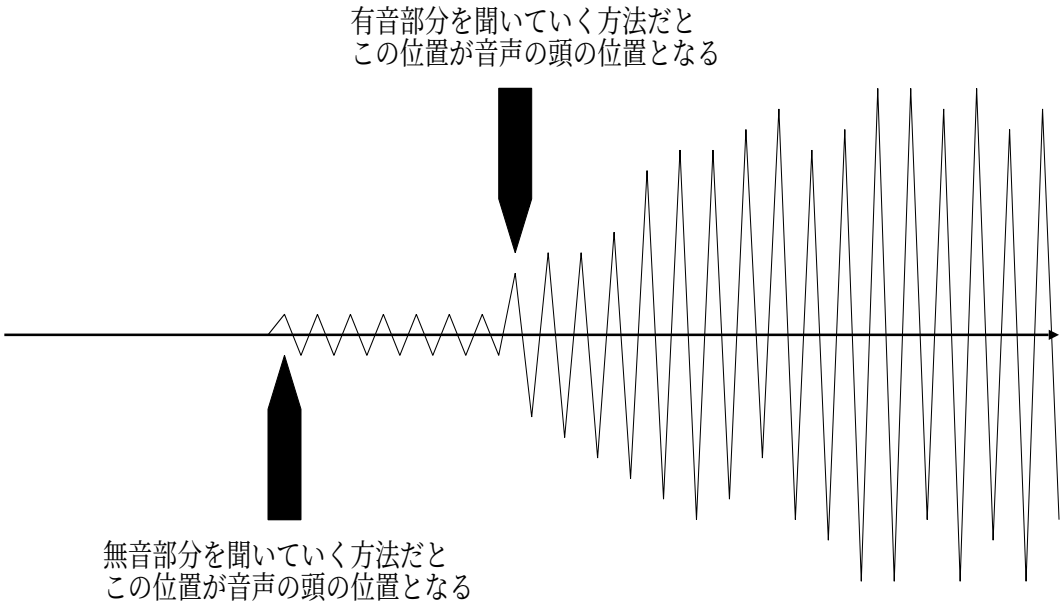

Fig. 25 音声の頭の決定位置の違い

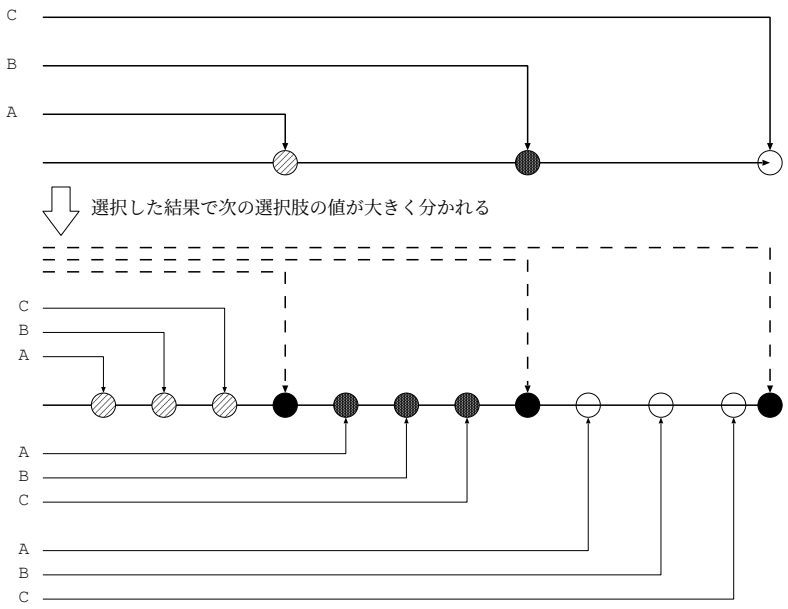

Fig. 26 選択した結果による違い