Suffix Array を用いた高速 STD のための検索閾値の調整手法

三浦成一[†1]　桂田浩一[†1]　入部百合絵[†2]　新田恒雄[†1†3]

本論文では，Suffix Array を用いた高速音声検索語検出における，検索閾値の調整手法を提案する．筆者らが提案してきた検索手法では，検索キーワードの長さに比例して検索時間が指数的に増加するという問題に対応するために，検索キーワードをいくつかに分割して検索する方法を採っている．このとき，分割せずに検索した場合と同様の検索結果が得られるよう分割キーワードに与える検索閾値の調整を行っている．これは，閾値を調整しないと検索性能が悪化することがあるためである．筆者らはこれまでは分割キーワードに付与する検索閾値を等しく与えていたが，その場合分割キーワードによっては膨大な数の検索結果が得られることがあり，検証に多くの時間がかかるという問題があった．そこで本研究では，分割キーワードの検索で得られる検索結果の数を推測して，これが各分割キーワードで均等になるように閾値を与えることで検索結果の総数を減らし，検索速度向上の実現を目指す．また，検索結果の数は検索キーワードごとに異なるため，検索キーワードを構成する音素列から検索結果の数を推測することで，推測精度を向上させる手法についても検討する．評価実験の結果，検索速度に関しては従来手法と比べて提案手法では平均して約 12%の速度向上がみられた．さらに検索キーワードを構成する音素列を考慮した検索結果数の推測を行うことで，検索速度は平均して約 17%向上した．

Adjustment of search threshold value assigned to sub-keywords in the fast STD using a suffix array

SEIICHI MIURA[†1]　KOUICHI KATSURADA[†1]
YURIE IRIBE[†2]　TSUNEO NITTA[†1†3]

This paper provides how to control the threshold values assigned to the search keywords in the fast spoken term detection using a suffix array. In our method, a search keyword is divided into some sub-keywords and they are searched for instead of the original keyword to avoid the problem that search time increases exponentially with the length of the search keyword. In the keyword division process, we give the adequate threshold values to the sub-keywords to get the search results same as the original one. For this purpose, in the previous study, we have assigned a single threshold value to each sub-keyword. However, there are some cases that a huge number of search results are obtained in some sub-keyword searches, and their verification processes take a long time. In this study, we reduce the search time by reducing the number of search results that are obtained by searching for sub-keywords. For this purpose, we estimate the number of search results in the sub-keyword search and assign proper threshold values to sub-keywords so as to equalize the number of search results. In addition, we propose a technique to improve the performance of estimation by taking account of the phoneme strings contained in the sub-keyword. An experimental result shows that improves search speed by around 12% with compared to the previous method. In addition, by controlling the threshold values according to the phoneme sequences included in the sub-keyword, we could improve search speed by around 17%with compared to the previous method.

1. はじめに

近年，コールセンターの応対記録や家庭用 HDD における録画映像，web 上での音声データや動画像データなどが増えつつあり，大量のデータが蓄積されている．こうしたデータは web サイト同様に検索機能なしには利用しづらくなってきており，音声検索に対する需要が大きくなってきている．これらのデータを有効活用する一つの方法が音声検索語検出である．音声検索語検出(STD: Spoken Term Detection)とは与えられたキーワードの出現箇所を音声データ内から検索する技術を指す．音声検索語検出に関する研究は近年盛んに行われており[1]，2006 年に NIST の主催でベンチマークテストが行われた[2]のを始めとして，日本においても NTCIR-9 のタスクに組み込まれるなど[3], 共通のタスクによる客観的な評価が行われ始めている．特にここ数年の間に，数十～数千時間の音声データを対象に数ミリ秒～数秒で結果を出力する非常に高速な検索法が提案されており，音声検索語検出における重要な課題の一つとなっている[4,5,6]．筆者らも Suffix Array に DP マッチングを適用することによって，非常に大規模な音声ドキュメントから高速にキーワードを検出する手法を提案している[7,8]. しかし, Suffix Array だけでは検索キーワードと完全に一致する部分しか検索できず，認識誤りに対応できない．そこで, 類似区間を検出できる DP マッチングを Suffix Array 上で行う，テキスト曖昧検索のアルゴリズム[9]を導入している．このアルゴリズムは，Suffix Array を木構造と見なし，木を辿りながら DP マッチングを行うものである．その際，キーワードとかけ離れた，見込みのないパスの枝を刈るため効率よく木を探索できる．また，この手法では，検索キーワードの分割，反復深化的探索などの技術を複合的に利

†1 豊橋技術科学大学
Toyohashi University of Technology
†2 愛知県立大学
Aichi Prefectural University
†3 早稲田大学
Waseda University

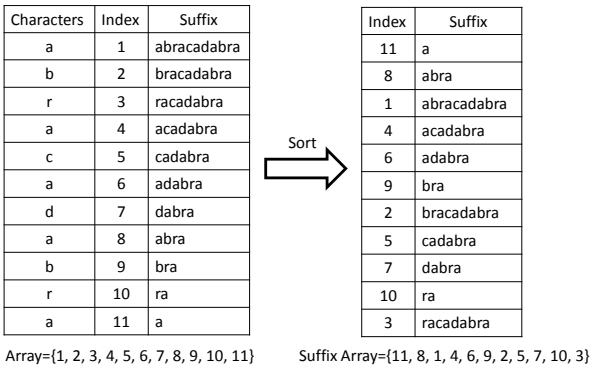

図 1　Suffix Array の構築方法

Figure 1　How to construct a suffix array.

	a	i	u	e	o	k	s	t	…
高舌性	-	+	+	-	-	+	-	-	
低舌性	+	-	-	-	-	-	-	-	
前方性	-	-	-	-	-	-	+	+	
後方性	+	-	+	-	+	+	-	-	
破裂性	-	-	-	-	-	+	-	+	
:									

図 2　弁別特徴テーブルの一部

Figure 2　A fragment of a distinctive phonetic feature table.

用することで，非常に高速な音声検索語検出を実現している．

上述の各技術のうち，検索キーワードの分割は，キーワードの長さの増加による探索領域の指数爆発を防ぐために導入したものである．テキスト曖昧検索のアルゴリズムでは，DP マッチングの実行中に枝刈りの閾値内のすべてのパスが保持されるため，閾値が大きいと探索空間及び処理時間が指数関数的に増加することが，山下らによって確認されている．閾値は検索キーワードの長さに比例して増加させる必要があるため，検索キーワード長に対して指数的に処理時間が増大する．そこで，この問題を解決するためにキーワードを分割し，分割キーワードを元のキーワードの代わりに検索する手法を導入している．この手法は Navarro らが文字列検索を対象に提案したキーワードの分割検索法[10,11]と同様の分割検索を音声検索に組み入れたものであり，検索キーワードを適切な長さに分割することによって，木構造で表された Suffix Array の探索領域が少なくなり，探索時間が短縮される．また，各分割キーワードの検索結果が得られた後に検証作業を行って元のキーワードの検索結果と同じ結果が得られるようにしている．しかし, Navarro らは文字列の検索を主目的としているため，DP マッチングの距離尺度が編集距離であることを前提としている．これに対して，筆者らの手法では音素間距離を実数値に拡張できる点が異なる．この分割検索の際に，筆者らのこれまでの手法では検索閾値を分割キーワードに等しく与えていたが，分割キーワードによっては膨大な数の検索結果が得られる場合があった．分割キーワードの検索結果が多いと検証作業に時間がかかるため，最終的な検索結果の出力の遅延につながっていた．

そこで本研究では，各分割キーワードに現れる検索結果の数を予測し，閾値を適切に調整して検索結果の総数を削減させることで，高速な音声検索語検出を実現する方法を提案する．筆者らがこれまで提案してきた検索手法では閾値を少しずつ増加させながら繰り返し検索を行う反復深化的探索を用いている．そこで反復深化的探索の実行中に，一回前の繰り返しで得られた分割キーワードの候補数から，現在の繰り返しにおける検索の候補数を推測する．この推測に基づいて，検索結果の数が均等になるように閾値を増減させることによって, 検索結果の総数を削減する. また，検索結果の数はキーワードによって大きく異なる. そこで，キーワードを構成する音素列に存在するトライグラムから，閾値の増減に対する検索結果数の増加量を推測することで推測精度を向上させ，更なる高速化を実現する方法についても検討する．

以下，本論文の構成について述べる．第 2 章では，従来の音声検索手法の概要について述べる．第 3 章では，従来の検索手法の問題点とその改善手法について述べ，第 4 章で比較評価を行う．第 5 章では，本論文のまとめと今後の課題について述べる．

2. Suffix Array を用いた高速な音声検索語検出

2.1 手法の概要

Suffix Array（接尾辞配列）[12]は，テキスト中の全ての音素に対する index を格納した配列を，Suffix（接尾辞）の辞書順にソートしたものである．図 1 に Suffix Array の構築例を示す. Suffix Array はソート済みのデータ構造であるため，検索キーワードを効率的に見つけ出すことができるが，オリジナルの Suffix Array では完全一致検索を想定している．このため，誤認識を含む音声認識結果を対象とする場合には何らかの曖昧検索技術を導入する必要がある．そこで本手法では山下らによって提案された，Suffix Array に DP マッチングを適用する方法[9]を導入して音声の誤認識に対応している．検索で用いる DP マッチングの累積距離の定義式は以下の通りである．

$$P_{i,j} = \min \begin{cases} P_{i-1,j} + D \\ P_{i-1,j-1} + d(a_i, b_j) \\ P_{i,j-1} + I \end{cases} \quad (1)$$

ここで，a_iは検索キーワード$a_1 a_2 \ldots a_k$中の音素，b_jは系列$b_1 b_2 \ldots b_k$中の音素，$P_{i,j}$は$a_1 a_2 \ldots a_k$と$b_1 b_2 \ldots b_k$の累積距離，$d(a_i, b_j)$はa_i，b_j間の局所距離を表す．D と I はそれぞれ脱落，挿入ペナルティである． DP マッチングの局所距離には音素弁別特徴[13]を利用している．音素弁別特徴とは調

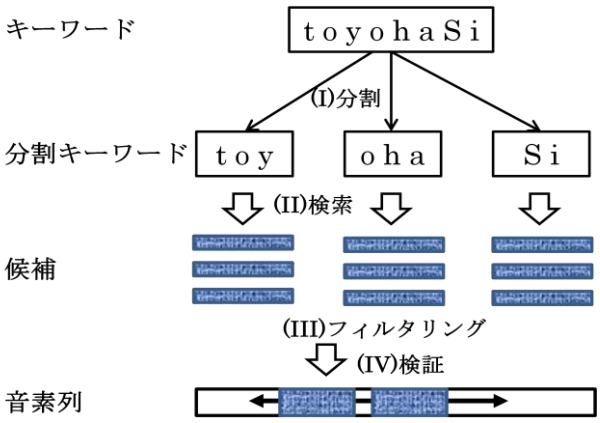

図 3　キーワード検索の流れ

Figure 3　The flow of keyword search.

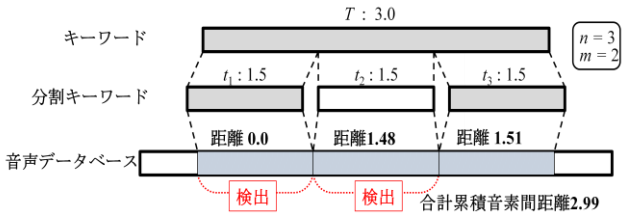

図 4　式(2)を用いた分割キーワードの閾値

Figure 4　A threshold value assigned to each sub-keyword by using expression (2)

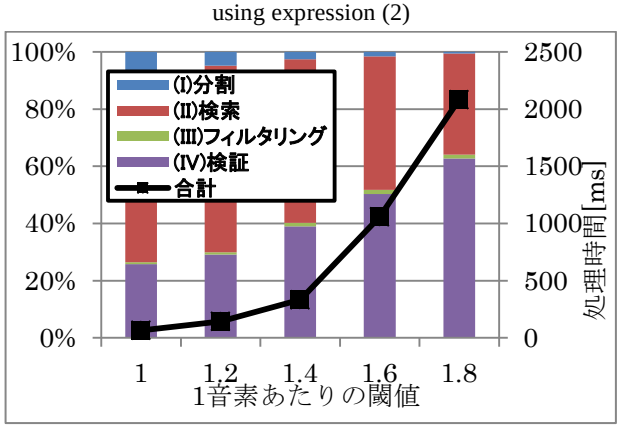

図 5 検索時間に占める各ステップの割合

Figure 5　The rate of each step in the search time

音様式・調音位置によって音素を特徴付けしたもので，図 2 に示すように+または－を取る 15 次元の素性により音素が表される．各音素間でこの素性のハミング距離を求め，局所音素間距離としている．

2.2　キーワード分割

Suffix Array 上で DP マッチングを用いてキーワードを検索する場合，DP マッチングの実行中に枝刈りの閾値内のすべてのパスが保持されるため，閾値が大きいと探索空間および処理時間が指数関数的に増加することが，山下らによって確認されている．閾値は検索キーワードの長さに比例して増加させる必要があるため，検索キーワード長に対して指数的に処理時間が増大する．そこで，この問題を解決するためにキーワードを分割し，分割キーワードを元のキーワードの代わりに検索する手法を導入する．

本手法ではキーワードを分割した場合に分割しない場合と同一の検索結果が得られるよう，図 3 に示す 4 ステップからなる検索を行う．検索は (I)分割，(II)検索，(III)フィルタリング，(IV)検証の各ステップから構成される．(II)検索のステップでは，式(2)のt_sで示される値を各分割キーワードの閾値として設定する．すなわち，各分割キーワードに与えられる閾値は等しい．

$$t_s = \frac{T}{n-m+1} \tag{2}$$

ここでTはキーワード全体の閾値，nはキーワードの分割数，mは検出されるべき分割キーワードの数である．式(2)の導出過程については文献[14]を参照されたい．図 4 に$n=3$，$m=2$，$T=3.0$の場合の例を示す．3 つの区間の距離はそれぞれ 0.0，1.48，1.51 で合計累積音素間距離が 2.99 である．したがってこの区間は検出されなければならない．式(2)に従うと$t_s=1.5$となり，距離が 1.5 より小さい二つの分割キーワードが検出されるため，この区間は検出される．もしT_sが式(2)の値より小さい場合，一つの区間しか検出されない場合があり得るため，閾値以下のキーワードを見落とさないためには式(2)のように設定する必要がある．

2.3　反復深化的探索

本手法では検索の際に閾値を低く設定すると，精度の高い（音素誤りの少ない）検索結果が短い処理時間で得られる．一方，閾値を高く設定すれば，より多くの認識誤りを許容した検索が行われ，多くの正解をみつけることができるが，同時に正解精度は低下し，前節で述べたように検索時間が指数的に増加する．そこで反復深化的な探索アルゴリズムを導入し，まず低い閾値で検索して正確な検索結果を即座にユーザに提示し，先に提示した結果をユーザが確認している間に閾値を上げて検索する方法を採用する．

3.　分割キーワードに与える閾値の調整手法

3.1　従来の分割キーワード検索の問題点

2.2 節で説明した検索の 4 つのステップのうち，処理時間のほとんどを占めているのが（II）検索・(IV）検証のステップである．図 5 に 10 音素から 18 音素の検索キーワード合計 30 個を検索した時の各ステップの処理時間の割合と合計を示す．1 音素あたりの閾値が増加して処理時間が増えるにつれ，(IV)検証の処理時間の割合が増加している事が分かる．従って（IV）検証の処理時間を削減することで全体の処理時間を大きく抑えることができると考えられる．（IV）検証の処理時間は（II）検索で得られる候補の数に依存する．なぜなら（II）検証は候補の数だけ行う必要があるからである．従って（II）検索において大量の候補が得られた場合，検証の処理時間は飛躍的に増大してしまう．筆者らはこれまで各分割キーワードに与える閾値を等しく

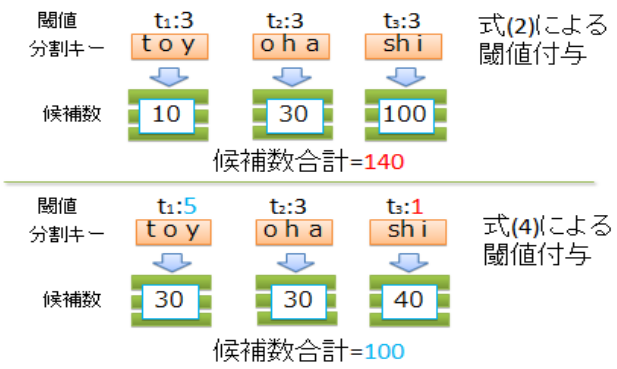

図 6　式(4)による分割キーワードの閾値

Figure 6　A threshold value assigned to each sub-keyword by using expression (4)

与えていた．しかしこの方法では（II）検索の処理で得られる候補数に大きなばらつきが生じる場合があった．このため，一部の分割キーワードの検索で大量の候補が得られてしまい，合計の処理時間が増加するという問題が生じていた．

3.2 分割キーワードに与える閾値の調整手法

前節の問題を解決するために，各分割キーワードの候補数にばらつきが出ないようにすることを考える．そのために，式(2)によって各分割キーワードの閾値を等しく与えるのではなく，個別に与える方法を検討する．筆者らは文献[14]において，閾値Tのキーワードをn個に分割して，そのうちm個以上の分割キーワードを検出するという条件の下で，各分割キーワードの閾値$t_1 \ldots t_n$が条件(3)を満たしていれば必ず(IV)検証の処理によって元のキーワードと同じ検索結果が得られることを示している．

$$\text{任意の} n-m+1 \text{個の} t_1, \ldots, t_{n-m+1} \text{について，} \quad t_1 + \cdots + t_{n-m+1} \geq T \tag{3}$$

文献[8]において筆者らは$m=1$のときに最も高速に検索ができることを示している．また，DP マッチングにおける探索空間と検索時間を削減するために，閾値の総数を最小にすることで DP マッチングにおける探索空間と検索時間を削減できることから，式(4)を使用することにする．

$$t_1 + \cdots + t_n = T \tag{4}$$

なお，先行研究では，本研究と同様に Suffix Array 上でキーワードを分割し，DP マッチングを行う場合の理論的な分析が Navarro らによって行われている．Navarro らも本論文と同様に，キーワードを分割検索しても検索結果が変化しない条件を示している．以下に Navarro らが示した 2 つの定理について説明する．

[定理 1]

$d(A,B) < k$ となるような 2 つの文字列A,Bに対し，$A = A_1 x_1 A_2 x_2 \cdots x_{k+s-1} A_{k+s}$とする．ここで，$A_i, x_i$は文字列，$d(A,B)$は$A,B$間の距離，$k$は編集距離を尺度とするエラーの数であり，$s \geq 1$である．このとき，少なくとも$s$個の部分文字列$A_{i_1} \cdots A_{i_s}$が$B$に現れる．さらに，それらの$B$上での相対距離は$A$上でのそれと$k$より大きくなることはない．

[定理 2]

$d(A,B) < k$ となるような 2 つの文字列A,Bに対し，$A = A_1 x_1 A_2 x_2 \cdots x_{j-1} A_j$とする．ここで，$A_i, x_i$は文字列，$d(A,B)$は$A,B$間の距離，$k$は編集距離を尺度とするエラーの数であり，$j \geq 1$である．$k_i$を$\sum_{i=1}^{j} k_i \geq k - j + 1$を満たす非負の値の集合とすると，最大$k_i$の誤りの$A_i$が少なくとも 1 つ$B$に出現する．

定理 1 はキーワード$B = B_1 B_2 \cdots B_{k+s}$を分割キーワード$B_1$，$B_2$，…，$B_{k+s}$に分割して検索することを想定した時に，少なくとも$s$個の分割キーワードが完全一致検索できることを示している．定理 1 の証明はAからBへの最大k回の変換を考えれば明白である．各変換は最大 1 つのA_iに影響を及ぼすため，少なくともs個は変換されずに残ることになる．なお，定理 1 のsは式(3)におけるmと見做すことができる．定理 1 はキーワード全体の検索では曖昧検索を行っているものの，分割キーワードの検索では先に述べたように完全一致検索を念頭に置いている．したがって各分割キーワードについても曖昧検索を行う本研究に定理 1 を適用することはできない．

一方，定理 2 は分割キーワードの曖昧検索に関する条件を示したものであり，本研究の検索方法と類似している．定理 2 の証明は簡単である．あらゆるA_iがk_iより大きな誤りを含んでBとマッチするならば，合計の距離は$(k-j+1)+j = k+1$より小さくなることはない．しかし定理 2 では最大k_iの誤りのA_iが少なくとも 1 つ B に出現する場合（式(3)における$m=1$）の条件を示しており，m>1 の場合にどのような条件が満たされるかについては議論されていない．

これに対して，筆者らの手法ではキーワードを任意のn個に分割して，m個以上の分割キーワードを検出するための条件を示しており，これは Navarro らの定理を拡張したものであると言える．また，Navarro らは文字列の検索を主目的としているため，定理 1 及び定理 2 は距離尺度が編集距離であることを前提としている．このため，距離を実数とした場合に定理 1 及び定理 2 は成り立たなくなる．図 8 および図 9 にその例を示す．それに対して筆者らの手法では音素間距離（距離尺度）を実数値（例えば混同行列から算出される値）とする場合にも適用できる点が異なる．

本報告では式(4)を満たす範囲で$t_1 \sim t_n$を調整し，(II)の検索時に得られる候補数がばらつかないようにする方法を検討する．図 6 に本手法で目指す閾値調整の例を示す．ここ

表 1　3gram テーブル

Table 1　3-gram Table

3gram/閾値	0	1	2	・・・
aaa	340	340	9078	・・・
aai	7531	7531	21487	・・・
aau	206	206	16450	・・・
aae	858	858	18902	・・・
・・・	・・・	・・・	・・・	・・・

例:”oUsei”の閾値 1→閾値 2 における増加率

3gram/閾値	0	1	2
oUs	743	743	13787
Use	18223	18223	55557
sei	626	626	65395
平均	19174.6	**6530.667**	**44913**

増加率α=44913/6530.667=**6.877**

$$C_i = C'_{i1}e^{a_1(\lfloor t_i/p \rfloor - \lfloor t'_i/p \rfloor)} + C'_{i2}e^{a_2(t_i - t'_i)} * 6.877$$

図 7　増加率の求め方

Figure 7　Method of calculating the rate of increase

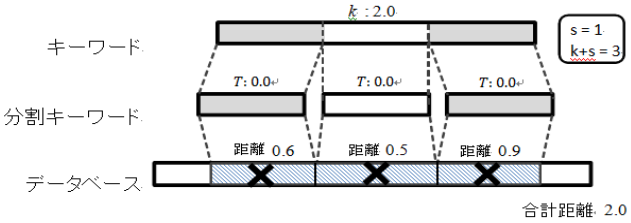

図 8　定理 1 での距離尺度を実数値としたときの検索例

Figure 8　Example of when distance scale is real value in Theorem 1

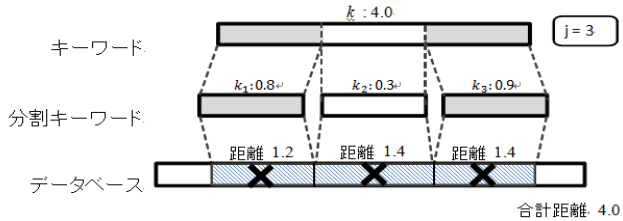

図 9　定理 2 での距離尺度を実数値としたときの検索例

Figure 9　Example of when distance scale is real value in Theorem 2

では$T = 9$，$n = 3$，$m = 1$とし，”toyohashi”という音素列を”toy”，”oha”，”shi”に分割して検索する場合を考える．式(2)に従うと図 6 上部に示すように$t_1 = t_2 = t_3 = 3$となるが，この場合”toy”という分割キーワードについてはあまり候補が得られず，一方で”shi”については非常に多くの候補が得られている．そこで式(4)を満たす範囲で”toy”の閾値を高めに設定し，”shi”の閾値を低めに設定することで候補数のばらつきを少なくし，図 6 下部に示すように候補数の総数を削減することを目指す．

このとき式(4)を満たす範囲で各分割キーワードの閾値をどのように設定すればよいかという問題が生じる．本手法では，反復深化的探索を用いて閾値を少しずつ上げながら検索を繰り返していることから，1 回前の繰り返しの際に得られた各分割キーワードの候補数から，現在の繰り返しにおける検索の候補数を推測し，閾値を調整する方法を検討する．

候補数を予測するにあたって，まず閾値を変化させた場合に，候補数がどのように変化するかを考える．候補数は概ね Suffix Array の探索領域に比例する．つまり，候補数は閾値に対して指数的に変化すると予測できる．従って，分割キーワードiの候補数C_iは，式(5)のように推測できる．

$$C_i = C'_i e^{a(t_i - t'_i)} \tag{5}$$

ここでC'_iはiの反復深化的探索における 1 回前の繰り返しでの候補数，t_iは分割キーワードiの検索で用いる閾値，t'_iは分割キーワードiの 1 回前の繰り返しで用いた閾値，aは候補数の増加カーブを表す定数である．候補数のばらつきを抑えるには，$C_i = \cdots = C_n$となるように各閾値を設定すればよい．これに式(5)を代入し，次の式(6)が導かれる．

$$C'_1 e^{a(t_1 - t'_1)} = \cdots = C'_n e^{a(t_n - t'_n)} \tag{6}$$

このように導いた式(6)と式(4)を解析的に解くと，t_iは以下の様に求められる．分割キーワードの検索ではこの値を使用する．

$$t_i = \frac{ln\frac{C'_1 \cdot C'_2 \cdot \cdots \cdot C'_n}{C'^m_i}}{a \cdot n} + t'_i + \frac{T - T'}{n} \tag{7}$$

3.3　閾値の調整手法の改良

前節の式(5)を用いた候補数の推測では精度が不十分な場合がある．なぜなら検索の際に行う DP マッチングで用いる距離は二つの方法で定義されているためである．1 つは 2.1 節で述べた式(1)の$d(a_i, b_j)$にあたる弁別特徴のハミング距離に基づく定義であり，これは置換誤りの音素間距離の算出に用いられている．もう 1 つは式(1)のDおよびIにあたる挿入・脱落ペナルティである．挿入・脱落ペナルティは検索性能の点から実験的に求めている．この二種類の距離は候補数という観点からは無関係であるため，式(5)のように 1 つの式で閾値を表すと正確に候補数を推測できない場合があった．そこで，本節では二種類の距離定義を考慮した候補数の推測方法について検討する．

まず，式(5)および式(6)で候補数の増加カーブを表すために用いた定数aを，(a)挿入・脱落ペナルティの増加に対応する定数a_1と，(b)置換誤りに対応する定数a_2の二種類で表すことにする．

その上で，式(5)と式(6)を以下のように変更する．ここでC'_{i1}は分割キーワードiの 1 回前の繰り返しにおける挿入・

脱落のみの（置換誤りが全くない）候補の数，C'_{i2}は 1 回前の繰り返しにおける置換誤りを含む候補の数，pは挿入・脱落ペナルティの値，a_1，a_2は定数，その他は式(5)，式(6)のときと同様である．

$$C_i = C'_{i1} e^{a_1(\lfloor t_i/p \rfloor - \lfloor t'_i/p \rfloor)} + C'_{i2} e^{a_2(t_i - t'_i)} \quad (5')$$

$$C'_{11} e^{a_1(\lfloor t_1/p \rfloor - \lfloor t'_1/p \rfloor)} + C'_{12} e^{b_2(t_i - t'_i)} = \cdots = C'_{n1} e^{a_1(\lfloor t_n/p \rfloor - \lfloor t'_n/p \rfloor)} + C'_{n2} e^{a_2(t_n - t'_n)} \quad (6')$$

式(5')及び式(6')の第 1 項は挿入・脱落誤りのみを含んだ候補数を推測する項である．挿入・脱落誤りのペナルティは定数pであるため，現在の閾値t_iが 1 回前の検索での閾値t'_iより大きいpの整数倍以上の値になったときのみ第 1 項の値は増加する．式(5')及び式(6')の第 1 項の指数部$a_1(\lfloor t_1/p \rfloor - \lfloor t'_1/p \rfloor)$はこの性質を表している．

例として，$t_i = 2.99$，$t'_i = 1.0$，$p = 3$の場合を考える．$\lfloor t_i/p \rfloor - \lfloor t'_i/p \rfloor = 0$となるため，現在の式(5')の第 1 項は$C'_{11}$になる．一方，$t_i = 3.0$，$t'_i = 1.0$，$p = 3$の場合は$\lfloor t_i/p \rfloor - \lfloor t'_i/p \rfloor = 1$となるため，現在の式(5')の第 1 項は$C'_{i1}$よりも大きくなる．

3.4 キーワードの音素列を考慮した閾値の調整手法

キーワードの候補数は，キーワードを構成する音素列にも依存している．なぜなら，音素ごとに出現頻度が異なることに加えて，出現頻度の高い音素からの音素間距離も音素ごとに異なるからである．そこで，キーワードの音素列を考慮に入れた候補数の推測を行うことで，精度の高い推測を行うことができると考えられる．

本研究では，キーワード中のトライグラムから閾値の増加に対する候補数の増加量を推測することを検討する．あらかじめすべてのトライグラムの検索結果数を表 1 のようなテーブルで持っておき，検索の際にテーブルを参照し，各トライグラムの 1 回前の検索での閾値と，現在の検索での閾値での検索結果数の統計的な平均値を求める．それらの値からキーワードの閾値の変化に対する増加率αを求め，式(5')，式(6')に組み込む(式(5'')，式(6''))．

$$C_i = C'_{i1} e^{a_1(\lfloor t_i/p \rfloor - \lfloor t'_i/p \rfloor)} + C'_{i2} e^{a_2(t_i - t'_i)} * \alpha \quad (5'')$$

$$C'_{11} e^{a_1(\lfloor t_1/p \rfloor - \lfloor t'_1/p \rfloor)} + C'_{12} e^{b_2(t_i - t'_i)} * \alpha_1 = \cdots = C'_{n1} e^{a_1(\lfloor t_n/p \rfloor - \lfloor t'_n/p \rfloor)} + C'_{n2} e^{a_2(t_n - t'_n)} * \alpha_n \quad (6'')$$

具体例として，“oUsei”というキーワードを 1 回目は閾値 1 で検索し，2 回目は閾値 2 で検索するときのαの求め方を図 7 に示す．“oUsei”に含まれるトライグラムは“oUs”，“Use”，“sei”の 3 種類である．これらの 各閾値での検索結果の数の平均を求めると，閾値 1 のときの平均は 6550.677，閾値 2 のときの平均は 44913 となる．44913 を 6550.677 で割ると 6.877 という値が得られ，この値が“oUsei”における閾値が 1 から 2 に増加した時の増加率となる．このαを式(5'')のαとして用いる．

4. 評価実験

4.1 実験環境と実験の概要

実験は Intel Core 2 Duo プロセッサ 3.00GHz，メインメモリ 6GHz を搭載した PC で行った．実験で用いた音声データは日本語話し言葉コーパス（CSJ，Corpus of SpontaneousJapanese）の 606 時間分の音声を使用した．これらの音声は，大語彙連続音声認識エンジン Julius を用いて音素列に変換した．Julius で用いた音響モデルは不特定話者 PTM（Phonetic Tied-Mixture）Triphone モデルを，言語モデルは Web から学習した語彙 6 万のモデルを用いた．共に Julius ディクテーション実行キットに付属のものである．

まず式(5)の定数a，式(5')の定数a_1，a_2を統計的に求めるために，$p = 3$として，変換した音声データの音素列からランダムに選んだ5～9音素の検索キーワード各200個を検索した．この結果から定数a，a_1，a_2を求めたところ，以下のような値になった．

$$a = 0.7816 \quad (8)$$

$$a_1 = 0.5936 \quad (9)$$

$$a_2 = 0.7148 \quad (10)$$

実験では閾値を式(2)に従って均等に調整した場合（均等），式(4)と式(6)を用いて候補数が平均化するよう閾値を調整した場合（推測方法 1），式(4)と式(6')を用いて閾値を調整した場合（推測方法 2），そして式(4)と式(6'')を用いて閾値を調整した場合（推測方法 3）を比較した．検索キーワードは NTCIR-9 SpokenDoc テストコレクションの STD(Spoken Term Detection)ALL タスク用検索語 50 個のうち，筆者らの手法においてキーワード分割を行う 10 音

素から 18 音素の検索語 30 個を使用した．なお，一音素あたりの閾値が 0.8 以下の場合に関しては閾値調整を行わなくても十分に高速であったため，閾値の調整は行っていない．

4.2 検索結果数の推測精度の評価

表 2，表 3 および表 4 に推測方法 1，2，3 における各閾値での候補数の推測の精度を示す．表の「実測値/推測値の対数」は推測値と実測値の比率の対数をとったもの，「分散」はその分散を示す．したがって「実測値/推測値の対数」，「分散」ともに 0 に近いほど性能がいいと言える．ここで推測方法 1 と推測方法 2 を比較すると，「実測値/推測値の対数」については推測方法 1<推測方法 2<推測方法 3 の順に性能が良くなっていることが分かる．したがって，候補数を 2 種類に分け，検索キーワードの音素列を考慮した推

表 2　推測方法 1 の候補数の推測精度

Table 2　Estimation of number of candidates method 1

音素当閾値	実測値/推測値の対数	分散
1	1.093	0.744
1.2	0.698	0.328
1.4	0.592	0.355
1.6	0.691	0.376
1.8	0.369	0.242

表 3　推測方法 2 の候補数の推測精度

Table 3　Estimation of number of candidates method 2

音素当閾値	実測値/推測値の対数	分散
1.0	0.999	0.600
1.2	0.572	0.137
1.4	0.466	0.169
1.6	0.659	0.174
1.8	0.325	0.106

表 4　推測方法 3 の候補数の推測精度

Table 4　Estimation of number of candidates method 3

音素当閾値	実測値/推測値の対数	分散
1.0	0.373	0.632
1.2	0.336	0.158
1.4	0.126	0.171
1.6	0.393	0.345
1.8	0.144	0.092

測を行うことで，精度については推測方法 1 と比べて推測方法 2，推測方法 3 が上回っている事が分かる．しかし，推測方法 2 と推測方法 3 を比較すると，閾値 1.8 の部分以外で推測方法 2 が下回ってしまっていることが分かる．このことから, 全体的な推測精度は推測方法 3 の方が良いが, 推測が外れてしまった場合，推測方法 3 の推測値は大きく外れてしまうということが分かる．

4.3 候補数と検索速度の評価

表 5，表 6 に 4 つの手法の検索キーワードの合計候補数を，表 7 に平均処理時間（単位[s]）を示す．表 5，表 6 および表 7 から，均等手法と比べて推測方法 1 が，推測方法 1 と比べて推測方法 2, 推測方法 3 が良い結果を得ている事が分かる．表 7 に示すように，平均して約 15%検索時間を短縮できていることから，提案する候補数推定が効果的であることが確かめられた．しかし，推測方法 2 と推測方法 3 を比較した場合，推測方法 3 が推測方法 2 の性能を下回る場合があることが分かる．表 3，表 4 の結果から全体的な候補数の推測精度は推測方法 3 が上回っていること，分散は推測方法 3 のほうが悪いことから，性能が悪化した理由として 2 つの理由が考えられる．1 つは推測方法 2 の時点で分割キーワードに最適な閾値が与えられていた場合が多いということである．表 8 にその例を示す．表 8 は“新婚旅行”という検索キーワードを 2 分割して検索した時の，音素当り閾値 1.0 の時の各分割キーワードの候補数の推測値と実測値である．推測方法 2 と推測方法 3 の推測値はそれぞれ異なり，推測方法 3 の方が精度が高い．しかし，各分割キーワードに与える閾値は変わらなかったため検索結

表 5　各手法の合計候補数の比較(1/2)

Table 5　Comparison of total number of candidates among 4 methods(1/2)

音素当閾値	均等	推測方法 1
1.0	1,194,537	947,778
1.2	3,520,509	2,764,925
1.4	8,194,119	7,481,690
1.6	25,787,601	23,986,702
1.8	70,936,897	60,164,889

表 6　各手法の合計候補数の比較(2/2)

Table 6　Comparison of total number of candidates among 4 methods(1/2)

音素当閾値	推測方法 2	推測方法 3
1.0	870,133	810,474
1.2	2,447,371	2,465,236
1.4	6,675,219	6,785,943
1.6	23,512,520	22,998,089
1.8	58,784,651	58,871,910

表 7　各手法の平均検索時間の比較

Table 7　Comparison of average search time among 4 methods

音素当閾値	均等	推測方法 1	推測方法 2	推測方法 3
1.0	0.200	0.198	0.198	0.186
1.2	0.587	0.478	0.378	0.387
1.4	1.149	1.063	0.890	0.938
1.6	3.088	2.934	2.757	2.542
1.8	6.704	6.021	5.776	5.904

表 8　キーワード“新婚旅行”検索時の候補数の各推測値と実測値

Table 8　Measured value and the estimated value of number of candidates when search “Sinkonryoko”

	推測方法 2		推測方法 3	
	推測値	実測値	推測値	実測値
分割キー1	21514	39440	35241	39440
分割キー2	13727	22451	28505	22451
合計	35241	61891	65399	61891

表 9　キーワード“ペットボトル”検索時の各推測方法における候補数（括弧内は推測値）

Table 9　Number of candidates among 2 methodswhen search “Pettobotoru”

	推測方法 2	推測方法 3
分割キー1	1,151,910(1,026,709)	3,538,870(2,545,940)
分割キー2	2,640,490(2,963,635)	1,080,707(1,132,096)
合計	3,792,400(3,990,344)	4,619,577(3,678,036)

果の候補数は変わらなかった．このように，推測方法 3 で推測方法 2 よりも精度の高い推測をしても与える閾値は推測方法 2 で与えたものと同様であるために，候補数が削減されなかったということが考えられる．また，推測が外れている部分で推測方法 3 は推測方法 2 よりも大きく外れている場合があったため，推測方法 3 によって十分な高速化が実現できなかった場合もあった．検索結果の数は閾値に対して指数的に変化する．このため，推測が外れると大量の候補が得られることがある．表 9 にその例を示す．表 9 は，“ペットボトル” という検索キーワードを 2 分割して検索した時の，音素当り閾値 1.8 の時の各分割キーワードの候補数の実測値である．推測方法 2 での検索時に得られた候補数と比べ，推測方法 3 での検索時に得られた候補数は約 80 万以上も多くなってしまっている．これらが性能悪化の原因であると考えられる．このことから，今後の課題としてキーワードを構成する音素列から候補数を推測する手法の改良を行う必要がある．本論文では 1 回前の検索でのトライグラムの情報のみを参照したが，それ以前のトライグラムの情報も参照するなど，より多くの情報から推測を行うことで精度の高い推測を行うことができると考えられる．また，本研究では検索時間の改善を行うことができたが，依然として高閾値での検索では検索時間の指数的爆発を防ぐまでには至らなかった．今後は高閾値での検索において他手法に切り替えて検索を行うといった方法も検討する必要があると考える．

5. おわりに

本論文では，筆者の研究室で提案してきた Suffix Array を用いた高速音声検索手法について，分割キーワードに与える閾値を均等に与えるのではなく，各分割キーワードにそれぞれ適切な閾値を与えて検索することで高速性を向上させる手法を提案した．第 3 章で従来の筆者らが提案してきた検索システムの問題点，検索閾値の調整方法について述べ，第 4 章で提案手法の評価を行った．

分割キーワードに与える閾値の調整手法として，本研究では反復深化的探索を利用して低閾値時の検索結果の候補数から高閾値時の検索結果の候補数を予測し，各候補数にばらつきが出ないように閾値を調整することで全体の候補数を減少させ検索時間を削減するという手法を提案した．実験の結果，検索閾値を均等に与える手法より，検索結果の候補数を推測して閾値を個別に調整して検索を行う手法が有効であることが確認できた．また，DP マッチングにおける距離定義を考慮して，候補数を 2 種類に分けて推測を行うことで推測の精度を向上し，更に検索速度を向上させることができた．更に，キーワードの音素列を考慮した閾値の調整を行うことで，より精度の高い候補数の推測を行うことができた．しかし，音素列を考慮せずに検索を行う場合と比べて大きく推測が外れてしまう場合があるという問題点もあり，今後は音素を考慮した閾値の調整手法に関しては更に改良を行う必要がある．

本手法は適合率が高く信頼性が高い検索結果を高速に出力できるという利点がある一方，再現率を高くするために閾値を緩めると検索候補，検索時間が指数的に増加するという問題がある．本論文では検索閾値の最適化について検討したが，今後はより精度の高い候補数の推測方法を検討する他に，キーワード分割方法の最適化，他手法との組み合わせや，複数のキーワードを用いた検索についても検討したい．

謝辞

本研究は SCOPE の委託研究の一部である．

参考文献

[1] 秋葉友良：音声ドキュメント検索の現状と課題，情報処理学会研究報告，Vol.2010-SLP-82，No.10，pp.1-8(2010).
[2] Fiscus, J. G., Ajot, J., Garofolo, S. H. and Doddingtion, G.: Results of the 2006 spoken term detection evaluation, Proc. SIGIR'07 Workshop, pp.51-57 (2007).
[3] Akiba, T., Nishizaki, H., Aikawa, K., Kawahara, T. and Matsui, T.: Overview of the IR for Spoken Documents Task in NTCIR-9 Workshop, Proc. NTCIR-9 Workshop Meeting, pp.223-235 (2011).
[4] Pinto, J., Szoke, I, Prasanna, S. R. M. and Hermansky, H.: Fast approximate spoken term detection from sequence of phonemes, Proc. SIGIR'08 Workshop, pp.28-33 (2008).
[5] Wallace, R., Vogt, R. and Sridharan, S.: Spoken term detection using fast phonetic decoding, Proc. ICASSP2009, pp.2135-2138 (2009).
[6] Kanda, N., Sagawa, H., Sumiyoshi, T. and Obuchi, Y.: Open-vocabulary keyword detection from super-large scale speech database, Proc. 2008 IEEE Workshop on Multimedia Signal Processing, pp.939-944 (2008).
[7] Katsurada, K., Teshima, S. and Nitta, T.: Fast keyword detection using Suffix Array, Proc. InterSpeech2009, pp.2147-2150 (2009).
[8] Katsurada, K., Sawada, S., Teshima, S., Iribe Y. and Nitta, T.: Evaluation of Fast Spoken Term Detection Using a Suffix Array, Proc. InterSpeech2011, pp.909-912 (2011).
[9] 山下達雄，松本祐治：Suffix Array を用いたフルテキスト類似用例検索，情報処理学会研究報告，Vol.1997-NL-97，No.85，pp.83-90 (1997).
[10] G. Navarro and R. Vaeza-Yates, “A hybrid indexing method for approximate string matching,” Journal of Discrete Algorithms, Vol.1, no.1, pp.205-239, 2000.
[11] G. Navarro, R Baeza-Yates, E. Suinen, and J. Tarhio, “Indexing methods for approximate string matching,” IEEE Data Engineering Bulletin, vol.24, no.4, pp.19-27, 2001.
[12] Manber, U. and Myers, G.: Suffix Arrays: a new method for on-line string searches, SIAM Journal on Computing, Vol.22, No.5, pp.935-948 (1993).
[13] Fukuda, T. and Nitta, T.: Orthogonalized distinctive phonetic feature extraction for noise-robust automatic speech recognition, IEICE Trans., Vol.E87-D, No.5, pp.1110-1118 (2004).
[14] 桂田浩一，入部百合絵，新田恒雄：Suffix Array を用いた高速 STD におけるキーワード分割に関する理論的検討，情報処理学会研究報告，Vol.2011-SLP-89，No.16，pp.1-6 (2011).