Vol.2010-SLP-80 No.11
2010/2/13

大規模音声ドキュメントを対象とした高速キーワード検索システムとその評価

手島 茂樹[†1]　入部 百合恵[†1]
桂田 浩一[†1]　新田 恒雄[†1]

我々は Suffix Array を用いたテキスト曖昧検索アルゴリズムを音声検索に適用し，音素単位のマッチングを行うことで，大規模な音声ドキュメントに対する高速なキーワード検索を行う手法を提案してきた．本報告では転置インデックスを用いた検索手法と提案手法の比較評価を行う．比較は検索速度と未知語検索の性能について行った．未知語検索性能については，情報処理学会音声言語情報処理研究会の音声ドキュメント処理ワーキンググループで策定されている，Spoken Term Detection(STD) のためのテストコレクションを用いた．比較実験の結果，既存手法に対して提案手法が応答性や未知語検索性能で優れていることを確認した．また，実装の改善により処理速度を数倍高速化したので併せて報告する．

Evaluation of fast spoken term detection system for large scale document

Shigeki Teshima,[†1] Yurie Iribe,[†1]
Kouichi Katsurada[†1] and Tsuneo Nitta[†1]

We have developed a fast spoken term detection(STD) method for large scale spoken documents. The method employs a phoneme-based string matching algorithm using a suffix array. This paper compares it with an existing method based on inverse index. Comparison is carried out on the detecting speed and performance of an out of vocabulary(OOV) queryset described in the test collection for STD developed by Spoken Document Processing Working Group in SIG-SLP(Spoken Language Processing). The proposed method outperforms the existing method on response time and performance at the speech task containing OOV terms. We also report on the improvement that realizes severalfold search speed by changing programming language.

1. はじめに

ブロードバンド回線の普及など情報通信技術の発展により，Web 上で音声や動画像のコンテンツを利用する機会が増えると共に，コンテンツ数も急激に増加している．これら Web 上の音声データを効率よく利用するには音声検索技術が必要になる．しかし，従来の音声検索研究は性能向上に主眼を置くものが多く，高速化を目指したものは少ない．近年，高速な音声検索手法が幾つか提案されているが[1)2)]，音声 DB の規模が大きくなると，DB に見合う規模の索引データを作成しなければならない欠点があった．また，高速な二次記憶装置が必要となりコスト面から望ましくないといえる．

我々は先に，Suffix Array を用いた高速キーワード検索手法を提案し，この中で反復深化探索により正確な検索結果から順に，結果を高速に出力できることを示した[3)]．本報告では，既存手法として転置インデックスを用いた音声検索手法との比較評価を行う．また実装方法を改良して一層の高速化を実現した．

以下，第 2 節では Suffix Array を用いたテキスト曖昧検索の手法について解説し，第 3 節でこれを音声検索に適用するための提案手法を述べる．次に第 4 節で評価実験について述べる．最後に第 5 節で本報告のまとめと今後の課題について述べる．

2. Suffix Array を用いたテキスト検索

2.1 Suffix Array

Suffix Array(接尾辞配列)[4)] とは，テキスト中の全ての suffix(接尾辞) を辞書順にソートしたもので，テキスト検索で検索キーワードを効率的に見つけ出すためのデータ構造である．例えば，“abracadabra” というテキストに対して Suffix Array を構築すると，**図 1** のようになる．図中の index はその suffix がテキストの何文字目から始まるかを示す．ここでキーワード “bra” が出現する位置を検索したい場合，Suffix Array を二分探索すると index が 8 と 1 の位置に出現することが効率的に得られる．

ソートされた index のみを保持すれば良いこの方式は，必要なデータ領域が小さく，任意の文字列の出現位置を文字単位で検索できるという特徴がある．

†1 豊橋技術科学大学
Toyohashi University of Technology

2.2 Suffix Array を用いたテキスト曖昧検索

山下ら[5] は Suffix Array を用いたテキスト曖昧検索のアルゴリズムを提案している．このアルゴリズムは，Oflazer による辞書類似検索を行う Error-tolerant Recognition アルゴリズム[6] を Suffix Array を用いて全文曖昧検索に拡張したものである．

山下らのアルゴリズムでは，Suffx Array を木構造に見立てて探索を行う．木構造の根から全てのパスに対して DP マッチングを行い，各パスと検索キーワードとの累積距離を求める．その際，累積距離がある閾値を越えたら，そのノード以下の部分木の探索を打ち切る“枝刈り” を行う．この枝刈りを行うことで高速な曖昧検索を実現している．

枝刈りを行うかどうかを判断する Cut-off 距離は次式で定義される．

$$cutdist(m) = \min_{1 \le k \le K} P_{k,l}$$

m は現在のノード，K はキーワード長であり，$P_{k,l}$ は DP マッチングによるキーワード $a_1 a_2..a_k$ と系列 $b_1 b_2..b_l$ の間の距離を表す．$b_1 b_2..b_l$ は根からノード m までに辿った枝に設定された文字の系列である．

探索において枝が刈られることなく，検索キーワードとの距離 $P_{K,l}$ が閾値以下となるノードに到達したら，そのノードを根とする部分木に属する suffix の index をキーワードの出現位置として出力する．例として，テキスト”abracadabra” からキーワード “bra” を閾値を 1 として検索したときの途中経過を**図 2** に示す．“ac” の枝と “ada” の枝は Cut-off 距離が閾値を越えたため枝刈りが行われ，それ以降は探索されない．また，“bra” の枝は検索キーワードとの距離が閾値内であるため，この枝の部分木に属する “bra” と “bracadabra” の index が検索結果として出力されている．

3. Suffix Array の音声検索への適用

3.1 LVCSR 音素出力と調音特徴距離の利用

提案手法は音声データに対して音声認識処理を施し，その結果得られる音素列から Suffix Array を構築して検索を行う．音声データからの音素列取得には LVCSR(Large Vocabulary Continuous Speech Recognition) を用いる．

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．

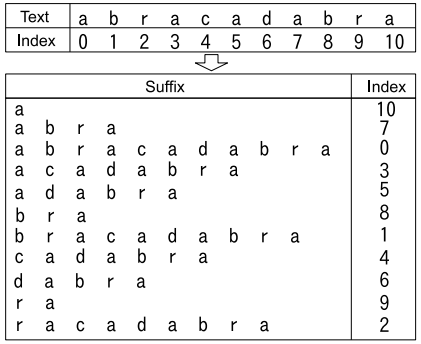

図 1 Suffix Array
Fig. 1 Example of suffix array

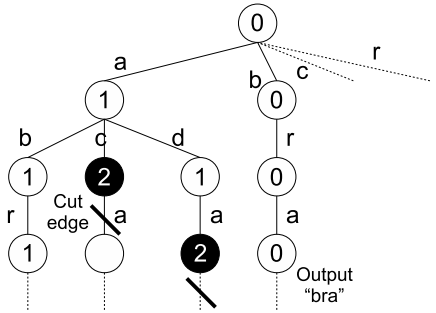

図 2 Suffix Array の探索
Fig. 2 Keyword search on suffix array

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

ここで，a_i は検索キーワード $a_1 a_2..a_K$ の音素，b_j は系列 $b_1 b_2..b_l$ の音素，$d(a_i, b_j)$ は a_i, b_j 間の局所距離である．

音声認識では音素によって誤り易さが異なる．そこで，音素間の音響的距離を適切に表す調音特徴[7] から求めた距離を局所距離 $d(a_i, b_j)$ に用いることにした．調音特徴とは調音様式・調音位置から音素を弁別したもので，$+$ または $-$ を取る 15 次元の素性により音素を定義している．本研究では各音素間でこの素性のハミング距離を求め，音素間距離とする．

3.2 キーワードの分割検索

2.2 節で説明したアルゴリズムは，枝刈りの閾値に対して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．これは閾値が増加すると探索範囲が一気に広がるためである．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て検索する手法を導入する．

分割検索によって処理時間の増大は回避できるが，キーワード中の音素認識誤りは一様ではないため，分割後のキーワード (分割キー) の一部に検出されないものが生じる場合がある．極端な場合には，作成された分割キーのうち閾値内で一つしか検出されない場合がある

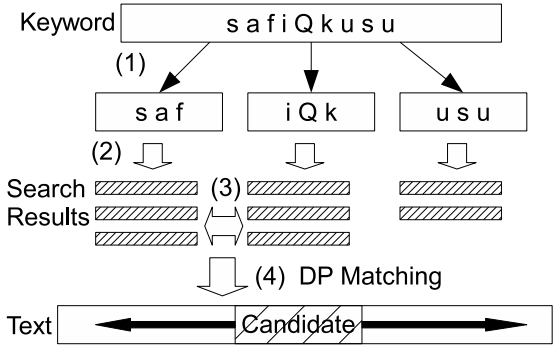

図 3 キーワード検索の流れ
Fig. 3 Process of keyword search

ため，検出された各分割キーを候補として前後の音素を調べ，キーワード全体が閾値内でマッチするかどうかを再度検証する必要がある．

しかし，Suffix Array の検索で得られた候補全てに対して検証のための DP マッチングを行うと，その回数が膨大になり処理に大きな時間が必要となる．そこで，必ず二つ以上の分割キーが検出されるよう，音素当たり閾値 t' を次式のように設定する．ここで p は分割数，t は元の閾値である．

$$t' = \frac{p}{p-1}t$$

ただし上式に従うと分割数が 2 になる場合には音素あたりの閾値が 2 倍になり，分割キーの閾値が分割しない場合のキーワード全体の閾値と同じになる．このため，分割を行うと逆に速度低下を招くと考えられる．そこで，分割する場合は 3 分割以上にすることとする．

本手法の流れをまとめると**図 3** のようになる．まず，(1) 検索キーワードが長い場合は固定長サイズで 3 分割以上にし，(2) 分割キーごとに Suffix Array の検索を行い，候補を検出する．続いて，(3) 任意の 2 つの分割キーのそれぞれの候補のうち，テキスト上の出現位置が近い候補の組を見つける．最後に，(4) 見つかった組の出現位置において検索キーワード全体の DP マッチングを行い，距離が閾値以下ならば最終的な検索結果とする．

3.3 反復深化探索

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，多くの正解を見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再検索する方法を採用する．

Average threshold	Phoneme sequence
0.0	h i t o r i g u r a sh i
0.4	h i t o r i k a r a sh i
0.6	h i t o r i k a r a f u
0.8	h i t o r i g a m o ch i
1.0	h i t o n i i k u n i sh i
1.2	f u t a r i k u n o sh i

図 4 閾値ごとの誤り系列例
Fig. 4 Example of phoneme sequences within certain threshold

4. 比較評価実験

4.1 実験環境

実験は Intel Core 2 Duo プロセッサ 3.33GHz，メインメモリ 8GB を搭載した PC で行った．音素間距離の平均値は 5.7 であった．したがって，例えば 1 音素当たりの枝刈りの閾値を 1.0 としたとき，平均で 5.7 文字に一つの誤りを許容することになる．例として，“h i t o r i g u r a sh i” の誤り系列を閾値の平均距離毎に**図 4** に示す．

4.2 比較対象

比較対象には音素 N-gram の転置インデックスを用いる検索手法を採用した．この手法は神田らの手法[1] を基にしている[⋆1]．転置インデックスとは文書検索などで一般的に用いられている索引付け手法であり，文書から抽出した見出し語とその出現位置をマップしたものである．音声検索に用いる場合，検索対象の音声ドキュメント音素列へ変換し，見出し語を音素 N-gram として音素列の転置インデックスを作成する．検索の際は，検索キーワードを音素 N-gram に分解し，それぞれの N-gram で転置インデックスを引き，N-gram が

⋆1 神田らは N-gram を検索する際，N-gram の出現順序や音響的な特徴を考慮しないことで検索を高速化している．しかし事前に行った実験の結果，出現順序を考慮しても速度の変化は少なかったため，本実験では出現順序を考慮して検索し，精度の向上を図った．

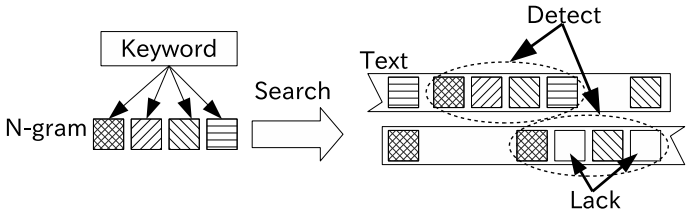

図 5 比較対象手法
Fig. 5 Existing method

表 1 認識結果
Table 1 Recognition result of corpus

Target	Correct rate	Accuracy
Core	76 %	72 %
All	75 %	71 %

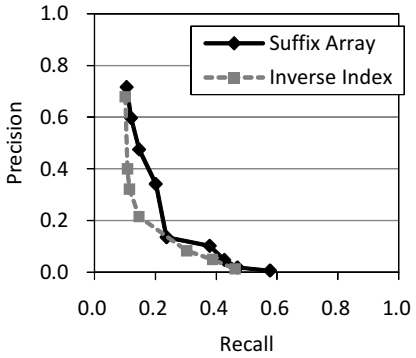

図 6 コア講演用未知語セットの検索結果
Fig. 6 Result of searching for OOV queries on core speeches

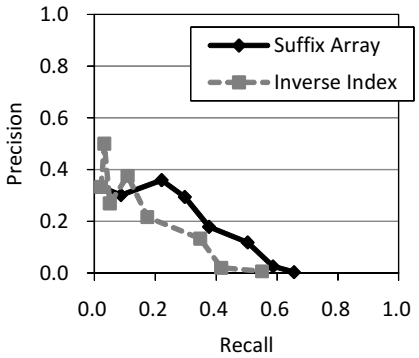

図 7 全講演用未知語セットの検索結果
Fig. 7 Result of searching for OOV queries on all speeches

多く現れる部分を**図 5** に示すように検索結果として検出する．このとき，キーワード中の N-gram の一部が欠落することを許すことで音声認識誤りを許容した検索を行う．欠落を許容する量を閾値で定め，この閾値を変化させることで検索結果を制御できる．閾値は検索語の N-gram が欠落する割合とする．すなわち，閾値は 0 から 1 の値を取り，例えば 0.3 の場合は 3 割の N-gram が欠落した箇所までが検索結果として出力される．

N-gram は N=3(トライグラム) として実装し，実験を行った．

4.3 未知語検索の比較

情報処理学会音声言語情報処理研究会 (SIG-SLP) で提案されている，Spoken Term Detection(STD) の評価用テストコレクション[9] を用いて，未知語に対する検索性能の評価を行った．このテストコレクションは CSJ を検索対象ドキュメントとし，既知検索語セット (全講演用，コア講演用) や未知検索語セット (全講演用，コア講演用)，簡易性能評価用セット等の検索語のセットを用意している．全講演とは CSJ に収録されている学会講演・摸擬講演 2702 講演 (604 時間) を指し，コアとはそのうち詳細なラベル付けがなされた 177 講演を指す．これらの検索語セットのうち，全講演用およびコア講演用の未知検索語セットを用いて検索実験を行った．

未知語セットは低頻度の単語から，全講演用，コア用それぞれ 50 個が設定されている．テストコレクションは，コア以外の講演を学習データとして言語モデルを構築し，音声認識を行うことを想定している．この言語モデルの辞書はカットオフ 4 で作成されるため，出現回数が 3 回以下の単語は登録されず未知語となり，この低頻度未知語の中から検索語が選出されている．しかし，コア講演用未知未知検索語セットのうち 4 つの検索語は出現回数が 4 回以上であることが分かったため，これらの検索語はセットから外し，46 個の未知検索語で実験を行った．

検索対象音声の認識には前節同様に Julius を利用した．言語モデルはテストコレクションの規定に従い，コア以外の学会講演・摸擬講演 2525 講演から作成した語彙約 27000 の単語 3-gram モデルを用いた．

認識結果の音素認識率，音素正解精度は**表 1** の通りである．ここで，言語モデルはコア以外の講演から作成しているため，コア講演に関してはオープンな評価であるが，全講演に関してはオープンな評価でないことに注意されたい．

検索結果の再現率，適合率を**図 6**，**図 7** に示す．図 6 はコア用未知語セットを検索した結果，図 7 は全講演用未知語セットを検索した結果である．

図 6，図 7 から，殆どの場合で提案手法は比較手法より良いか同程度の性能が得られていることが分かる．これは，提案手法が調音特徴による音響的距離を考慮している効果だと考えられる．

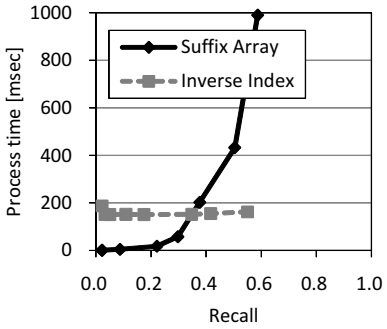

図 8 検索速度の比較
Fig. 8 Comparison of process time

4.4 検索速度の比較

全講演用未知検索語セットを CSJ の学会講演・摸擬講演 2702 講演 (604 時間) から検索した際の処理時間を**図 8** に示す.

図 8 より，再現率が低い (閾値が低い) 場合には提案手法の方が短い時間で検索できていることが分かる．特に，比較手法では閾値に関わらず常にある程度の処理時間を要するのに対し，提案手法は低閾値時 (適合率が高く，再現率が低い場合) には一瞬といえる時間で検索できている．この点から，提案手法は反復進化探索を適用するこで，応答性に優れた検索システムを実現できているといえる．

一方，再現率を増加させると提案手法の処理時間は指数的に増加するため，再現率が高い (閾値が高い) 部分では処理時間が逆転している．

4.5 大規模音声ドキュメントに対する検索

大規模音声ドキュメントに対する処理速度を評価するため，約 1 万時間分の音素列を対象として検索を行った．検索対象は毎日新聞記事データ集[10] の新聞記事データから音素列を作成[⋆1] して用いた．新聞記事データは，漢字仮名混じり文を MeCab[11] を用いて仮名文にした後，規則的に音素列へ変換した．

変換した音素列から 2000, 4000, 6000, 8000, 10000 時間分の音素列およびその Suffyx Array を作成し，それぞれを検索したときの処理時間を測定した．検索キーワードには，検索対象の音素列から 6, 12, 18, 24 音素の系列を無作為に各 100 個抽出して用いた．各キーワード長ごとの検索対象長に対する処理時間の変化を**図 9** に示す．(i)，(ii) はそれぞれ閾値が 0.6, 1.0 の場合である．

また，先の報告[3] で行った同様の実験の結果を**図 10** に示す．図 10 では閾値 1.0，24 音素キーワードで 10000 時間を検索した場合，およそ 6 秒の処理時間を要しているが，本実験ではおよそ 1 秒で検索できている．他の条件を見ても 4〜6 倍高速化されている．これは実装言語を変更したことに因るものが大きいと考えられる．以前は C#(Microsoft .NET Framework) を用いて実装していたが，これを C++で再実装した．C#では配列アクセス

⋆1 4.4 節で述べた CSJ コーパスの音素数を基に 1 万時間分を算出

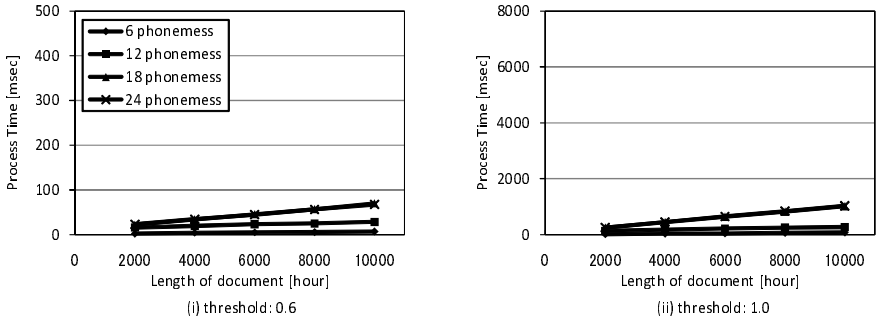

図 9 大規模ドキュメントに対する処理時間 (改善後)
Fig. 9 Process time for large scale document(improved)

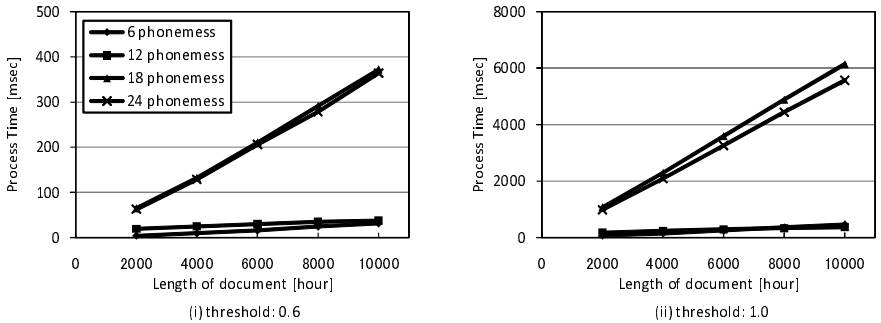

図 10 大規模ドキュメントに対する処理時間 (改善前)
Fig. 10 Process time for large scale document(base)

時に境界チェックが自動的に行われるが，C++では行われないため，その分高速化されたと考えられる．

5. ま と め

本報告では，先に提案した Suffix Array を用いて音素単位の音声検索を高速に行う手法について比較評価を行った．比較対象として N-gram 転置インデックスを用いる手法を用いて，検索速度および未知語に対する検索性能を評価した．速度比較の結果，低閾値時には提案手法よりも大幅に少ない処理時間で検索でき，応答性に優れることを確認した．未知語検索においては，提案手法はほぼ全ての場合で比較手法よりも良いか同程度の性能が得られた．また，実装を改善した結果，検索速度を 4〜6 倍高速化できた．

今後は，キーワード分割の条件や，等しい大きさで分割できない場合の分割方法などについて検討したい．

参 考 文 献

1) N.Kanda, H.Sagawa, T.Sumiyoshi and Y.Obuchi : Open-Vocabulary Keyword Detection from Super-Large Scale Speech Database, IEEE MMSP 2008, pp.939-944 (2008).
2) K.Thambiratnam and S.Sridharan : Dynamic Match Phone-Lattice Searches For Very Fast And Accurate Unrestricted Vocabulary Keyword Spotting, ICASSP 2005, vol.1, pp.465-468 (2005).
3) 手島茂樹, 桂田浩一 and 新田恒雄 : Suffix Array を用いた高速なキーワード検索, 情報処理学会研究報告, vol.2009-SLP-77, no.3, pp.1-6 (2009).
4) U.Manber and G.Myers : Suffix arrays: a new method for on-line string searches, SIAM J.Computing, vol.22, no.5, pp.935-948 (1993).
5) 山下 達雄 and 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 NL, vol.97, no.85, pp.83-90 (1997).
6) K.Oflazer : Error-tolerant Tree Matching, COLING-96: The 16th International Conference on Computational Linguistics, pp.860-864 (1996).
7) T.Fukuda and T.Nitta : Orthogonalized Distinctive Phonetic Feature Extraction for Noise-Robust Automatic Speech Recognition, IEICE Trans., vol.E87-D, no.5, pp.1110-1118 (2004).
8) 李 晃伸 : 大語彙連続音声認識エンジン Julius ver.4, 情報処理学会研究報告 SLP, vol.2007, no.129, pp.307-312 (2007).
9) 伊藤慶明ほか : 音声中の検索語検出のためのテストコレクション構築 -中間報告-, 情報処理学会研究報告, vol.2009-SLP-78, no.4, pp.1-8 (2009).
10) 毎日新聞社 : CD-毎日新聞データ集 (1997-2000).
11) 工藤 拓 : MeCab : Yet Another Part-of-Speech and Morphological Analyzer, http://mecab.sourceforge.net/ (2008).