Vol.2009-SLP-77 No.3
2009/7/17

Suffix Array を用いた高速なキーワード検索

手島 茂樹[†1] 桂田 浩一[†1] 新田 恒雄[†1]

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，音声データベースに対する高速かつ使用データ領域が小さいキーワード検索を実現する．キーワード長に対して処理時間が指数的に増加する問題を解決するため，キーワードの分割検索法を提案した．また，高精度な検索結果を高速に提示するため，反復深化探索のアルゴリズムを導入した．CSJ 男女話者 600 時間分の音声データに対して検索実験を行い，キーワード分割の有効性，および，反復深化探索の適応性を評価した．さらに，連続 DP マッチングと比較して高速に検索できることを確認した．

Fast keyword spotting using suffix array

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

We present a fast and space-saving keyword spotting method for spoken documents. The method employs the phoneme-based approximate string matching algorithm using a suffix array. To solve the exponential explosion problem of process time with length of keywords, we propose a keyword division method and show effectiveness of the method experimentally. An iterative lengthening search algorithm is used to retrieve accurate results rapidly. We confirmed that the method can retrieve accurate results fast from the CSJ database containing 600 hour speech data of male and female speakers. We also confirmed that this approach is faster than the one using conventional continuous DP-matching.

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

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，コンテンツの数も急激に増加している．これらの Web 上の音声データを効率的に利用するには音声検索技術が必要になるが，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 が大規模になると，これらの手法では 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．このため高速な二次記憶装置が必要となり，コスト面からは望ましくない．

本稿では，必要なデータ領域が比較的小さい Suffix Array を導入し，データ領域が小さく高速な，音声に対するキーワード検索手法を提案する．本手法では，Suffix Array 上で DP マッチングによる音素単位のマッチングを行う．キーワード長に対して処理時間が指数的に増大する問題を解決するため，キーワードを分割して検索する手法を導入し，処理時間を抑制する．また，精度の高い結果を高速に提示するため，反復深化探索のアルゴリズムを導入する．

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 節で提案手法の評価を行う．最後に第 5 節で本稿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)[3)] 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．

2.2 Suffix Array を用いたテキスト曖昧検索

山下ら[4)] 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

Vol.2009-SLP-77 No.3
2009/7/17

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[5] 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 $a_1a_2..a_k$ と系列 $b_1b_2..b_l$ の間の距離を表す．$b_1b_2..b_l$ 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 音素出力と音素弁別特徴距離の利用

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．

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

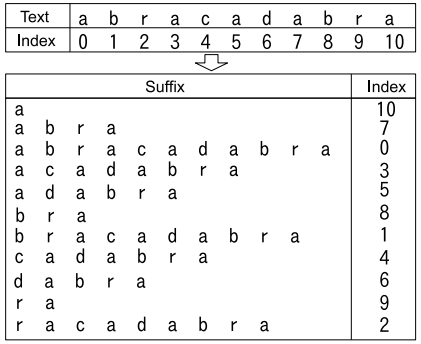

図 1 Suffix Array
Fig. 1 Example of suffix array

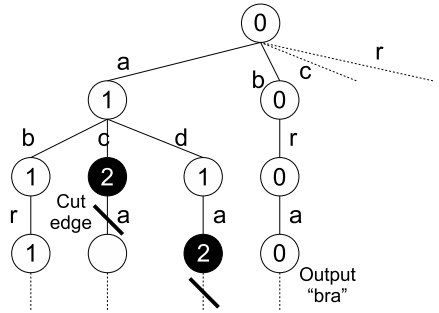

図 2 Suffix Array の探索
Fig. 2 Keyword search on suffix array

ここで，a_i は検索キーワード $a_1a_2..a_K$ の音素，b_j は系列 $b_1b_2..b_l$ 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音素弁別特徴[6] 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．音素弁別特徴とは調音様式・調音位置から音素を弁別したもので，+ または − を取る 15 次元の素性から音素を定義している．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

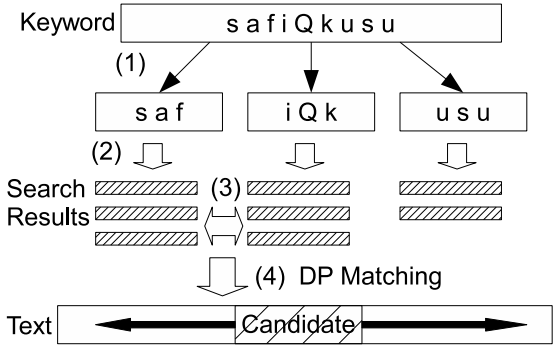

図 3 キーワード検索の流れ
Fig. 3 Process of keyword search

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. 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．検索対象には CSJ(Corpus of Spontaneous Japanese) の男女話者 600 時間分の音声データを，Julius[7] を使用して認識した結果を用いた．認識に用いた音響モデルは不特定話者 PTM(Phonetic Tied-Mixture) Triphone モデル，言語モデルは Web から学習した語彙 6 万の単語 3-gram モデルであり，双方とも Julius Dictation Kit に付属しているものである．

認識の結果，音素認識率は 73%，音素正解精度は 69%であった．認識結果の音素列から Suffix Array を作成したところ，そのサイズは約 81MB であった．

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 検索キーワードの分割サイズに関する予備実験

最適な検索キーワードの分割サイズを求めるために予備実験を行った．実験では枝刈りの閾値を 1.0 とし，6〜30 音素のキーワードを各 10 個，計 250 個を検索したときの平均の処理時間を求めた．閾値を 1.0 としたのは，それ以下の閾値よりもキーワード長ごとの処理時間の差が大きいためである．分割サイズを 4〜10 音素としたそれぞれの場合について検索を行ったところ，**図 5** のような結果が得られた．

この予備実験では，分割サイズを 6 音素とする場合が最も高い処理速度を示したため，以

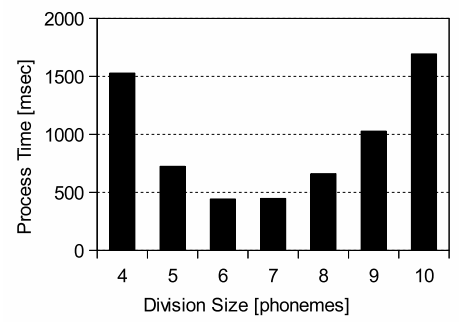

図 5 分割サイズに関する予備実験
Fig. 5 Preliminary experiments of split size

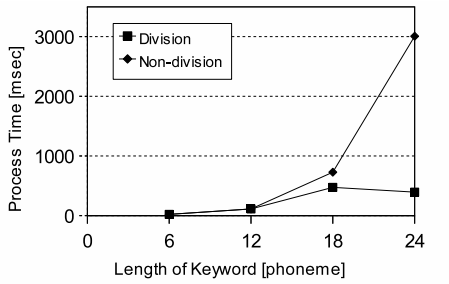

図 6 分割/非分割時の処理時間
Fig. 6 Processing time of keyword split/unsplit

降の実験はこの分割サイズを用いて行った.

4.3 キーワード分割検索の有効性に対する実験

分割検索の有効性を検証するため，検索キーワードの分割を行う場合と行わない場合の処理時間の測定を行った．枝刈りの閾値は 1.0 とし，検索キーワードには 6，12，18，24 音素の名詞をそれぞれ 100 個用いた.

結果を**図 6** に示す．分割数が 2 以下になる 6, 12 音素では分割を行わないため，処理時間は同じになっている．18, 24 音素の検索では分割を行うことで処理時間の増加を抑えられており，キーワードの分割が有効であることが分かる.

4.4 検索性能評価

検索性能を評価するために，音素当たりの枝刈りの閾値を変化させて検索を行い，それぞれの閾値に対する検索結果の再現率，適合率および処理時間の測定を行った．検索キーワードには 6，12，18，24 音素の名詞をそれぞれ 100 個用いた．結果を**図 7** に示す．図中の (a)〜(d) はそれぞれ，6, 12, 18, 24 音素のキーワードを検索した場合である.

図 7 より，(a)〜(d) のいずれの場合も閾値 0.2 以下では 20msec 以下の処理時間で検索できており，特に (b)(c)(d) では検索結果は高い適合率を保っている．従って，低い閾値から反復して検索を行い，順にユーザに提示する手法は有効であるといえる.

(a)〜(d) の結果を比較すると，(c) および (d) では閾値が大きくなると (b) と比べて急激に処理時間が増大している．これは (c)，(d) では 6 音素の分割キーを検索しており，このような短いキーワードでは分割キーごとの検索結果が膨大な数になるためであると考えられる．これに対して (b) は分割数が 2 となるために分割しておらず，12 音素という比較的

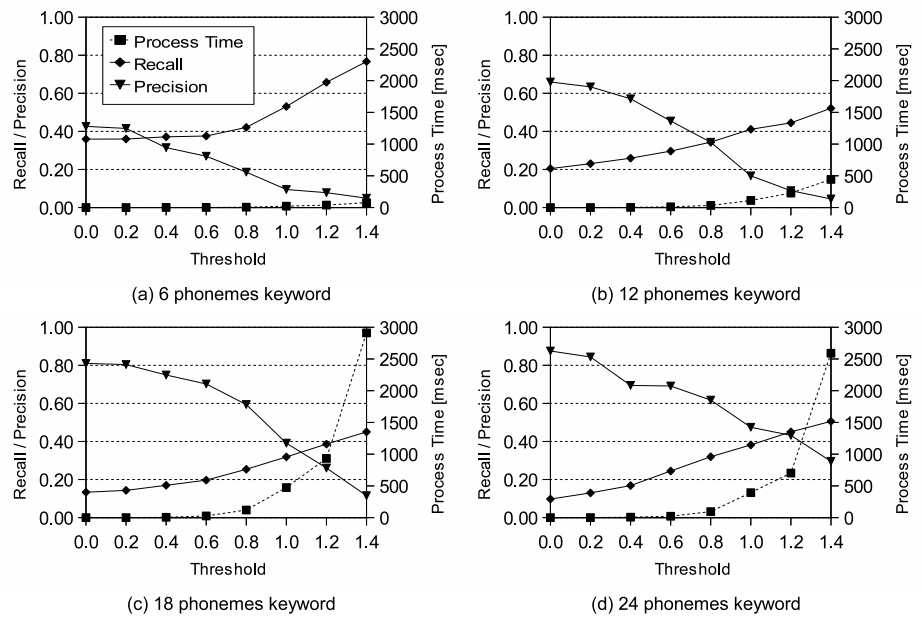

図 7 検索性能
Fig. 7 Search performance

長いキーワードで検索している．実際に，閾値 1.0 の場合の検索結果の数は 12 音素の場合は平均 107 個なのに対し，6 音素の場合は 9124 個となっている.

この結果からは一見分割しない方が閾値を上げた時の処理時間が緩やかであるため望ましいように見えるが，実際には図 6 に示したように，分割をしないと処理時間そのものは大きくなる．これらの性質から，最も望ましい処理法は閾値を上げた時に分割/非分割を制御することであると考えられる．この点については今後検討したい.

4.5 連続 DP マッチングとの比較

提案手法と Suffix Array を用いない一般的な連続 DP マッチングの検索速度の比較を行った．24 音素のキーワードを検索したときの結果を**図 8** に示す．Suffix Array を用いない連続 DP マッチングでは，検索対象の音素列の先頭から末尾までを走査しながら DP マッチングを行い，検索キーワードが閾値以下の累積距離で現れる箇所を探す.

図 8 より，提案手法は連続 DP マッチングよりも短い時間で検索できていることが分か

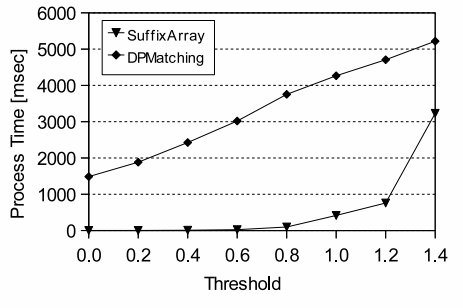

図 8 連続 DP マッチングとの比較
Fig. 8 Comparison with Continuous DP Matching

る．特に低い閾値においては，連続 DP マッチングはある程度の処理時間が必要となっているのに対し，提案手法は非常に高速に検索できている．このように短時間で検索結果を提示できるという点は，反復深化探索を適用する上で重要な要素である．

一方で，連続 DP マッチングは閾値の増加に比例して処理時間が増加するのに対し，提案手法は指数的に増加している．このため，閾値をより大きくしていった場合，提案手法よりも Suffix Array を用いない連続 DP マッチングの方が高速になると予測される．このことから，閾値に応じて提案手法と Suffix Array を用いない連続 DP マッチングを選択的に用いる手法が有効であると考えられる．今後はこの二手法の選択を決定するアルゴリズムも検討したい．

4.6 検索対象データ長を変化させたときの処理時間の変化

検索対象データ長を変化させたときの検索処理時間の変化を測定した．検索対象には，毎日新聞記事データ集[8] の新聞記事データから約 1 万時間分[⋆1]の音素列を作成して用いた．新聞記事データは，漢字仮名混じり文を MeCab[9] 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．また，音素当たりの閾値は 0〜1.0 の場合を測定した．

⋆1 4.1 節で述べた CSJ コーパスの音素数を基に算出

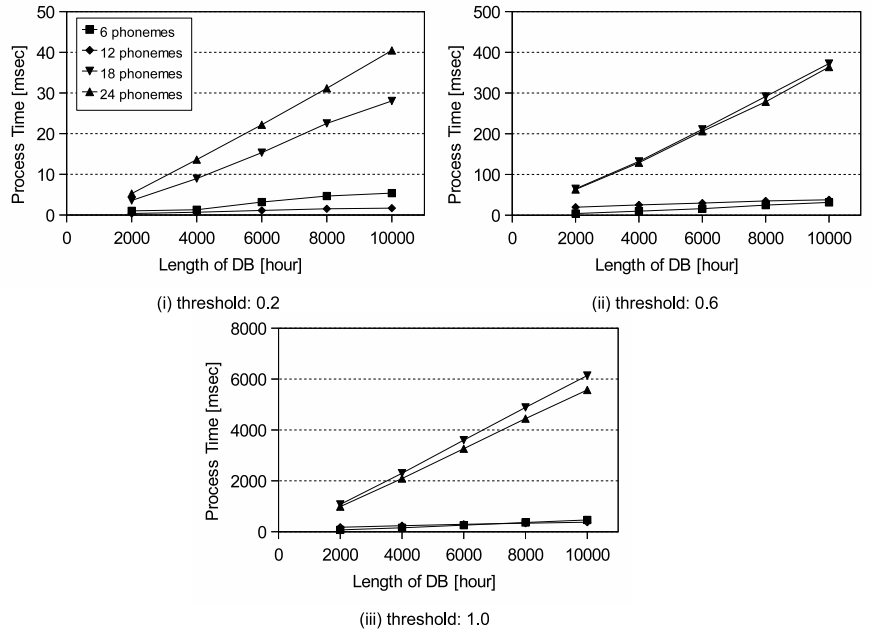

図 9 検索対象長に対する処理時間の変化
Fig. 9 Process time increase against length of DB

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)，(iii) はそれぞれ閾値が 0.2, 0.6, 1.0 の場合である．どの閾値においても，分割を行った 18, 24 音素のキーワードは検索対象長に比例して処理時間が増加してる事が分かる．処理時間の内訳を見ると，Suffix Array の検索よりも検証のための DP マッチングに要する時間が支配的であり，処理時間の増加は検証に要する時間の増加に因るものであった．そこで，Suffix Array の検索結果として得られる候補の数を調べたところ，検索対象データ長に比例して増加してることが分かった．これらのことから，検索対象長に比例して候補の数が増加することで，検証に要する時間も比例して増大し，全体の処理時間を増加させていると考えられる．

5. ま と め

本稿では，Suffix Array を用いて音素単位の音声検索を高速に行う手法を提案し，評価実

Vol.2009-SLP-77 No.3
2009/7/17

験から精度の高い検索結果を短時間でユーザに提示できることを示した．これは反復深化探索アルゴリズムの導入に適した特性であると言える．また，検索キーワード長に対して処理時間が大きく増大する問題点を，キーワードの分割検索により解決した．実験により，一般的な連続 DP マッチングと比較して高速に検索できることを確認した．さらに実験により，検索対象データ長が大きくなると分割検索において検証に要する時間が支配的になり，処理時間が検索対象に比例することが分かった．

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の検討，連続 DP マッチングとの組み合わせの方法を考えたい．

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) 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).
4) 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).
5) K.Oflazer : Error-tolerant Tree Matching, COLING-96: The 16th International Conference on Computational Linguistics, pp.860-864 (1996).
6) 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).
7) 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).
8) 毎日新聞社 : CD-毎日新聞データ集 (1997-2000).
9) 工藤 拓 : MeCab : Yet Another Part-of-Speech and Morphological Analyzer, http://mecab.sourceforge.net/ (2008).