論文 / 著書情報
Article / Book Information

論題(和文)	WFST音声認識デコーダの開発とその性能評価
Title(English)	
著者(和文)	大西 翼, ディクソン ポール, 古井 貞煕
Authors(English)	Oonishi Tasuku, Paul Dixon, SADAOKI FURUI
出典(和文)	情報処理学会 研究報告, Vol. , No. 2007-SLP-68, pp. 1-6
Citation(English)	, Vol. , No. 2007-SLP-68, pp. 1-6
発行日 / Pub. date	2007, 10
権利情報 / Copyright	

WFST音声認識デコーダの開発とその性能評価

大西 翼† ディクソン ポール† 古井貞熙†

†東京工業大学大学院 情報理工学研究科 計算工学専攻

〒 152–8552 東京都目黒区大岡山 2–12–1–W8–77

Email: {oonishi, dixonp, furui}@furui.cs.titech.ac.jp

本稿では，実用的な音声認識デコーダの実現に向けて東京工業大学で開発が行われている，WFSTを利用した音声認識デコーダについて，概要とその性能について述べる．本デコーダでは，スケーラビリティを向上させるために，省メモリ化としてon-the-fly合成とdisk-based search，高速化として，GPUを利用した音響尤度計算の実装が行われている．この他にも，実用化に向けた様々な機能が実装されている．これらについての詳細を述べる．また，WFST音声認識で問題となるメモリ消費量の増大を解決するために，本デコーダで行われている省メモリ化について，CSJを利用して性能評価を行った。その結果，on-the-fly合成を行うことで最大で60%以上のメモリ消費量の削減をまたdisk-based searchを行うことで最大で60%以上のメモリ消費量の削減を確認した．さらに，これらのアプローチを組み合わせることで，すべてのWFSTを事前に合成した場合と比較して，80%程度のメモリ消費量の削減を確認した．これらの実験により，本デコーダの省メモリ化についてのアプローチの有効性を示した．

Development and evaluation of a WFST-based speech decoder

Tasuku Oonishi, Paul R. Dixon, and Sadaoki Furui

Department of Computer Science, Tokyo Institute of Technology

2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8552 Japan

Email: {oonishi, dixonp, furui}@furui.cs.titech.ac.jp

This paper presents an overview of the Weighted Finite State Transducer (WFST) based speech decoder being developed at Tokyo Institute of Technology and illustrates the performance via evaluations on the Corpus of Spontaneous Japanese. The decoder has a rich feature set including on-the-fly composition, disk-based search and a new method for accelerating acoustic likelihood calculations using graphics hardware. To provide flexibility there is a highly configurable front-end, batch or live operating modes and lattice generation. Experiments were conducted to evaluate the memory consumption in various configurations. By using either on-the-fly composition or a disk-based search network a memory reduction of more than 60% was achieved. Furthermore, a combination of these techniques with additional factoring of the WFST reduced the memory consumption by over 80%.

1 はじめに

東京工業大学では，経済産業省「情報家電センサー・ヒューマンインターフェイスデバイス活用技術開発・音声認識基盤技術」プロジェクトの一環として，高精度な音声認識デコーダの開発を行っている．本プロジェクトには，複数の大学・企業が参加しており，各機関で音声認識技術の実用化に向けて，音声分離や音声／非音声の判別，言語モデル改良などの要素技術の研究開発を行っている．したがって，我々が担当するデコーダには，これらの様々な技術と柔軟に連携ができるような設計が求められている．

一方，近年，WFST(Weighted Finite State Transducer)を利用した音声認識手法の研究が行われ，高速で高性能な音声認識手法として注目されている [1]．これまでに，Juicer [2] やSOLON [3] などのWFSTに基づくデコーダの開発が行われ，その有効性が報告されている．WFSTに基づく音声認識では，探索に先だち，音響モデルや言語モデル，単語発音辞書などの構成要素を合成してまとめあげ，一つの巨大なWFST形式のネットワークを構築する．認識はこのネットワークを探索することで進められる．従来の認識手法に比べて，探索ネットワークを保持するためのメモリ量を多く必要とする反面，モデルの融合が事前に行われるこ

とから，探索時に動的なモデル融合を行う必要がなく，高速なデコーディングが可能となる．また，様々な形式のモデルや辞書を扱う必要が生じても，最終的に WFST の形式でネットワークに変換できれば，モデルに応じてデコーダ自体を変更するといった必要がないため，柔軟なデコーダが実現できる．

このような観点から，我々も WFST を利用したデコーダの開発を行っている．その初期段階として，スケーラビリティーの向上を狙い，省メモリ化や高速化などのための，様々な機能の実装と検討を行っている．本稿では，本デコーダの概要と，取り入れた機能の説明，それらの性能の評価結果について報告を行う．

2 WFST を利用した音声認識

WFST(Weighted Finite State Transducer) とは，与えられた入力記号列に対して状態遷移を繰り返し，それに対応した出力記号列と重みを出力する有限状態オートマトンの一種である．

WFST を利用した音声認識では，まず音響モデルや言語モデル，単語発音辞書などをそれぞれ個別に WFST の形式に変換する．次に，基本演算の一つである合成 (composition) 演算を施して WFST 同士をまとめ，複数のモデルを組み込んだ一つの WFST を生成する．合成に際して，最小化 (minimization) や決定化 (determinization) などの演算を施すことにより，すべてのモデルが考慮されたネットワーク全体に対して最適化が行われ，効率的な探索ネットワークを生成することができる．これにより，高速で高精度な音声認識を実現することができる．

代表的な大語彙連続音声認識を例にすると，探索ネットワークは以下のような 4 つの WFST を合成して構築される．

H: HMM の状態から文脈依存音素への WFST
C: 文脈依存音素から文脈非依存音素への WFST
L: 文脈非依存音素から単語への WFST
G: 単語から単語 N-gram への WFST

合成演算を $\circ$ で表現すると，全ての WFST を合成して構築された探索ネットワークは，以下のように表現される．

$$H \circ C \circ L \circ G \tag{1}$$

3 東京工業大学 WFST デコーダ

入力音声は，フロントエンドを通して特徴ベクトルに変換され，デコーディングに利用される．探索は，フレーム同期型の 1 パス探索であり，第一位仮説からの尤度差と保持仮説数の上限値を用いた枝刈りを行っている．認識結果は 1-best や単語ラティス形式で出力する．認識結果の出力方式には，探索の終了時に一度に出力を行う「バッチ型」と，探索途中で確定した単語列を順次出力する「逐次型」を選択することができる．

省メモリ化の対策として，on-the-fly 合成 [4, 5, 6]，disk-based search [7]，の 2 つについて実装を行った．高速化への対策としては，音響尤度計算の高速化を狙って，GPU (Graphics Processing Unit) の利用について検討した．

以降では，実装を行った，フロントエンド設計，逐次デコーディング，省メモリ化手法，高速化手法，ラティス生成について順に詳しく述べる．

3.1 柔軟なフロントエンドの設計

本デコーダでは，Sphinx [8] で利用されている，多段フィルタによるフロントエンド設計を採用している．例えば，入力音声は，「窓掛け」や「FFT」などの個別の処理フィルタに順次通されることで，MFCC などの特徴ベクトルへ変換される．

ユーザは，利用目的に応じて設計した処理フィルタを，容易に取り入れることができる．例えば，動画から画像特徴量への変換フィルタを作成することで，デコーダを動画認識やマルチモーダル音声認識に利用することが容易にできる．

3.2 逐次デコーディング

音声への字幕付与などのアプリケーションでは，発話から単語列確定までの遅れ時間の短縮が非常に重要になる．そのため，全ての発話が終了した後に最尤となる単語列を確定するのではなく，発話途中で早期に単語列を確定することが求められる．

この早期に単語列を確定する手法として，保持している複数の仮説の単語履歴に対し，履歴中の先頭からの部分単語列が共通になった段階で，その単語列を出力する手法 [9]，過去の仮説単語履歴と比較する手法 [10]，推定された無音区間毎に単語列を出力する手法 [11] などがある．本デコーダでは [9] の手法に基づいて逐次音声認識を行っている．この手法について簡単に説明する．

図 1 に，共通する単語履歴の検出に基づく早期仮説確定の例を示す．図では，時刻 $t = 0 \sim 3$ における探索を示しており，黒丸は探索ネットワーク中の各時刻におけるアクティブな状態を表している．したがって，図は $t = 3$ の時点で 2 つの仮説が生き残っている状態を示している．$t = 0$ から 2 の太線のパスは，$t = 3$ で生き残っている全ての仮説の共通の履歴となっており，この共通の単語履歴は $t = 3$ 以降の探索でも変化しない．そこで，このような共通履歴部分が確定した段階で，該当する単語系列の出力を行う．この手法は，最終的な認識結

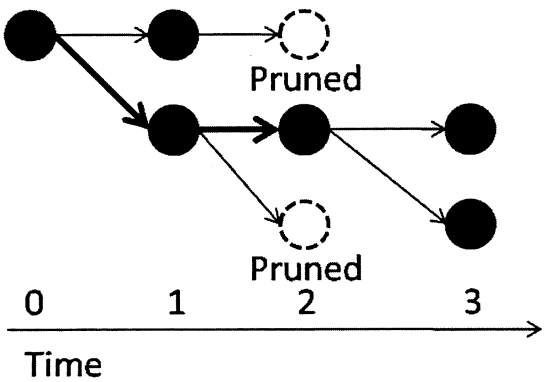

図 1: 共通履歴の検出による早期仮説確定

果に影響を与えないため，認識率の低下を起こさずに，早期に単語列を確定することが可能である．

3.3 省メモリ化

WFST による音声認識では，肥大化した探索ネットワークの読み込みに伴う，メモリ使用量の増大がしばしば問題となる．その対策として，1) 事前のネットワーク構築の段階ですべての WFST を合成せず，一部の WFST については，探索中に動的に合成するようにして，読み込む探索ネットワークの肥大化を防ぐ手法 (on-the-fly 合成 [4, 5, 6])，2) 認識時に探索ネットワーク全体をメモリ上に読み込むのではなく，ディスク上に展開しておき，必要分だけを随時メモリ領域に読み込んで利用する方法 (disk-based search [7]) の 2 つについて検討を行った．

3.3.1 On-the-fly 合成

どの部分の WFST を事前の探索ネットワークの合成に利用するかによって，様々な方式が考えられるが，本稿では，以下の 2 つの式で表される合成について検討を行った．式中の括弧で示される部分が事前に合成される WFST である．

$$(H \circ C \circ L_{uni}) \circ G_{tri/uni} \quad (2)$$

$$H \circ (C \circ L \circ G) \quad (3)$$

L_{uni} は単語辞書に unigram 確率を付与した WFST，$G_{tri/uni}$ は trigram 確率を unigram 確率で割った値を持つ WFST である．

式 (2) の方式は Dolfing らの研究に基づいている [4]．この方式では，unigram の確率を付与した WFST を，事前に構築した静的な探索ネットワーク中に組み込むことで，早期に単語の統計量を探索に利用することができ，先読みの効果を付与することができる．探索は，到達した状態において，次に遷移する状態や入力記号列などを合成演算に基づいて動的に生成することで行われる．この方式ではメモリ使用量を少なく抑えられる反面，合成演算を状態ごとに行うため，その計算に伴うオーバヘッドが問題になる可能性がある．

式 (3) では，HMM のトポロジーが反映されている H の部分を探索時に動的に融合する．ただし，本デコーダでこの合成方式を用いる場合には，HMM としてスキップなしの left-to-right 型を扱うこととした．このような制限を与えると，H の WFST は各文脈依存音素が，一本のパスで表現されるような非常に簡単な構造となり，合成は「H 上の該当する音素のパスを事前に合成されたネットワークに当てはめる」という操作のみで行うことが可能である．これは一般的にデコーダで利用されている「ネットワーク状態遷移の動的な展開」と同じ処理を行うことにより実現できる．したがっていわゆる合成演算を必要としない．このため，式 (2) の説明中にあるような計算に伴うオーバヘッドを削減する効果が期待できる．

3.3.2 Disk-based search

Willett らは，WFST を利用した音声認識における省メモリ化の対策として，探索ネットワーク全体をディスク上に展開し，探索で到達した状態ごとに，必要となる情報のみをメモリ上に読み出す手法の提案を行っている [7]．我々も同様の処理の実装を行った．

我々の実装では，状態ごとに，「次の遷移先の状態番号」「各遷移に伴う入力記号，出力記号，スコア（重み）」などをメモリ上に読み込み，探索に利用する．探索時にこれらの情報を素早くディスク上のネットワークから読み込むため，「状態番号」と「その情報が格納されているディスク上の位置」の関連を表すテーブルを作成し，探索時に利用している．探索が進んで不要となった情報は，メモリ上から消去し，探索の進展に伴って増加する占有メモリ領域の増加を抑えている．

3.4 高速化

混合ガウス分布を音響モデルとして利用する音声認識では，ガウス分布の混合数の増加に伴い，音響尤度計算に多くの時間を要する．このため音響尤度計算を効率的に行うことは，高速な音声認識に非常に重要である．

近年，グラフィックスカードに搭載されたGPU(Graphics Processing Unit) の浮動小数点速度が， CPU のそれと比較して飛躍的に向上しており，将来的には，汎用的な計算プロセッサとしてGPU が広く利用されることが予想される．

我々は高速に音響尤度を計算する一つのアプローチとして，GPU を利用した音響尤度計算手法を提案し，その実装を行った．このアプローチでは，高速な演算ユニットを利用して正確に音響尤度計算を行うため，近似的な計算により計算量を削減するアプローチと違い，認識率の劣化なしに高速に音響尤度を計算することができる．これにより，最大で 25%程度の認識時間が削減されることを確認した [12].

3.5 ラティス生成

本デコーダは音声検索，リスコアリング処理を利用したアプリケーションなどとの親和性を高めるため，ラティス形式での出力を行うことを可能にしている．ラティスは WFST 形式であり，仮説展開の際に同時に生成される．ラティスを構成する要素単位は，事前に合成されたネットワークに依存しており，例えば，$H \circ C \circ L \circ G$ を用いて探索を行った場合には，HMM の状態を構成要素とした単語ラティスが得られ，$H \circ (C \circ L \circ G)$ では，(文脈依存) 音素を構成要素とした単語ラティスが得られる．なおラティス生成は， [13] と同様の手法で実装を行った．

4 性能評価実験

本デコーダの機能の中で，省メモリ化に焦点を当て性能評価実験を行った．

実験には，日本語話し言葉コーパス (Corpus of Spontaneous Japanese) [14] を用いた．学習用データとして，音響モデルには 967 学会講演，言語モデルには学会講演と模擬講演の計 2,682 講演を用いた．音響特徴量には，フレームシフト 10ms, 分析窓幅 25ms の MFCC 12 次元+Δ MFCC 12 次元+Δ Δ MFCC 12 次元+Δ 対数パワー+Δ Δ 対数パワーの計 38 次元を用いた．音響モデルには 3,000 状態 16 混合の triphone HMM を用い，言語モデルには語彙サイズ 55,000 単語の trigram を用いた．

評価データには，男性の学会講演 10 講演 (perplexity = 65.9, 未知語率= 0.52%) を用いた．実験は Intel Xeon 1.86GHz 2GB メモリの計算機を利用した．

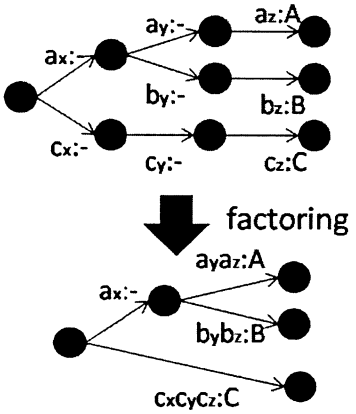

図 2: factoring を行った WFST の例

4.1 on-the-fly 合成の効果

表 1 に使用したネットワーク形態ごとの平均メモリ使用量を，図 3 に認識時間と単語正解精度との関係を示す．認識時間は RTF (Real Time Fatcor) で示す．

結果中に fac とあるのは，WFST のネットワークのサイズを削減する処理である factoring [1] を施したことを示している．factoring とは，ある状態とある状態を結ぶパス中に複数の状態が存在する状況を考えたときに，1) そのパス中の全ての状態が，そのパス以外の状態との遷移経路を持たず，しかも，2) パスから出力される記号が唯一である，という条件を満した場合に，該当パスを一つの状態遷移として新たに定義して，ネットワーク内の状態数を削減する処理である．WFST に factoring を行った場合の例を図 2 に示す．

表 1 の上段は，factoring を行わなかった場合のメモリ消費量を示しており，on-the-fly 合成 を行っていない場合 ($H \circ C \circ L \circ G$)，式 (2) に基づく on-the-fly 合成を行った場合 ($(H \circ C \circ L_{uni}) \circ G_{tri/uni}$) 式 (3) に基づく on-the-fly 合成を行った場合 ($H \circ (C \circ L \circ G)$) をそれぞれ示している．どちらの on-the-fly 合成を用いた場合においても 50% 以上メモリ消費量が削減されていることが分かる．表の下段は，それぞれの手法に対して，事前に合成した静的なネットワークの部分を factoring した場合の結果を表している．on-the-fly 合成を用いていない $H \circ C \circ L \circ G$ の場合であっても，factoring により，大きくメモリ消費量が削減されていることが分かる．on-the-fly 合成を factoring と併用した場合には，factoring を行わない場合に比べて効果は小さくなるが，組み合わせの効果が得られ，ベースラインの $H \circ C \circ L \circ G$ に比べて， 60% 以上のメモリ使用量の削減を達成した．

図 3 の認識性能の結果を見ると，式 (2)

表 1: on-the-fly 合成によるメモリ消費量削減の効果（上段: factoring なし，下段: factoring あり）

Network type	Memory usage (MB)
$H \circ C \circ L \circ G$	560
$(H \circ C \circ L_{uni}) \circ G_{tri/uni}$	260
$H \circ (C \circ L \circ G)$	250
fac$(H \circ C \circ L \circ G)$	260
fac$(H \circ C \circ L_{uni}) \circ G_{tri/unit}$	190
$H \circ$fac$(C \circ L \circ G)$	220

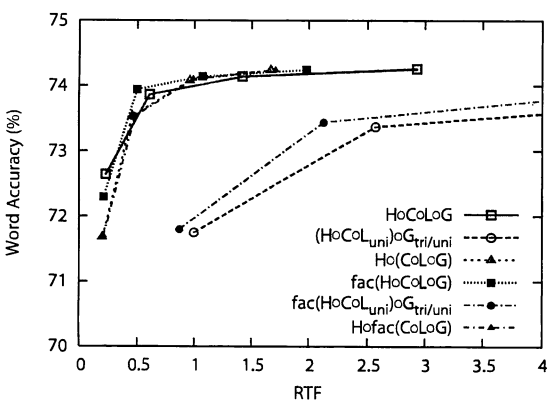

図 3: 動的な合成を行った場合の認識率と認識時間との関係

に基づく on-the-fly 合成を行った場合 ($(H \circ C \circ L_{uni}) \circ G_{tri/uni}$) には，ベースラインの $H \circ C \circ L \circ G$ に比べ，認識時間が多くかかっており，RTF が 4 となってもまだ最高値に達していない様子が分かる．これは，動的な合成に伴う合成演算のオーバヘッドの影響が大きいためであると考えられる．一方，式 (3) に基づいて H を動的に合成した場合には，認識時間の増加はほとんど見られず，収束時の認識性能の違いもほとんど見られなかった．これは H の構造を制限して合成時の計算を簡略化した効果により，オーバーヘッドに伴う計算時間の増加を抑えて，WFST の合成を行うことができたためであると考えられる．

以上より，今回の実装では，式 (3) に基づく $H \circ (C \circ L \circ G)$ の on-the-fly 合成が効果的であり，性能の劣化なくメモリ使用量を削減することができた．また，factoring と併用することで，さらにメモリ使用量を抑えることができることが示された．

表 2: Disk-based search によるメモリ消費量削減の効果

Netowork type	Memory usage(MB)
fac$(H \circ C \circ L \circ G)$	260
$H \circ$fac$(C \circ L \circ G)$	220
fac$(H \circ C \circ L \circ G)$ + disk	130
$H \circ$fac$(C \circ L \circ G)$ + disk	100

4.2 Disk-based search による効果

前節の結果でメモリ消費量の削減が大きかった fac$(H \circ C \circ L \circ G)$ と $H \circ$fac$(C \circ L \circ G)$ のネットワークについて，disk-based search を行った場合の効果について検討した．表 2 の上段は，disk-based search を利用しなかった場合のメモリ消費量を示している．下段は，fac$(H \circ C \circ L \circ G)$ のネットワークに Disk-base search を行った場合，H を動的に合成するネットワークを用いて disk-based search を行った場合のメモリ使用量を示している．これから，disk-based search を行うことにより，fac$(H \circ C \circ L \circ G)$ のネットワークを用いた場合で 50%程度，H を動的に合成するネットワークを用いた場合で 60%程度のメモリが削減されていることが分かる．

図 4 を見ると，fac$(H \circ C \circ L \circ G)$ のネットワークを用いて disk-based search を行うことで，認識率の収束値付近に到達する時間が 30%程度増加していることが分かる．同様に H を合成した場合も，同程度の認識時間の増加が見られる．これは，ディスクアクセスに伴うオーバヘッドが発生したためであると考えられる．

以上より，今回の実装では disk-based search を利用することで，最大で 60%程度のメモリ使用量が削減できることが確認できた．また，H を動的に合成するネットワークを用いることで，更なる省メモリ化を実現し，全てのネットワークを事前に合成した場合と比べ，80%以上のメモリ消費量が削減できることを確認することができた．

5 まとめ

本稿では，東京工業大学で開発が行われている，WFST を利用した音声認識デコーダの概要とその性能の評価を行った．その中で，本デコーダで利用されている省メモリ化，高速化などの種々のアプローチを紹介した．具体的には，省メモリ化として，on-the-fly 合成と disk-based search について，高速化として GPU を利用した音響尤度計算法について，また柔軟なフロントエンド設計や逐次デコーディング，ラティス生成について述べた。

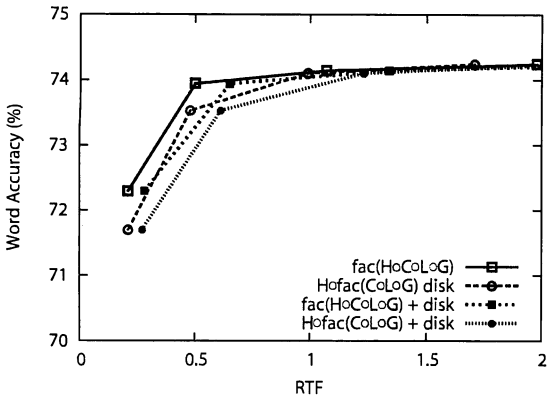

図 4: Disk-based search を行った場合の認識率と認識時間の関係

WFST 音声認識で問題となるメモリ消費量の増大を解決するために本デコーダで行われているアプローチについて，CSJ を利用して性能評価を行った。その結果，on-the-fly 合成を用いることで最大 60%以上のメモリ消費量の削減を，また disk-based search を行うことで，最大で 60%以上のメモリ消費量の削減を確認することができた．さらに，それらを組み合わせることで，全ての WFST を事前に合成した場合と比べて，80%以上のメモリ消費量の削減を確認することができた．これにより，本デコーダの省メモリ化についてのアプローチの有効性を示した．

今後は on-the-fly 合成の効率的実装及びアルゴリズムの検討を行うと共に，本プロジェクト内の他テーマ技術との連携を行い，実用的な音声認識デコーダの実現に向けた検討を進めていきたい．

謝辞

本研究は経済産業省「情報家電センサー・ヒューマンインターフェイスデバイス活用技術開発・音声認識基盤技術」プロジェクトの一環として行った．

参考文献

[1] M. Mohri et al., "Weighted finite-state transducers in speech recognition," Computer Speech and Language, vol.16, no.1, pp.69–88, 2002.

[2] D. Moore et al., "Juicer: A weighted finite-state transducer speech decoder," Proc. MLMI'06, 2006.

[3] T. Hori, "NTT Speech recognizer with Out-Look On the Next generation; SOLON," Proc. Communication Scene Analysis, 2004.

[4] H. J. G. A. Dolfing and I. L. Hetherington, "Incremental language models for speech recognition using finite-state transducers," Proc. ASRU, 2001

[5] T. Hori et al., "Generalized fast on-the-fly composition algorithm for WFST-based speech recognition," Proc. Interspeech, pp. 847–850, 2005.

[6] D. A. Caseiro et al., "A specialized on-the-fly algorithm for lexicon and language model composition," IEEE Transactions on Audio, Speech, and Language Processing, vol.14, no.4, pp. 1281–1291, 2006.

[7] D. Willett et al., "Time and memory efficient Viterbi decoding for LVCSR using a pre-compiled search network," Proc. Eurospeech, pp.847–850, 2001.

[8] P. Lamere et al., "Design of the CMU sphinx-4 decoder," Proc. ICSLP, pp.1181–1184, 2003.

[9] P. F. Brown et al., "Partial traceback and dynamic programming," Proc. ICASSP, pp.1629-1632, 1982.

[10] 今井　亨 他, "最ゆう単語列逐次比較による音声認識結果の早期確定," 電子情報通信学会論文誌 D-II, vol.J84-D-II, no.9, pp.1942–1949, 2001.

[11] 河原達也 他, "話し言葉音声認識のための言語モデルとデコーダの改善," 情報処理学会研究報告, vol.2001, no.55, pp.15–22, 2001.

[12] ポール ディクソン 他, "WFST を用いた音声認識デコーダの機能拡張," 秋季音響論, pp.105–106, 2007.

[13] A. Ljolje et al., "Efficient general lattice generation and rescoring," Proc. Eurospeech, pp.1251–1254, 1999.

[14] K. Maekawa et al., "Corpus of spontaneous Japanese: Its design and evaluation," Proc. ISCA and IEEE Workshop on Spontaneous Speech Processing and Recognition, pp.7–12, 2003.