論文 / 著書情報
Article / Book Information

論題(和文)	WESTデコーダにおける on-the-fly 合成の高速化に関する検討
Title(English)	Fast on-the-fly composition for a WEST-based decoder
著者(和文)	大西 翼, ディクソン ポール, 岩野 公司, 古井 貞煕
Authors(English)	Tsubasa Oonishi, Paul R. Dixon, Koji IWANO, Sadaoki FURUI
出典(和文)	日本音響学会2008年秋季講演論文集, , No. 1-1-14, p. 33-34
Citation(English)	, , No. 1-1-14, p. 33-34
発行日 / Pub. date	2008, 9

WFST デコーダにおける on-the-fly 合成の高速化に関する検討*

◎大西 翼, ディクソン ポール, 岩野 公司, 古井 貞熙 (東工大)

1 はじめに

近年，新しい音声認識の枠組みとして Weighted Finite State Transducer(WFST) を用いた音声認識手法 [1] が提案されている．これを利用した音声認識では HMM などのモデルの情報をすべて WFST の形式で表現し，それらを合成演算により一つに合成することで探索ネットワークを構築する．そのため探索ネットワークが肥大化し，認識時に大きなメモリ量が必要となる場合がある．またモデルの情報を変更する際には探索ネットワークを再構築する必要があるためモデル変更に伴う大きなオーバヘッドが発生する．

この問題を解決するために認識時に合成演算を行うことで動的に探索ネットワークを生成する「on-the-fly 合成」手法が提案されている [2]．しかしながら，on-the-fly 合成を用いて生成される探索ネットワークには十分な最適化が施されていないため認識速度が低下するといった問題がある．過去には Caseiro らによって合成時に最適化処理を行うことで効率的な探索ネットワークを生成する手法が提案されている [3]．そこで我々は Caseiro と同様に合成時に探索ネットワークの最適化処理を行う on-the-fly 合成手法を，現在開発を進めている WFST デコーダ「T^3 Decoder(Tokyo Tech Transducer-based Decoder)」 [4] に実装し，その性能評価を行った．本論文では，その結果について報告する．

2 WFST 音声認識におけるネットワーク構築

WFST を利用した音声認識では，利用するモデル情報 (HMM，単語辞書，N-gram など) をそれぞれ WFST の形式で表現する．次にそれらの WFST に合成演算を施すことで，すべてのモデル情報を組み込んだ一つの WFST を生成し，それを探索ネットワークとして利用する．通常の大語彙連続音声認識では，HMM，トライフォン，単語辞書，N-gram の情報が WFST として表現される．それらを H，C，L，G として表し，合成演算を $\circ$ と表すと探索ネットワークは $H \circ C \circ L \circ G$ で表現される．

3 on-the-fly 合成の高速化

on-the-fly 合成を行う場合には，すべての状態を一度に生成するのではなく，探索に必要な状態のみを随時生成する．そのため最小化演算など WFST に含まれるすべての状態の情報を利用した最適化処理を on-the-fly 合成時に実行することは不可能となる．過去に Caseiro によって on-the-fly 合成時に実行可能な最適化処理として「dead-end 状態の回避処理」,「dynamic pushing」などが提案されている [3]．以下では，本稿で行う「dead-end 状態の回避処理」,「dynamic pushing」について述べる．

3.1 dead-end 状態の回避処理

二つの WFST L および R を合成演算 [1] により合成した場合，生成される WFST $L \circ R$ の状態には，非最終状態で，かつ遷移先の状態が一つも存在しない状態 (dead-end 状態) が生成される場合がある．dead-end 状態は，最終状態への最短パスの探索に不要な，無駄な状態である．このため on-the-fly 合成時に dead-end 状態の生成を回避する手法が提案されている [3]．この手法では事前に WFST L の各状態について，将来出力される記号集合 (先読み記号集合) を求める．on-the-fly 合成時には，この先読み記号集合と WFST R の各状態の入力記号とのマッチングを行うことで，生成される状態が dead-end 状態に到達するかどうかを判定し，無駄な状態の生成を未然に回避している．我々も，これと同様の考えにより dead-end 状態の回避処理を実現している．

3.2 dynamic pushing

pushing は，初期状態に近い状態に重みを再配置する WFST の基本演算の一つである [1]．これにより WFST の重みを探索の早期で利用することが可能となるため，探索の効率化が実現できる．pushing 演算では，まず各状態に対して最終状態への最短パス重みを「先読みスコア」として設定する．そして各状態遷移における開始状態と到達状態の先読みスコアの差を，状態遷移の出力重みに加えることで初期状態に重みを近づける．しかし on-the-fly で合成を行う場合には，ある状態から最終状態への最短パス重みを求めることは不可能である．そのため Caseiro によって on-the-fly 合成時に利用可能な pushing アルゴリズム (dynamic pushing) が提案されている [3]．この手法では L と R の on-the-fly 合成を行う場合の先読みスコアを以下の基準により決定する．1) 合成の対象となる L 上のある状態への遷移の出力記号と，R 上の状態への遷移の入力記号とのマッチングにより合成された状態では，先読みスコアを 0 とする．2) 合成の対象となる L 上のある状態について，その状態への遷移の出力が ϵ 記号の場合には，その状態の先読み記号集合を集める．合成対象となる R 上の状態から次状態への遷移のうち，入力記号がこの先読み記号集合に含まれるものを選び出し，その中の最小重みを，合成した状態の先読みスコアとする．

しかし，この手法では pushing 後の WFST の重みの配置に曖昧性を生じる場合がある．その例を Fig.1 に示す．Fig.1 の各状態上の [ ] 内の数字は，先読みスコアを表す．WFST L_1 と R_1 を合成し dynamic pushing を行う際に，先に (0,0) からの遷移先状態を生成した場合と先に (1,0) からの遷移先状態を生成した場合とで，合成される WFST として (1)，(2) の 2 つが考えられる．これは合成後の状態 (2,0) の先読みスコアが一意に決定されないためである．一方，同じ WFST を filter[1] と呼ばれる WFST と共に合成し，dynamic pushing を行った場合，Fig.2 のような WFST が生成される．この場合，L_1 の出力 ϵ 記号を ϵ_2 に置き換えた WFST $L_1^{'}$ と，R_1 の入力 ϵ 記号を ϵ_1 に置き換えた WFST $R_1^{'}$ を利用する．これから先読みスコアに曖昧性があった状態 (2,0) が (2,2,0),(2,0,0) のように区別され，先読みスコアが一意に決定されることがわかる．我々は，この filter を用いた dynamic pushing を実装している．

4 性能評価実験

実験には日本語話し言葉コーパス (Corpus of Spontaneous Japanese) を用いた．学習用データとして，音

*Fast on-the-fly composition for a WFST-based decoder by Tasuku Oonishi, Paul R. Dixon, Koji Iwano, Sadaoki Furui (Tokyo Institute of Technology)

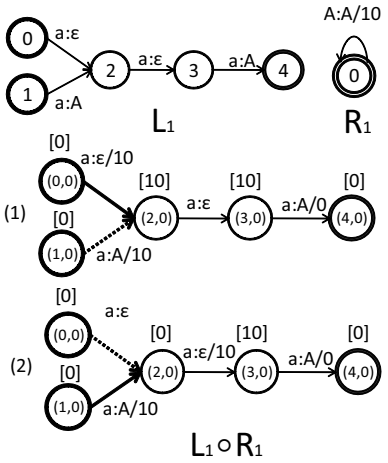

Fig. 1　dynamic pushing により生成される WFST

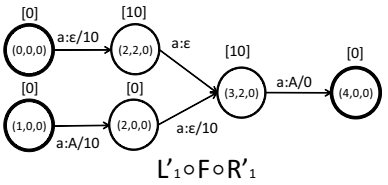

Fig. 2　filter を利用した dynamic pushing

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+ΔΔ MFCC 12 次元+対数パワー+Δ対数パワー+ΔΔ対数パワーの計 39 次元を用いた．音響モデルには 3,000 状態 32 混合の triphone HMM を用い，言語モデルには語彙サイズ 65,000 単語の trigram を用いた．評価データには，テストセット 1 の 10 講演を用いた．実験には Intel Core2 2.4GHz 2GB メモリの計算機を使用した．全ての WFST には factoring 処理 [1] を行った．評価する WFST の組み合わせとして $H \circ C \circ L \circ G$, $(H \circ C) \circ (L \circ G)$, $(H \circ C \circ L) \circ G$, $(H \circ C) \circ L \circ G$ を評価した．() 内の WFST は，事前に合成する WFST を表している．$H \circ C \circ L \circ G$ は，on-the-fly 合成を行わない場合を表し，$(H \circ C) \circ L \circ G$ は，H，C の WFST を事前に合成し，それと L，G の WFST との 2 回の on-the-fly 合成を行うことを表す．

Fig.3 にそれぞれの WFST について dead-end 状態の回避処理を行った場合の認識精度と認識時間 (RTF) の関係を表す．これから，L の WFST と言語制約を表す WFST G を事前に合成する $(H \circ C) \circ (L \circ G)$ の場合には，実時間でほぼ認識率が収束していることがわかる．一方，G を単独で on-the-fly で合成する $(H \circ C \circ L) \circ G$ や $(H \circ C) \circ L \circ G$ の場合には大きな速度低下が発生することがわかる．この場合には言語制約の情報を用いた最適化処理が行われなかったために，仮説の枝刈りを有効に行うことができなかったためであると考えられる．

次にそれぞれの WFST の組み合わせに dynamic pushing の処理を併用した場合の認識性能について Fig.4 に示す．dynamic pushing を併用した場合，$(H \circ C \circ L) \circ G$ や $(H \circ C) \circ L \circ G$ を用いた場合で大幅な認識速度の改善が見られる．

以上より，これらの最適化処理を行うことで，すべての WFST の組み合わせで実時間での認識精度の収束性が確認された．

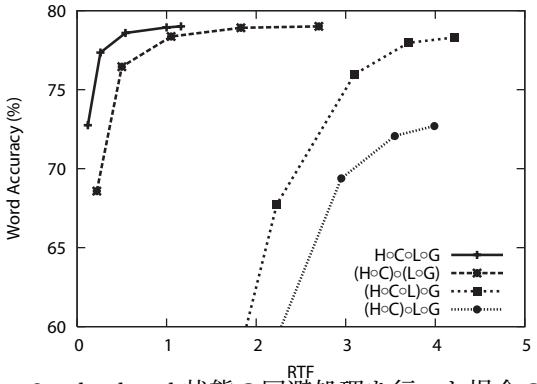

Fig. 3　dead-end 状態の回避処理を行った場合の認識性能

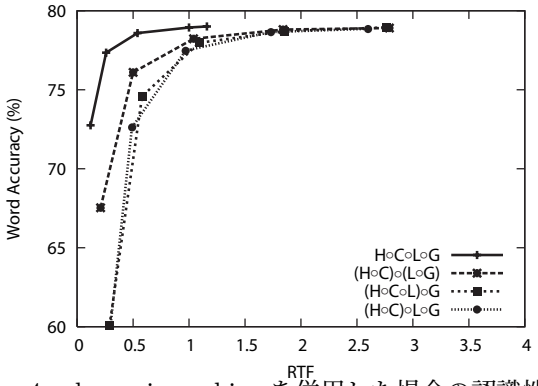

Fig. 4　dynamic pushing を併用した場合の認識性能

5　まとめ

本稿では，高速 on-the-fly 合成を実装し，様々な WFST の組み合わせにおける性能評価を行った．これにより，事前に言語制約を表す WFST と単語辞書を表す WFST を合成する場合には，dead-end 状態の回避処理を行うだけで，実時間で認識精度が収束することを確認できた．さらに，言語制約のモデルを単独の WFST として on-the-fly 合成する場合には，dynamic pushing の処理を併用することで，実時間で認識精度が収束することを確認できた．これからすべての WFST の組み合わせにおいて，on-the-fly 合成時に動的な最適化処理を行うことで，実時間で認識精度が収束することが確認できた．

謝辞　本研究は経産省「情報家電センサー・ヒューマンインターフェースデバイス活用技術開発・音声認識基盤技術」プロジェクトの支援により行った．

参考文献

[1] M.Mohri et al. Computer Speech and Language, vol16, no.1, pp. 69–88, 2002.

[2] H.J.G.A. Dolfing and I. L. Hetherington. Proc. IEEE Workshop on ASRU, pp.194–197, 2001.

[3] D.A.Caseiro et al. IEEE Transactions on Audio, Speech, and Language Processing, vol.14, no.4, pp.1281–1291, 2006.

[4] Paul R. Dixon et al. Proc. IEEE Workshop on ASRU,pp.443–448,2007.