Internal Use Only（非公開）

TR－SLT－0071

連続音声認識システムにおける探索処理の効率化と省メモリ化手法
Efficient Decoding and Memory Size Reduction for Continuous Speech Recognition

小窪 浩明 Hiroaki Kokubo
林 輝昭 Teruaki Hayashi
山本 博史 Hirofumi Yamamoto
菊井 玄一郎 Genichiro Kikui

2004.4.30

概要

ATRの連続音声認識プログラムATRIUMSに実装されている探索処理の効率化，省メモリ化手法について解説する．連続音声認識の仮説探索の高速化，省メモリ化を実現するためには，生成される仮説数の削減が有効である．ATRでは，(1)同音語の言語尤度の共有化，(2)開始時刻の代表処理による先行単語の共有化，(3)枝刈りにより不要となった単語仮説の効率的な削除手法を導入することで，生成される仮説数の削減を図っている．また，木構造辞書に関しては，各ノードに付与する言語スコアに近似手法を導入することでテーブルサイズを削減し省メモリ化を図っている．評価実験では，単語グラフに登録される単語仮説の仮説数削減と木構造辞書のfactoringテーブルのメモリ削減についてその効果を報告する．

（株）国際電気通信基礎技術研究所
音声言語コミュニケーション研究所
〒619－0288「けいはんな学研都市」光台二丁目2番地2 TEL：0774－95－1301

Advanced Telecommunication Research Institute International
Spoken Language Translation Research Laboratories
2-2-2 Hikaridai "Keihanna Science City" 619-0288,Japan
Telephone:+81-774-95-1301
Fax :+81-774-95-1308

©2004（株）国際電気通信基礎技術研究所
©2004 Advanced Telecommunication Research Institute International

目次

1 はじめに

音声認識機能を搭載したカーナビゲーション [1] や携帯型音声通訳の試作機 [2] など，音声インタフェースの応用範囲はしだいに広がっている．今後，携帯型音声通訳機など，小型機器をプラットフォームとした連続音声認識機能のニーズは高くなると予想される．連続音声認識システムは，認識対象語の大語彙化と音響モデル，言語モデルの高精細化により，計算機に求められるリソースの要求水準は高い．一方で，ハードウエアの拡張が容易ではない PDA やカーナビゲーションなどの民生機器では，システムリソースの制約を強く受けることになる．そのため，連続音声認識システムの省メモリ化は重要な研究課題である．

本報では，ATR の連続音声認識システムである ATRIUMS に実装されているデコーダの探索の高速化，省メモリ化手法について報告する．

本報の構成は以下の通りである．第 2 章では ATIRUMS のデコーダの構成について概要を述べ，第 3 章では単語仮説数の削減手法として ATRIUMS が採用している (1) 同音語の言語尤度の共有化手法，(2) 音素環境近似，(3) 単語グラフ内の不要仮説の効率的削除手法について解説する．第 4 章は木構造辞書の各ノードに付与する言語スコアテーブルのデータサイズ削減に関して解説する．第 5 章では，単語グラフに登録される単語仮説の仮説数削減と木構造辞書の factoring テーブルのメモリ削減について評価実験の結果を報告する．第 6 章はまとめである．

2 デコーダの構成

連続音声認識システムのデコーダ [4] について説明する．本デコーダは 2 パス探索である．第 1 パスでは，木構造辞書を用いた単語仮説の探索と，生成された単語仮説を単語グラフへ登録する．この時，先行単語末の発音が同じである単語仮説の開始時刻を一つに絞り込む音素環境近似 [4] を用いることで単語仮説数を削減している．第 2 パスでは，単語グラフに含まれる単語仮説の再評価 (リスコアリング) と pruning を行う．

2.1 木構造辞書

大語彙認識システムでは，辞書を木構造音素ネットワークで表現することが一般的である．木構造表現は，単語の先頭部分の辞書ノードを複数の単語で共有することでノード数を削減し，探索効率を向上させる．対象とするデコーダではさらに，同音異義語を区別せずに木構造辞書を作成することでノード数の削減をはかっている．木構造辞書を使用する場合，単語を共有している語頭部分の辞書ノードで単語が特定できずに言語スコアを適用するタイミングが遅れるのを防ぐため，ノードを共有する単語の中で最大の尤度を言語スコアとして与える factoring をおこなう [5]．言語モデルにクラス bigram を用い，先行単語 v の属する先行クラスを C_v とすると，辞書ノード s における factoring スコアの値は，

$$p(s|C_v) = \max_{w_k \in s} p(w_k|C_{w_k}) \cdot p(C_{w_k}|C_v) \tag{1}$$

となる．ただし，w_k は，ノード s が共有している単語，C_{w_k} は w_k の属するクラスである．

2.2 単語グラフの生成

対象とするデコーダは第 1 パスで木構造辞書を用いた時間同期の探索を行い，生成した単語仮説を単語グラフに登録する．

デコーダは木構造辞書の先頭ノードから終端ノードへ時間同期の遷移を繰り返しながら仮説の展開を進める．終端ノードに到達した単語仮説は単語グラフに登録され，再び辞書の先頭ノードから後続単語仮説の探索を開始する．このとき，仮説数削減を目的とした音素環境近似 [4] を導入している．音素環境近似は，同一時刻，同一辞書ノード，同一 HMM 状態で，かつ先頭音素のアルフォンが同じ単語仮説のうち最大尤度を与える仮説のみを残してそれ以外の仮説を削除することで単語仮説の絞り込を行う．また，仮説数の増加を抑制するために尤度幅一定のビームサーチを行い尤度の低い仮説を棄却する (pruning)．

2.3 単語仮説の再評価

第 2 パスでは，第 1 パスで生成された単語グラフの再評価を行う．第 1 パスでは区別しなかった同音異義語を展開し，言語重みの値を変えて再計算を行う．個々の単語仮説を起点に，接続が許される任意の経路に対して，forward-backward アルゴリズムに基づき，文頭からの累積尤度と文末までの累積尤度をそれぞれ計算し，この単語仮説を含む文頭から文末までの累積尤度の最大値を求める．この累積尤度を用いて尤度幅一定の pruning を行い，単語グラフから尤度の低い仮説を削除する．

3 仮説数の削減手法

デコード過程の効率化には生成される仮説数の削減が有効である．本章では，仮説数の削減を目的とする3つの手法について紹介する．

3.1 同音語の言語尤度の共有化

同音語の言語尤度の共有化 [4] について説明する．通常，木構造辞書を用いる探索の場合，同音語の終端で正確な言語尤度を与えるために，同音の終端を最終音素もしくは null 音素で分岐させていた．同音音素を分岐させた場合の単語仮説の増加は，後続する単語仮説の先行単語に直結し，辞書ノード s に対応する言語尤度

$$Q_L(s,v) = \max_{u \in E_s} P(u|v) \tag{2}$$

の計算量の増加をもたらす．(ここで u は，辞書ノード s から到達可能な単語セット E_s の要素である．) この問題を解決するために，第1パスでは同音語の言語尤度を共有化し，第2パスで正確な値を与える．第1パスでは，先行単語 v が属するクラス $C(v)$ (ここでは発音が同じ単語を同じクラスとしている) と当該単語 w が属するクラス $C(w)$ に含まれる単語対を用いて次式を計算する．

$$Q_L(S(w),v) = \max_{k,l} P(w_k|v_l) \tag{3}$$

ここでは，単語 w_k は単語 w と同じクラス $C(w)$，単語 v_l は単語 v と同じクラス $C(v)$ に属する．この場合，$Q_L(S(w),v)$ は真の値 $P(w|v)$ と同じか大きいので，共有化を行うことによる取りこぼす危険性は少ない．また，第2のパスで共有化を外し正確な言語尤度を与えれば，第1パスにおいて真の値より高い尤度を与えられたために生き残った単語仮説を単語グラフから削除することができる．

3.2 開始時刻の代表処理による先行単語の共有化

文仮説のコンパクトな表現手法として単語グラフが提案されている [3]．単語グラフ生成時の単語仮説削減手法には，同一単語，同一時刻の単語仮説に対して，先行単語が同じであれば開始時刻を一つに絞り込む単語対近似 (word pair approximation) と呼ばれる手法がよく用いられている [3]．ATR では先行単語末の発音が同じである単語仮説の開始時刻を一つに絞り込む音素環境近似 (cross-word context approximation)[4] を提案しているが，ATRIUMS には単語対近似が採用されている．

単語対近似は，同一時刻 t，同一辞書ノード s，同一 HMM 状態 h，同一先行単語 v の単語仮説のうち最尤の単語仮説のみを残してそれ以外の単語仮説を削除することによって，単語仮説の開始時刻を絞り込む．この時，尤度の比較条件は，先行単語 v が同じ場合に限られる．図1に単語対近似の例を示す．

一方，音素環境近似 (cross-word context approximation) は，「先行単語末の発音が同じであれば開始時刻は同じになる」という仮定である．この仮定に基づいて単語末の発音が同じ先行単語を共有化する．音素環境近似は，先行単語の代わりに先行単語末の発音を用いるため，先行単語末の発音が同じであれば，一つの単語仮説に複数の先行単語を持たせることが可能である．音素環境近似のもっとも簡単は実現方法は，先行単語の末尾の音素のみ利用することで

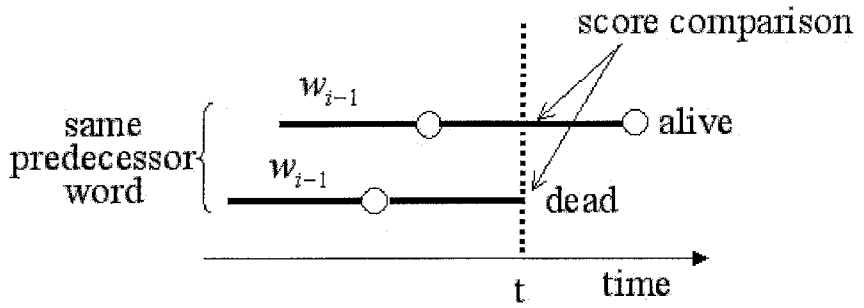

図 1: 単語対近似

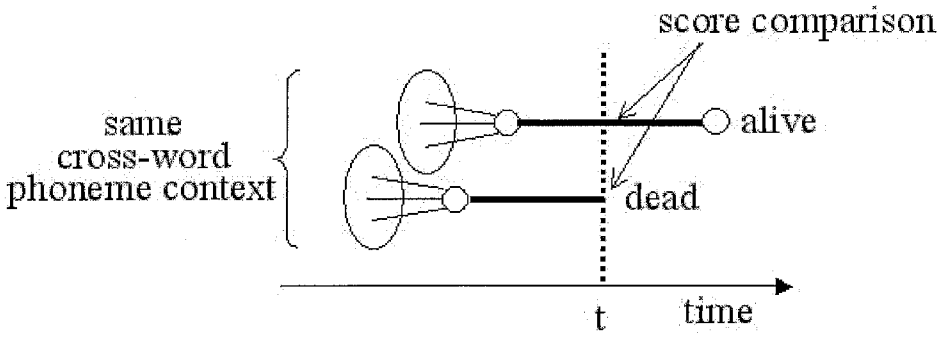

図 2: 音素環境近似

ある．音素環境依存の音響モデルを使用する場合，先行単語の末尾の音素は後続単語の先頭の異音 (allophone) に反映されているので，単語仮説の先頭の異音に応じて先行単語をまとめれば良い．図 2に単語対近似の例を示す．

3.3 単語グラフ内の不要仮説数の削除

不要な単語仮説

単語グラフ内で不要な単語仮説が出現する過程について説明する．ここでは問題を単純化するために，音素環境近似を省略し，辞書ノードに対応する HMM の状態数も 1 と仮定して説明する．

図 3に時刻 t における単語グラフと単語仮説の後続仮説として辞書ノード上に生成された部分仮説を示す．図中，●はアクティブな部分仮説，◎は pruning により棄却された部分仮説を示している．これらの部分仮説は単語グラフ中の先行単語仮説へのリンクを持っている．この例では，単語仮説 **W1** から展開された部分仮説は木構造辞書のノード 2 とノード 4 とノード 5 でアクティブとなっている．また，単語仮説 **W2** から展開された仮説はノード 3 で pruning される．この時，単語仮説 **W1**， **W3** はアクティブな部分仮説からリンクを張られているの

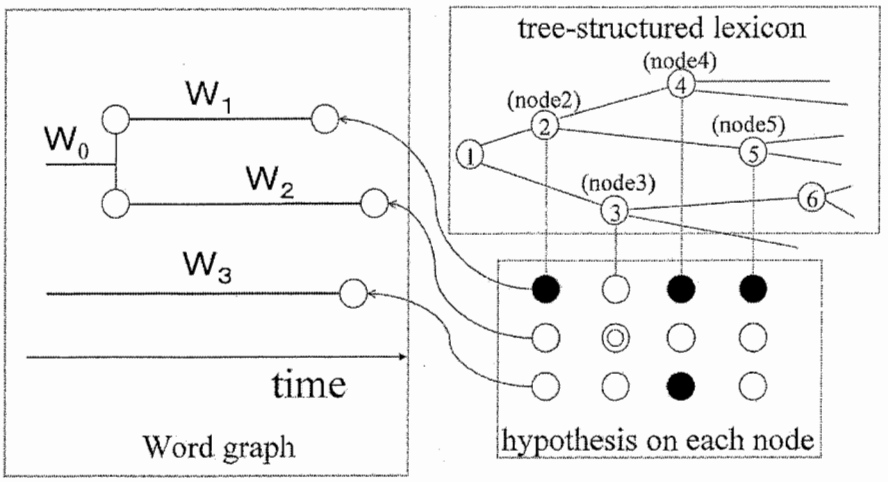

図 3: 単語仮説と後続する部分仮説

に対して，単語仮説 **W2** にはアクティブな部分仮説からのリンクは存在しない．このため，時刻 t 以降に単語仮説 **W2** を先行単語とする単語仮説が生成されることはない．つまり，単語仮説 **W2** は文末まで到達する経路の存在しない不要な仮説とみなすことができる．このような不要な単語仮説を単語グラフより削除し開放されたメモリを再利用 (GC) することで，メモリ空間の効率的な利用を図ることが可能となる．

後続仮説数の管理機能を用いた単語仮説のガーベッジコレクション

文末までの到達経路が存在しない不要な単語仮説を特定するためには，その単語仮説から生成されたすべての部分仮説が棄却されていることを確認しなければならない．文献 [6] では，アクティブな部分仮説から先行単語へとトレースバックすることで，不要な単語仮説を特定し，GC をおこなっている．ただし，処理のオーバーヘッドを軽減するために，トレースバックを伴う GC は 50 フレーム (0.5 秒) 間隔で実行する．

ところで，1 つの単語仮説から展開される後続単語の部分仮説は多数存在しているため，単語グラフ内に存在する単語仮説の数に比べてアクティブな部分仮説の数は圧倒的に多い．したがって，アクティブな部分仮説からトレースバックするよりも，単語仮説自身で後続する仮説数を管理し不要な単語仮説を特定する戦略をとる方がより効率的である．この考えに基づいて，不要となった単語仮説をトレースバックなしに特定し，時間同期で単語グラフから削除する方法について説明する．

時刻 t_i の時点で，単語グラフに登録されている単語仮説と辞書ノード上に生成される部分仮説を図 4に示す．本手法では，単語グラフに生成された各単語仮説に対して，後続する部分仮説数 (**n_hypo**) と単語仮説数 (**n_sucwd**) との 2 つの属性を持たせる．**n_hypo** は分析フレーム同期のカウンタであり，奇数フレームと偶数フレームとで値の符号を反転させる．つまり，符号を除いた絶対値が後続する部分仮説数である．例えば，前フレームで負の値でカウントした場合，現フレームで後続する部分仮説が生成されると，負の値をクリアした後に正の値でカウントを始める．そのため，**n_hypo** の符号が負の値のままである単語仮説は現フレームでアクセスされなかったことが判定できる．また，後続する部分仮説を生成する毎に +1，prun-

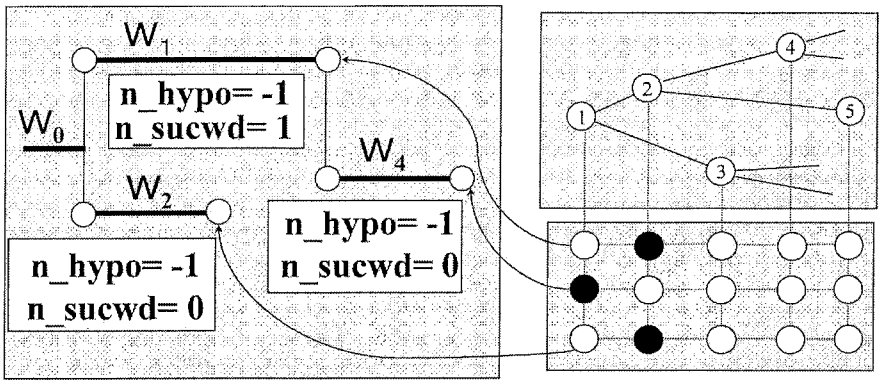

図 4: 属性を付加した単語仮説

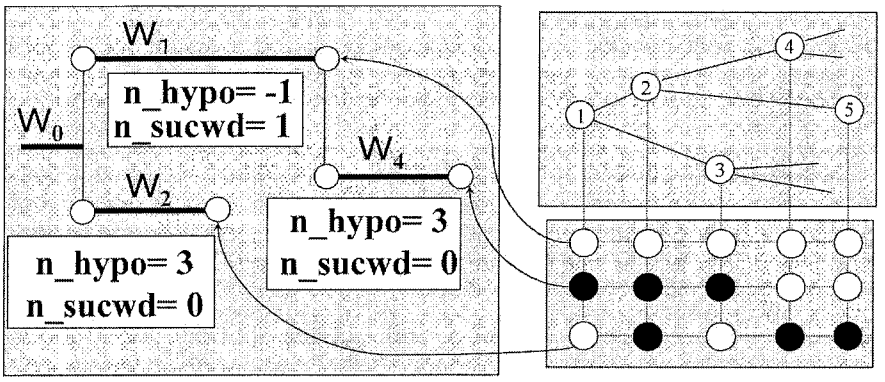

図 5: 部分仮説の展開

ing による部分仮説の棄却毎に -1 と n_hypo の絶対値をカウントしていくことで，生成した後続仮説のうち pruning 後に生き残った仮説数が求められる．つまり，n_hypo の符号が異なるものは，現フレームで参照されなかった単語仮説，n_hypo=0 のものは，後続仮説の生成による参照はあったものの，生成された後続仮説がすべて棄却された単語仮説を示す．このいずれの場合についても，この単語仮説に後続するアクティブな部分仮説は存在しない．これに対し，n_sucwd は分析フレームに非同期であり，後続する単語仮説に変化が生じた際に値を更新する．

図 4の例で，n_hypo は負の値で表現されており，例えば，単語仮説 **W1** には単語仮説が 1 つと，部分仮説が 1 つ後続している．単語仮説 **W4** には，後続する単語仮説はなく，部分仮説が 1 つ後続している．

以下，図 4を現在生成されている仮説の状態であると想定して処理の手順を説明する．

[ステップ 1 : 部分仮説の展開] 図 4で生成されている個々の部分仮説に対し，フレーム同期で仮説の展開が行われる (図 5)．例えば，単語仮説 **W4** を先行単語としノード 1 に生成された部分仮説は，自分自身 (ノード 1) と後続ノード (ノード 2，ノード 3) に仮説を展開す

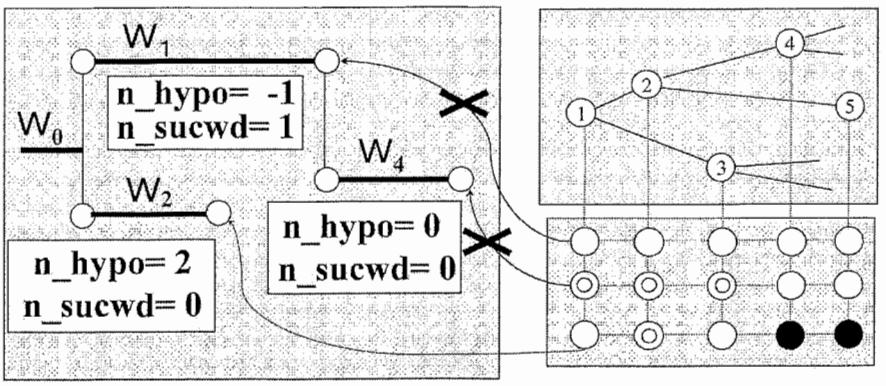

図 6: 部分仮説の pruning

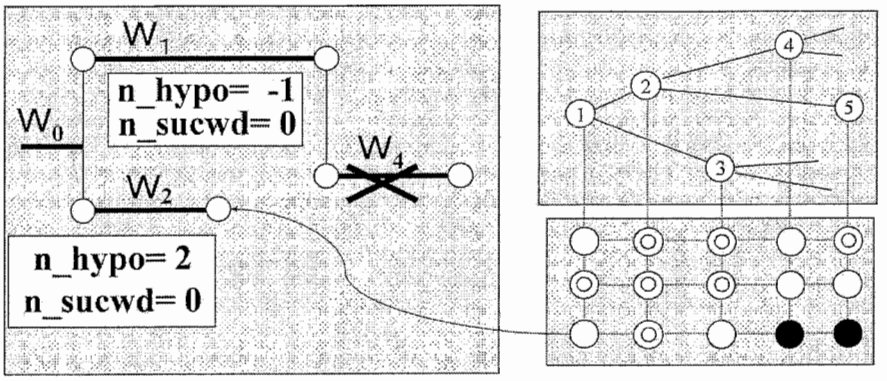

図 7: 不要な単語仮説の削除

る．この時，単語仮説 **W4** のカウンタ **n_hypo** は仮説が展開される毎にその絶対値をインクリメントしていく．ただし，カウンタの符号が異なる場合には，値をクリアした後カウントを開始する (前フレームでは負の値でカウントしているため，現フレームでは正の値となる)．図 5の単語仮説 **W4** の例では，合計 3 つの部分仮説を生成したため **n_hypo=3** となる．一方，展開後の尤度がしきい値を下回る場合には仮説は生成されない．例えば，単語仮説 **W1** を先行単語としノード 2 に生成された部分仮説は，自分自身 (ノード 2) と後続ノード (ノード 4，ノード 5) に仮説を展開することが期待されるが，いずれも展開後の尤度が小さいために仮説生成は行われない．このとき **n_hypo** の符号は負のままであり，現フレームでは仮説生成がなされなかったことがわかる．

[ステップ 2 : 部分仮説の **pruning**] ビームサーチでは，現時点での最大尤度を基準として設定されたビーム幅よりも低い尤度の部分仮説は pruning により削除される．この時，部分仮説が削除される毎に先行単語仮説のカウンタ **n_hypo** の絶対値をディクリメントしていく．図 6の単語仮説 **W4** では，生成した 3 つの部分仮説のすべてが pruning により削除されるため，**n_hypo=0** となり，この単語仮説から後続する部分仮説は存在しない

ことがわかる．一方，単語仮説 **W2** は生成された 3 つの部分仮説のうち 1 つが pruning により削除され，**n_hypo=2** となる．つまり，単語仮説 **W2** を先行単語とするアクティブな部分仮説は 2 つ存在していることが示される．

[ステップ 3：不要な単語仮説の削除] 単語仮説 **W1** は，現フレームでは正の値をとるべき **n_hypo** の符号が負のままである．したがって，現フレームで後続する部分仮説の生成はなかったことを意味し，後続する単語仮説がない場合にはこの単語仮説も削除の対象になり得る．ただし，現時点では後続する単語仮説 **W4** が存在するため削除の対象とはならない．

単語仮説 **W4** は **n_hypo=0** となり，後続する部分仮説は存在しない．更に，後続する単語仮説も存在しない **(n_sucwd=0)** ため，この単語仮説は単語グラフから削除される．この時，単語仮説 **W4** の先行単語仮説である単語仮説 **W1** の **n_sucwd** をディクリメントする．この手続きによって，単語仮説 **W1** の後続単語仮説数は 0 となり，後続する部分仮説も存在しないため，この単語仮説も削除することができる (図 7)．

以上のような処理を行うことにより，単語グラフから不要となった単語仮説をトレースバックなしに削除し，時間同期でメモリ空間を再利用することが可能となる．

4 木構造辞書の省メモリ化手法

単語仮説数の削減に関して，単語辞書の木構造化による単語仮説のマージがよく知られている [5]．この手法は単語辞書を木構造化することで，単語の先頭からの音素系列が共通となる部分までの音響尤度計算を共有化する．この際，共有化された単語の言語尤度のうち最大値を共有仮説の言語尤度に割り当てることで言語尤度の共有化も図られる (factoring)．デコード時に factoring を逐次計算する際の処理量増加を避けるために，factoring の値を予めテーブル化しておく手法が有効である．ただし，共有する単語が異なる辞書ノードは factoring の値も異なるため，factoring の値をあらかじめテーブル化しておくと，多くのメモリを消費してしまうという問題がある．この問題を解決するために，factoring 値を unigram で近似する方式 (unigram factoring)[10] あるいは，アクセス頻度の高い上位レイアのみをテーブル化し，下位レイアについては逐次計算とキャッシングを併用する方式 [11] などが提案されている．ATR では，factoring で計算される bigram スコアに対して深さ依存の unigram 近似および POS(品詞) 等によるクラス bigram を用いた近似 [12] を導入することでテーブルサイズのコンパクト化を図っている．

4.1 factoring テーブルのメモリサイズ

式 (1) で示されるように，factoring スコアの値はノードの持つ共有単語リストに依存する．このため共有単語リストが異なる辞書ノード毎に factoring テーブルを用意する必要がある．

bigram の先行エントリー数を N_f，異なる共有単語リストを持つ辞書ノード数を N_n とし，言語尤度を 4 バイトで表現すると，必要となる factoring テーブルのメモリサイズ M_t(バイト) は

$$M_t = 4 \cdot N_f \cdot N_n \tag{4}$$

で計算することができる．対象とする認識システム (クラス bigram) の例では，$N_f = 700$, $N_n = 15,000$ であった．この条件で必要なメモリは 42MB となる．また，クラス bigram の代りに辞書エントリー数 30,000 の単語 bigram($N_f = 30,000$) を使用した場合を想定すると，必要なメモリ領域は 1.8GB となる．この場合，すべての辞書ノードに対する factoring スコアを予めテーブル化しておく手法は現実的ではない．

4.2 factoring スコアの近似

factoring テーブルのサイズを削減するために，単語が確定する終端ノードを除くすべてのノードで factoring スコア値を unigram で近似する手法 (unigram factoring) が提案されている [10]．unigram factoring では式 (1) を次式で近似する．

$$p(s|C_v) \approx \max_{w_k \in s} p(w_k) \cdot p(C_{w_k}) \tag{5}$$

この近似により，1 ノードあたり先行クラス数分のサイズを必要とした factoring スコアのテーブルをスカラー値として持つことができる．ATR ではこの unigram factoring の拡張として，factoring スコアに対する 2 つの近似手法を導入している．

深さ依存の unigram 近似

木構造辞書では，語頭に近いノードでより多くの単語を共有している．このため従来の unigram factoring で，近似により不適切な pruning がなされた場合，語頭に近いノードほど認識

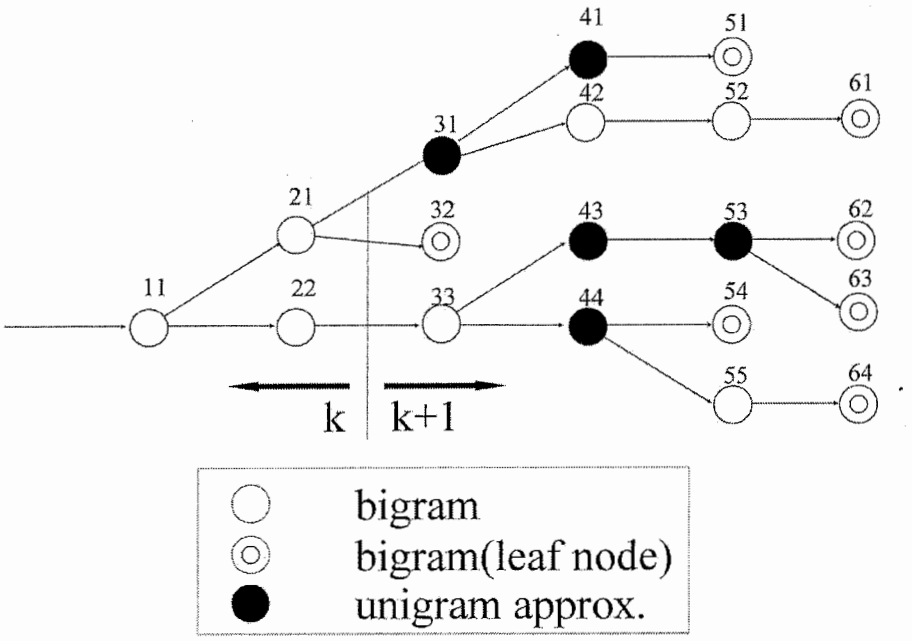

図 8: factoring テーブルの深さ依存 unigram 近似

性能の劣化に及ぼす影響が大きいと予想される．我々の提案手法では，このような性能劣化を最小限に抑えるため，pruning による影響の大きい語頭から k 音素 (階層) 目までのノードと言語スコアが確定する終端ノードは近似を行わないことにする (以下，本方式を深さ依存 unigram 近似と呼ぶ)．

深さ依存 unigram 近似の模式図を図 8に示す．図中◎は終端ノードを表し，●で示すノードに対して unigram の近似を行っている．◎と○で示されるノードは近似を行わない．ノード間の遷移に分岐を持たないノードは遷移先のノードと同じ共有単語を持っているため，factoring テーブルを共有する．例えば，ノード 42 とノード 52 は終端ノード 61 と同じテーブルを共有している．このように，近似を行わない辞書ノードと同じ factoring テーブルを共有するノードに関しては，k 音素以降のノードであっても unigram 近似を行う必要はない．

bf POS bigram による近似

対象としている認識システムでは，クラス数 700 のクラス bigram に基づいて factoring テーブルを作成している．これに対して，クラス数の小さい POS bigram(クラス数 90) から作成した factoring テーブルで置き換えを行う．factoring テーブルのサイズは先行クラス数に比例するため，クラス数に応じたテーブルサイズの削減が期待できる．この際，深さ依存 unigram 近似と同様，終端ノードおよび，終端ノードとテーブルを共有しているノードについては POS bigram による置き換えはしない．

表 1: 実験条件

評価音声	・旅行会話ドメインの 42 片側会話音声 [14] 男性 17 名, 女性 25 名, のべ 551 発話
特徴量	・16kHz サンプリング (16bit) ・フレーム周期 10ms，フレーム長 20ms ・12 次 MFCC と対数パワー，および それらの一次回帰係数 (計 26 次元)
音響モデル	・音素環境依存 HMnet[15] ・1400 状態 5 混合 (男性用モデル) ・1400 状態 15 混合 (女性用モデル)
辞書	・エントリー数 約 32,000
言語モデル	・多重クラス複合 N-gram[13] ・クラス数 700 ・POS(品詞) bigram ・クラス数 90
学習用言語コーパス	・旅行会話文 のべ 1,600,000 語 [14]
デコード	・1 パス 時間同期ビタビサーチ ・2 パス 言語重みを変更したフルサーチ
プラットフォーム	・Linux マシン ・Pentium4 1.5GHz,2GB memory

5 評価実験

これまで述べた手法の有効性を確認するための音声認識実験について報告する．本章では，報告者が行った評価実験のみを述べる．同音語の言語尤度共有化，および単語仮説における単語対近似に関する評価は，文献 [4] を参照されたし．

5.1 実験条件

実験条件を表 1に示す．音響モデルは，文献 [15] で使用した GD(gender dependent) モデルである．音響モデルの混合数は事前実験に基づき，男女それぞれに対し最適な値となっている．実験に使用した辞書のエントリー数は約 32,000，言語モデルはクラス数 700 の多重クラス複合 N-gram[13] である．また，factoring テーブルの近似に使用する POS bigram(クラス数 90) は，多重クラス複合 N-gram と同じ言語コーパス [14] を用いて作成した．評価用音声データには旅行会話ドメインの 42 片対話音声 [14] を用いた．評価音声の平均発話長は 3.6 秒である．実験は Pentium4(1.5GHz) の Linux マシン上でおこない，このマシンで測定した CPU 時間を発声時間で正規化した値を RTF(real time factor) として処理量の指標とした．

5.2 単語仮説のガーベッジコレクション

不要となった単語仮説数に対するガーベッジコレクション (GC) の効果を確認するために行った評価実験について説明する．

1 発話のデコードにおいて生成される単語仮説数の時間推移を図 9に示した．横軸は発話経

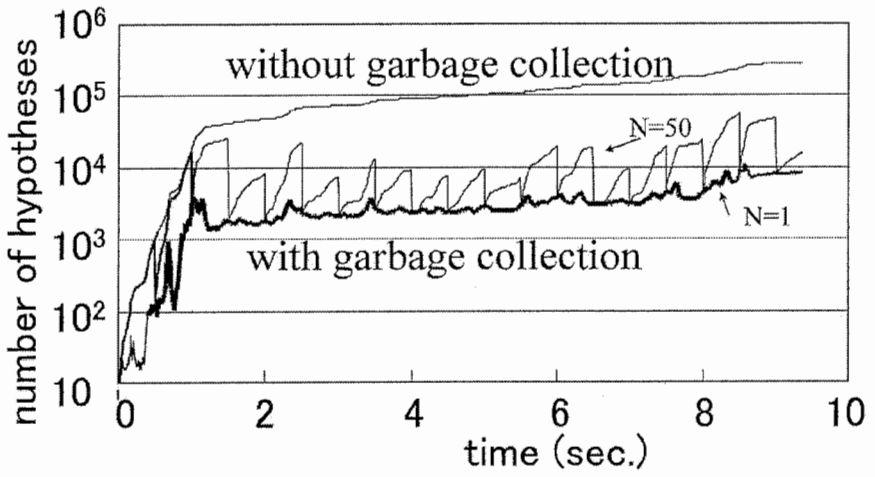

図 9: 生成される単語仮説数の時間推移

表 2: ガーベッジコレクションを搭載したデコーダの評価

decoding		word accuracy	memory(MB) Max	memory(MB) mean	RTF
without GC		87.1%	127	11.4	1.43
proposed		87.1%	6.9	0.8	1.53
trace back	N=1	87.1%	6.9	0.8	1.60
	N=2	87.1%	8.9	1.0	1.57
	N=8	87.1%	14.9	1.5	1.53
	N=25	87.1%	30.2	2.8	1.50
	N=50	87.1%	55.3	4.0	1.47

過時間，縦軸はその時点で単語グラフに存在する単語仮説の数である．実験では，GC を行わない場合と分析フレーム毎 (10ms 間隔) に GC を行った場合，更に，文献 [6] と同条件で 50 フレーム毎 (0.5 秒間隔) で GC を行った場合の 3 つの条件を比較した．GC のない従来のデコードでは，デコードが進むにつれて単語仮説数が単調増加し，10^5 を超える単語仮説が生成されている．また，GC を行った場合に関しては，毎フレーム GC を行ったグラフ (N=1) と 50 フレームごとに GC を行ったグラフ (N=50) を比較すると大きな差が見られる．毎フレーム GC を行った場合には，単語仮説数の上限は 10^4 未満となり大幅な仮説数の削減が見込まれるのに対し，50 フレーム毎に GC を行った場合は，GC を停止している 0.5 秒の間に不要となった単語仮説が累積することで，より多くのメモリ空間を必要としてしまう．

のべ 551 発話の評価音声を用いた認識結果を表 2に示す．GC については，トレースバックを用いる場合とトレースバックを用いない提案手法を比較した．比較項目は，word accuracy，単語仮説生成で消費されるメモリ量 (最大値，平均値)，RTF の 3 つである．

はじめに，GC なしと本提案手法について比較する．word accuracy はいずれの条件とも

87.1%と一致し，GCによる認識性能の劣化は認められない．この時のRTFは，GCなしが1.43に対して提案手法は1.53と約7%の処理量増加となった．消費メモリサイズに関しては，最大値で127MBから6.9MBへと1/18以下のサイズに，平均値で11.4MBから0.8MBへと1/14以下のサイズになった．

次に，トレースバックを用いてGCを行う方式と提案手法とを比較する．消費メモリ量はトレースバックを用いた場合(N=1)と提案手法との間に差はなかった．これは，トレースバックを用いる場合にも後続仮説への参照を記憶するためのエリアが必要となるためである．処理量に関しては，トレースバックを用いて毎フレームGCを行う場合(N=1)がRTF=1.60であるのに対し，提案手法はRTF=1.53と，提案手法を用いることでGCの計算負荷を軽減できることが確認された．一方，50フレーム毎にトレースバックを用いてGCを行う場合(N=50)では，RTF=1.47と提案手法よりも低処理量であるものの，消費される最大メモリ量は提案手法の6.9MBに対し53.3MBと大きく増加しており，十分なメモリ削減はなされていない．また，提案手法と同じRTFとなるGC間隔(N=8)で比較しても，提案手法のほぼ2倍近いメモリを消費している．

以上の結果，本提案手法を用いることにより，トレースバックを用いてGCを行う方式(N=1)に比べ少ない処理量で，仮説生成に伴う消費メモリを平均で1/14以下に削減できることが示された。

5.3 factoringテーブルのメモリサイズ削減

不要な単語仮説に対するGCを実装したデコーダを用い，factoringテーブルの近似手法を評価する．実験ではfactoringテーブルを以下の3条件で近似し，factoringのメモリサイズとword accuracyについて比較した．

(1) 深さ依存unigram近似 (DD unigram approx.)
(2)POS bigram近似 (POS approx.)
(3) これら2つの近似を併用 (composite)

言語モデルに多重クラス複合N-gramを用いた場合の実験結果を表3に示す．baselineはfactoringテーブルの近似を行わない条件での結果である．table sizeは，factoringテーブルの生成によって消費されたメモリサイズである．なお，処理量はfactoringテーブルの近似手法によらずRTF=1.5であった．また，POS bigram近似との比較のため，多重クラス複合N-gramに代えてPOS bigramを使用した場合も評価した (POS bigram)．

深さ依存unigram近似では，近似による認識性能の低下を最小限に抑えるため，pruningの影響の大きい先頭からk音素目までのノードは近似を行わない．k=0の場合は従来のunigram factoringと等価である．k=0でも15MBのメモリが必要な理由は，第1パスで同音異義語を区別しないため，終端ノードでbigramのfactoringテーブルを生成する場合があるためである．kを小さくしていくほど近似するノード数が増え，メモリサイズが減少していく．同時に，近似によるword accuracyの低下も生じているものの，劣化の程度は大きくない．たとえば，k=2の条件ではfactoringテーブルに必要なメモリサイズは28MBと，近似を行わない場合に対して1/2のメモリサイズとなった．この時のword accuracyの低下は0.5%(87.1% → 86.6%)である．

言語モデルをPOS bigramに置換した場合，word accuracyは84.7%と大きな性能劣化をおこしている．これに対し，POS bigramを用いた近似では，word accuracyは87.0%とほとんど劣化はみられなかった．また，factoringテーブルのメモリサイズは19MBと，baseline

表 3: factoringテーブルの近似に関する評価結果（多重クラス複合 N-gram）

		table size	word accuracy
baseline		56MB	87.1%
DD unigram approx.	k=0	15MB	86.2%
	k=1	21MB	86.4%
	k=2	28MB	86.6%
	k=3	39MB	86.8%
POS approx.		19MB	87.0%
composite	k=1	16MB	86.6%
	k=2	17MB	86.5%
	k=3	18MB	86.3%
POS bigram		19MB	84.7%

に対して 34% のサイズに削減できた．

POS bigram 近似と深さ依存 unigram 近似の併用方式は，深さ k までの非終端ノードは POS bigram で近似，それ以降の非終端ノードは unigram で近似する．この条件では，POS bigram 近似単独よりもメモリサイズの削減は計られているものの，word accuracy の劣化はわずかながら大きい．

次に，言語モデルに単語 bigram を用いて同様の評価を行った．単語 bigram の学習には，多重クラス複合 N-gram の学習と同一のコーパスを用いた．また，辞書は共通である．実験結果を表 4に示す．単語 bigram の場合，factoring テーブルの近似を行わない条件では factoring テーブルのサイズが肥大化してデコーダが動作しない．このため，baseline の word accuracy には木構造辞書の全ノードで factoring スコアを逐次計算してデコードした結果を示した．RTF は factoring テーブルに近似を用いた条件では 1.4 であったのに対し，factoring スコアを逐次計算した baseline では 22.6 となった．

factoring テーブルに深さ依存 unigram 近似を用いた場合，k を増やすと factoring テーブルのメモリサイズは急激に増加し，k=3 ではデコーダが動作しなかった．また word accuracy の劣化は，多重クラス複合 N-gram の場合よりもわずかに大きい．factoring テーブルの近似に POS bigram 近似を用いた場合，factoring テーブルの占めるメモリサイズは 19MB となり，word accuracy の劣化は見られなかった (86.3% → 86.6%)．深さ依存 unigram 近似と POS bigram 近似の併用では，factoring テーブルのメモリサイズは POS bigram 近似単独よりも小さくなるものの，word accuracy は baseline に比べ 0.3% ～ 0.7% 低下している．

以上の結果，いずれの言語モデルを使用した場合においても，factoring テーブルに近似を用いることでメモリサイズの削減が図られ，word accuracy の劣化は 1% 以下であった．特に factoring テーブルの近似手法として POS bigram による近似を用いた場合には，word accuracy の劣化をほとんど起こさずにメモリサイズを削減できることがわかった．

表 4: factoring テーブルの近似に関する評価結果（単語 bigram）

		table size	word accuracy
baseline		N/A	86.3%
DD unigram approx.	k=0	15MB	85.6%
	k=1	47MB	85.3%
	k=2	253MB	85.9%
POS approx.		19MB	86.6%
composite	k=1	16MB	85.6%
	k=2	17MB	85.9%
	k=3	18MB	86.0%

6 むすび

ATRの連続音声認識プログラムATRIUMSに実装されている探索処理の効率化，省メモリ化についてまとめた．連続音声認識の仮説探索の高速化を実現するためには，生成される仮説数の削減が有効である．ATRでは，(1)同音語の言語尤度の共有化，(2)開始時刻の代表処理による先行単語の共有化，(3)枝刈りにより不要となった単語仮説の効率的なメモリ再利用の手法を導入することで，生成される仮説数の削減を図っている．また，木構造辞書の各ノードに付与する言語スコアに近似手法を導入することでテーブルサイズを削減し省メモリ化を図っている．

評価実験では，枝刈りにより不要となった単語仮説の効率的なメモリ再利用と木構造辞書の言語スコアテーブルの削減に関して調査した．時間同期ビーム探索では部分仮説のpruningにより，文末まで到達することのない不要な単語仮説が単語グラフ中に多く存在し，メモリ空間の肥大化をまねいている．これを解消するため，単語仮説に後続する仮説数を管理する属性を追加することによって，トレースバックなしで不要な単語仮説を特定し，ガーベッジコレクション(GC)によるメモリの再利用を図っている．評価実験の結果，従来では単語仮説生成に最大127MBのメモリを必要としていたのに対して，メモリの再利用を行うことで6.9MBと単語仮説生成に伴うメモリ消費量を1/18以下に抑えることができた．また，処理量に関して，トレースバックにより不要な単語仮説を特定する方式ではRTF=1.60であったのに対しATRの採用方式ではRTF=1.53であった．GCなし(RTF=1.43)からの処理の増加量はそれぞれ12%，7%であり，提案方式を用いることでGC処理のオーバーヘッドを抑えることが可能となった．

木構造辞書の各ノードに割り当てられるfactoringテーブルのメモリサイズ削減に関して，factoringテーブルに格納するbigramスコアをPOS bigramで近似した．言語モデルに多重クラス複合N-gramを用いた評価では，認識性能の劣化をほとんど起こすことなしに，factoringテーブルに必要なメモリサイズを56MBから19MBへと削減することができた．また，言語モデルとして単語bigramを用いた場合についても，同様の効果を確認した．

参考文献

[1] 赤羽 誠, “カーナビゲーションと音声技術”, 音響誌,vol.54,no.3,pp.223-228,Mar. 1998.

[2] 大淵 康成, 北原 義典, 小泉 敦子, 松田 純一, “マイコン向け音声認識技術を用いた携帯型音声通訳機”, 信学論 (D-II),vol.J83-D-II,No.11,pp.2309-2317,Nov. 2000.

[3] H.Ney, X.Aubert, “A word graph algorithm for large vocabulary, continuous speech recognition”, Proc. ICSLP'94, pp.1355-1358, Yokohama,Japan, Oct. 1994.

[4] 清水 徹, 山本 博史, 政瀧 浩和, 松永 昭一, 匂坂 芳典, “大語彙音声認識のための単語仮説数削減”, 信学論 (D-II),vol.J79-D-II,No.12,pp.1355-1358,Dec. 1996.

[5] G.Antoniol, F.Brugnara, M.Cettolo, M.Federico, “Language model representations for beam-search decoding”, Proc. ICASSP95', pp.588-591, Detroit,USA, May 1995.

[6] S.Ortmanns, H.Ney, X.Aubert, “A word graph algorithm for large vocabulary continuous speech recognition”, Computer, Speech and Language, Vol.11, No.1, pp.43-72, January 1997.

[7] D. Willett, E. McDermott, Y. Minami, S. Katagiri, “time and memory efficient viterbi decoding for LVCSR using a precompiled search network”, Proc. Eurospeech, pp.847-850, Aalborg,Denmark, Sep. 2001.

[8] 小窪 浩明, 林 輝昭, 山本 博史, 菊井 玄一郎, “後続仮説数の管理機能を用いた単語グラフの仮説数削減”, 音講論,3-5-11,pp.163-164,Mar. 2002.

[9] 小窪浩明, 林 輝昭, 山本 博史, 菊井 玄一郎, “連続音声認識システムにおける factoring テーブルのコンパクト化と不要単語仮説のガーベッジコレクション”, 電子情報通信学会論文誌, Vol.J86-D-II,No.6,pp.787-794, 2003.

[10] V. Steinbiss, B. Tran, H Ney, “Improvements in beam search”, Proc. ICSLP'94, pp.2143-2146, Yokohama,Japan, Oct. 1994.

[11] 今井 亨, 小林 彰夫, 佐藤 庄衛, 安藤 彰男, “逐次 2 パスデコーダを用いたニュース音声認識システム”, 信学技報,SP99-129,pp.85-90,Dec. 1999.

[12] 小窪 浩明, 山本 博史, “木構造ネットワークサーチにおける bigram factoring テーブルの削減”, 音講論,1-5-18,pp.35-36,Sep. 2000.

[13] 山本 博史, 匂坂 芳典, “接続の方向を考慮した多重クラス複合 N-gram 言語モデル”, 信学論 (D-II),vol.J83-D-II,No.11,pp.2146-2151,Nov. 2000.

[14] 内藤 正樹, 山本 博史, シンガー ハラルド, 中嶋 秀治, 中村 篤, 匂坂 芳典, “対話音声を対象とした連続音声認識システムの試作と評価”, 信学論 (D-II), vol.J84-D-II,No.1,pp.31-40,Jan. 2001.

[15] 内藤 正樹, Harald Singer, 山本 博史, 中嶋 秀治, 松井知子, 塚田 元, 中村 篤, 匂坂 芳典, “旅行会話タスクにおける ATRSPREC の性能評価”, 音講論,3-1-9,pp.113-114,Oct. 1999.

[16] 小窪 浩明, 山本 博史, “木構造辞書を記録した記憶媒体，及び木構造辞書の言語スコアテーブル作成プログラム”, 特願 2004-07402, 2004.