英日統計的機械翻訳における語順優先探索デコーダ

岩越隼人† 山本幹雄†

統計的機械翻訳では翻訳候補の中から確率が最大である候補を出力する．これまでにも効率的に探索するデコーダはいくつか提案されているが，我々は *greedy* デコーダに着目した．*greedy* デコーダの英日翻訳における欠点は単語の並べ替えに関して局所解に陥りやすいことにあると我々は考える．そこで訳語選択と単語の並べ替え部分を分離して，より語順を重視した語順優先探索デコーダを提案する．その結果，翻訳精度 *BLEU*，*NIST* において向上が確認できた．

Reordering Priority Decoder for Statistical Machine Translation

Hayato Iwakoshi† and Mikio Yamamoto†

In statistical machine translation, a decoder searches the translation hypothesis with the highest probability in the large search space. We focused on *greedy* decoders and found out that they tend to fall into local maximum of reordering when they translate sentences between languages very different on word order. This paper describes the reordering priority decoding algorithm which divides greedy searches into reordering phase and the generation phase for target words. The experiment results of English-to-Japanese translation show the reordering priority decoder is superior to normal *greedy* decoders.

1. はじめに

統計的機械翻訳では確率モデルが翻訳候補に確率を与え，デコーダが探索空間の中から確率が最大である翻訳候補を探索する．この探索は探索空間が膨大であるため，様々な工夫が必要である．これまでに効率的に探索する手法として $A*$ 探索，*DP* 探索，*greedy* 探索などを用いたデコーダが提案されている[4),7),10)]．本論文では翻訳速度，翻訳精度の点で比較的優れている Germann らが提案した *greedy* デコーダ[4)] に着目し，英日翻訳のように大幅な語順の入れ替えが必要な場合の改良法を検討する．

まず，大幅な語順の入れ替えが必要な場合のために，語順優先探索デコーダを検討した．もともと *greedy* デコーダは局所解に陥る可能性があるが Germann らの近傍の定義は仏英翻訳を意図しているため，英日のように大きな語順の変更が必要な言語間の翻訳には適さない．単語の並べ替えが局所解に陥っているためである．

† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Systems & Information Engineering, University of Tsukuba

そこで我々は単語の並べ替えに着目し，*greedy* デコーダの単語並べ替え部分と訳語選択部分を分離して語順決定の *greedy* 探索と訳語決定の *greedy* 探索の 2 重ループの構造にすることを提案する．ただし，単に 2 重ループの構造にすれば探索空間が広がりすぎ，ノイズとなる確率の高い誤った翻訳候補までが探索空間に入ってくる．これは現在の翻訳モデルはあまり精度が良くないことに問題がある．そこで木構造による制約を用いて制限をかけることでノイズを除去する．*greedy* デコーダを 2 重ループの構造とし，特に語順に関する探索空間を広げ，木構造によって単語の並べ替えを制約した語順優先探索デコーダを本論文で検討する．

次に単語間の対応がつきにくい言語どうしの翻訳の場合への対処として，*Zero Fertility* 制限（以下，F_Z 制限と呼ぶ）を検討した．一般に日本語文の形態素数は対応する英語文の単語数より多い傾向があり，英日翻訳で *IBM* モデルを用いる場合，F_Z 単語☆リストをかなり大きくとる必要がある[4)]．しかし F_Z リストを大きくすると探索空間が膨大になり探索に失敗する．そこで各訳語に直後に来やすい F_Z 単語をリストに

☆ Fertility（繁殖数）：目的言語の 1 単語に対応する原言語の単語数，F_Z: Fertility がゼロであること

しておくことで，リストを小さくし，かつ，網羅的に F_Z 単語を挿入可能にする F_Z 制限を提案する．

評価実験では *greedy* デコーダ，語順優先探索デコーダを比較した．さらに，これらのデコーダに F_Z 制限を組み込んだ場合も評価実験を行った．また単語ベースでの評価実験だけでなくフレーズベースにした場合も比較をした．その結果，一貫して語順優先探索は *greedy* 探索より高い翻訳精度が得られた．また，語順優先探索においては，ナイーブな単語並べ替え手法に比べて，構文を考慮した並べ替えは効率的に翻訳精度を改善できることを示した．また，語順優先探索と組み合わせることで F_Z 制限を使ってさらに翻訳精度を改善することができた．

2 章では統計的機械翻訳の概要を述べ，3 章で語順優先探索デコーダと F_Z 制限について述べる．4 章では木構造を用いた単語並べ替え制約の英日翻訳における有効性を予備実験し，5 章でデコーダの評価実験を行う．

2. 統計的機械翻訳

2.1 統計的機械翻訳の概要

統計的機械翻訳では原言語の文 E から目的言語の文 J に翻訳する場合，与えられた E に対して確率 $P(J|E)$ を最大にする $\hat{J}$ を選ぶことで英語を日本語に翻訳する．さらにベイズの定理により，$P(J)P(E|J)$ を最大にするものを探せばよい．ここで $P(J)$ は言語モデル，$P(E|J)$ を翻訳モデルという．

$$\hat{J} = \arg\max_J P(J|E) = \arg\max_J P(J)P(E|J)$$

以下，翻訳モデル，デコーダを簡単に説明する．

2.2 翻訳モデルの概要

翻訳モデルの中でも代表的なのが *IBM* モデルである．*IBM* モデルは Brown ら[2] が提案したモデルで，順番に複雑になるモデル 1 からモデル 5 までがある．本研究では *GIZA++*[8] の実装によるモデル 4 を用いた．*IBM* モデル 4 では主に翻訳確率，繁殖確率，歪確率の 3 つが使われる．以下に原言語を英語，目的言語を日本語として，簡単に説明する．

(1) 翻訳確率：$t(e|j)$
日本語単語 j が英語単語 e に翻訳される確率

(2) 繁殖確率：$n(\phi|j)$
日本語単語 j が ϕ 個の英語に翻訳される確率

(3) 歪確率：$d(x - x'|class(j_{y-1}), class(e_x))$
y 番目に位置する日本語単語に対応する英語の単語が相対的にどの位置に移動するかをモデル化する確率．j_y は y 番目に位置する日本語単語，j_{y-1} はその直前の日本語単語とし，それぞれの日本語単語に対応する英語単語を e_x と $e_{x'}$ とする．x と x' は英文中の位置である．$class()$ は単語の品詞などのカテゴリを表す．

実際の文に対しては各単語ごとの上記 3 種の確率の積によって $P(E|J)$ を近似する．

2.3 デコーダの概要

$P(J)P(E|J)$ が最大となる $\hat{J}$ を出力するのがデコーダである．全候補を調べれば $P(J)P(E|J)$ が最大になるものを必ず見つけられるが，候補数が膨大なため現実的ではない．英語の 1 単語が日本語の 1 単語に翻訳されるとして，英語文が L 単語からなる場合，各英単語が M 個の翻訳候補を持つと，訳語の並び方は $L!$，さらに各訳語に M 個の候補があるので候補数は $L! \times M^L$ となり非常に膨大な数となる．さらに，英語には対応しない日本語（“は”，“を”，“に” など），逆に日本語には対応しない英語（“a”，“the” など）なども考慮すると全候補数はさらに増えることになる．

膨大な探索空間において最も確率の高い翻訳候補を効率良く見つけることがデコーダの課題である．この課題を解決するために探索の効率化，探索空間の制限，フレーズの利用などが行われる．これらはあわせて使うことでより質の良い翻訳結果が得られる．以下で簡単にそれぞれを説明する．

効率的に探索する手法として *A** 探索，*DP* 探索，*greedy* 探索などを用いたデコーダが提案されている[4),7),10)]．*A** 探索，*DP* 探索は *greedy* 探索よりも大幅に翻訳時間がかかるので長い文の翻訳には向いていない．一方，Germann らが提案した *greedy* デコーダ[4)] は比較的高速であるが，語順が似ている言語間の翻訳を基本として提案されたため，英日翻訳にそのまま用いても十分な翻訳精度を得ることはできない．

探索空間を制限する代表的な手法は目的言語の文と原言語の文がほぼ線形に対応していると仮定し，単語の並べ替えに制約する *IBM* 制約[1)] である．しかしながら単語の並べ替えがかなり多く必要である英日翻訳には向いていない．そのほか，木構造を用いて単語を並べ替えるという制約がある．任意の 2 分木のノードを反転させる☆ことで単語を並べ替える *ITG* 制約[12)] や，入力文の構文解析木のノードを反転させる構文木制約[13)] などである．これらは単語列をまとめて移動させることができるので英日翻訳に向いていると考えられる．

フレーズベースの翻訳では原言語の単語をある程度

☆ ノード以下の右部分単語列と左部分単語列の交換

の長さのフレーズにまとめ，フレーズごと翻訳し目的言語のフレーズを生成する．一般にフレーズベースとは単語ベースの翻訳モデルをフレーズベースに拡張したモデルを意味する[6),11)]．しかしながら本論文ではデコーダの改良に限定しているので，訳語選択や単語の並べ替えの操作をフレーズ単位で行うことによるデコーダの改良として考える．

以降では本論文の基本となる Germann らが提案した *greedy* デコーダを説明し，その問題点について述べる．そして，その問題点を解決するように訳語選択の操作と単語の並べ替えの操作を分離した語順優先探索デコーダを提案する．さらにまだ残っている問題点を指摘し，F_Z 制限を提案する．

3. 語順優先探索デコーダ

3.1 *greedy* デコーダ

本節では Germann ら[4)] が仏英翻訳を基本として提案した *greedy* デコーダを説明する．これは局所探索の 1 つである *greedy* 探索を用いたデコーダである．

greedy デコーダは以下のように動作する．1 つの翻訳候補（親）から親を少し変更した翻訳候補群（近傍，子）を生成する．もし子の中で最も確率が高い翻訳候補が親よりも確率が高い場合その翻訳候補を次の親とする．親よりも高い確率の子がない場合終了する．近傍（子）の作り方は以下である．

- 訳語を 1 カ所もしくは 2 カ所変更する
- 訳語を 1 カ所変更し，同時に F_Z の単語（列）を追加，変更，削除する
- F_Z の単語（列）を削除する
- 任意の 2 つの重ならない部分単語列を交換（= swap）する

以上のような操作で近傍を生成することで局所解に陥りにくいとされている．比較的近傍を広く考慮することによって局所解に陥りにくいようになっている．

5 章の評価実験で用いるベースラインシステムは上記で説明した German らのアルゴリズムを基本的に用いた．ただし，1 カ所だけ Germann らが提案した *greedy* デコーダから変更した点がある．Germann らの *greedy* デコーダでは翻訳元には対応しない単語（F_Z の単語）を単語単位で挿入しているが，本論文では単語列単位で挿入する．英日翻訳の場合，日本語は形態素単位で扱うため英語に対応しない日本語単語の数が多くなるためである．

3.2 英日翻訳における *greedy* デコーダの問題点

3.1 節の *greedy* デコーダは仏英翻訳を基本に提案されたため，英日翻訳には適さない部分もある．仏英翻訳と英日翻訳で大きく違う点の 1 つは語順であり，仏英翻訳に対し，英日翻訳ではかなり多くの単語の並べ替えが必要となる．Germann らが提案した *greedy* デコーダでは訳語の変更と語順の変更を同時に近傍に生成する 1 重の *greedy* 探索であることから語順の変更が起こりにくい傾向がある．いったん，言語モデルでの確率が高いような単語列の並びができると，その単語列を壊すような並べ替えは確率が低くなってしまい，次世代の親として選ばれにくいからである．

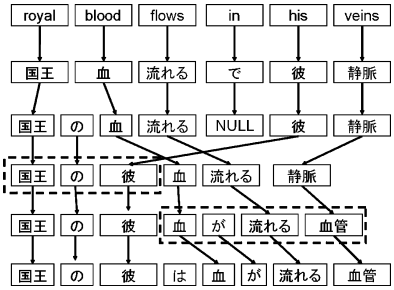

図 1 *greedy* デコーダの翻訳失敗例
Fig. 1 Example of translation by *greedy* decoder.

その例を**図 1** に示す．ここでは “国王の彼” や “血が流れる血管” のような単語列がいったん親として選ばれると，これらの単語列を壊すように並べ替えると確率が下がってしまう．この例では単語の並べ替えは 1 回しか起こっていない．

語順の大きく違う翻訳では図 1 の例のように特に語順に関して局所解に陥る可能性がある．この問題を緩和するために次節では語順を重視するように拡張した語順優先探索デコーダを提案する．

3.3 語順優先探索デコーダ

本節では並べ替えを重視した語順優先探索デコーダを提案する．3.1 節の *greedy* デコーダは訳語の変更と語順の変更を同時に近傍に生成する 1 重の *greedy* 探索である．1 重の *greedy* 探索であるから語順の並べ替えが起こりにくいと我々は考えた．そこで，訳語選択部分と単語並べ替え部分を分離させ，語順決定の *greedy* 探索と訳語決定の *greedy* 探索の 2 重ループの構造にすることを提案する．これにより 1 重目の語順決定の *greedy* 探索で単語の並べ替えを強制的に行うことができる．

しかしながら単に 2 重ループにすれば探索空間が膨大になってしまう．たとえば，入力英文の長さが 20 単語の場合，任意の 2 つの重ならない部分単語列の交換はシミュレーションによると約 7,000 通りあり，それぞれの英語単語列で訳語決定の *greedy* 探索をすることを繰り返すことになる．現在のモデルの精度はあま

A: **語順決定のgreedy探索**

1: 初期操作として入力英語文を構文解析をすることで構文解析木を得る

2: 上記で得られた構文解析木を初期の親として以下を繰り返す

2-1:親と1箇所のみノードの状態を変え(反転)単語を並び替えることで複数の子(反転するノードが違う)を生成する

2-2:各子を訳語決定の局所探索へと渡し訳語決定を行い確率を得る

2-3:子の確率の中で最も高い確率が親の確率を上回る場合は最も確率の高い子 を次の親 とする. もし親よりも確率の高い子がいない場合, 操作2を 終了し出力する

B: **訳語決定のgreedy探索**

1: 初期の訳語として最も翻訳確率が高い訳語候補を採用し親とする

2: 以下の手順でつくり出せる子の中で最も高い確率が親の確率より 高ければその子を次の親として繰り返す. もしそうでなければ終了

2-1: 親の訳語を1箇所変更する

2-2: ZeroFertility の単語(列)を1箇所追加, 変更, 削除する

図 2 語順優先探索デコーダのアルゴリズム
Fig. 2 Reordering Priority Decoding algorithm.

り良くないことから，探索空間が広い場合にノイズとなる確率の高い誤った翻訳候補までが探索空間に入ってくる．

そこでノイズをできるだけ入らないように，語順決定の *greedy* 探索では構文木制約を用いて近傍を小さくする．これにより単語の並べ替えで任意の 2 つの重ならない部分単語列を交換する操作の無駄を大幅に削減することができる．また，訳語決定の *greedy* 探索では訳語と F_Z 単語列の変更パターンを少なくした．具体的な手順を**図 2** に示す．

実際には初期化を 3 回行って最良の候補を選択した．1 回目の初期状態としてすべてのノードをそのまま，2 回目の初期状態としてすべてのノードを反転，3 回目の初期状態としてランダムにノードを反転させて初期の親とした．

3.4 F_Z 制 限

本節では F_Z の単語列の挿入に関する制限を加えた F_Z 制限について説明する．英語文から日本語文に翻訳した場合，正しい翻訳結果は入力英文より約 1.3 倍ほど長くなる．このことから特に英日翻訳では F_Z の単語列の挿入が大きな問題となることが分かる．原言語の単語を翻訳した訳語と F_Z の単語列の組合せで正しい翻訳結果を得るには多くの F_Z の単語列が必要となる．これを 1 つのリストにまとめればリストが非常に大きくなってしまう．F_Z 単語列リストは大きくする必要があるが，リストを大きくすると探索空間は膨大になるため十分な大きさのリストを用意することができず，翻訳に失敗する．

そこで，各訳語ごとに F_Z 単語列リストを持たせることで F_Z の単語列リストを小さくする．その例を**図 3** に示す．“眠り” には直後に来やすい “に”，“の” という F_Z 単語列リストを持たせる．“眠っ” には直後に来やすい “て”，“て いる”，“て しまっ た” などの F_Z 単語列リストを持たせる．このように各訳語の直後に来やすい単語列だけをリストに入れることで各 F_Z 単語列リストは非常に小さくすることができる．

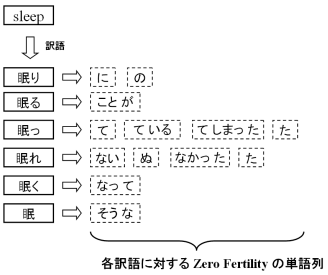

図 3 各訳語に F_Z 単語列リストを持たせる例
Fig. 3 Examples of F_Z words list.

また各訳語特有の直後に来る単語列も網羅できる．

F_Z 制限では挿入する F_Z 単語列は直前の訳語に依存するので訳語の変更と F_Z 単語列の変更を別に操作することはできない．そこで以下のようにして *greedy* デコーダと語順優先探索デコーダに F_Z 制限を組み込む．F_Z 制限付き $\dot{g}\dot{r}\dot{e}\dot{e}\dot{d}\dot{y}$ デコーダでは以下の操作で近傍を生成する．

- 訳語を 1 カ所もしくは 2 カ所変更し，その直後には変更した訳語の F_Z の単語列リストの最も頻度が高い単語列を挿入する．
- 訳語を 1 カ所変更し，同時に直後の F_Z 単語（列）を追加，変更，削除する．
- F_Z の単語（列）を削除する．
- 任意の 2 つの重ならない部分単語列を交換する．

F_Z 制限付き語順優先探索デコーダでは以下の操作で訳語決定の局所探索における近傍を生成する．

- 訳語を 1 カ所変更し，同時に直後の F_Z 単語（列）を追加，変更，削除する．

4. 木構造による制約に関する予備実験

4.1 予備実験手順

4.1，4.2 節では予備実験として，英日翻訳時に木構造による制約が正しい並べ替えをどの程度生成可能であるかを定量的に調べる．英日対訳コーパスに *GIZA++* [8] によってアライメント（単語対応）を付け，それを元に英語単語列を日本語の語順に合わせて

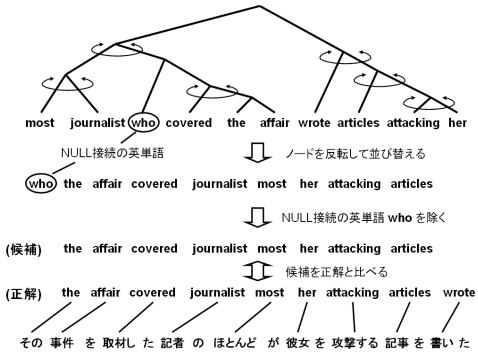

図 4 木構造のノードの反転による並べ替えと正解との比較例
Fig. 4 Example of comparison reordering by parsing tree with reference.

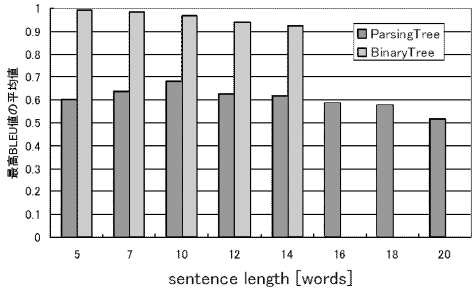

図 5 候補中の最高 *BLEU* 値の平均値
Fig. 5 Average of the highest *BLEU* in hypotheses.

並べ替える．このとき IBM モデルでは多対 1 のアライメントを仮定しており，日本語 1 単語に複数の英語単語が対応している場合は，それらの英語の語順は保持する．また，日本語に接続しない単語は削除する．この日本語の語順に並べ替えた英語単語列を正解として，以下の 2 つの操作で入力英文が正解の並びになるかを検証した．

- 任意の 2 分木による並べ替え（*ITG* 制約）[12]
  - まず入力英文に対し英語単語を葉とするあらゆる形の 2 分木を生成する．次に各 2 分木においてノードを反転させることで単語を並べ替える．反転させるノードの組はあらゆるものを試す．
- 構文解析木による並べ替え（構文木制約）[13]
  - まず入力英文に対し構文解析をすることで構文解析木を得る．次に構文解析木のノードを反転させることで単語を並べ替える．任意の 2 分木による並べ替えとは違い構文解析木 1 つのノードを反転させるので候補は非常に少なくてすむのが利点である．

これらの操作で生成される各英語単語列に対し，正解文との *BLEU* 値[9] を算出し最も高い *BLEU* 値を出力する（*BLEU* は 5.1 節を参照）．1 カ所でも構文解析誤りが生じると正解との完全一致が困難となるため，語順の部分評価が可能である *BLEU* 値を評価指標として選んだ．**図 4** にノードの反転による並べ替えの様子と日本語の語順をした正解の例を示す．

4.2 予備実験結果

長さ 5 単語から 20 単語まで各 100 文のテストセットで実験を行い，各文の *BLEU* 値の平均によって比較した．ただし任意の 2 分木を用いる場合，候補数が膨大なため 14 単語までの実験しかできなかった．英文の長さごとの平均 *BLEU* 値を**図 5** に示す．任意の 2 分木を用いた場合のグラフを *Binary Tree*，構文解析木を用いた場合のグラフを *Parsing Tree* と表記する．

任意の 2 分木を用いた並べ替えでは *BLEU* 値が 0.9 以上の高い値となった．しかしながら候補数は 14 単語で 1.4 億個にもなり非常に膨大になってしまう．構文解析木を用いた並べ替えでは *BLEU* 値が約 0.6 前後と低くなってしまった．この原因の多くは *GIZA++*[8] がつけた単語対応または構文解析誤りである．しかし，候補数は 14 単語で約 8 千個，20 単語でも約 50 万個程度なのでデコーダに埋め込みやすいというメリットがある．

木構造を利用して単語の並べ替えを制約しても，ある程度日本語の語順を作り出せることが分かった．性能としては構文木制約が *ITG* 制約よりも劣っているものの，計算量では構文木制約が *ITG* 制約より大きく優れている．本論文では，翻訳時間を重視して構文木制約の方を用いることにする．

5. 評価実験

5.1 実験条件

学習コーパスを**表 1** に示す．学習コーパスをもとに *GIZA++*[8] を用いて構築した *IBM* モデル 4 を評価実験で用いた．*IBM* モデルの概要は本論文の 2.2 節，詳しくは文献 2) を参照．また学習コーパスから独立したテスト用コーパスから長さ 5〜6，7〜8，9〜10，11〜12，13〜14，15〜16，17〜18，19〜20 単語からなる英文，各 100 文ずつ抜き出し，オープンなテストセットを作った．また，テスト用コーパスからランダムに抽出した 1,000 文（平均英語単語長：9.47，平均日本語単語長：12.36）のテストセットも用いた（長さごとのテストセットと 1,000 文のテストセットは重複あり）．また学習コーパス，テストコーパスから独立した開発用テストセットとして 1,000 文（平均英語文長：9.46，平均日本語文長：12.26）を用いた．各対

表 1 学習コーパス
Table 1 Training corpus.

	英語	日本語
文数	471,848	
単語数	5,408,318	6,752,796
語彙数	79,075	69,368
平均文長	11.46	14.31

訳文の英文を入力に使い，日本語文を正解文として用いた．これらの対訳コーパスは新聞記事[14]や辞書例文から構成されている．構文解析木を利用するためにパーザとして *Collins parser*[3]を利用した．評価実験では 1 つの英語単語に対し，最大 10 個の訳語を用意した．F_Z 制限を用いない場合は，訳語の直後に考慮する F_Z 制限を 50 個用意した．F_Z 制限を用いる場合は，各訳語にその訳語の直後に考慮する F_Z 制限を最大 10 個用意した．このとき，リストを用意できない訳語に関しては直後に F_Z 制限は挿入されにくいと考え，F_Z 制限は挿入しないことにする．

評価指標として翻訳精度と翻訳速度を各デコーダで比較する（CPU: Xeon，2.4 GHz の計算機で実験）．翻訳精度は以下の基準を用いた．

- BLEU[9]
  翻訳結果と正解文の *ngram* の一致率を幾何平均し算出される．ただし，翻訳結果が正解文よりも短い場合はペナルティを与える．*BLEU* のとりうる範囲は 0〜1 である．
- NIST[5]
  翻訳結果と正解文の *ngram* の一致率に重みをつけて算術平均し算出される．*NIST* も同様に，翻訳結果が正解文よりも短い場合はペナルティを与える．*NIST* のとりうる範囲は 0 以上である．

本論文では 4-*gram*ベースでの *BLEU*，*NIST* 評価法を用いており，参照訳は 1 つである．*BLEU*，*NIST* ともに値が大きいほど翻訳精度が高いことを表す．

5.2 比較デコーダ

評価実験で比較するデコーダを述べる．我々は単語ベースのデコーダだけではなく，フレーズベースのデコーダでも評価実験を行った．一般にフレーズベースとは単語ベースモデルを拡張したモデルを意味するが，ここでは訳語選択や単語の並べ替えの操作をフレーズ単位で行うことを意味する．まず，フレーズ辞書を構築する必要がある．構築法は Och ら[6]が提案した手法を用いた．実際に学習コーパスから抽出した例を**表 2** に示す．フレーズベースのデコーダはフレーズ辞書を用いて入力単語列をフレーズ列へ分割する．フレーズ辞書を参照しながら分割数最少法で分割を決める．フレーズを 1 つの単語と見なして *greedy* デコーダ，語順優先探索デコーダへフレーズ列を渡す．入力単語列を分割する例を**図 6** で示す．

表 2 フレーズ抽出例
Table 2 Example of phrase.

英語フレーズ	日本語フレーズ
late in the 12th century	12 世紀末
until late in the 12th century	12 世紀末 まで
in England	イギリス で
employed in England	イギリス で 使わ
was employed in England	イギリス で 使わ れ
employed in	で 使わ
was employed in	で 使わ れ
was employed	使わ れ

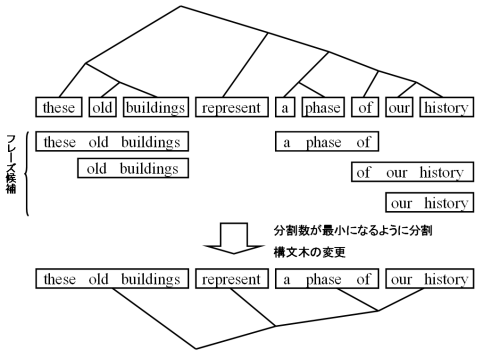

図 6 フレーズへの分割例
Fig. 6 An example of phrase segmentation.

greedy デコーダと語順優先デコーダを基本とし，単語ベースとフレーズベース，F_Z 制限あり，なしのパターンを組み合わせ，以下の 8 種類のデコーダを比較する．

- 単語ベース *greedy* デコーダ
- F_Z 制限付き単語ベース *greedy* デコーダ
- フレーズベース *greedy* デコーダ
- F_Z 制限付きフレーズベース *greedy* デコーダ
- 単語ベース語順優先探索デコーダ
- F_Z 制限付き単語ベース語順優先探索デコーダ
- フレーズベース語順優先探索デコーダ
- F_Z 制限付きフレーズベース語順優先探索デコーダ

上記の 8 種類に加え，*greedy* デコーダのナイーブな拡張として *greedy*（4*swap*）デコーダを評価実験に加える．*greedy*（4*swap*）デコーダは 1 重の *greedy* 探索のまま，*greedy* デコーダの語順に関する近傍を大きく広げたデコーダである．単純な *greedy* デコーダの語順に関する近傍は「任意の 2 つの重ならない部分単語列を交換」で定義されていたが，*greedy*（4*swap*）では「最大 4 つの重ならない部分単語列の交換」で定義

する．すなわち，以下の 3 種類の交換を行う．

- 任意の 2 つの重ならない部分単語列の交換
- 任意の 3 つの重ならない部分単語列の交換
- 任意の 4 つの重ならない部分単語列の交換

訳語に関する近傍は *greedy* と同じである．*greedy* に比べて *greedy*（*4swap*）の語順に関する近傍は大きく広がり，語順に関する局所解に陥ることを回避できる可能性が出てくる．

5.3 評価実験結果と考察

表 3，**図 7** に 1,000 文のテストセットにおける *BLEU* 値，翻訳速度を示す．ただし，表 3 の翻訳速度は 1 文あたりの時間であり，図 7 の翻訳速度は 1,000 文全体の時間である．単語ベース，フレーズベース，F_Z 制限あり，なしのどの点においても，一貫して，語順優先探索デコーダは *greedy* デコーダよりも *BLEU*，*NIST* ともに高い結果となった．

また，語順優先探索デコーダは F_z 制限と組み合わせることでさらに翻訳精度を改善できることが分かる．これはより適切な F_Z 単語列を用意することで重みの大きい適切な内容語が選ばれるようになったためと考えられる．さらに，F_Z 制限により，*greedy* デコーダでは速度が向上するが翻訳精度が低下する一方で，語順優先探索デコーダでは逆に速度が若干悪化するが翻訳精度は改善できる．*greedy* デコーダでは F_Z 制限によって近傍が小さくなった効果が直接現れている．一方，語順優先探索デコーダではもともと訳語決定の近傍を小さくとっているうえに，F_Z 制限では訳語の変更と F_Z の単語列の変更を別に行えないために，近傍が大きくなったので翻訳速度が若干悪化した．

また，語順優先探索において，ナイーブな単語並べ替え手法に比べて，構文を考慮した並べ替えは効率的に翻訳精度を改善できることが分かった．単語ベース *greedy* デコーダと比べ単語ベース語順優先探索デコーダは翻訳精度が上がっているが，翻訳速度の悪化は 2 倍程度にとどまっている．これに対し，単語ベース *greedy*（*4swap*）デコーダは単語ベース *greedy* デコーダの翻訳精度が下がっているうえに，翻訳時間が 15 倍ほどかかっている．*greedy*（*4swap*）デコーダはナイーブな手法で語順に関する近傍を大幅に広げたために翻訳速度が大幅に悪化するばかりか，質の悪い確率の高い翻訳候補（ノイズ）までが近傍に入ってしまい，翻訳精度まで悪化していると考えられる．このような傾向が出てくるのは翻訳モデルの精度が高くないことに原因があると考えられる．しかし，語順優先探索デコーダは構文木制約を用いることによって，このようなノイズまでを翻訳候補として考慮してしまうことをうまく避けながら，探索範囲の拡大に成功しているといえる．

表 3 1,000 文テストセットでの各デコーダの *BLEU* 評価
Table 3 *BLEU* of each decoders in 1,000 sentences test set.

デコーダ	*BLEU*	*NIST*	Time (sec./文)
単：*greedy*	0.042	2.79	11
単：*greedy(4swap)*	0.044	2.75	165
F_z：単：*greedy*	0.037	2.93	1.5
単：語順	0.061	2.98	20
F_z：単：語順	0.070	3.32	38
句：*greedy*	0.074	3.39	2.6
句：*greedy(4swap)*	0.076	3.36	9.6
F_z：句：*greedy*	0.073	3.66	0.29
句：語順	0.088	3.72	2.0
F_z：句：語順	0.092	3.82	2.1
ISI ReWrite Decoder	0.046	2.59	—
市販ソフト	0.093	3.44	—
Web 翻訳	0.085	3.29	—

greedy：*greedy* デコーダ，語順：語順優先探索デコーダ，
単：単語ベース，句：フレーズベース，F_Z：F_Z 制限付き

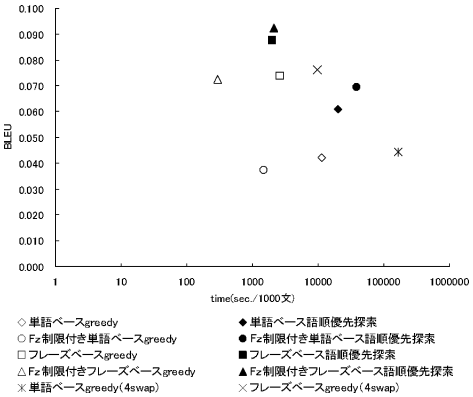

図 7 各デコーダの *BLEU* 評価，1,000 文の翻訳速度
Fig. 7 *BLEU* and time of each decoders.

また参考として市販ソフトと *Web* の翻訳サービスの *BLEU*，*NIST* を表 3 に載せた．4 つの市販ソフトと 3 つの *Web* 翻訳のうち最も *BLEU* が高かったものをそれぞれ載せた．

単語ベースのデコーダにおける入力英語長ごとのテストセットに対する *BLEU* を**図 8** に示す．また，翻訳時間を**表 4** に示す．単語ベースにおいては *greedy* デコーダより語順優先探索デコーダが *BLEU* 値が高いことが分かる．これは語順をより重視して 2 重ループの構造にしたためである．その分，*greedy* デコーダの約 2 倍の翻訳時間がかかってしまっているが，近傍を小さくとることで 2 倍程度に抑えられたといえる．

また図 8 より，単語ベースにおける F_Z 制限は入

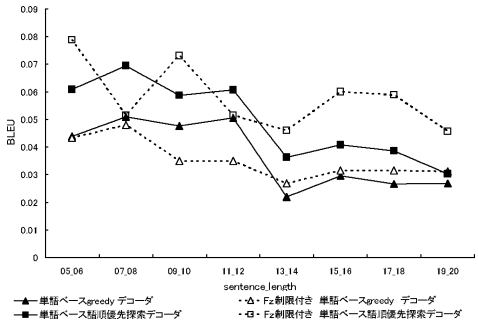

図 8 単語ベースデコーダの *BLEU* 評価
Fig. 8 *BLEU* of word based decoders.

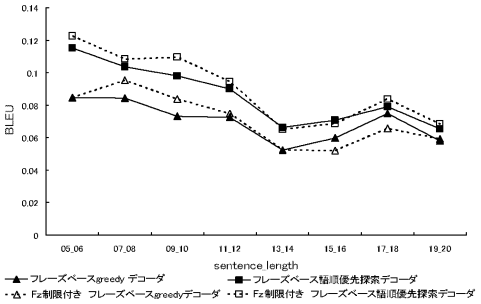

図 9 フレーズベースデコーダの *BLEU* 評価
Fig. 9 *BLEU* of phrase based decoders.

表 4 各デコーダの評価実験結果（翻訳時間，秒）
Table 4 Time of each decoder.

単語数	単：G	単：語順	句：G	句：語順
5–6	0.79	0.43	0.29	0.48
9–10	5.20	5.00	0.92	1.51
13–14	17.11	24.53	3.34	4.69
17–18	49.43	93.56	10.33	10.92
単語数	F_z：単：G	F_z：単：語順	F_z：句：G	F_z：句：語順
5–6	0.14	0.65	0.21	0.29
9–10	0.72	9.76	0.32	0.90
13–14	2.26	50.86	0.65	3.91
17–18	6.21	203.45	1.27	10.67

G：*greedy* デコーダ，語順：語順優先探索デコーダ，
単：単語ベース，句：フレーズベース，F_Z：F_Z 制限付き

力英文が長い場合に *BLEU* 値の向上へとつながったことが分かった．特に語順優先探索デコーダにおいてはより顕著にその効果が見られる．長いテストセットではより多くの F_Z 単語の挿入をする必要があるために，特に長いテストセットで F_Z 制限の効果が現れた．

フレーズベースにおける入力英語長ごとのテストセットに対する *BLEU* を**図 9** に示す．各デコーダともに単語ベースに比べ *BLEU* 値が向上していることが分かる．また表 4 から分かるように各デコーダともに単語ベースに比べ，フレーズベースにすることにより翻訳時間も飛躍的に向上している．フレーズベースにおいても *greedy* デコーダより語順優先探索デコーダがより *BLEU* 値が高いことが分かる．また F_Z 制限を加えても単語ベースの場合ほどの差はあまり見られなかった．これはフレーズを利用することである程度 F_Z 制限の効果がすでに入っていると考えられる．

各デコーダの特性が確認しやすい翻訳例を**表 5** に示す．1 つ目の翻訳例ではあまり単語の並べ替えが起こっていないことから *greedy* デコーダは翻訳に失敗している．2 つ目の翻訳例では “*were sent in to*”，“*support us*” がフレーズとなって翻訳精度の向上に貢献している．しかしながらフレーズベースでも *greedy* デコーダではあまり単語の並べ替えが起こらずに失敗している．3 つ目の翻訳例では “氷” の F_Z リストに “の ように” があり，全体的にスムーズな翻訳結果に至っている．

表 5 翻訳例
Table 5 Example of translation.

入力	the drug will take effect in a few minutes
正解：	数分 で 薬 が 効い て くる
単：*greedy*	麻薬 数分 かかる 効果
単：語順	その 薬 の 効果 は 数分 かかる
句：*greedy*	その 薬 は 数 分 で 実施 さ れる 見通し
句：語順	その 薬 は 数 分 で 実施 さ れる 見通し
入力	Mounted troops were sent in to support us
正解：	騎馬 隊 が われわれ の 支援 に 送りこま れ た
単：*greedy*	騎馬 隊 派遣 の 支持 者 たち
単：語順	騎馬 隊 を 派遣 し た 指示 者 たち
句：*greedy*	騎馬 隊 の ため 送り込まれた 我々 を 支持 する
句：語順	騎馬 隊 は われわれ を 支援 する ため 送りこま れ た
入力	immerse one’s hand in the icy cold water
正解：	氷 の ように 冷たい 水 に 片手 を 突っこむ
単：語順	冷たい 水 は 氷 に 手 を つける
句：語順	冷たい 冷たい 水 に 自分 の 手 を つける
F_z：単：語順	氷 の ように 冷たい 水 に 手 を つける
F_z：句：語順	氷 の よう に 冷たい 水 に 自分 の 手 を つける

6. おわりに

本論文ではモデルは従来の *IBM* モデルを使い，デコーダの改良で翻訳精度の向上を試みた．比較的近い言語を基本としていた Germann らが提案した *greedy* を語順決定の *greedy* と訳語決定の *greedy* の 2 重ループにすることで語順の大きく違う英日翻訳に適するように拡張した．これにより *BLEU* 値を向上させることにつながった．また，各訳語に F_Z 単語列リストを持たせて個々のリストを小さくし，かつ網羅的に F_Z 単語列を持つようにする F_Z 制限を提案した．単語ベース語順優先探索デコーダでは長いテストセットに対して顕著にその効果が見られた．ただし，フレーズベースに拡張した際にはフレーズは F_Z 単語を含んでいる

ことからあまりその効果が見られなかった．

今後の課題としてビーム探索法[10]やスタックデコーダ[7]との比較が考えられる．また，正解に対して翻訳結果が短すぎることから長さを考慮する必要があると考えられる．特に F_Z 単語の挿入がより促進されることが必要である．

謝辞 （独）情報通信研究機構知識創成コミュニケーション研究センターの内山将夫主任研究員には本研究を進めるにあたり様々なアドバイスをいただきました．ここに記して感謝いたします．

参 考 文 献

1) Berger, A.L., Brown, P.F., Pietra, S.A.D., Pietra, V.J.D., Gillett, J.R., Kehler, A.S. and Mercer, R.L.: Language translation apparatus and method of using contex-based translation models, United States patent, patent number 5510981.
2) Brown, P.F., Pietra, S.A.D. and Pietra, V.J.D.: The Mathematics of Statistical Machine Translation: Parameter Estimation, *Computational Linguistics*, Vol.19, No.2, pp.263–311 (1993).
3) Collins, M.: Head-Driven Statistical Models for Natural Language Parsing, *Computational Linguistics*, Vol.29, No.4, pp.589–637 (2003).
4) Germann, U., Jahr, M., Knight, K., Marcu, D. and Yamada, K.: Fast Decoding and Optimal Decoding for Machine Translation, *Proc. 39th Annual Meeting on Association for Computational Linguistics*, pp.228–235 (2001).
5) NIST: Automatic Evaluation of Machine Translation Quality Using N-gram, Co-Occurrence Statistics. http://www.nist.gov/speech/tests/mt/
6) Och, F.J. and Ney, H.: The Alignment Template Approach to Statistical Machine Translation, *Computational Linguistics*, Vol.30, No.4, pp.417–449 (2004).
7) Och, F.J., Ueffing, N. and Ney, H.: An Efficient A* Search Algorithm for Statistical Machine Translation, *Proc. Data-Driven Machine Translation Workshop*, pp.55–62 (2001).
8) Och, F.J.: Training of statistical translation models. http://wasserstoff.informatik.rwth-achen.de/Colleagues/och/software/GIZA++.html
9) Papineni, K., Roukos, S., Ward, T. and Zhu, W.J.: Bleu: a Method for Automatic Evaluation of Machine Translation, *Proc. 40th Annual Meeting on Association for Computational Linguistics*, pp.311–318 (2002).
10) Tillmann, C. and Ney, H: Word Reordering and a Dynamic Programming Beam Search Algorithm for Statistical Machine Translation, *Computational Linguistics*, Vol.29, No.1 (2003).
11) Watanabe, T., Sumita, E. and Okuno, H.G.: Chunk based Statistical Translation, *Proc. 41st Annual Meeting of the Association for Computational Linguistics*, pp.303–310 (2003).
12) Wu, D.: A Polynomial-Time Algorithm for Statistical Machine Translation, *Proc. 34th annual meeting on Association for Computational Linguistics*, pp.152–158 (1996).
13) Yamada, K. and Knight, K.: A Syntax based Statistical Translation Model, *Proc. 39th Annual Meeting on Association for Computational Linguistics*, pp.523–530 (2001).
14) 内山将夫，井佐原均：日英新聞の記事および文を対応付けるための高信頼性尺度，自然言語処理，Vol.10, No.4, pp.201–220 (2003).

(平成 18 年 1 月 5 日受付)
(平成 18 年 9 月 14 日採録)

岩越 隼人

昭和 55 年生．平成 18 年筑波大学大学院システム情報工学研究科修士課程修了．同年朝日新聞社（株）入社．現在，同社製作本部社員．

山本 幹雄（正会員）

昭和 61 年豊橋技術科学大学大学院修士課程修了．同年（株）沖テクノシステムズラボラトリ研究開発員．昭和 63 年豊橋技術科学大学情報工学系教務職員．平成 4 年同助手．平成 7 年筑波大学電子・情報工学系講師．平成 10 年同助教授．博士（工学）．自然言語処理，音声言語情報処理の研究に従事．電子情報通信学会，言語処理学会，人工知能学会，音響学会，ACL 各会員．