英日統計的機械翻訳における構文解析木を用いた単語並び替え制約の検討

岩越 隼人　　山本 幹雄

筑波大学 {hayato@milab.,myama@}is.tsukuba.ac.jp

1 はじめに

統計的機械翻訳において翻訳候補の中から確率が最大になる候補を探索するのがデコーダの役割であるが、候補数が膨大になるので全解探索は不可能である。そのため、探索範囲をしぼって準最適解を見付けられるような greedy デコーダ [1]、DP 探索デコーダ [4] などが研究されて来た。これらのデコーダでは妥当な制約によって探索範囲を合理的に絞り込むことが重要である。

最近では翻訳候補を絞り込むために構文構造を明示的に利用する手法が注目されている。一般にいかなる言語においても単語の依存関係を表す構文木を作った場合に木の枝が交差することはほとんどないと言われている。また、翻訳の前後で文の意味は変わらないことから単語の依存関係は保存される。この 2 つのことから原言語文における構文木の枝が交差することのないように単語の並び替えを限定しても目的言語の構文木へと変換可能となる。この考え方は 2 分木を利用した ITG [7] や Syntax ベースのデコーダ [3] などで利用されている。しかしながら、これらのデコーダは独自の確率モデルを用いていることが多く、デコーダ部分だけの比較が困難である。本稿では翻訳モデルを一般的な $IBM\ model$ [2] としたデコーダにおいて構文構造を利用することを検討する。以下、まず最初に木構造による制約の下での単語並び替えを定量的に検証した後、実際のデコーダを作成・比較し構文構造利用の有効性を示す。

2 木構造による制約の有効性に関する予備実験

2.1 予備実験手順

本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び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 $GIZA++$ [5] 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並び替える。このとき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び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 分木による並び替え [7]
  - 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び替える。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び替え [3]
  - 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。ここではパーザーとして $Apple\ Pie\ Parser$ [8] を使用した。次に構文解析木のノードを反転させることで単語を並び替える。任意の 2 分木による並び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。以下、この制約を「構文木制約」と呼ぶ。

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 [6] を算出し最も高い $BLEU$ 値 を出力する。1 箇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 1 にノードの反転による並び替えの様子と日本語の語順をした正解の例を示す。

2.2 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。英文の長さ毎の平均 $BLEU$ 値 を図 2 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 分木を用いた並び替えでは $BLEU$ 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び替えでは $BLEU$ 値 が約 0.6 前後と低くなってしまった。この原因の多くはアライメント又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び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。任意の 2 分木のを用いれば精度が高いが並び替え候補は膨大になってしまい、デコーダに利用する際に翻訳時間という点で有望ではない。以下では構文解析木による並び替え (構文木制約)

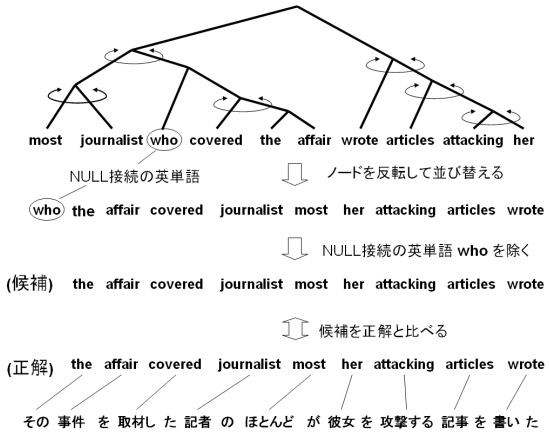

図 1: 木構造のノードの反転による並び替えと正解との比較

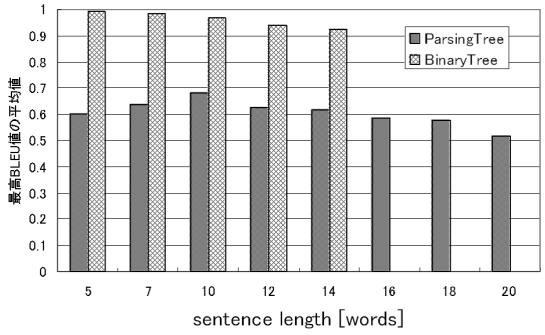

図 2: 候補中の最高 $BLEU$ 値 の平均値

を実際のデコーダに組み込んだ翻訳実験について述べる。

3 構文解析木を利用したデコーダ

3.1 試作したデコーダの概要

構文木を利用した並び替え制約を 4 節で評価するために以下の 4 つのデコーダを試作した。以下の節で詳しく述べる。

- [1] を模した $Greedy$ デコーダ (3.2 節)
- 並び替えに構文木制約をつけた $Greedy$ デコーダ (3.3 節)
- 並び替え (構文木制約付き) と訳語決定の探索を分離した $Greedy$ デコーダ (3.4 節)
- 上記の訳語決定の部分を DP としたデコーダ (3.5 節)

3.2 局所探索デコーダ

greedy デコーダ [1] を模したデコーダを作成した。以下、これを局所探索デコーダと呼ぶ。近傍（子）の作り方は以下である。

- 訳語を 1 箇所もしくは 2 箇所変更する
- 訳語を 1 箇所変更し、同時に $fertility$ が 0 の単語 (列) を追加・変更する
- $fertility$ が 0 の単語 (列) を 1 つ削除する
- 任意の 2 つの重ならない部分単語列を置換する

上記の操作で作り出される全てのパターンだけ子を作る。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。

3.3 構文解析木を考慮した局所探索デコーダ

前節で説明した局所探索デコーダでは任意の単語列を入れ替えるが、この部分を入力英文の構文解析木のノードの反転による単語の並び替えに限定する。近傍 (子) の作り方は前節の局所探索デコーダと以下の部分を変更する。

・ 任意の 2 つの重ならない部分単語列を置換する
→・ 親のノードの状態 (正 or 反転) を 1 箇所だけ変えて単語列を並び替える

この方法を以下「構文木局所探索デコーダ」と呼ぶ。

3.4 構文木局所探索×局所探索デコーダ

単語を並び替える局所探索と訳語を決定する局所探索の 2 重ループによって探索を行う。並び替え部と訳語決定部を分けて操作することで単語の並び替えを強制的に行うようにする。具体的な手順を以下に示す。

- 並び替えの局所探索
  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
- 訳語決定の局所探索
  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全ての子の中で最も高い確率が親の確率よりも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
  （a）親の訳語を 1 箇所変更する

(b) $fertility$ が 0 の単語 (列) を 1 箇所追加・変更する
(c) $fertility$ が 0 の単語 (列) を 1 箇所削除する

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全てのノードをそのまま、2 回目の初期状態として全てのノードを反転、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。

3.5 構文木局所探索× DP デコーダ

前節と大まかな動きは同じであるが訳語決定には DP を使用する。単語の並びを決定してから訳語を選ぶので比較的単純な DP を用いて効率良く訳語の最適解を求めることができる。

4 評価実験

4.1 学習コーパスとテストセット

3 節で述べた 4 つのデコーダと $ISI\ Rewrite\ Decoder(thorough \quad greedy)$ [1] について翻訳の時間と精度を測定した。翻訳モデルは英日対訳コーパスから $GIZA++$ [5] を用いて $IBMmodel4$ を学習した。言語モデルは対訳コーパスの日本語部分から CMU ツールキットを用いて $trigram$ モデルを学習した。学習データの詳細を以下に示す。

- 対訳コーパス約 47 万文ペア (英:約 540 万単語, 日:約 675 万　単語)
- 内訳:英日辞書例文 31 万文ペア、読売新聞 $DailyYomiuri$13 万文ペア [10]、プロジェクト杉田玄白 2 万文ペア [11]、日経ビジネスレター 1.6 万文ペア

あらかじめ対訳コーパスのランダムな 9 割を翻訳モデルの学習用に、残りの 1 割をテスト用に分けた。さらに、テスト用の対訳コーパスの中から長さ 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 文ずつ抜きだし、$open\ test\ set$ を作った。各対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。また構文解析木を利用するためにパーザーとして $Collins\ parsr$ [9] を利用した。各デコーダでは各入力単語の訳語として上位 10 個を考慮する。また $fertility$ が 0 の単語列は上位 50 個使用した。

4.2 実験結果と考察

各デコーダと $ISI\ Rewrite\ Decoder$ の $BLEU$ 値と翻訳時間をそれぞれ図 3、表 1 に示す。図 3 中の単語並び替えに構文木制約を使う単純なデコーダ (構文木局所探索デコーダ) と使わない単純なデコーダ (局所探索と $ISI\ Rewrite\ Decoder$) の $BLEU$ 値がほぼ同等であることから、構文木制約の探索範囲は妥当であることが分かる。しかし、表 1 より構文木制約を用いても (構文木局所探索)、用いなくても (局所探索)、翻訳時間に違いはない[1)]。これはどちらのデコーダも親から生成される子供のうち、訳語変更による分が圧倒的多数を占めるためである。例えば長さ 10 単語入力で単語の並び替えに関する子の割合は 1%程度である。また、これらのデコーダは訳語決定と単語の並び替えを同時に行うため訳語決定が進めば単語の並び替えが起こりにくい傾向が見られた。英語日本語間では比較的大きく単語が移動することからこの点が問題となってくる。

単語数	ISI	局	木局	局×局	局× D
5-6	0.24	1.96	1.90	1.26	16.87
7-8	0.55	5.84	5.61	6.58	56.00
9-10	1.21	12.94	12.19	22.24	120.91
11-12	2.20	23.79	23.75	58.53	235.49
13-15	3.93	40.58	37.41	109.02	526.27
15-16	6.57	70.14	68.73	216.66	1209.46
17-18	11.03	110.39	102.60	293.42	1575.88
19-20	15.96	167.78	151.70	552.63	1469.74

表 1: 英文 1 文あたりの翻訳時間（秒）
局所探索 : 局、構文木局所探索 : 木局、構文木局所探索×局所探索 : 局×局、構文木局所探索× DP : 局× D と略記する

訳語決定と単語の並び替えを分離したデコーダ (構文木局所探索×局所探索、構文木局所探索× DP) は分離しない場合と比べ $BLEU$ 値がかなり上がっていることがわかる。これは単語の並び替えを促進させた影響であると考えられる。また訳語決定と単語の並び替えを分離すると局所探索の 2 重ループになってしまい翻訳時間が長くなってしまうが構文木局所探索×局所探索は局所探索の 3 倍程度である。

また、構文木局所探索× DP と比べ構文木局所探索×局所探索の精度がさほど落ちないことから訳語決定だけであれば局所探索でも十分である。それよりも単語の並び替えが本実験の対象 (英日)・レベルでは重要性が大きい。

4.3 翻訳例

局所探索、構文木局所探索×局所探索、構文木局所探索× DP デコーダの翻訳例を表 2 に示す。翻訳例 1 の局所探索は単語の並び替えが 1 回も起こらず翻訳に失敗している。翻訳例 2 の局所探索の翻訳結果では任意の単語列を置換することにより ” 空港の周囲 ” と ” 騒音 ” の目的語の間に ” 減らす ” という動詞が入っており構文構造がおかしくなっている。他のデコーダでは構文解析木を考慮している

[1)] $ISI\ Rewrite\ Decoder$ は非常に高速だがインプリメント技術等の差もあると考えられるので時間的比較には用いなかった

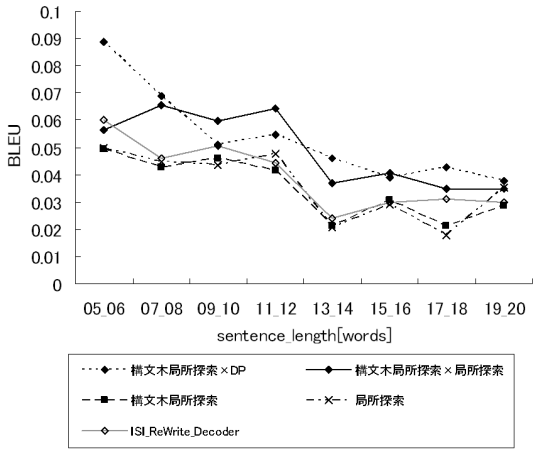

図 3: 各デコーダの BLEU 値

翻訳例 1	
英語文	a game of mixed doubles
正解文	混合 ダブルス の 試合
局	試合 は 混合 ダブルス
局×局	混合 ダブルス の 試合
局× D	混合 ダブルス の 試合
翻訳例 2	
英語文	reduce noise disturbance around an airport
正解文	空港 周辺 の 騒音 を 減らす
局	空港 の 周囲 を 減らす 騒音 を かき乱す
局×局	空港 周辺 の 騒音 騒動 を 減らす
局× D	空港 周辺 の 騒音 障害 を 減らす
翻訳例 3	
英語文	the elephant gave a loud trumpet
正解文	象 が 大きな 鳴き声 を あげ た
局	騒々しい トランペット その 象 を 与え た
局×局	騒々しい トランペット を 与え た ゾウ
局× D	声高 に 自慢 し た ゾウ

表 2: 各デコーダの翻訳結果例(表記については表 1 を参照)

のでそのような並び替えを回避できている。しかし、翻訳例 3 のように意味不明な翻訳結果はまだまだ多く、慣用的な表現にはほとんど対応できていないことがわかる。

5 おわりに

英日翻訳において木構造を利用した並び替えで日本語の語順を生成できるか検証してみたところある程度有効であることが分かった。また構文解析木を用いていくつかのデコーダを作成し実験してみたところ、構文解析木のノードを反転させて並び替えることで、任意の単語列を置換して並び替える操作の無駄を省くことができた。また訳語変更と単語の並び替えを同時に行う局所探索では訳語変更による近傍が選ばれがちであり、単語の並び替えがあまり起こらない。訳語決定と単語の並び替えを別に処理することで強制的に並び替えを行うと、翻訳精度の向上につながった。今後の課題としては今回試すことができなかった DP によるデコーダ (単語並び替えも行う) [4] と比較することである。

謝辞

対訳コーパスの一部を提供して頂いた情報通信研究機構の内山将夫氏に厚く御礼を申し上げます。

参考文献

[1] U.Germann, M.Jahr, K.Knight, D.Marcu, K.Yamada. Fast Decoding and Optimal Decoding for Machine Translation. Proc. of the Conference of the Association for Computational Linguistics, 2001

[2] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra. The Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics, 25. pp.263-311, 1993.

[3] Kenji Yamada, Kevin Knight. A Syntax based Statistical Translation Model. Proc. of the Conference of the Association for Computational Linguistics. pp.303-310, 2002.

[4] Christoph,Tillmann., Hermann,Ney. Word Reordering and Dynamic Programming Beam Search Algorithm for Satistical Machine Translation. Computational Linguistics, 29. pp.97-133, 2003.

[5] Franz,Josef,Och. Training of statistical translation models. http://wasserstoff.informatik.rwth-aachen.de/Colleagues/och/software/GIZA++.html

[6] Papineni,K., Roukos,S., Ward, T. and Zhu,w.-J. Bleu:a Method for Automatic Evaluation of Machine Translation. Proc. of the Conference of the Association for Computational Linguistics. pp.311-318, 2002.

[7] Dekai Wu. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. Computational Linguistics 23(3). pp.377-403, 1997.

[8] S. Sekine, R. Grishman. A corpus-based probabilistic grammar with only two non-terminals. Fourth International Workshop on Parsing Technology 1995.

[9] Michael Collins. Head-Driven Statistical Models for Natural Language Parsing. Computational Linguistics. pp. 589-637, 2003.

[10] 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, 10:4. pp.201-220, 2003.

[11] 内山将夫. 日英対訳文対応付けデータ. http://www2.nict.go.jp/jt/a132/members/mutiyama/align/index.html