英日翻訳における語順について

磯崎　秀樹
日本電信電話株式会社
NTT コミュニケーション科学基礎研究所

英語と日本語の語順が大きく違うことが、英日翻訳が困難な理由の一つであることは、よく知られている。階層的句を用いた統計的機械翻訳を用いても、大きな語順の変更は難しく、語順の問題は十分解決できていない。英文の書き換えによって英日の語順の相関がどのように変化するか調べたので、その実験結果について報告する。

1　はじめに

英語は代表的な SVO (Subject-Verb-Object) 型言語で、日本語は代表的な SOV 型言語である。世界の言語の多くは、このどちらかの型に属する。SVO 型と SOV 型では、語順がまったく違うが、従来の統計的機械翻訳 (SMT) は、このような大きな語順の違いに十分対処ができなかった。

既存手法としては、構文解析や依存構造解析の解析結果を用いて、順序を入れ替えるかどうか機械学習する方法 [3, 4] がある。

一方、SVO 型言語から SOV 型言語への統計的機械翻訳に特化した前処理として、SVO 型言語の文の語順をルールベースの前処理で SOV 型に書き換える方法 [1, 2] が、最近相次いで発表されている。これらの手法は、いずれも、SVO 型の文の用言を最後に持っていくことによって、語順を SOV 型に近づける。

たとえば Peng Xu らの方法 [1] では、head の品詞 L、重み W (= 0, ±1,±2)、向き O (= NORMAL or REVERSE) からなる (L, W, O) という 3 つ組みのルール 20 個で英語の依存構造木を書き換える。この手法では、

```
Living is exciting because we do n't know
what the future has .
```

という文は

```
because we the future what has know n't do
Living exciting is .
```

となる。これは

我々 は 未来 が 何 を 持っている か わから ない ので 人生 は エキサイティング だ 。

という直訳の語順に近い。ただし、because と「ので」の位置は大きく異なる。

Gumwon Hong らの手法 [2] も Xu らの手法に似ているが、英語にない助詞に相当する疑似語 (pseudo word) を加えるところが大きな違いである。助詞推定については、鈴木久美ら [5] も提案を行っている。

2　提案手法

本稿で提案する手法は、機械学習ではなく、ルールベースの手法である。ただし、これまでの手法が、日本語を「SOV 型言語」ととらえていたのに対して、「**head-final 型言語**」ととらえる。SOV 型というと用言の位置にだけ注目してしまうが、head-final 型という表現では、用言かどうかではなく、head かどうかが重要になる。これによって、書き換えルールが、品詞によらず、次の一つだけになる。

> Q. 英文を **head-final 型言語**の語順に近づけるためにはどうしたらよいか？
>
> A. **head を最後に持っていけばよい**。もう少し正確に言えば、構文木の各非終端ノードにおいて、その子ノードのリストの中で、head を最後に移動すればよい。

用言という制約が外れた影響は大きい。たとえば前述の英文の because 節の head は because である。したがって、because は because 節の最後に移動する。これはちょうど、日本語の「ので」の位置に相当する。

以下では、このように、head を後ろに移動した英文のことを **Head-Final English (HFE)** と呼ぶ。HFE を実現するため、英文を enju 2.3.1[6] で構文解析し、その XML 出力を読んで、トップダウンで再帰的に head の位置を移動するプログラムを作成した。

たとえば、John saw a beautiful girl yesterday. という文を enju で解析した結果を S 式で略記すると

```
(S (NP John)
   (VP* (VP* saw*
             (NP a (NX beautiful girl*)))
        (NP yesterday)))
```

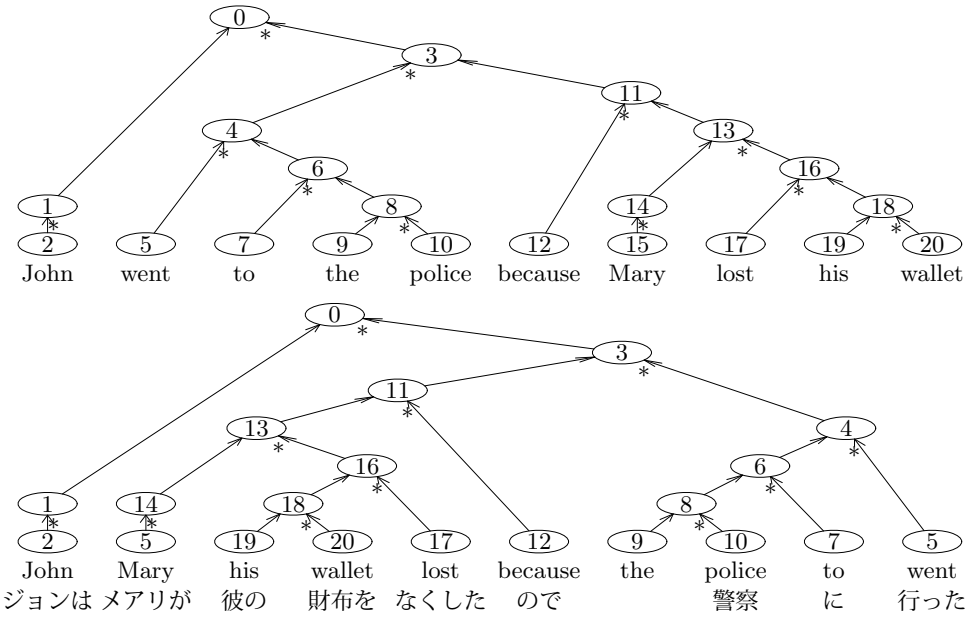

図 1: because を含む文の構文木を head (*印) の後方への移動により書き換える

となる。ここで*は head を表す。それぞれの非終端ノードにおいて、head を後ろに持っていくと、

```
(S (NP John*)
   (VP* (NP yesterday)
        (VP* (NP a (NX beautiful girl*))
             saw*)))
```

したがって、HFE は以下のようになる。

```
John yesterday a beautiful girl saw .
```

これは以下の直訳と同じ語順である。

ジョン は 昨日 美しい 少女 を 見た 。

助詞に相当するものがないので、助詞相当語として、動詞の arg1（「は」「が」に対応）を表す_va1 と、arg2（「を」「に」に対応）を表す_va2 を入れる。また、冠詞 the, a, an はほとんどの場合、日本語に相当するものがないので削る。これによると、

```
John went to the police because Mary lost
his wallet.
```

は図 1 上のように解析されるので、HFE は

```
John _va1 Mary _va1 his wallet _va2 lost
because police to went .
```

となる。これを逐語訳すると

ジョン は メアリ が 彼の 財布 を なくした
ので 警察 に 行った 。

となって、自然な日本語が得られる。

3 実験の設定

提案手法の効果を調べるために、内山らの読売新聞対訳データ [7] のうちの 5 万文を使って、以下の要領で HFE の語順と日本語の語順の相関を調べた。

- 英文は enju で解析して HFE に変換。日本語は mecab で単語分割。ただし、単語の過分割をさけるため、数字を中心に、書き換え処理をしている。
- HFE を英文の代わりに作った対訳データに対して、GIZA++[8] を適用し、トレーニングデータの各文について単語対応を表すファイル (A3.final.gz) を求める。なお、単語の対応がなるべく正しく推定されるように、対訳データに現れる 4.2 万語の英語の訳を ALT-J/E の翻訳辞書から得て加えた。
- A3.final.gz のうち、翻訳辞書部分を除く各文について、順位相関係数の一種、Kendall の τ を計算し、τ の平均 $\bar{\tau}$ を求める。

Kendall の τ について説明しておく。n 個の要素のリストに対して、そのリストに含まれる各要素 2 個に注目したときに、そのペアが昇順に並んでいるかいなかに

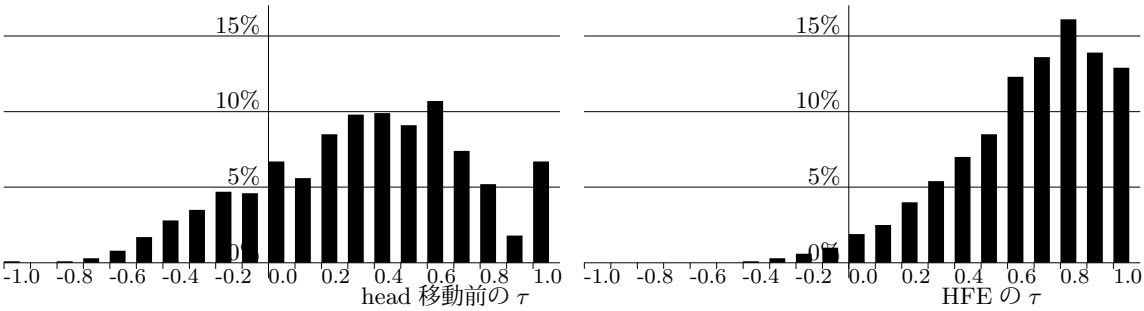

図 2: 順位相関係数 τ の分布

よって τ は以下のように定義される。

$$\tau = \text{昇順ペアの数/全ペアの数} \times 2 - 1$$

たとえば、[4,2,1,3] というリストを考えると、全ペア数は ${}_4C_2 = 6$ である。このうち、昇順のものは (2,3) と (1,3) の 2 つしかないので、$\tau = 2/6 \times 2 - 1 = -0.333$ となる。リストの長さが 1 以下の場合は、τ に意味がないので、$\bar{\tau}$ の計算に含めない。

また、ここでは単語の対応を自動で求めているが、1 文中に同じ単語が複数回出現していると、日本語のひとつの語に複数の単語が対応づけられてしまうことがある。このような単語は、τ の計算に含めない。

なお、他のデータを用いた予備実験の結果から、以下の微修正を加えている。

- A and B などの並列構造（コーディネーション）が B and A と書きかえられるのは問題なので、コーディネーション・ノードの直下の head は移動させない。（enju の出力には、コーディネーションが明示されている。）
- 丸括弧（）でくくられた部分は削除。
- enju は受動態の文では、意味を考慮して arg1 と arg2 を入れ替えてしまうが、　忠実な翻訳という観点から、arg1 と arg2 を構文通りに戻しておく。
- 日本語の助動詞を生成するためのヒントとして、英語の動詞の直後にその品詞を補助語_VBG 等として出力。
- 連体修飾されている単語で助詞相当語が競合する場合、外側を優先。たとえば John bought a toy that was popular in Japan. の toy は、bought の arg2 だが、was の arg1 でもあり競合する。この場合は外側の_va2 を採用。
- 「は」に対応することの多い文の最初の_va1 を_va0 に、訳されないことの多い be 動詞の_va2 を_va3 に書き換え。

表 1: HFE と日本語の語順の順位相関係数の平均値

採点対象	$\bar{\tau}$
head 移動なし	0.366
HFE	0.686
HFE のうち日本語が 30 語以下の文 (64%)	0.719

4　実験結果

実験結果を表 1 に示す。全体では $\bar{\tau} = 0.686$ となった。語数が多いほど構文解析を失敗しやすいので、採点対象を日本語が 30 語以下のものに制限すると、$\bar{\tau} = 0.719$ と若干良い。

τ の分布は図 2 の通りである。左は head を移動させなかったときの分布であり、右は移動させた HFE における分布である。HFE では、τ が 1.0 に近づいてきていることがわかる。

τ が低い文を調べると、以下の問題が見うけられた。

- 自動で求めた単語対応の誤り。
- 長い文。とくにコンマなどで区切られた長い文で、構文解析が失敗している。
- 時間表現。during ... や last autumn のような時間を表す表現が、日本語では文頭に来ているが、HFE では文の途中に現れるために τ が低い。
- 数値。hundreds of people や 30 to 50 など、数値を含む表現には、head を移動しない方がよいパターンがある。
- 倒置。たとえば日本語が「ジョンが〜した。」となっていれば τ が高いのに、「〜したのはジョンだ。」となっているために τ が低い。
- 疑問文。Did John ...?の場合、Did が文末に移るが、enju の出力では (Did John) で VX（不完全な動詞句）という非終端ノードが作られるので、Did が主語 John を文末に連れて行って、... John did

? という HFE になる。多くの場合「ジョンは…しましたか？」となっているので τ が低い。

- 説明の追加。読売新聞データは、日本語を英語に訳したものであるため、日本語にない説明が英語で加わっていることがある。

5 考察

enju は豊富な情報を出力してくれるので、今回必要な情報は苦労せずに得られたが、提案手法の汎用性を調べるため、別の構文解析器として Charniak & Johnson の reranking parser [9] (以下 CJ) を利用してみた。HFE を生成する、という観点から見ると、CJ と enju には以下の違いがあることがわかった。

- CJ は head か否かの情報を直接出力しないが、McClosky の PyInputTree [10] を使えば得られる。
- CJ は arg1, arg2 の情報を出力しないので、自分でカバーしなければならない。
- enju の出力は 2 分木だが、CJ は Penn Treebank 準拠の比較的フラットな構造を出力する。

2 分木にするために enju が挿入する中間ノードは、案外重要である。たとえば、Xu の手法 [1] では、

```
John hit the ball but Sam threw the ball.
```

という英文が

```
John the ball but Sam the ball threw hit.
```

となってしまうので、これを避けるため、

```
John the ball hit but Sam the ball threw.
```

となるように、移動範囲が句読点や接続詞を越えない、という制限を加えているが、提案手法では、enju の出す中間ノードが移動範囲の制約になっていて、このような制限を加えなくてよい。

6 まとめ

enju を用いて英文を構文解析し、head-final 化して得られる Head-Final English は、日本語の語順と高い相関を持っていることが判明した。なお、すでに述べたように、単語対応は自動で求めているので間違いがあるし、複数の単語に対応しているものは計算から除外しているので、人手で単語対応を調べれば、もっと正確な評価ができるであろう。また、機械学習を用いた既存手法との比較や、翻訳結果の性能向上については、別の機会に報告したい。

謝辞

統計翻訳システムについて、渡辺太郎、須藤克仁、塚田元の各氏に、enju について、東京大学辻井研の宮尾祐介氏に相談にのっていただきました。感謝いたします。

参考文献

[1] Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och: Using a Dependency Parser to Improve SMT for Subject-Object-Verb Languages, Proc. of NAACL-HLT 2009, pp.245–253, 2009.

[2] Gumwon Hong, Seung-Wook Lee, and Hae-Chang Rim: Bridging Morpho-Syntactic Gap between Source and Target Sentences for English-Korean Statistical Machine Translation, Proc. of ACL-IJCNLP 2009 Short Papers, pp.233–236.

[3] Kenji Yamada and Kevin Knight: A Syntax-based Statistical Translation Model, Proc. of ACL-2001, pp.523–530.

[4] Chris Quirk et al.: Dependency Treelet Translation: Syntactically Informed Phrasal SMT, Proc. of ACL-2005, pp.271–279.

[5] 鈴木久美 and Kristina Toutanova: 機械翻訳における日本語格助詞の生成、言語処理学会年次大会発表論文集, pp.384–387, 2007.

[6] Tsujii Laboratory: Enju – A Practical HPSG parser, http://www-tsujii.is.s.u-tokyo.ac.jp/enju/, 2007.

[7] Masao Utiyama and Hitoshi Isahara: Reliable Measures for Aligning Japanese-English News Articles and Sentences. Proc. of ACL 2003, pp.72–79.

[8] Franz Josef Och and Hermann Ney: A Systematic Comparison of Various Statistical Alignment Models, Computational Linguistics, Vol. 29, No. 1, pp.19–51 March 2003.

[9] Eugene Charniak and Mark Johnson: Coarse-to-fine n-best parsing and MaxEnt discriminative reranking, Proc. of the ACL 2005, pp.173–180.

[10] David McClosky: PyInputTree, A Pythonic wrapper of Eugene Charniak's InputTree structure from his parser using SWIG. http://www.cs.brown.edu/~dmcc/software/PyInputTree/