翻訳自動評価と日本語の語順の自由度の関係について

磯崎 秀樹　高地 なつめ
岡山県立大学
isozaki@cse.oka-pu.ac.jp

平尾 努
日本電信電話株式会社
hirao.tsutomu@lab.ntt.co.jp

要旨

機械翻訳の改善には、翻訳の品質を評価することが重要である。しかし、人手で評価を行うと時間がかかるので、「翻訳自動評価法」が利用される。欧米言語間の翻訳自動評価では、BLEU がデファクト・スタンダードだが、日英や英日のような、語順のまったく異なる言語の間の翻訳では、BLEU と人手評価との相関が非常に低い。そこで以前我々は、RIBES（ライビーズ）という翻訳自動評価法を考案した。RIBES は人間が作成した手本となる訳（参照訳）と機械訳の語順の近さに基づいて翻訳品質を評価するもので、日英や英日の特許翻訳で人手評価と高い相関がある。しかし、英日翻訳では、日本語の文を比較することになり、日本語の語順が比較的自由という常識と矛盾しているのではないか、という疑問が生じる。そこで本稿では、参照文を自動で増やす方法を提案する。

1　はじめに

統計的機械翻訳 [6] では、翻訳品質を BLEU[7] という自動評価法で測定することが一般的であり、BLEU のスコアが高くなるようにチューニングを行う。しかし、日英や英日のような語順のまったく異なる言語の間の翻訳品質評価では、BLEU と人手評価の相関がきわめて低く、BLEU がよくなるようにチューニングしても、人間にはよくなったように見えない。

そこで我々は、RIBES（ライビーズ）[5, 8] という翻訳自動評価法を考案し、NTCIR-7 の特許翻訳タスクのデータ [1] でその性能を確認した。この手法では、参照訳と機械訳の語順の近さを順位相関係数 Kendall の τ で測定する。

NTCIR-9, 10 の特許翻訳の概観論文 [3, 2] によれば、NTCIR-7 に比べて翻訳機の性能は全体的に向上してきているが、RIBES は日英でも英日でも、人手評価との高い相関を維持している。

しかし、ここで一つの疑問がある。NTCIR の自動評価では、基本的に参照訳が一つしか使われていないのに、RIBES は人手評価と高い相関を示すのである。BLEU による評価では、参照訳が複数必要とされていることを考えると、これは不思議である。

しかも、英日翻訳の場合、日本語での語順の評価となるが、日本語は語順が比較的自由と言われている。たとえば「太郎がパリでフランス料理を食べた。」は以下のように 6 通りの入れ替えが可能である。

1. 太郎が/パリで/フランス料理を/食べた。
2. 太郎が/フランス料理を/パリで/食べた。
3. パリで/太郎が/フランス料理を/食べた。
4. パリで/フランス料理を/太郎が/食べた。
5. フランス料理を/太郎が/パリで/食べた。
6. フランス料理を/パリで/太郎が/食べた。

このことから、英日翻訳で参照訳の日本語がたったひとつであるのに、RIBES のような語順を評価する指標が人手評価と高い相関を示しているのは、不思議なことである。

2　係り受け木からの文の生成

前述の例のような語句の入れ替えがどうしてできるのか考えてみよう。「太郎がパリでフランス料理を食べた。」は、図 1 の係り受け木で表せる。この木を見ると、前掲の 6 通りの語順は、用言「食べた」を修飾する 3 つの文節「太郎が」「パリで」「フランス料理を」の順列 $3! = 6$ に他ならないことがわかる。

参照訳の係り受け木があれば、このような順列を自動生成して参照訳を自動で増やすことができるのではないか？そうすれば、これまで RIBES で低い点しか得られなかったが、人手評価ではよい訳に、高い点を与えられて、さらに人間の評価に近づくことができるかもしれない。

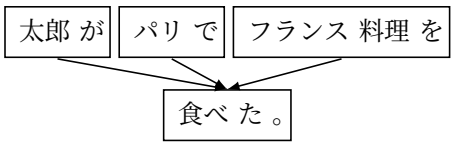

図 1: 「太郎がパリでフランス料理を食べた。」の係り受け木

2.1 POST-ORDER による文の生成

ここで日本語が Head Final な語順であることを思い出してみよう。Head とは修飾される語句であり、Head Final とは、修飾する語句をすべて出力したあとに、修飾される語句を出力する語順である。

係り受け木から Head Final な語順を生成しようとした場合、「**POST-ORDER な順序**」で出力すればよいことがわかる。POST-ORDER とは、木の走査において、子ノードをすべて処理してから親ノードを処理する順序のことである。

POST-ORDER では、子ノードをどういう順番で処理するかは決まらない。したがって、子ノードが n 個あれば、$n!$ 通りの順序が発生する。

もっと複雑な文「太郎が回転寿司でお寿司を食べたあとに歌舞伎を見た。」の係り受け木は図 3 のようになる。この場合、「食べた」を修飾する語句は「太郎が」「回転寿司で」「お寿司を」の 3 つ。「見た」を修飾する語句は「あとに」「歌舞伎を」の 2 つ。したがって、この係り受け木から POST-ORDER で出力できる Head Final な文は、以下の $3! \times 2! = 12$ 個となる。

1. 太郎が/回転寿司で/お寿司を/食べた/あとに/歌舞伎を/見た。○
2. 太郎が/お寿司を/回転寿司で/食べた/あとに/歌舞伎を/見た。○
3. 回転寿司で/太郎が/お寿司を/食べた/あとに/歌舞伎を/見た。○
4. 回転寿司で/お寿司を/太郎が/食べた/あとに/歌舞伎を/見た。○
5. お寿司を/回転寿司で/太郎が/食べた/あとに/歌舞伎を/見た。○
6. お寿司を/太郎が/回転寿司で/食べた/あとに/歌舞伎を/見た。○
7. 歌舞伎を/太郎が/回転寿司で/お寿司を/食べた/あとに/見た。×
8. 歌舞伎を/太郎が/お寿司を/回転寿司で/食べた/あとに/見た。×
9. 歌舞伎を/回転寿司で/太郎が/お寿司を/食べた/あとに/見た。×
10. 歌舞伎を/回転寿司で/お寿司を/太郎が/食べた/あとに/見た。×
11. 歌舞伎を/お寿司を/回転寿司で/太郎が/食べた/あとに/見た。×
12. 歌舞伎を/お寿司を/太郎が/回転寿司で/食べた/あとに/見た。×

右の○×は、第一著者が分かりやすい自然な文と感じたかどうかを表す。○はわかりやすい自然な文、×はわかりにくい不自然な文である。真ん中のあたりの数文については、判断が分かれるかもしれないが、最初の方がわかりやすく自然で、最後の方がわかりにくく不自然であることには合意していただけるものと期待する。

```
def allPerms(list):
  if len(list) <= 1: return [list]
  all = []
  for one in list:
    rest = [c for c in list if c != one]
    rper = allPerms(rest)
    all.extend([([one] + p) for p in rper])
  return all
```

図 2: 与えられたリストのすべての順列を生成する再帰的関数の Python による実装

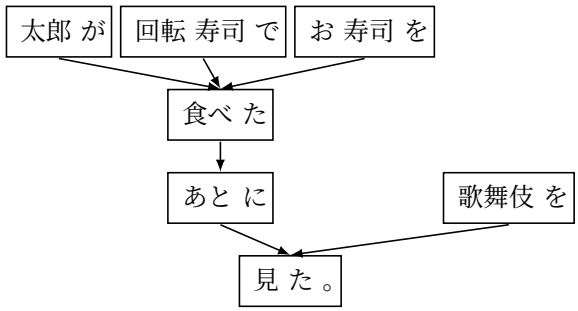

図 3: 「太郎が回転寿司でお寿司を食べたあとに歌舞伎を見た。」の係り受け木

これらの文は、図 2 のような順列生成関数により、係り受け木から自動生成できる。

2.2 生成された文の満たすべき制約

たとえば 9 番の「歌舞伎を/回転寿司で/太郎が/お寿司を/食べた/あとに/見た。」が不自然に感じられるのはなぜだろうか？

「歌舞伎を見た」の「歌舞伎を」が前すぎて、それを回収する述語「見た」が出てくる前に、同じ「を格」の「お寿司を」が出ている。そしてこの「お寿司を」は直後の「食べた」に回収されているが、先に出てきている「歌舞伎を」はまだ回収されていない。これが不自然に感じられる理由であろう。×印の他の文が不自然に感じられるのも同じ理由であろう。

同様に「が格」や「に格」などについても同じ現象が起きることが予想される。「太郎が回転寿司でお寿司を食べたあとに次郎が歌った。」であれば、「次郎が太郎が回転寿司でお寿司を食べたあとに歌った。」が生成されるはずだが、これはやはり不自然である。

このような不自然な文は、参照訳として利用することができない。そこで、このような不自然な文が生成されないように、制約を設ける。どのような制約を設ければよいだろうか？

ここでの問題は、ある連用修飾句 M が、本来修飾している用言 Y0 より前の用言 Y1 より前にあると、M の修飾先が Y1 であるように思えることにある。これを防ぐには、M を Y0 より前にある他の用言よりも前に出さないようにすればよい。

用言といってもいろいろあり、それぞれの用言によって取る格助詞が違う。たとえば、形容詞が「を格」を取ることは考えにくい。したがって、形容詞の前に動詞の「を格」が来ても、問題はないと予想される。たとえば「新しい家に犬を連れて行った。」を「犬を新しい家に連れて行った。」と並び替えても問題はない。

同じことは「を格」を取らない動詞にも適用できるだろう。たとえば「咲く」は「を格」を取らないので、「花が咲く前にその桜を見た。」を「その桜を花が咲く前に見た。」と言っても問題はない。どの動詞がどの格をとるかは、日本語語彙大系 [4] の構文体系のような辞書を利用することができる。

したがって、各用言を修飾する助詞句は、先行する他の用言で、同じ助詞句を取りうるものよりも前に出すべきではない、という制約が考えられる。これを「**助詞制約**」と呼ぶことにしよう。本稿では構文体系のような言語リソースに頼らない、単純な方法を用いる。

単純助詞制約

先行する用言の前に格助詞句を出してはいけない。ただし、「を格」の格助詞句は、先行する形容詞の前に出してもよい。

もちろん、用言の修飾句は、格助詞句だけではなく、他の助詞で終わるものや、用言の連用形で終わるもの、副詞などがあるが、複雑になるので、今回は語順の入れ替えから除外することにした。

ここでは、各用言において、同じ用言を修飾する「が」「を」「に」「で」「と」の格助詞句が連続している場合にのみ、その間で順序を入れ替える処理を行うプログラムを作成した。

このプログラムを図 3 の係り受け木に適用すると、前述の 1〜6 が生成され、7〜12 は生成されなかった。

3 実験

以上の語順入れ替えアルゴリズムを、京大コーパスに適用して、どれくらい入れ替えられるかを試してみたところ、引用符「」の扱いで問題が見つかった。京大コーパスでは、引用符は、その前後のどちらかの文節に組み込まれているが、それでは語順の入れ替えを行ったときに括弧が一緒に動くので、おかしなことになる。たとえば、以下のような引用文を含む文がある。

…/「政権に/影響を/及ぼす/ことには/ならない。/…

ここで〈「政権に〉と〈影響を〉を文節単位で入れ替えると、

…/影響を/「政権に/及ぼす/ことには/ならない。/…

となって、「影響を」が引用符からはみ出してしまう。

そこで、語順を入れ替える前に、引用符の前後で文節を区切りなおす処理を追加した。

最初の 1134 文に適用したところ、生成された文数の分布は表 1 のようになった。なお、原文が制約を満たしていないことがあるが、ここではかならず出力している。これは、すでに述べたように、ここでは「単純助詞制約」を利用しているからである。

生成された語順の数	1	2	3	6	7	8	24	25
原文の数	804	225	26	37	8	1	2	1

表 1: 「単純助詞制約」による京大コーパスの語順の入れ替え

これによると、原文以外の語順が生成されたのは 330 文で、1134 文の 29%であった。つまり、71%の文では、語順の入れ替えを行うことができなかった。また、語順の数の平均は 1.54 であった。

この結果から、日本語の語順は比較的自由だとはいえ、わかりにくい語順を除こうとすると、7 割もの文で原文以外の語順を生成できないことがわかった。

しかし、一部の文は、多数の語順の入れ替えが可能である。参照文 1 つの RIBES による評価では、これらの文が低い評価を受けている可能性が考えられる。

NTCIR 特許翻訳では、参照訳と参加者の翻訳結果だけでなく、それらを人手評価した結果も公開されている。この参照訳から係り受け木を作り、POST-ORDER で並べ替えた文を生成して、前述の制約を満たす文だけを出力することで参照訳を増やせば、複数参照訳で自動評価ができる。

まず、NTCIR-7 で人手評価の対象となった 100 文を CaboCha で解析し、調べたところ 100 文中 40 文に問題があり、修正を行った。同様に NTCIR-9, 10 のデータでも、参照訳を係り受け解析した結果を修正し、各翻訳自動評価手法の人手評価との相関がどう変わるか実験する予定である。

4 まとめ

本稿では、日本語の語順が比較的自由であるにもかかわらず、たった一つの参照訳の RIBES のスコアが、人手評価と高い相関を取れる理由について考察した。その結果、Head Final な日本語の語順を係り受け木から生成しようとすると、POST-ORDER による出力を行えばよいことがわかった。

しかし、その中にはわかりにくい文も含まれているので、同じ用言を修飾する、連続した格助詞句の語順の入れ替えだけに限定し、さらに、先行する他の用言の前には格助詞句を出さない、という制約を儲けたところ、7 割の文で原文以外の語順を生成できないことが判明した。

これは、「日本語の語順は比較的自由であるにもかかわらず、語順の近さを測定する RIBES が、参照訳 1 つで人手評価と高い相関を持つ不思議」に対する理由づけとなっているであろう。

本稿で提案した制約によれば、長い修飾句が前に、短い修飾句が後ろになる傾向があるが、この傾向は経験的に知られている [9]。

本稿では「は」など、他の助詞について述べなかった。別の機会に発表したい。

また、NTCIR の人手評価の対照となっている文の参照文の係り受け木のデータを作成中であり、このデータを用いて参照文を自動生成で増やし、翻訳自動評価手法の評価（メタ評価）を行う予定である。

なお、本稿で用いた、CaboCha の XML 出力から係り受け木を自動描画する LaTeX のマクロは、
`http://softcream.oka-pu.ac.jp/tex-macros-jp/`
で公開している。また。RIBES のコードは
`http://www.kecl.ntt.co.jp/icl/lirg/ribes/` にある。

謝辞

本研究は、翻訳自動評価の改良にかかわる岡山県立大学と NTT との共同研究によって行われた。本共同研究を支援してくださった NTT コミュニケーション科学基礎研究所の関係者の方々に感謝する。

参考文献

[1] Atsushi Fujii, Masao Uchimura, Mikio Yamamoto, and Takehito Usturo. Overview of the patent machine translation task at the NTCIR-7 workshop. In *Working Notes of the NTCIR Workshop Meeting (NTCIR)*, 2008.

[2] Isao Goto, Ka Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K. Tsou. Overview of the patent machine translation task at the NTCIR-10 workshop. In *Working Notes of the NTCIR Workshop Meeting (NTCIR)*, 2013.

[3] Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K. Tsou. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Working Notes of the NTCIR Workshop Meeting (NTCIR)*, 2011.

[4] 池原悟, 宮崎正弘, 白井諭, 横尾昭男, 中岩浩巳, 大山芳史, 林良彦. 日本語語彙大系. 岩波書店, 1997. http://www.kecl.ntt.co.jp/mtg/resources/GoiTaikei/.

[5] Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, and Masaaki Nagata. Automatic evaluation of translation quality for distant language pairs. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 944–952, 2010.

[6] Philipp Koehn. *Statistical Machine Translation.* Cambridge University Press, 2009.

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pp. 311–318, 2002.

[8] 平尾努, 磯崎秀樹, Kevin Duh, 須藤克仁, 塚田元, 永田昌明. RIBES: 順位相関に基づく翻訳の自動評価法. In *Proc. of the Annual Meeting of the Association for Natural Language Processing*, pp. 1115–1118, 2011.

[9] 山下裕子, 近藤公久. 言語的制約と long-before-short 語順傾向. 電子情報通信学会信学技報 TL2011-19, 2011.