対訳辞書を使わない対訳コーパスからの機械翻訳知識の自動獲得

大倉 清司

東京大学大学院総合文化研究科言語情報科学専攻

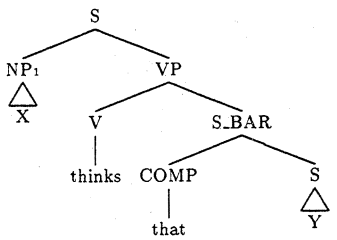
1 はじめに

本研究では、対訳辞書を使わずに、文対応のついた対訳コーパスに構造的なタグをつけたものから機械翻訳のための翻訳知識を自動獲得する方法を提案する。

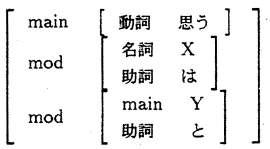
対訳辞書を使わない、ということは重要である。一般的に任意の言語間の翻訳知識を獲得しようとしたとき、その2言語の対訳辞書があるのは稀である。そのとき、対訳辞書を作成してから翻訳知識を獲得するよりも、対訳コーパスだけから対訳辞書と文法を同時に獲得できればよい。

翻訳知識とは、2言語間における単語レベル、句レベル、節レベルの構造的な対応のことである。構造的な対応とは、ある言語のある構造は他の言語ではどういう構造に対応するか、という知識である。例えば、次のような知識を獲得できればよい。

(1) a.



b.



この論文で提案する方法の特徴は次の通りである。

- 対訳辞書を使わない。よって、対訳辞書が存在しない任意の言語間の機械翻訳知識が獲得できる。
- 構造的な知識を獲得することによって、これまでの研究で難しいとされてきた複文を翻訳するための知識も獲得できる。

この研究では、英語・日本語の翻訳規則の抽出を行った。実際にコーパスを使って実験してみたところ、提案する方法で単語レベル、句レベルの構造的な翻訳知識が獲得できることがわかった。

次節では、本研究で提案する方法を説明する。3節では、実際にその方法をシステムとして実現し実際のコーパスで実験した結果について述べる。4節では、考察と将来への展望を述べる。

2 機械翻訳知識獲得の方法

前節で述べた構造的な翻訳知識の獲得は、

1. 部分構造対応付け
2. 変数置き換え

の順に行なう。

2.1 部分構造対応付け

(1)のように、英語の構造を木構造で、日本語の構造を素性構造であらわすとすると、対訳があるときに、木構造の部分と素性構造の部分が対応している可能性が高い。部分構造対応づけは、部分木と部分素性構造を対応づける方法である。例えば、

(2) a.
```
S(NP(PRP(I)),VP(VBD(went),
  PP(TO(to),NP(NNPX(NNP(Los),
  NNP(Angeles)))),
  NP(DT(the),JJ(other),NN(day))))
```

b.
```
[[main,[[いく | 動:五ワ用b,いっ],
  [た | タ=終,た]]],
  [mod,[[名詞,私],[ハ=係助,は]],
  [mod,[[時副詞名詞,先日]]],
```

[mod,[[名詞, ロサンジェルス],
[へ = 格助, へ]]]]

という対訳文から、

(3) a. PP(TO(to),NP(NNPX(NNP(Los),
NNP(Angeles))))<--->
[[名詞, ロサンジェルス],[へ = 格助, へ]]

b. NP(NNPX(NNP(Los),NNP(Angeles)))
<---> [[名詞, ロサンジェルス]]

などの知識が獲得できる。特に、次のような仮定ができる。

(4) 仮定　2つ以上の (英語の文の構造を表した) 木構造が同じ部分木を持つ時、その英語の文に対応する日本語の文の構造の中にも同じ構造が存在する可能性がある。

部分木の中で、特に、その葉が全て単語である (つまり、部分木の終端記号が全て単語で、カテゴリーを含まない) 部分木を、葉が単語である部分木と呼ぶことにする。形式的には、次のように定義する。

(5) 葉が単語である部分木 木 T の次のような1つながりのサブグラフ U である。
(1) U の全てのノードについて、対応する T のノードの全ての子を持ち、
(2) U は2つ以上のノードからなる。

部分木に対応しうる日本語の構造は、もとの文の対訳の日本語の構造の部分であるといえるので、その日本語の構造の部分構造をすべて列挙し、その部分構造がその部分木に対して統計的に高い割合で現れていれば、それを英語の部分構造と日本語の部分構造の対応とすることにする。

まとめると、次のようになる。

1. 対訳文、つまり英語の文<---> 日本語の文があるとき、英語の文構造の葉が単語である部分木それぞれについて、日本語の文構造の部分素性構造を全て列挙し、それを仮の対応とする。コーパスの全ての対訳文に対してこれを行い、同じ対応の数を数える。
2. 同じ部分木に対するそれぞれの部分素性構造の出現数の割合を計算する。
   (a) 部分木に対応すると仮定された部分素性構造の出現数を、同じ部分木に対応付けられた全ての部分素性構造の出現数の総和で割る。
3. それに対し、部分木に関係なく、それぞれの部分素性構造がどれだけの割合で出現しているかを計算する。
   (a) ある部分素性構造の全体の出現数は、部分木に関係なく、同じ部分素性構造の数を総和することによって求まる (a)。
   (b) 全ての部分素性構造の全体の出現数は、その総和である (b)。
   (c) (a) の (b) に対する割合を求める。
4. もし、仮の対応が正しければ、ある部分木についてのその部分素性構造の出現割合は、全体的なその部分素性構造の出現割合よりも大きくなるはずである。逆に、もし仮の対応が間違っていれば、それは小さくなるはずである。前者の場合を対応とする。

この方法では、英語の前置詞のように訳語がたくさんあるものは対応がとりにくい。反対に訳語が決まっている専門用語などは容易に獲得できる。コンピュータは言語に関する知識は全くもたないのだが、結果として、これはそれぞれ内容語 (content words)、機能語 (function words) の区別がついていることになる。文脈に応じて訳語が変わるものについては、変数置き換えによって抽出できる。

2.2 変数置き換え

ある構造の一部が他の構造に相当している時、その構造は他の構造を含む、という。(3a) は、英語の構造においても、日本語の構造においても、(3b) を含んでいる。このように、一方の対応の英語の構造、日本語の構造がそれぞれ、他方の構造のそれぞれの構造を含む時、その対応は他方の対応を含む、と呼ぶ。この場合、(3a) の対応の中で (3b) に相当する所を次の方法で変数に置き換えることができる。なお、変数といっても数ではなく構造に置き換えられるものだが、ここでは変数と呼び、X で表す。

a. 木構造における変数置き換え　木構造 $T_A = A(\gamma)$ と $T_B = B(\alpha T_A \beta)$ がある時 (α、β は0個以上の、γ は1個以上の木構造を指す)、T_B の T_A による変数置き換えの結果を、

$$B(\alpha A(X)\beta)$$

とする。$T_A = T_B = A(\gamma)$ の時は、

$$A(X)$$

とする。

b. 素性構造における変数置き換え　素性構造 $FS_A = [[Attribute_i, Value_i]]$ と $FS_B = [\alpha FS_A \beta]$ がある時 (α、β は0個以上の素性構造を指す)、FS_B の FS_A による変数置き換えによる結果を、素性構造 FS_B の中にある属性 - 値のペアの値を変数に置き換えた構造、つまり、

$$[\alpha[Attribute_i, X_i]\beta]$$

とする。$FS_A = FS_B = [[Attribute, Value]]$ の時は、

$$[[Attribute, X]] \text{ とする。}$$

c. 対応の変数置き換え　対応 $C_A = T_A$ <---> FS_A, $C_B = T_B$ <---> FS_B があり、C_B が C_A を含む時、C_B の C_A による変数置き換えの結果は、T_B を T_A で木構造における変数置き換えした構造と FS_B を FS_A で素性構造による変数置き換えした構造を対応づけたものである。

(3a) の (3b) による変数置き換えの結果は次の通りである。

(6) PP(TO(to),NP(X))
<---> [[名詞,X],[へ = 格助, へ]]

これらは、そのまわりの文脈があってはじめてその位置にあることができる、という制約を含んだものである。

一般に、2つの対応関係（英語 <---> 日本語）

C_1: E_1 <---> J_1
C_2: E_2 <---> J_2

があるとき、英語の部分木に関して

(7) a. E_1 が E_2 を完全に含んでいる (これをこれから $\supset_{TREE}$ で表す) 場合
b. E_1 が E_2 に完全に含まれている (これをこれから $\subset_{TREE}$ で表す) 場合
≡ E_1 と E_1 が同じ (これをこれから $\equiv_{TREE}$ で表す) 場合

の3通り、日本語の素性構造に関しても同様に3通りあるので、9通りの可能性がある。それぞれの可能性に対する処理は、表1のように分けることができる (1, 2, 3)。

最初に挙げた例は、1の場合であり、2つの対応関係の片方が他方に含まれる、という場合である。

表 1: 2つの対応関係の関係の可能性

	$\supset_{TREE}$	$\subset_{TREE}$	$\equiv_{TREE}$
$\supset_{FS}$	1	———	2
$\subset_{FS}$	———	1	2
$\equiv_{FS}$	2	2	3

(8) a. NP(NNPX(NNP(Los),NNP(Angeles)))
<---> [[名詞, ロサンジェルス]]
b. PP(TO(to),NP(X))
<---> [[名詞,X],[へ = 格助, へ]]

を翻訳規則とする。

表1のときの処理をまとめると、次のようになる。

1 $\supset_{TREE}$ かつ $\supset_{FS}$ の場合、
E_1 を E_2 によって変数置き換えをした結果と、J_1 を J_2 によって変数置き換えをした結果を対応づけ、
その結果と C_2 を翻訳規則とする。
$\subset_{TREE}$かつ $\subset_{FS}$ の場合も同様。

2 $\supset_{TREE}$ かつ $\equiv_{FS}$ の場合、E_1 を E_2 によって変数置き換えをした結果と、J_1(または J_2) を対応づけ、この結果と C_2 を翻訳規則とする。
$\subset_{TREE}$ かつ $\equiv_{FS}$ の場合も同様。
$\equiv_{FS}$ かつ $\supset_{FS}$ の場合、E_1(または E_2) と、J_2 を J_1 によって変数置き換えをした結果を対応づけ、この結果と C_1 を翻訳規則とする。
$\equiv_{FS}$ かつ $\subset_{FS}$ の場合、$\subset_{FS}$ かつ $\equiv_{FS}$ の場合も同様。

3 $\equiv_{TREE}$ かつ $\equiv_{FS}$ の場合、その対応を翻訳規則とする。

他の場合は無視する。

3 実験

前節で説明した方法をシステムとして実現し、実際に実験を行なった。実験に使ったコーパスは、講談社和英辞書例文 [8] の一部 10,000 文である（短いものから選択）。

3.1 タグ付け（構文解析）

今回の実験では、英語の文の解析には Apple Pie Parser [4] を、日本語の文の解析には QJP[6] を用いた。

3.2 結果

3.2.1 英単語の訳語獲得の結果 (一部)

3 @ DT(Every) @ [[名詞, だれ],[デ = 格助, で],[モ = 係助, も]]
0 @ JJ(bandylegged) @ [[名詞, 蟹股],[デ = 格助, で]]
0 @ JJ(beautiful) @ [[美しい — 形: ク終, 美しい]]
2 @ JJ(beautiful) @ [[いい — 形: ク体, いい]]
0 @ JJ(beautiful) @ [[名詞, お美しい]]

3.2.2 句レベルの対応関係獲得結果 (一部)

0 @ PP(IN(In),NP(DT(a),NN(word)))
@ [[数名詞, 一],[名詞, 言],[デ = 格助, で]]
0 @ PP(IN(In),NP(NN(fact))) @ [[副詞名詞, 事実]]
0 @ PP(IN(by),NP(RB(all),NN(means)))
@ [[副詞, どうして],[モ = 係助, も]]
0 @ PP(IN(by),NP(RB(all),NN(means)))
@ [[副詞名詞, 絶対]]
0 @ PP(IN(for),NP(NN(murder)))
@ [[名詞, 殺人罪],[デ = 格助, で]]

3.2.3 変数による置き換え

今回の実験では部分対応付けの精度がよくなく、変数による置き換えはいい結果が出なかった。

4 まとめ

部分構造対応づけの実装が不十分なため、単語レベル、句レベルの知識しか獲得できなかった。

この知識を使って実際に機械翻訳する方法も考える必要があるが、基本的に、例による翻訳 [3] によって実現できるだろう。

提案した方法によれば、単語レベル、句レベル、節レベルの言語全般的な対応関係が獲得できるため、今まで難しいといわれていた複文に関する翻訳知識の獲得も可能になると思われる。これは、変数置き換えによって、

```
VP(V(thinks),S_BAR(COMP(that),S(X)))<--->
 [[main,[[動詞, 思う]]],
  [mod,[[main,X],[助詞, と]]]]
```

という対応知識が獲得でき、しかも、

```
  S(NP(X),VP(V(likes),NP(Y)))
<---> [[main,[[動詞, 好きだ]]],
       [mod,[[名詞,X],[助詞, は]]],
       [mod,[[名詞,Y],[助詞, が]]]]
```

のような対応関係も獲得できるので、これを組み合わせて、

```
VP(V(thinks),S_BAR(COMP(that),
                   S(NP(X),VP(V(likes),NP(Y)))))
 [[main,[[動詞, 思う]]],
  [mod,[[main,[[動詞, 好きだ]]],
        [mod,[[名詞,X],[助詞, は]]],
        [mod,[[名詞,Y],[助詞, が]]]
  [助詞, と]]]]
```

という対応も導き出せる。

理論的には、部分構造の対応付けの精度がよければ、提案する変数置き換えの方法によって熟語やコロケーションなど獲得しにくい翻訳規則まで獲得できると思われる。

参考文献

[1] Bod, R.: *Enriching Linguistics with Statistics: Performance Models of Natural Language*, ILLC dissertation series 1995-14, 1995.

[2] Cicekli, I. and H. Altay Güvenir: Learning Translation Rules From A Bilingual Corpus, *Proc. of NEMLAP-2* (1996).

[3] Sato, S.: MBT2: A Method for Combining Fragments of Examples in Example-based Translation, *Artifitial Intelligence*, Vol.75, pp.31-49 (1995).

[4] Sekine, S.: Manual of Apple Pie Parser, version 5.7 (1996) terminals, *Fourth International Workshop on Parsing Technology*, pp.216-223 (1995).

[5] 大倉清司：タグ付きコーパスからの統語・意味的知識の自動獲得, 信学技報 NLC95-75 (1996).

[6] 亀田雅之：軽量・高速な日本語解析ツール『簡易日本語解析系Q__ＪＰ』, 言語処理学会 第1回年次大会発表論文集, pp.349-352 (1995).

[7] 北村美穂子, 松本裕治：対訳コーパスを利用した翻訳規則の自動獲得, 情報処理学会論文誌, Vol37, No.6, pp.1030-1040 (1996).

[8] 清水 護, 成田成寿：和英辞典, 講談社学術文庫 (1979)