用例ベース処理を用いたパターンベース翻訳システム

渡辺日出雄 武田浩一
日本IBM東京基礎研究所
{watanabe,takeda}@trl.ibm.co.jp

1 はじめに

WWWの普及により外国語で書かれたテキストに出会う機会が非常に増えてきた。このような状況のため、自動翻訳ソフトがかなり使われるようになってきている。Web 年で物事が動くと言われるこの世界では、如何に迅速に未知語や新たな言い回しに対応できるかが重要である。このような考えに基づき、我々はシステムの向上が容易な**パターンベース翻訳** [5, 6] という手法を考案し、それに基づく PalmTree と呼ばれる翻訳システムを実現した。[1]

パターンベース翻訳の基本的なアイディアは、ソース側 CFG 規則とその対応するターゲット側の CFG 規則のペアである翻訳パターンを大量に用意し、翻訳パターンのソース側部分を用いて構文解析をし、構文解析が終わると同時に同期導出によりターゲット構造を生成するというものである。

このパターンベース翻訳では、システムの訳質向上のための処理のかなりの部分を新たなパターンの追加により行うことが出来るため、カスタマイゼーションが非常に容易であるという利点がある。しかし、幾つかの問題点があるのも事実である。一つには、大量の翻訳パターンを使うことから、構文解析で使う文法ルールの数が膨大となり処理速度の点で不利であるということである。我々は、この問題に対処するため幾つかの枝刈り手法を考案し実装した。[8] もう一つの問題は、完全マッチを用いているため、一つの翻訳パターンのカバーする範囲が狭く、翻訳パターンの量が急速に増えてしまう。この増加ペースを押さえるため、一つの翻訳パターンが様々なバリエーションで使われるようにする必要がある。本論文では、この問題点に対処するため用例ベース手法 [1, 3, 7] でパターンベース手法を拡張した**用例パターンベース翻訳手法**について述べる。

[1] このシステムは弊社製品「インターネット翻訳の王様」の翻訳エンジンとして使用されている。

2 パターンベース翻訳

まず、パターンベース翻訳について簡単に説明しておくことにする。パターンベース翻訳手法とは、ソース側の CFG 規則とそれと対応する CFG 規則のペアを翻訳パターンとし、このソース側の CFG 規則を用いて構文解析を行い、解析結果を構成するソース側 CFG 規則と翻訳パターン中で対を成すターゲット側 CFG 規則から同期導出によりターゲット側構造を構築するというものである。

翻訳パターンは一般に以下のような形式をしている。

$$SR_1 \ldots SR_n \Leftarrow SL\ TL \Rightarrow TR_1 \ldots TR_m$$

ここで、SR_i はソース側右辺項、SL はソース側左辺項、TR_j はターゲット側右辺項、TL はターゲット側左辺項を表わす。それぞれの項は、

単語:品詞:IDX:属性

からなる。ここで、単語 (終端記号) と品詞 (非終端記号) はどちらか一つがある場合と両方ある場合が許される。IDX は右辺と左辺及びソース側とターゲット側の対応関係を表わす。属性としては、マッチング条件などを指定することが出来る。以下は、翻訳パターンの例である。

(p1) take:VERB:1 a look at NP:2 ⇒ VP:1
VP:1 ⇐ NP:2 wo(dobj) miru(see):VERB:1
(p2) NP:1 VP:2 ⇒ S:2 S:2 ⇐ NP:1 ha VP:2
(p3) PRON:1 ⇒ NP:1 NP:1 ⇐ PRON:1

(p1) は英語の慣用表現 "take a look at," の翻訳パターンであり、(p2) と (p3) はそれぞれ一般の文法的な翻訳パターンをあらわしている。これらの翻訳パターンがあるときに、入力文"She takes a look at him."の翻訳を示したのが図 1 である。

構文解析は通常の CFG パージングの手法を用いる。例えば、チャートパージングでは、

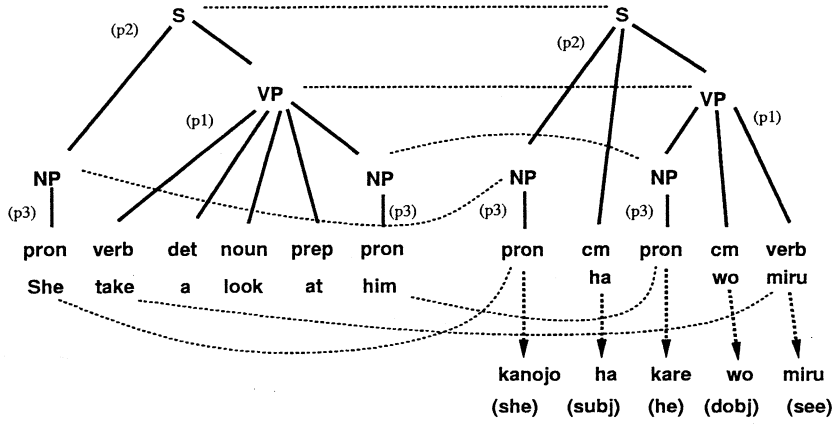

図 1: Translation Example by Pattern-based MT

(1) ある不活性アークの規則の左辺項 T_A とマッチする項を最左右辺項 T_B として持つ規則から新たなアークを作成する。

(2) ある不活性アークAの始点を終点とする活性アークBがあるとき、Aの規則の左辺項 T_A とBの規則の最左右辺活性項 T_B がマッチすれば、AとBから新たなアークを作成する。

というのが、基本動作である。マッチするかどうかは、通常 T_B の品詞・単語・マッチング属性等が T_A のものとマッチするかどうかで決定される。通常のCFG規則を構成する項は単語（終端記号）か品詞（非終端記号）のどちらかであるが、パターンベース翻訳では両者があっても良い。そこで、両者がある場合は、AND条件でマッチングをチェックするということにしている。

3 用例パターンベース翻訳

パターンベース翻訳手法 [5, 6] では、完全マッチを採用しているため、それぞれの翻訳パターンが使われる表現に限りが有り、使用効率の点で問題がある。そこで、用例ベース処理で一般に用いられているファジーマッチを導入することにより、翻訳パターンの使用効率を高めることにした。ここでは、用例ベース処理によるパターンベース翻訳の拡張の仕方について述べる。

3.1 用例ベース構文解析

まず、基本となる構文解析アルゴリズムを用例を用いて処理するように拡張する。基本的には、語彙化文法[2] による構文解析でのマッチングで完全マッチではなくファジーマッチを使うことになる。これを**用例ベース構文解析**と呼ぶことにする。

用例ベース構文解析の翻訳パターン及びそれを構成するソース側CFG規則は前述のものと同じであるが、以下のような拡張を施してある。翻訳パターンのソース言語側規則の右辺項が語彙化項[3] であり、ターゲット側規則の右辺に対応項がある場合、その項をファジーマッチ項と呼び、そうでない項を完全マッチ項と呼ぶことにする。また、単語をシングルクオートあるいはダブルクオートで囲むことにより、明示的にそれぞれ完全マッチ項あるいはファジーマッチ項であると指定することも出来る。

例えば、以下の翻訳パターンの例では、*make* はターゲット側に対応項を持つので通常はファジーマッチ項であるが、シングルクオートで囲まれているためこの場合は完全マッチ項である。*a* と *decision* はターゲット側に対応項を持たないので完全マッチ項である。

'make':VERB:1 a decision ⇒ VP:1
VP:1 ⇐ ketsudan-suru:1

[2] ただし、ここでは全てが語彙化された規則を用いるという意味ではない。
[3] 単語を持つ項

用例ベース構文解析アルゴリズムでは前述のCFG構文解析の動作を以下のように拡張する。ここで、T_Aの品詞をPOS_A、単語をLEX_A[4]とする。更に、右辺項T_Bの品詞をPOS_B、単語をLEX_B[5]とする。この時上記(1)、(2)の処理で、T_AとT_Bが以下の条件のどれかを満たしたときにマッチすることにする。

(1) T_BにPOS_BとLEX_Bが両方存在する場合、

(1-1) LEX_BがLEX_Aと同じであり、かつ、POS_BもPOS_Aと同じである。

(1-2) T_Bがファジーマッチ項であり、LEX_BとLEX_Aの意味的距離がある閾値内であり、かつ、POS_BとPOS_Aが同じである。

(2) T_BにLEX_Bだけが存在する(POS_Bが指定されていない)場合、

(2-1) LEX_BとLEX_Aが同じである。

(2-2) T_Bがファジーマッチ項であり、LEX_BとLEX_Aの意味的距離がある閾値内である。

(3) T_BにPOS_Bだけが存在する(LEX_Bが指定されていない)場合、POS_BとPOS_Aが同じである。

3.2 ルールの優先度

パージングでは多くの競合する解が残るが、それらの解の優先順位は通常解を構成する規則のコストの和により決められる。用例ベース処理では、このコストが類似度を反映したものになる。よって、マッチングコストとして元々規則に付けられているコストに加えて、ファジーマッチによる類似度を反映した意味距離コスト(類似度が高いほどコストは低くなる)を付加する。意味距離コストは、以下の不等式を満足するように設定する。

$$(1\text{-}1) < (1\text{-}2),(2\text{-}1) < (2\text{-}2) < (3)$$

(1-2)と(2-1)に与えるコストはファジーマッチの際の意味的距離の閾値に応じて決定される必要がある。

前述の語彙化規則優先原理に加えてこの意味距離コストを導入することにより、よりきめ細かく規則の優先度を決定することが可能である。

[4] 全てのマッチした項は、単語を継承しているとする。
[5] この場合は、単語がある場合とない場合がある。

3.3 ターゲット側規則の修正処理

用例ベース翻訳手法を用いているので、翻訳パターンのターゲット側をそのまま使ってターゲット構造を作ることは出来ない。[6] よって、ターゲット側規則を入力に合わせる以下のような修正処理が必要となる。

翻訳パターンのターゲット側の右辺項t_tが単語w_tを持ち、t_tがソース側に対応する項t_sを持つ場合、t_sが入力単語w_iとマッチし、かつ、w_tがw_iの訳語でない場合、w_tをw_iの訳語で置き換える。

4 翻訳例

図2に用例ベース処理を導入したパターンベース翻訳での翻訳例を示す。この例では、以下の翻訳パターンが使われている。

(p2) NP:1 VP:2 ⇒ S:2 S:2 ⇐ NP:1 ha VP:2
(p3) PRON:1 ⇒ NP:1 NP:1 ⇐ PRON:1
(p4) take:VERB:1 a bus:2 ⇒ VP:1
VP:1 ⇐ basu:2 ni noru:VERB:1

翻訳パターン(p4)は“taxi”と“bus”が意味的に近いので、“take a taxi,”の部分とマッチする。これらの翻訳パターンのターゲット部分を結合することにより、“PRONはバスに乗る”という翻訳が生成される。この翻訳で“バス”は対応する入力単語“taxi”の正しい訳語でないので、正しい訳語“タクシー”に変更される。更に、PRONは“I”の訳語“私”に置き換えられる。こうして、最終的に“私はタクシーに乗る。”という翻訳が得られることになる。

5 議論

この用例パターンベース翻訳手法は、同期文法[2, 4]と用例ベース翻訳[3, 7]を組み合わせた方式と言うことが出来る。同期文法の考え方を用いることにより、従来用例ベース手法はトランスファー処理(やその他の翻訳の一部の処理)に適用範囲が限定されてきたが、解析から生成までを一貫して簡潔なモデルで処理出来るようになった。

パターンベース翻訳の利点は、一つの翻訳パターンの追加により、従来型システムの解析・トランスファー・生成のプロセスに同時に影響を与えることが出来る点にある。これにより、訳質の向上が非常に楽に行える

[6] なぜなら、ファジーマッチを用いているため、ターゲット側規則で使われている単語の対応するソース側単語がマッチした入力単語と同じとは限らないからである。このような場合、ターゲット側単語を対応する入力単語の訳語に変更する必要がある。

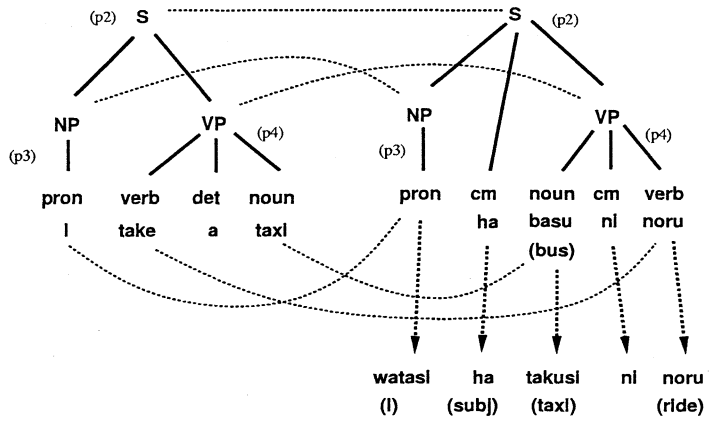

図 2: 用例ベース処理による翻訳例

ことになった。ただ、かなり容易になったとは言え、翻訳パターンには細かなマッチング属性[7] 等を記述する必要があるため、一般の人が記述できるほどに簡単ではない。我々は、高校生程度の英語と日本語の文法的知識があれば翻訳パターンを記述できることを目指して、簡易レベルの翻訳パターンの記述形式を導入した。基本的には、記述するパターンの句の種類 (名詞句、動詞句、等) とソース側単語列[8] とターゲット側単語列を記述すれば良い。そして、この簡易レベル翻訳パターンを内部記述レベルに変換するパターンコンパイラを開発した。例えば、以下の (q1) という簡易レベル翻訳パターンは、(q2) の内部レベル翻訳パターンに変換される。

(q1) [VP] make a big shot =
素晴らしい ショットを 打つ
(q2) make:Verb:1 a big shot ⇒ VP:1 VP:1
⇐ 素晴らしい ショットを 打つ:Verb:1

このパターンコンパイラと用例パターンベース翻訳手法により、用例ベース翻訳の究極の目標である翻訳例を入力するだけで訳質が向上していくというものにかなり近づくことが出来た。

[7] 本論文で紹介した翻訳パターンでは紙面の都合で細かなマッチング属性等を省略している

[8] 名詞句とマッチする変数 (#と記述する) を書けるようにもなっている

6 おわりに

本論文では、翻訳パターンの使用効率改善により翻訳パターンベースの量的爆発を押さえるため、用例ベース処理を導入した用例パターンベース翻訳手法について述べた。

CFG パージング+同期導出アルゴリズム、パターンコンパイラ、及び、用例ベース手法を組み合わせた用例パターンベース翻訳システムにより人間の翻訳処理に近いシステムの構築が可能であることを示した。

参考文献

[1] Nagao, M., "A Framework of a Mechanical Translation between Japanese and English by Analogy Principle," Elithorn, A. and Banerji, R. (eds.) : *Artificial and Human Intelligence,* NATO 1984.

[2] Rambow, O., and Satta, S., "Synchronous Models of Language," Proc. of the 34th of ACL, pp. 116–123, June 1996.

[3] Sato, S., and Nagao, M. "Toward Memory-based Translation," Proc. of 13th COLING, August 1990.

[4] Shieber, S. M., and Schabes Y., "Synchronous Tree-Adjoining Grammars," Proc. of the 13th COLING, pp. 253–258, August 1990.

[5] Takeda, K., "Pattern-Based Context-Free Grammars for Machine Translation," Proc. of 34th ACL, pp. 144–151, June 1996.

[6] Takeda, K., "Pattern-Based Machine Translation," Proc. of 16th COLING, Vol. 2, pp. 1155–1158, August 1996.

[7] Watanabe, H. "A Similarity-Driven Transfer System," Proc. of the 14th COLING, Vol. 2, pp. 770-776, 1992.

[8] 渡辺、武田、「パターンベース翻訳システム : PalmTree」第５５回情報処理学会全国大会講演論文集（２）, pp. 80-81, 1997