日本語表現の柔軟な照合

黒橋 禎夫 *1*2　酒井 康行 *1
*1 京都大学大学院 情報学研究科　*2 科技団 さきがけ研究21
{kuro,sakai}@pine.kuee.kyoto-u.ac.jp

1　はじめに

計算機で自然言語を扱う上での最大の問題は，自然言語の自由度，冗長性をいかにして吸収するかという問題である．本論文では，そのための第一歩として，言語表現間の意味的な照合，すなわち，表層的に異なる二つの表現の意味的な近さを判定する新しい方法を提案する．ここでは，意味的に近い表現を類義表現とよぶことにする．類義表現には，語，句，文などの様々なレベルにおいて，辞書的に定義されるものと，変換規則により結び付けられるものがある．類義表現を結び付けるルール群を整備し，それらを共有メモリを通してボトムアップ的に適用することにより，柔軟で強力な類義性判定システムを作成した．

2　類義表現の基本要素

日本語における類義表現の基本要素を以下のように分類・整理した．

語のレベルの類義

拳銃 ↔ けん銃 ↔ けんじゅう (かなと漢字)
ユーザー ↔ ユーザ (表記のゆれ)
登山 ↔ 山登り (同義語)
学生 ↔ 生徒 (類義語)

語と句/文

登山 ↔ 山に登る

機能的表現の有無・交代

明けかかる ↔ 明けはじめる (アスペクト)
飲んだはずだ ↔ 飲んだに違いない (モダリティ)
読める ↔ 読むことができる (可能)
赤 ↔ まっ赤 (強調)
金 ↔ お金 (丁寧)

体言と用言

帰り ↔ 帰る
帰宅 ↔ 帰宅する

助詞の有無・交代

日本車 ↔ 日本の車
東京転勤 ↔ 東京への転勤
問題を検討 ↔ 問題について検討
酒に酔う ↔ 酒で酔う

反意語の否定

明るい ↔ 暗くない
不完全だ ↔ 完全でない

受身・使役

AがBを捕まえる ↔ AにBが捕まえられる
AがBを捕まえる ↔ AにBを捕まえさせる

対義

AがBにCを預ける ↔ AからBがCを預かる
AがBにCを売る ↔ AからBがCを買う

対称

AがBと会う ↔ BがAと会う ↔ AとBが会う

単文の構造

太郎が怒った ↔ 怒った太郎 ↔ 怒ったのは太郎だ

文の接続表現

転んだ<u>ので</u>泣いた ↔ 転んだ<u>ために</u>泣いた
↔ 転ん<u>で</u>泣いた
逃げる<u>なら</u>撃つ ↔ 逃げ<u>れば</u>撃つ

冗長な (無くても推測できる) 表現

雷<u>が鳴って</u>驚いた ↔ 雷で驚いた
学校<u>の中の</u>プール ↔ 学校のプール

その他

演説する ↔ 演説をする ↔ 演説を行う
ぴったり ↔ ぴったりと
～する<u>こと</u>～ ↔ ～する<u>の</u>～

3　共有メモリを用いたボトムアップ・マッチング

3.1　概要

日本語の類義表現のマッチング・システムを構築するためには，まず，前節であげた類義表現の基本要素それぞれを扱うマッチング・プログラムを作成しなけ

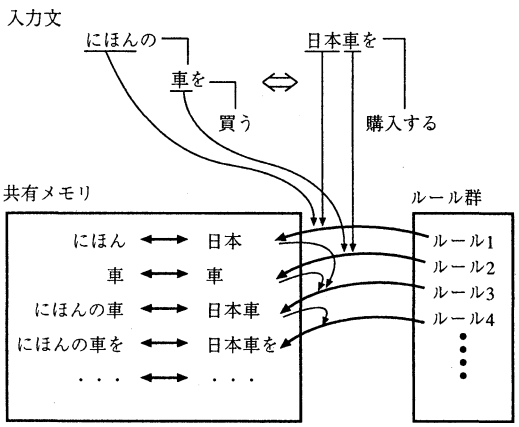

図 1: システムの概要

ればならない．そのためには，反意語や対義語についての辞書情報などが必要となるが，それらはきっちりと整備すればよいという問題で，それほど難しいことではない．

問題となるのは，実際の類義表現では，多くの場合，複数の基本要素が混在しており，それぞれに対するマッチング・プログラムをどのように統合して，全体が類義表現であることを判定すればよいかという問題である．たとえば,「にほんの車を買う」と「日本車を購入する」という類義表現では，かなと漢字，助詞の有無，類義語の 3 つの要素が混在している．

このような問題を扱うために，本論文では，共有メモリを用いる次のようなマッチングの枠組みを提案する．

1. 類義表現の基本要素のそれぞれを扱うルール (マッチング・プログラム) を作成する．

2. 入力表現間の部分的な対応 (類義表現) を保持する共有メモリを用いる．

3. ルール群が入力表現と共有メモリ中の対応から徐々に大きな対応を発見し，それを順次共有メモリに書き込む．すなわち，対応がボトムアップに発見されていく．

図 1 に，このシステムによって「にほんの車を買う」と「日本車を購入する」の照合を行う様子を示す．

このような枠組みであれば，ルールの適用順序を手続き的に制御する必要はなく，ルールの修正，追加も比較的容易となる．

3.2 ルール

ルールは類義表現の基本要素ごとに用意する．ルールが共通に行うことは，入力表現間の部分的な対応 (類義表現) を作り出して，共有メモリに書き込むことである．対応を作り出す際に，共有メモリ中の対応をもとにするかどうか (0 個，1 個，2 個) によってルールは 3 つに大別できる．以下では，それぞれについて具体的に説明する．

語間の対応を作り出すルール (共有メモリ中の対応を参照しない)

入力表現間のすべての語のペアに対して，それらが類義語，反意語，対義語であるかどうかを調べ，そうであればスコア，適当なフラグとともにその対応を共有メモリに書き込む．これは，マッチングの初めに行われるもので，共有メモリ中の対応を参照するものではない．

1. 同一の語であればスコア 1.0 の対応とする．

2. かな漢字の表記のバリエーションであれば，スコア 0.95 の対応とする．かな漢字の表記のバリエーションは JUMAN の辞書を用いる．

3. シソーラス中の 2 語の近さを最大値 0.95 のスコアに正規化し，スコア 0.5 以上であれば対応とする．シソーラスとしては，NTT「日本語語彙大系」を用いる [1].

4. 「帰る」と「帰り」のように，動詞と名詞 (=動詞連用形) の関係にあればスコア 1.0 の対応とする．

5. 「読む」と「読める」のように，動詞とその可能形の関係にあれば,〈可能〉というフラグをつけて，スコア 1.0 の対応とする．

6. 反意語，対義語であれば,〈反意〉,〈対義〉というフラグとともにスコア 1.0 の対応とする．これらのフラグは，この対応が拡張される際に他のルールで参照され，否定表現と打ち消されたり，対義の格助詞対応などの取り扱いで利用される．

共有メモリ中の対応を拡張するルール

共有メモリの対応に対して，入力表現中の前後の機能的表現を付与し，拡張した新たな対応を共有メモリに書き込むというルールがある．付与される機能的表

現には，助詞，接頭辞，接尾辞，助動詞などがあり，それぞれに個別のルールが用意されている．

たとえば，図1の例で，共有メモリ中の対応「にほんの車 ↔ 日本車」に対して，助詞を付与するルールが適用され，新たな対応「にほんの車を ↔ 日本車を」が作られる．

このようなルールでは，もとになる対応の長さを L_m，スコアを S_m，付与される表現の長さを L_f，スコアを S_f として，次式で計算されるスコアが新たな対応のスコアとなる．

$$\frac{L_m \cdot S_m + L_f \cdot S_f}{L_m + L_f}$$

ここで，対応の長さとは，自立語を1語，付属語を0.2語と数えたときの対応中の総語数であり，付与される表現の長さも同様に計算される．付与される表現のスコアはルールごとに定義されている．たとえば，助詞を付与するルールでは，同一の格助詞であれば1.0，提題・取り立て助詞同士も1.0，それ以外は0.0のように定義される．

上記の例の場合，「にほんの車 ↔ 日本車」の長さは4.2，スコアは0.98，付与される表現（ともに助詞「を」）の長さは0.4，スコア1.0であるから，新たな対応のスコアは $(4.2\times0.98+0.4\times1.0)/(4.2+0.4)=0.98$ となる．

共有メモリ中の2つの対応を併合するルール

最も基本的なルールは，共有メモリ中の連続する2つの対応を併合し，その結果を新たな対応とするものである（図2-a,b）．ここで，対応が連続であるとは，それぞれの入力表現において，対応に含まれる表現間が文節の係り受け関係にあるか，文節内部で隣接している場合をさす．

連続でない2つの対応を併合するルールとしては，能動文と受身文の用言と格要素をつなげるもの（図2-c），類義の接続表現で文の対応をつなげるもの（図2-d）などがある．

このような併合によって新たな対応が作られる場合にも，そのスコアは，長さに正規化されて次のように計算される．

$$\frac{L_{m1} \cdot S_{m1} + L_{m2} \cdot S_{m2} + L_f \cdot S_f}{L_{m1} + L_{m2} + L_f}$$

ここで，$L_{m1}, S_{m1}, L_{m2}, S_{m2}$ は併合される2つの対応の長さとスコアであり，L_f, S_f は挿入される接続表現の長さ，スコアである．挿入される接続表現のスコアはその類義の度合いに応じて各ルールにおいて定義されている．

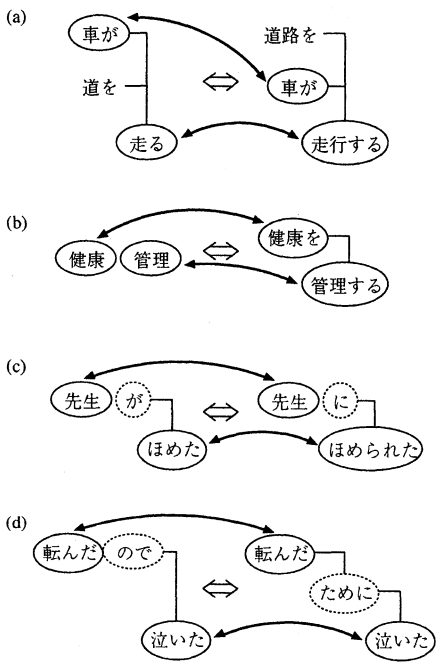

図 2: 対応の併合

3.3 国語辞典定義文を用いた再帰的マッチング

「登山が好きだ」と「山に登ることが好きだ」という類義表現の中には，「登山」と「山に登ること」という，語と文の対応が含まれている．このような類義表現を扱うために，本研究では国語辞典の定義文を用いる．国語辞典において「登山」の定義文が「山に登ること」と与えられていれば，「語間の対応を作り出すルール」の場合と同様に，入力表現間から直接この対応を見つけ出し，共有メモリに書き込めばよい．

しかし実際には，語の定義文が，入力表現の一部と完全一致するとは限らず（むしろまれであり），上記のような単純な国語辞典の利用ではあまりにも有効性が低い．たとえば，入力表現が「登山が好きだ」と「山にのぼるのが好きだ」であれば，単純な国語辞典の利用はたちまち失敗してしまう．

そこで，「登山」の定義文「山に登ること」と「山にのぼるのが好きだ」との間で，ここで述べているシステム自身を用いて柔軟な照合を行う必要があるということになる．そして，「山に登ること」が「山にのぼるの」の部分と類義表現である（スコア0.7以上である）ことがわかれば，その結果，すなわち「登山」と

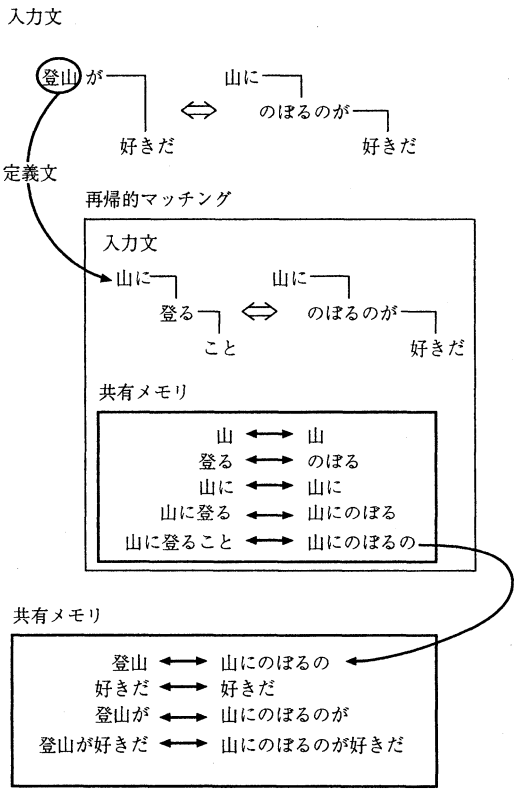

図 3: 国語辞典定義文による再帰的マッチング

「山にのぼるの」の対応をもとの共有メモリに書き込む (図 3).

このような定義文を介した類義の関係は入力表現中のすべての語について調べる必要がある．すなわち，入力表現中の各語について，その定義文ともう一方の入力表現との間で，本システムによる照合を行い，類義表現であればもとの共有メモリ書き込む．

ここで問題となるのは，このような定義文とのマッチングにおいてさらに定義文による置換を行うと，すなわち真に再帰的にシステムを適用すると，定義文による置換が爆発的に (何もしなければ無限に) 繰り返されるという問題である．この問題をふせぐために，現在のところ，国語辞典定義文による置換は 1 レベルのみ，すなわち，定義文とのマッチングでは定義文による置換は行わない，としている．

4 動作例

以下では，本システムの動作例を，共有メモリに書き込まれる対応とともに具体的に示す．

例 1: 「悪い」 ⇔ 「良くない」

1: 悪い ↔ 良い (1.0) ··· 〈反意〉
2: 悪い ↔ 良くない (1.0)
··· 1 を拡張し〈反意〉を解消

例 2: 「論文の執筆を行う」 ⇔ 「論文を執筆する」

1: 論文 ↔ 論文 (1.0)
2: 執筆 ↔ 執筆 (1.0)
3: 執筆を行う ↔ 執筆する (1.0)
··· 2 を拡張 (サ変関連表現)
4: 論文の執筆を行う ↔ 論文を執筆する (1.0)
··· 1 と 3 を併合

例 3: 「打鍵ミス」 ⇔ 「キーの打ち間違い」

再帰: 「鍵盤をたたく」 ⇔ 「キーの打ち間違い」
(「打鍵」の定義文)
R1: 鍵盤 ↔ キー (0.59) ··· シソーラス
R2: たたく ↔ 打ち (0.95) ··· シソーラス
R3: 鍵盤をたたく ↔ キーの打ち (0.80)
··· R1 と R2 を併合

1: 打鍵 ↔ キーの打ち (0.80) ··· R3 から
2: ミス ↔ 間違い (0.95) ··· シソーラス
3: 打鍵ミス ↔ キーの打ち間違い (0.86)
··· 1 と 2 を併合

5 おわりに

日本語表現における類義表現の基本要素を結び付けるルール群を整備し，それらを共有メモリを通してボトムアップ的に適用することにより日本語表現間の類義性を判定する枠組みを提案した．この枠組みの中で，国語辞典定義文による言い換えの関係を扱うことにより，従来のシソーラス等に限った類義表現の取り扱いに比べて，はるかに柔軟で強力な類義性判定が可能となった．

今後，反意語，対義語などの辞書情報の整備が必要である．また，本システムを情報検索のテストセットなどに適用し，実際上の有効性を検証する予定である．

参考文献

[1] 池原，宮崎，白井，横尾，中岩，小倉，大山，林 編: 日本語語彙大系, 岩波書店, 1997.