修 士 論 文

機械的に言い換えを実現するシステムの作成

指導教官　佐藤理史 助教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

佐藤 理

1999.2.15

目 次

第 1 章

序論

1.1 研究の背景と目的

近年インターネット環境が大幅に整備されたおかげで、WWW を中心として様々な文書が電子化され、ネットワーク上で容易に入手できるようになってきた。

しかし、それらの文書は必ずしも多くの人々にとってわかりやすい言葉で書かれているわけではない。たとえば、分野固有の表現（専門用語や特殊な言い回し）が使われている文書は、その分野の専門家以外には理解が困難であるし、また、政府や自治体の報告書のように、堅苦しく難解な言葉が好んで用いられている文書もある。

本研究では、このようなわかりにくい文書を、多くの人々にとってわかりやすい文書へと変換する (言い換える) 手法について研究し、それを機械的に行うシステムを作成する。

1.2 本論文の構成

本論文では、第 2 章で言い換えの具体例を検討し、それに基づき機械的な言い換えを実現する方法を検討する。第 3 章では作成した言い換えシステムについて述べる。第 4 章では作成した言い換えシステムを用いて、政府自治体文書を機械的に言い換える実験とその結果について述べる。第 5 章で結論を述べる。

第 2 章

言い換えの調査

本章ではある表現をよりやさしく言い換える具体例を検討し、それらの言い換えを機械的に実現する方法を検討する。

2.1 調査対象

どのような言い換えを行うかは、言い換えを行う対象となる文書の種類や分野に強く依存すると考えられる。ここでは、堅苦しく難解な表現が多いと考えられる政府、自治体、公的機関などの報告文書を対象として選び、これらを読みやすいようにするための言い換えを研究対象とする。

ここでは、まず、以下の 2 つの文書を選び、その文書にみられる難解な表現をやさしく言い換える作業を行った。

- 郵政省「インターネット上の情報流通ルールについて（報告書）」
  http://www.mpt.go.jp/pressrelease/japanese/new/980105j601.html
  (10764 バイト 5000 語強)
- 大蔵省「新しい金融行政のあり方について」[1]
  (7335 バイト 3600 語強)

この作業により 162 例の言い換え例が作成された。これらの例を表 2.1にように整理した。以下では、それぞれの言い換え例を示し、その機械的実現法について検討する。

[1]現在は、すでに WWW 上に存在しない。

言い換え種類		計
単語の削除		12
単語から単語		39
	活用変化なし 活用変化あり	(16) (23)
単語列から単語列		111
	名詞句から名詞句 動詞句から動詞句	(9) (102)
総計		162

表 2.1: 言い換え例

2.2 単語の削除

過剰な形容詞やまわりくどい言い回しをそのまま削除する言い換えが 12 例あった。以下に例を示す。

例 1 銀行行政は、おおむね制度の企画・立案、監督に整理される→
銀行行政は、制度の企画・立案、監督に整理される

このタイプの言い換えは、削除すべき単語リストを用意することで実現できる。

2.3 単語から別の単語への言い換え

単語を別の単語に言い換えたものが 39 例あった。

2.3.1 活用しない語の言い換え

言い換える単語が活用しない語の場合は、単に単語を置換すればよい。以下に例を示す。

例 2 利払費の増嵩に伴い → 利払い費の増加に伴い

この例では「増嵩」を「増加」に置き換えている。このタイプの言い換えは言い換え前の単語と言い換え後の単語の組 (単語言い換え辞書) を用意することによって実現できる。

2.3.2 活用する語の言い換え

動詞、形容詞などの活用する語を言い換える場合は、単なる単語の置換では不十分であり、活用形を調整する必要が生じる。

活用型が同じ場合

言い換え前の単語と言い換え後の単語が同一の活用型を取る場合は、活用形を変更する必要は生じない。以下に例を示す。

例 3 真摯な態度でのぞむ → まじめな態度でのぞむ

この場合、「真摯な」と「まじめな」の活用型はどちらもナ形容詞であるため、「まじめな」の活用形は「真摯な」の活用形を同じ活用形（この場合は、基本連体形）にすればよい。

このような言い換えは、2 つの単語の対応表をそれぞれの活用形に対して用意し、これらを全て言い換え辞書に登録しておくことによっても実現できる。しかし、この方法は辞書のサイズが大きくなるという欠点を持つ。

より巧みな実現法は、以下のように活用形を写像する方法である。

1. 言い換え辞書には、それぞれの標準形の組を登録する。

   例 真摯だ → まじめだ

2. 言い換えの対象となる文を形態素解析し、活用する語に対してはその活用型、活用形、標準形など情報を得る[2]。

   例 真摯な:真摯だ:形容詞::ナ形容詞:ダ列基本連体形

3. 標準形で言い換え辞書を引くことにより、言い換え先の語を得る。

   例 真摯だ → まじめだ

4. 言い換え先の活用形を元の語の活用形に合わせる。

   例 まじめだ →（ダ列基本連体形に変換）→ まじめな

[2] 本研究では、形態素解析システムとして、Juman[3] を用いる。形態素情報は、先頭から、表層形、標準形、品詞、品詞細分類、活用型、活用形を表す。

同じ活用形が存在する場合

言い換え前の単語と言い換え後の単語において、たとえ活用型が異なった場合でも、言い換え前の単語の活用形と同じ活用形が言い換え後の単語の活用型に存在する場合は、活用形を変更する必要は生じず、言い換え語の活用形をその活用形に合わせればよい。この場合は、上述の活用型が同じ場合と全く同じ方法で言い換えを実現できる。以下に例を示す。

例 4 子をもうける → 子を産む

この場合、「もうける」の活用型は母音動詞であり、「産む」は活用型が子音動詞マ行であり、活用型は異なる、しかし、「もうける」の活用形は基本形であり、子音動詞マ行の活用型にも基本形が存在するので、「産む」の活用形は基本形とすればよい。

活用形の調整が必要な場合

言い換え前の単語と言い換え後の単語の活用型が異なり、言い換え前の単語の活用形が言い換え後の単語の活用型に存在しない場合、言い換え後の単語の活用形を決定する必要が生じる。この決定のためには、表 2.2表 2.3表 2.4 に示すような異なる活用型間の活用形対応表を用意すればよい。以下に例を示す。

例 5 難解な問題を解く → 難しい問題を解く

この例の場合、「難解な」の活用型はナ形容詞、活用形はダ列基本連体形である。一方、「難しい」の活用型はイ形容詞イ段であり、この活用型にはダ列基本連体形はない。イ形容詞イ段型の活用型をとる形容詞は、名詞に接続する場合は基本形を取るので、この活用形は基本形とする必要がある。

2.3.3 名詞から形容詞への言い換え

ほとんどの単語の言い換えは、同じ品詞に属する単語間での言い換えであるが、例外的に異なる品詞の単語の言い換えが存在する。

本調査で見つかったのは、複合名詞を構成する名詞を形容詞に言い換えるものである。これは、当該名詞が形容詞語幹となる場合に限られる。以下に例を示す。

例 6 新規開拓先を探す → 新規の開拓先を探す

形容詞は名詞と異なり活用するので、その活用形を決定する必要が生じる。形容詞は名詞に接続する場合、活用型がナ形容詞の場合はダ列基本連体形を、活用型がナノ形容詞の場合はダ列特殊連体形をとるので、言い換え後の活用形は、このように決定すればよい。

言い換え元活用型が (ナ形容詞, ナノ形容詞, ナ形容詞特殊) で
言い換え先活用型が (母音動詞, 子音動詞カ行, 子音動詞カ行促音便形, 子音動詞ガ行, 子音動詞サ行, 子音動詞, タ行, 子音動詞ナ行, 子音動詞バ行, 子音動詞マ行, 子音動詞ラ行, 子音動詞ラ行イ形, 子音動詞ワ行, 子音動詞ワ行文語音便形) ならば異なる活用形の対応は以下の通り。

言い換え元活用形	言い換え先活用形
ダ列基本連用形	*
ダ列基本条件形	基本条件形
ダ列基本連体形	基本形
ダ列特殊連体形	基本形
ダ列タ形	タ形
ダ列タ系条件形	タ系条件形
ダ列タ系連用形	*
デアル列基本形	基本形
デアル列基本条件形	基本条件形
デアル列基本連用形	*
デアル列タ形	ダ列タ形
デアル列タ形条件形	ダ列タ系条件形
デアル列タ系連用形	*

表 2.2: 異なる活用型間の活用形対応表 1

言い換え元活用型 (ナ形容詞, ナノ形容詞, ナ形容詞特殊) で
言い換え先活用型 (イ形容詞アウオ段, イ形容詞イ段, イ形容詞イ段特殊) ならば異なる活用形の対応は以下の通り

言い換え元活用形	言い換え先活用形
ダ列基本連用形	基本連用形
ダ列タ系連用テ形	基本連用形
ダ列基本連体形	基本形
ダ列特殊連体形	基本形
ダ列基本条件形	基本条件形
ダ列タ形	タ形
ダ列タ系条件形	タ系条件形
ダ列タ系連用形	タ系連用形
デアル列基本形	基本形
デアル列基本条件形	基本条件形
デアル列基本連用形	基本連用形
デアル列タ形	タ形
デアル列タ系条件形	タ系条件形
デアル列タ系連用形	タ系連用形

表 2.3: 異なる活用型間の活用形対応表 2

言い換え元活用型 (ナノ形容詞) で
言い換え先活用型 (ナ形容詞) ならば異なる活用形の対応は以下の通り

言い換え元活用形	言い換え先活用形
ダ列特殊連体形	ダ列基本連体形

表 2.4: 異なる活用型間の活用形対応表 3

2.4　単語列から別の単語列への言い換え

単語列を別の単語列の置き換える言い換えが 111 例あった。これらの言い換えは、言い換える単語列の統語的性質に着目すると、名詞句の言い換えと動詞句の言い換えに分けられる。

2.4.1　名詞句から別の名詞句への言い換え

名詞句を別の名詞句へ言い換える場合は、名詞を名詞に置き換える場合と同様に、単に、単語列の置き換えを行えばよい。以下に例を示す。

例 7 競争促進 → 競争を促すこと

このような言い換えは、単語列を単語列に置換する言い換え規則を用意することによって実現できる。

2.4.2　動詞句から別の動詞句への言い換え

動詞句の言い換えにおいては、各種の難しい問題が発生する。ここでは、

1. 複合動詞相当句の言い換え
2. ヴォイスの調整を必要とする言い換え
3. 修飾句の調整を必要とする言い換え

の 3 つの場合に分けて、解決しなければならない問題を示す。

複合動詞相当句の言い換え

複合動詞相当句の言い換えは、動詞の言い換えと同様に活用形を調整する必要が生じる。以下に例を示す。

例 8 休業するに至った → 休業した

この例では「する」の活用形を「至った」の活用形を合わせ、「した」とする必要がある。

ヴォイスの決定が必要な言い換え

サ変名詞を動詞化する言い換えでは、各種の難しい問題が発生する [2]。例えば「サ変名詞+を+図る」は文脈によって「サ変名詞+する」に言い換えられたり、「サ変名詞+させる」に言い換えられたりする。以下に例を示す。

例 9 税制の改革を図る → 税制を改革する

例 10 組織の存続を図る → 組織を存続させる

「する」と「させる」のどちらに言い換えるかを正しく決定することは容易ではない。しかし、上記のように「〜+の+サ変名詞+を+図る」の場合は、「〜+の」が「〜+を」と言い換えられるため、

1. 「サ変名詞+する」が他動詞となるならば「する」（例 9）を、
2. 「サ変名詞+する」が自動詞となるならば「させる」（例 10）

とするのがよいと考えられる。

修飾句の調整が必要な言い換え

動詞化するサ変名詞が連体修飾句によって修飾されている場合、サ変名詞の動詞化と同時に、その連体修飾句を連用修飾句に変換する必要が生じる。以下に例を示す。

例 11 東京への出張をたびたび行う → たびたび東京へ出張する

例 12 たくさんの寄付を頻繁に行う → 頻繁にたくさん寄付する

例 13 大きな改訂を 3 回も行う → 3 回も大きく改訂する

連体修飾にはいろいろな形式が存在する。それらのリストとそれぞれを連用修飾に変換する方法を表 2.5に示す。但し、この表において、『名詞+へ格+「の」』は『名詞+ニ格』となる場合もある。

2.5　言い換えシステムの設計

以上の調査に基づき、言い換えシステムの設計においては以下のような方針を立てた。

1. 言い換えの対象となる文は、あらかじめ形態素解析し、その結果得られる形態素列を言い換え処理の対象とする。

連体修飾	→	連用修飾
形容詞		
イ形容詞基本形	→	イ形容詞基本連用形
ナ形容詞ダ列基本連体形	→	ナ形容詞ダ列基本連用形
名詞+「の」	→	名詞+「を」
名詞+格助詞+「の」	→	名詞+「格助詞」
名詞+格助詞相当句+「の」	→	名詞+「格助詞相当句」
名詞+格助詞相当句基本形	→	名詞+「格助詞相当句」
副詞+「の」	→	副詞

表 2.5: 連体修飾の連用修飾への言い換えルール

2. どのような言い換えを行うかは言い換え規則によって定義できるものとする。1つの言い換え規則は形態素列の置換規則とする。

3. 活用形の調整などを行う関数部を用意し、言い換え規則中でそれらを参照できるようにする。

第 3 章

言い換えシステム

3.1　言い換えシステムの概要

作成したシステムの概要を図3.1に示す。本システムは以下の 6 つの部分から構成される。

1. 前処理モジュール
   入力文を形態素解析し、形態素列に変換する。

2. 言い換えルールセット
   どのような言い換えを行うかを規則集合として記述したもの。言い換えシステムの動作を規定するプログラムに相当する。

3. ルールコンパイラ
   外部形式で記述された言い換えルールセットを内部形式に変換する。

4. 言い換えエンジン
   形態素列に対して内部形式に変換された言い換えルールセットを適用し、ルールに従った部分形態素列の置換を実行する。

5. 関数群
   言い換えルールで使用できるサブルーチン群。活用形の調整など行う関数が用意されている。

6. 辞書群
   サブルーチンで利用される辞書の集合。

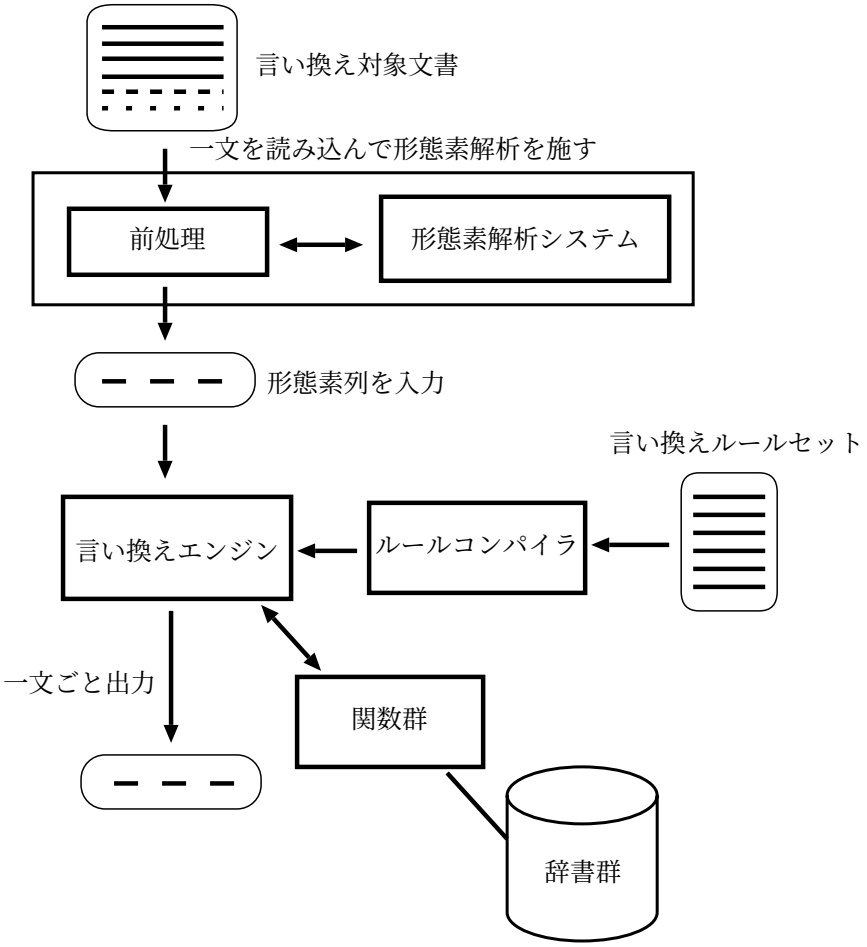

図 3.1: 言い換えシステムの概要

本システムは、入力として一文 (文字列) を受け取り、以下の処理を行なって言い換えた結果 (文字列) を出力する。

1. 入力された文を形態素解析し、形態素列に変換する。この処理は前処理モジュールによって行われる。
2. 形態素列に対して言い換えルールセットを適用し、適用後の形態素列を得る。 言い換えルールセットはあらかじめルールコンパイラによって内部形式に変換されており、言い換えエンジンは内部形式に変換されたルールセットに従って実際の言い換えを行う。
3. 得られた形態素列を文字列に変換し、出力する。

以下では、言い換えルールセットの記述形式 (外部表現)、ルールコンパイラ、言い換えエンジン、関数・辞書群の詳細について述べる。

3.2 言い換えルールセット

言い換えルールセットは、以下の要素から構成される。

1. 宣言
2. マクロ宣言
3. 言い換えルール

以下、それぞれの要素について述べる。

3.2.1 宣言

ルールセットの名称、システムの動作モード、システムが処理の対象とする形態素に関する情報などを指定する。宣言は以下のように記述する。

<宣言名称> (<値> ...)

宣言名称には、name、scan、restart、fields、values の 5 つがある。

name

言い換えルールセット系の名称を以下の書式で定義する。

name(ルールセット名)

scan

言い換えを行う際の走査方法を以下の書式で指定する。

scan(走査方法)

走査方法については 2 通りの方法がある。

forward 入力の前方から後方に向かって形態素列を走査しながら言い換えを試みる (図 3.2)

backward 入力の後方から前方に向かって形態素列を走査しながら言い換えを試みる (図 3.3)

restart

1 つの言い換えが実際に行われた後、次にどの位置から走査を再開するかを以下の書式で指定する。

restart(再開位置の指定の設定)

再開位置には次の 4 つがある。

none 再開しない。すなわち、ルールが一度でも適用されたら、その段階で言い換えルールセットの適用を終了する

skip 書き換えられた語の次の語から走査を再開する。すなわち、走査方法が forward ならば、書き換えられた語の直後から後方へ走査を再開する。操作方法が backward ならば、書き換えられた語の直前から前方へ走査を再開する (図 3.4)。

local 書き換えられた部分を再び走査する。すなわち、 走査方法が forward ならば、書き換えられた部分の先頭の語から後方へ走査を再開する。走査方法が backward ならば、書き換えられた部分の末尾の語から前方へ走査を再開する (図 3.5)。

global 最初から走査をやり直す。すなわち、走査方法が forward ならば、文の先頭から再び後方への走査を行なう。走査方法が backward ならば、文の末尾から再び前方へ走査を行なう (図 3.6)。

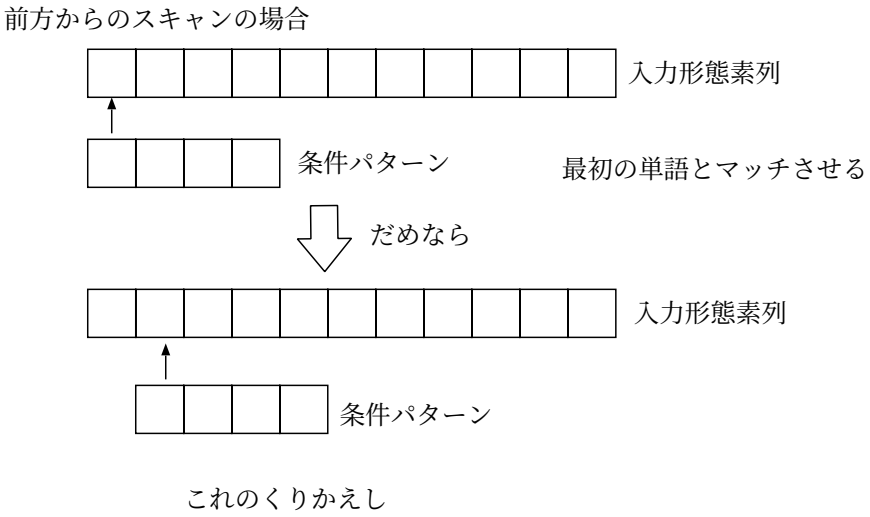

図 3.2: 前方からの走査

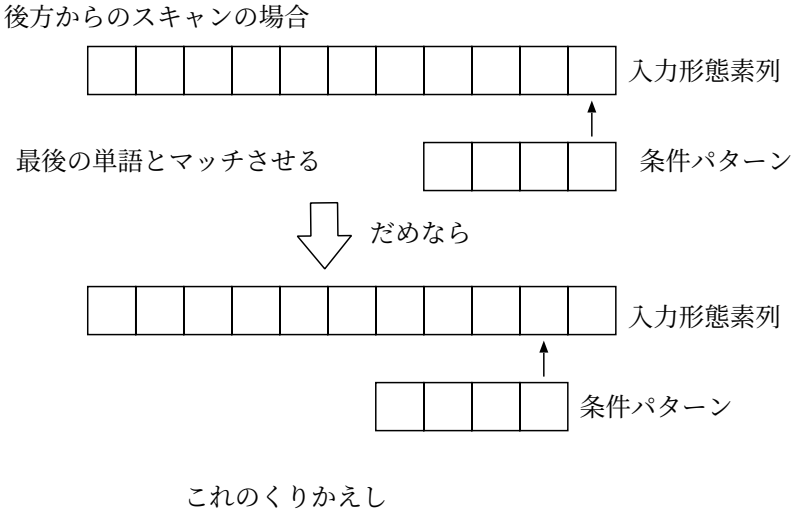

図 3.3: 後方からの走査

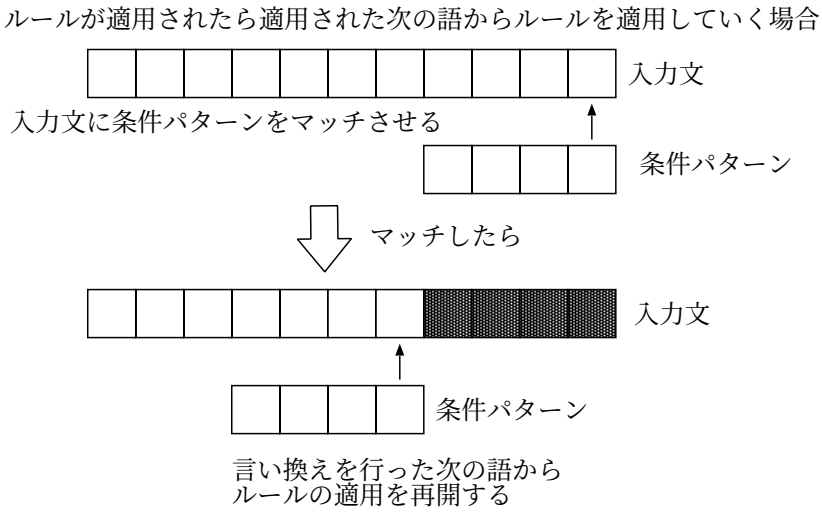

図 3.4: 走査方法が backward で再開位置が skip のケース

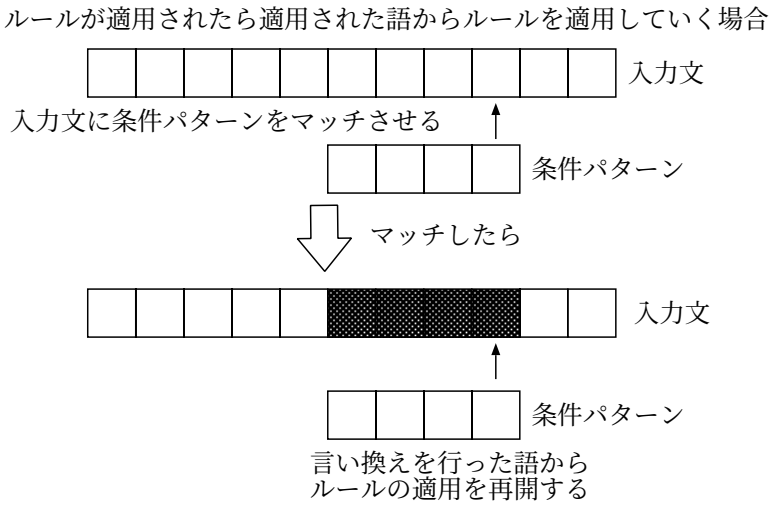

図 3.5: 走査方法が backward で再開位置が local のケース

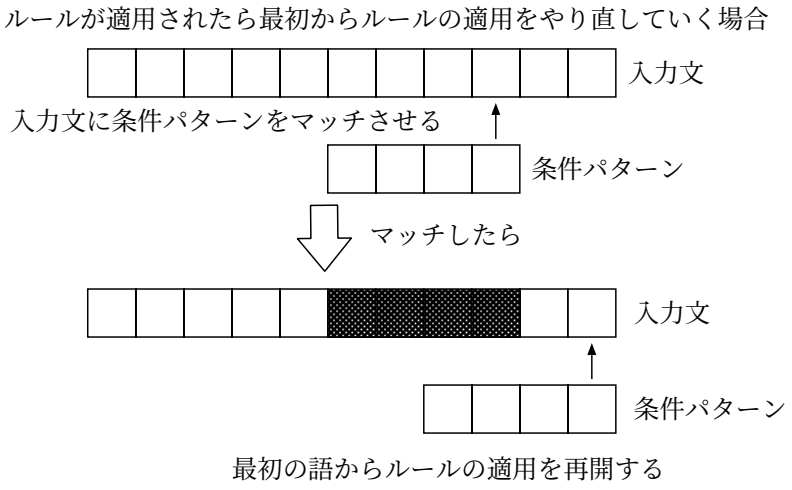

図 3.6: 走査方法が backward で再開位置が global のケース

fields

システムの処理対象となる形態素が、いくつのフィールドから構成されるのかを以下の書式で定義する。

fields(フィールド名のリスト)

フィールドの指定においては、ある特定のフィールドをデフォルトフィールドとして指定することができる。この指定は、フィールド名の先頭に '!' を付けることによって行う。デフォルトフィールドを指定することで、言い換えルール記述においてフィールド名を省略することができる。

本システムは形態素解析システムとして JUMAN [3] を用いることを仮定しているので、以下のように定義することになる。

fields(表層形, !標準形, 品詞, 品詞細分類, 活用型, 活用形);

すなわち、形態素は、表層形、標準形、品詞、品詞細分類、活用型、活用形の 6 つのフィールドから構成され、デフォルトフィールドは標準形である。

values

以下の書式で各フィールドの取りうる値を定義することができる。

values(フィールド名, <値> ...)

全てのフィールドに values を定義する必要はない。但し、あらかじめ values を定義することでデフォルトフィールド以外でもフィールド名を省略することができる。

3.2.2 言い換えルール

言い換えルールは「条件部」と「生成部」から構成されており、以下の書式で記述される。

```
条件部 --$>$ 生成部 ;
```

条件部

条件部は、ルールの適用条件を以下の書式で記述する。

パターン条件列 もしくは パターン条件列 && 付加条件列

パターン条件列は、書き換えの対象となる形態素列に対する条件を指定するためのもので、パターン条件の列として記述される。ここでパターン条件は、一つの形態素に対する条件を指定するもので、その形態素のそれぞれのフィールドが満たすべき条件の集合ををフィールド条件列として記述する。

フィールド条件は、形態素のある特定のフィールドが満たすべき条件を指定するもので、以下の 4 種類がある。

1. <フィールド指定> <比較記号> <文字列>
2. <フィールド指定> <照合記号> <正規表現>
3. <比較記号> <文字列>
4. <照合記号> <正規表現>
5. <文字列>

フィールド指定には以下の 2 つがある。

1. フィールド名 (文字列)
2. フィールド番号 (数字)

比較記号には以下の 2 つがある。

- =(等号)
- !=(不等号)

照合記号には以下の 2 つがある。

- =~(パターンにマッチするかどうか)
- !~(パターンにマッチしないかどうか)

フィールド指定が省略された場合は、デフォルトフィールドが仮定される。単に文字列が指定された場合は、その文字列が values によって定義される値ならば values で定義されたフィールド名が仮定される。その文字列が values によって定義されていないならば、デフォルトフィールド名が仮定される。

なお、これらとは別に、パターン条件には制御記号を付けることができる。制御記号には３種類のものがある。

1. ^ → この記号が付加されたパターン条件にマッチする形態素は、文の先頭の形態素でなくてはならない。
2. $ → この記号が付加されたパターン条件にマッチする形態素は、文の先頭の形態素でなくてはならない。
3. ! → この記号が付加されたパターン条件をベースとする。ベースは、照合の起点となる形態素 (フォーカスと呼ばれる) と照合する。

最後の制御記号は、パターン条件列のなかのどのパターン条件を、フォーカス形態素（照合の起点となる形態素）と照合するかを明示的に示すためのもので、もし、明示的に指定されなかった場合は、

- 走査方法が forward の場合は、パターン条件列の先頭のパターン条件
- 走査方法が backward の場合は、パターン条件列の末尾のパターン条件

がベースとなり、フォーカス形態素と照合される。

付加条件列は、パターン条件列以外に満たすべき条件を指定するためのもので、以下のように記述する。

関数 1, 関数 2, ...

これらの関数は (<flag> <value> ...) という値を返し、flag が true の場合、その条件は満たされたものと解釈する。なお、関数の返す値のうち、<flag>を除いた部分は、@変数（@1,@2,...）に格納され、生成部において参照可能である。（関数１の値は@1 に、関数２の値は@2 に格納される。）

生成部

生成部は、ルールの条件部が満たされた場合、パターン条件列にマッチした形態素列をどのような形態素列で置き換えるべきかを指定するもので、以下の書式で記述する。

生成パターン, 生成パターン, ...

生成パターンは次のいずれかである。

1. $変数：パターン条件列にマッチした形態素は、前から順に$1、$2 のような$変数として参照することができる。
2. @変数：付加条件列で呼び出した関数の返す値は、前から順に@1、@2 にような@変数として参照することができる。
3. 形態素マクロ：3.2.3節参照
4. 形態素：定数
5. 関数呼び出し：3.3節参照。関数の引数には、生成パターンが記述できる。

なお、言い換えルールの例は、3.4節に示す。

3.2.3 マクロ宣言

マクロには、形態素マクロとパターンマクロの 2 種類がある。形態素マクロは生成パターンとして用いることができ、パターンマクロはパターン条件として記述することができる。

形態素マクロ

形態素マクロは、よく使われる形態素を簡潔に表現するために、あらかじめマクロとして定義しておくものである。以下に書式を示す。

マクロ名 == 形態素 ;

以下に例を示す。

_も == も:も:助詞:副助詞:: ;

このようなマクロを定義することにより、生成パターン中で、も:も:助詞:副助詞:: と記述する代りに、_も と記述できる。これにより生成パターンをより簡潔に記述することができる。

パターンマクロ

パターンマクロは、複数のパターン条件列を一つのパターン条件としてまとめるためのものである。以下に書式を示す。

```
%マクロ名 ==
        パターン条件列 1 ||
        パターン条件列 2 ||
            …
        パターン条件列 n ;
```

以下に例を示す。

```
%連体修飾 ==
        名詞，名詞，%格助詞相当句，の ||
        名詞，%格助詞相当句，の ||
        名詞，名詞，的，な ||
        名詞，的，な ||
        名詞，名詞，の ||
        名詞，の ||
        副詞，の ||
        イ形容詞アウオ段:基本形 ||
        イ形容詞イ段:基本形 ||
        イ形容詞イ段特殊:基本形 ||
        ナノ形容詞:ダ列特殊連体形 ||
        ナノ形容詞:ダ列基本連体形 ||
        ナ形容詞:ダ列基本連体形 ||
        ナ形容詞:ダ列基本連体形 ||
        指示詞 ;
```

上の例ではパターンマクロ %連体修飾 は 15 個のパターン列を見かけ上 1 つの条件にまとめている。書き換えルールの条件部に %連体修飾 と記述した場合、それはこれら 15 個のパターン列のいずれかにマッチするという条件を指定したと解釈される。

関数名	内容	使用する辞書
form	ある形態素を指定した活用形に変更する	
adjust	ある形態素列と別の形態素との間の活用変化を調整する	
lookup	入力の語を元に辞書を引く	単語言い換え辞書 名詞形容詞対応辞書
suru_saseru	「サ変名詞+する」が自動詞か他動詞かを判断する	サ変動詞辞書
delete	語を削除する	
call	別のルールセットを呼び出す	

表 3.1: 関数・辞書群一覧

3.3 関数・辞書群

関数は、言い換えルールの条件部や生成部において、フィールド条件やマクロ変数などだけでは記述できない処理を実現するためのものある。

関数呼び出しは、以下の書式で記述される。

&関数名 (引数....)

本システムが提供する関数と、それが使用する辞書群を表 3.1に示す。以下ではこれらの関数と辞書について述べる。

3.3.1 form 関数

form 関数は、活用形と形態素を引数として取り、その形態素の活用形を指定した活用形に変更する。

&form(活用形, 形態素)

以下に例を示す。

例: &form(未然形, する:する:動詞::サ変動詞:標準形)
→ さ:する:動詞::サ変動詞:未然形

言い換え元	言い換え先
単語 (標準形)	表層形:標準形:品詞:品詞細分類:活用型:活用形

表 3.2: 単語言い換え辞書

この form 関数は、形態素解析システム JUMAN で用いている活用変化対応表を利用することによって実現されている。

3.3.2 adjust 関数

adjust 関数は、ある形態素列 (配列 A) の活用形をある形態素 B の活用形に合わせる。

&adjust(形態素 B, 配列 A)

以下に例を示す。

例: &adjust(行い:行う:動詞::子音動詞ワ行:基本連用形, する:する:動詞::サ変動詞:標準形)
→ し:する:動詞::サ変動詞:基本連用形

この関数は、form 関数で述べた活用形対応表を使用することによって実現されている。

3.3.3 lookup 関数

lookup 関数は入力に形態素と辞書名を取り、その入力形態素が辞書に登録されているかどうかを調べる。この関数は付加条件の 1 つとして以下のように記述することができる。

&lookup(形態素, 辞書名)

この関数は<フラグ, 値>という値を返す。引数として与えられた形態素が辞書に登録されていた場合は、true と言い換え先の形態素を返し、辞書に登録されていなかった場合は、false を返す。

なお、本システムではこの関数と単語言い換え辞書を用意することで単語間の言い換えを、名詞形容詞対応辞書を用意することで名詞から形容詞への言い換えを実現している。

単語言い換え辞書は表 3.2のような形式で記述される。

名詞形容詞対応辞書は名詞とナ形容詞の対応表であり、表 3.3のような形式で記述される。

この名詞形容詞対応辞書は、JUMAN の名詞辞書と形容詞辞書から機械的に作成した。

言い換え対象名詞	ナ形容詞
標準形	表層形:標準形:品詞:品詞細分類:活用型:活用形

表 3.3: 名詞形容詞対応辞書

3.3.4 suru_saseru 関数

suru_saseru 関数は、入力にサ変名詞をとり、そのサ変名詞が「する」を伴った場合に他動詞になるか自動詞になるかを調べ、他動詞の場合は「する」を、自動詞の場合は「させる」を出力する。この関数は生成パターンの 1 つとして以下のように記述することができる。

&suru_saseru(サ変名詞の形態素)

なお、他動詞か自動詞かの判断には、サ変名詞-動詞相当句対応辞書 [2] を用いた。

3.3.5 delete 関数

delete 関数は、語を削除する言い換え規則を記述する際に、生成パターンとして記述する。

```
条件部 --> &delete() ;
```

この関数は、空列を返すため、パターン条件列にマッチした形態素列が削除されることになる。

3.3.6 call 関数

call 関数は、別のルールセットをサブルーチンとして適用する場合に用いられる。以下に書式を示す。

&call(ルールセット名, 形態素列)

3.4 言い換えルールの例

3.4.1 単語から単語の言い換え

単語から単語の言い換えルールの例を示す。

まずもっとも簡単な名詞から名詞への言い換えルールについて例文 1 を基に示す。

例文 1 金額の増嵩→金額の増加

この言い換えを実現するのにもっとも単純な方法は、言い換えの組合わせを全てルールに記述することである（ルール例 1）。

ルール例 1 増嵩-->増加

しかし、この方法では記述するルール数が非常に多くなる。したがって、より上手な解決方法としてあらかじめ言い換えの組合わせを辞書に記述し、その辞書を引くことで言い換えを実現する（ルール例 2）。

ルール例 2 名詞 && lookup($1, “単語言い換え辞書”) --> @1;

ルール例 2 は、「品詞が名詞ならば、その名詞を基に単語言い換え辞書を引き、辞書中に対応する言い換え語があれば、言い換え語を出力する」というルールになる。

例文 1 は、名詞から名詞への言い換えであったが、これが活用する語の言い換えならば、活用変化を考慮する必要がある。

例文 2 春の陽気にいざなわれて→春の陽気にさそわれて

例文 2 の言い換えでは、「いざなわれる-->さそわれる」という言い換えルールが適用されるが、動詞の言い換えのためルール例 2 に活用変化に対応させる関数を付け加える。結果がルール例 3 である。

ルール例 3 品詞=~(動詞|形容詞) && lookup($1, “単語言い換え辞書”)
-->&adjust($1, @1);

ルール例 3 は、「品詞が動詞か形容詞ならば、その語を基に単語言い換え辞書を引き、辞書中に対応する言い換え語があれば、言い換え語を活用変化に対応させた上で出力する」というルールになる。

3.4.2 複合動詞相当句の言い換え

複合動詞相当句の言い換えルールについて、例文 3,4 を基に示す。

例文 3 頂上に到達するに至った→頂上に到達した

例文 4 峠を越えるに至った→峠を越えた

言い換えた部分を切り出してルールを記述すると、ルール例 4,5 のように書ける。

ルール例 4 する, に, 至った --> した;

ルール例 5 越える, に, 至った --> 越えた;

ルール例 4,5 は、ルール例 3 のように活用変化の対応関数を利用することでルール例 6,7 のように書き直せる

ルール例 6 する, に, 至る --> &adjust($3, $1);

ルール例 7 越える, に, 至る --> &adjust($3, $1);

生成パターンから、条件部の第 1 要素は動詞であれば適用可能であることが分かるため、ルール例 6,7 はルール例 8 のようにまとめられる。

ルール例 8 動詞, に, 至る --> &adjust($3, $1);

3.4.3 ヴォイスの調整が必要な言い換え

ヴォイスの調整が必要な言い換えの言い換えルールについて、例文 5,6 を基に示す。

例文 5 組織の存続を図る→組織を存続させる

例文 6 税制の改革を図る→税制を改革する

例文 5,6 の条件部は次の様に記述できる。

条件部 名詞, の, サ変名詞, を, 図る

一方生成部の記述は、同じ条件部からサ変名詞の動詞化において 2 通りの記述がありうる。例文 5 は生成部 1 が、例文 6 は生成部 2 が対応する。

生成部 1 $1, _を, $3, _させる;

生成部 2 $1, _を, $3, _する;

ここで「する、させる」の区別のために&suru_saseru 関数を用意することで生成部 1,2 の記述を生成部 3 のように、1 つにまとめて記述することにする。

生成部 3 $1, _を, $3, &suru_saseru($3);

最後に「する、させる」は活用語の言い換えのため、活用変化の対応関数を用いて活用変化に対応させる（生成部 4）。

生成部 4 $1, _を, $3, &adjust($5, &suru_saseru($3));

以上より、条件部と生成部 4 を組み合わせて、ルール例 9 を導き出せる。

ルール例 9 名詞, の, サ変名詞, を, 図る --> $1, _を, $3, &adjust($5, &suru_saseru($3));

3.4.4 修飾句の言い換え

修飾句の言い換えルールについて例文 7,8 を基に示す。

例文 7 東京への出張を行う→東京へ出張する

例文 8 言い換えについての研究を行う→言い換えについて研究する

まずサ変名詞の動詞化について、言い換えルールを作成するとルール例 10 ように書ける。

ルール例 10 サ変名詞, を, 行う --> $1, _する;

サ変名詞の動詞化に伴うサ変名詞を修飾する部分の言い換えを、ルール例 10 に追加すると例文 7 からはルール例 11 が、例文 8 からはルール例 12 が書ける。

ルール例 11 名詞, へ, の, サ変名詞, を, 行う--> $1, $2, $4, _する;

ルール例 12 名詞, に, ついて, の, サ変名詞, を, 行う--> $1, $2, $3, $5, _する;

ルール例 11,12 で、下線部は連体修飾句から連用修飾句への言い換えを表している。ここであらかじめ連体修飾句を表す形態素列をマクロとして定義しておくことでルールの記述を簡潔にする。また、定義したマクロを利用して別のルールセットを適用する関数を用意する。以上から、ルール例 11,12 はルール例 13 のようにまとめて書ける。

ルール例 13 %連体修飾, サ変名詞, を, 行う -->
&call(連体修飾から連用修飾への変換, $1), $2, _する;

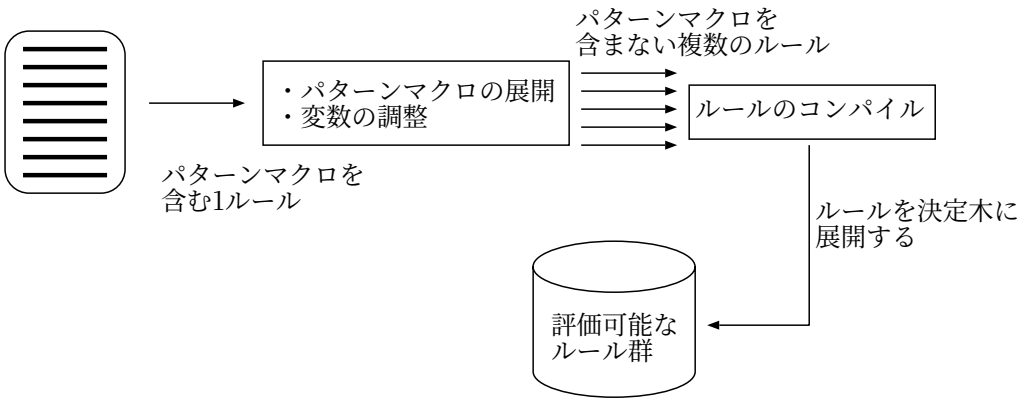

図 3.7: ルールコンパイラ

3.5 ルールコンパイラ

ルールコンパイラは、ファイルに外部形式で記述されたルールセットを内部形式に変換する処理を行う。この処理は以下の 3 ステップで実現されている。

1. ルールセットの読み込み、解析
2. マクロ (形態素マクロ、パターンマクロ) の展開
3. ルールセットの内部形式への変換

ルールコンパイラの概要を図 3.7に示す。

以下では、マクロ展開とルールセットの内部形式への変換について述べる。

3.5.1 マクロの展開

言い換えルールを内部形式に変換する前に、あらかじめルール中に含まれるマクロ呼び出しを全て展開しておく。

マクロには形態素マクロとパターンマクロの 2 種類ある。

形態素マクロの展開は、単なる文字列の置換である。

名詞, _が → 名詞, が:が:助詞:格助詞::

一方、パターンマクロは、複数の異なるパターン条件列をまとめたものであるため、これらを展開して複数のルールを作成することになる。パターンマクロは、見かけ上の長さは１であるが、展開された結果のパターン条件列は、長さが１とは限らない。このため、マクロ展開時に$変数の番号を付け変える必要が生じる。図 3.8に例を示す。

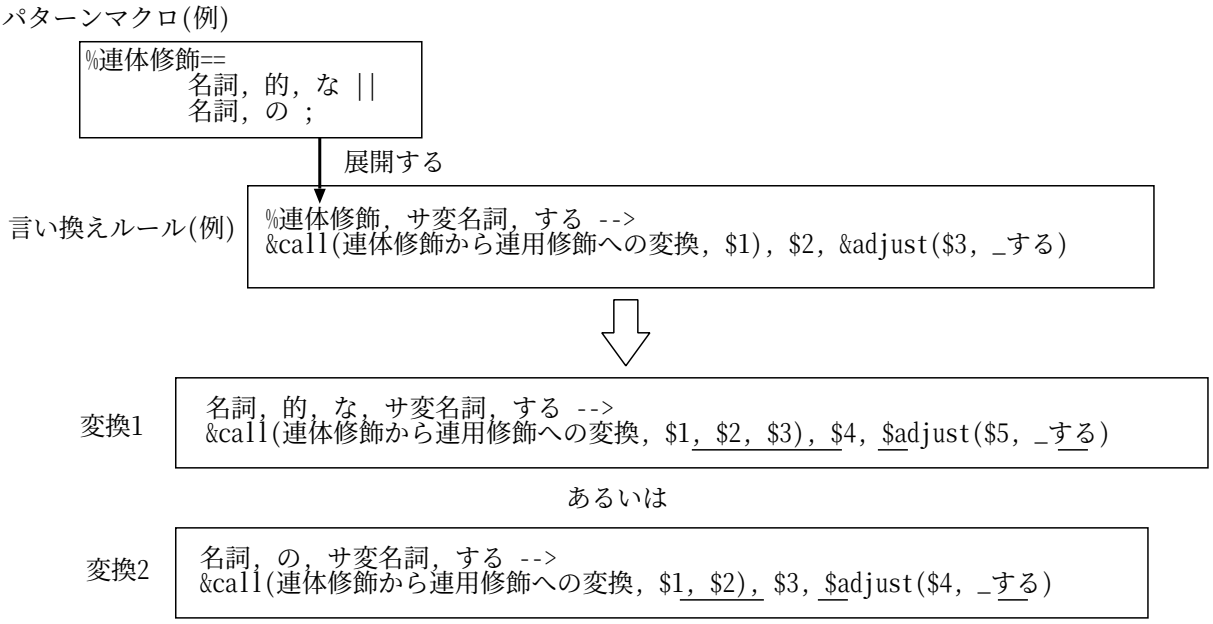

図 3.8: パターンマクロの展開

3.5.2 ルールセットの内部形式への変換

内部形式への変換は、以下の２つのことを行なう。

1. それぞれのルールのパターン条件列をフィールド条件（内部形式）の列に変換する。
2. 同一のフィールド条件をマージすることにより、複数のフィールド条件列から決定木を構成する。

以下の例を用いてこれらの処理について説明する。

ルール例 1 サ変名詞，を，行う --> $1，する ；

ルール例 2 会議，を，行う --> 相談:相談:名詞:サ変名詞::，す　　る ；

まず、それぞれのパターン条件列を、フィールド条件（内部形式）の列に変換する。

フィールド条件例 1 [-2, 3, eq, サ変名詞], [-1, 1, eq, を], [0, 1, eq, 行う]

フィールド条件例 2 [-2, 1, eq, 会議], [-1, 1, eq, を], [0, 1, eq, 行う]

ここで、例えば、[-2, 3, eq, サ変名詞] は、「フォーカス形態素から-2 の位置にある（2 つ前の）形態素の 3 番目のフィールド（品詞）の値がサ変名詞であるかどうかを調べなさい」と

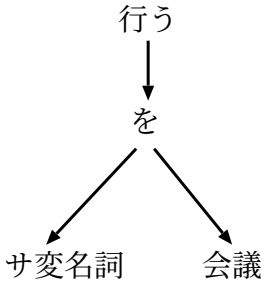

図 3.9: 決定木の例

いうことを意味する。なお、ここでは、scan モードとして、backward を仮定した。この場合、最後のパターン条件がベース（フォーカス形態素と照合すべきパターン条件）となる。

次の、これらのフィールド条件列をマージする。この例の場合、scan モードが backward であることを仮定しているので、後ろから条件をマージする。後ろ２つの条件は同一なので、図 3.9のような決定木が構成されることになる。この決定木は、次のようなプログラムだと解釈される。すなわち、

1. まず、[0, 1, eq, 行う] が成り立つかどうか調べる。成り立たなければ、ルールセットの適用は失敗である。

2. 次に、[-1, 1, eq, を] が成り立つかどうか調べる。成り立たない場合は、ルールセットの適用は失敗である。

3. [-2, 3, eq, サ変名詞] が成り立つかどうか調べる。成り立つ場合は、フィールド条件例 1 の条件が全て満たされた。（生成部を実行し、得られた形態素列で置換を行なう。）

4. もし、3 が成り立たない場合、[-2, 1, eq, 会議] が成り立つかどうか調べる。成り立つ場合は、フィールド条件例 2 の条件が全て満たされた。（生成部を実行し、得られた形態素列で置換を行なう。）

5. もし、4 が成り立たない場合、ルールセットの適用は失敗である。

3.6 言い換えエンジン

言い換えエンジンは、入力形態素列に対してルールセットを適用し、あるルールの条件部が満たされたならば、そのルールのパターン条件列にマッチした部分を、生成部で指定

される形態素列で置き換えることを行なう。

ルールの適用処理は、以下の手順で行なう。

1. 入力された形態素列を、指定された走査方法で走査する。この走査において、形態素列の形態素に一つづつ焦点を当てていく。これをフォーカスと呼ぶ。フォーカスを固定して、以下の処理を行なう。

   (a) 決定木を順に評価していき、条件部が全て満たされる規則を見つける。

   (b) もし、そのような規則が見つかった場合は、パターン条件列とマッチした形態素列を、その規則の生成部を評価して得られる形態素列で、置き換える。なお、以降の処理は、restart に従う。

   (c) もし、適用できる規則が見つからない場合は、このフォーカスに対する処理を終了する。(フォーカスを一つずらす)。

第 4 章

政府自治体文書の自動言い換え

どのような言い換えが適切であるかは、言い換えを行う対象となる文書の種類や分野に強く依存すると考えられる。ここでは、政府、自治体、公的機関が発行する報告文書を対象とし、そこに見られる難解な表現をより分かりやすい表現に言い換える規則集合を作成し、それを実際の文書に適用する実験を行った。

4.1 政府自治体文書の特徴

政府自治体の報告文書は、全体として堅苦しく難解な、いわゆる「役所文章」で書かれている [4]。これらの堅苦しさ・難しさの一つの原因は、難しい単語や表現、サ変動詞の多用によるものと考えられる。したがって、難しい単語をよりやさしい単語に置き換えたり、難しい表現をより簡単な表現に置き換えることによって、原文を読みやすい文に変換することができると考えられる。

以下に作成した言い換えルールを示す。

4.1.1 冗長な語句の削除

ルール 1 パターン列→[削除]

この規則は冗長な語句の削除である。削除する語句は以下の 12 パターンである。

- という形式がとられる
- いくこととなる
- と考えられる

真摯だ	まじめだ:まじめだ:形容詞::ナ形容詞:基本形
厳格だ	厳しい:厳しい:形容詞::イ形容詞イ段:基本形
有する	持つ:持つ:動詞::子音動詞タ行:基本形
厳格だ	厳しい:厳しい:形容詞::イ形容詞イ段:基本形
困難だ	難しい:難しい:形容詞::イ形容詞イ段:基本形
難解だ	難しい:難しい:形容詞::イ形容詞イ段:基本形
先般	さきごろ:さきごろ:名詞:時相名詞::

表 4.1: 単語言い換え辞書

- こととしている
- 傾向がある
- ことは否めない
- ことは否定しえない
- おおむね
- 全力で
- 積極的に
- いわゆる
- 極めて

4.1.2 単純な置き換え

ルール 2 単語→単語

この規則は単語言い換え辞書を用いて、よりやさしい単語に置き換えるものである。単語言い換え辞書には 37 組の言い換えを登録した。その一部を表 4.1に示す。

4.1.3 名詞句の言い換え

ルール 3 名詞+サ変名詞 → ナ形容詞ダ列基本連体形+サ変名詞

ルール 4 名詞+サ変名詞 → ナノ形容詞ダ列特殊連体形+サ変名詞

この規則は名詞形容詞対応辞書を参照し、名詞が形容詞の語幹となる場合に、「名詞+サ変名詞」を「ナ形容詞+名詞」に言い換える。

「名詞+『的』」はナ形容詞語幹に相当し、「名詞+化」はサ変名詞に相当する。これらの場合に対しても規則を作成した。

例:完全勝利→完全な勝利

4.1.4 動詞句の言い換え

ルール 5 サ変名詞+する+こと+が+可能だ→サ変名詞+できる

ルール 6 動詞+に+至る→動詞 (至るの活用形に合わせる)

これらの規則は複合動詞句を簡略にするためのものである。

ルール 7 連体修飾 (A)+名詞+サ変名詞+を+「する、図る、行う」→ (A を連用修飾に変換したもの)+ナ形容詞ダ列基本連用形+サ変名詞+する

ルール 8 連体修飾 (A)+名詞+サ変名詞+に+「努める」→ (A を連用修飾に変換したもの)+ナ形容詞ダ列基本連用形+サ変名詞+ する

これらの規則は、名詞がナ形容詞の語幹になるとき適用可能である

例:新規開拓を図る→新規に開拓する

なお

- 「名詞+『的』」はナ形容詞語幹に相当する
- 名詞+「化」はサ変名詞に相当する

- 格助詞「を」の他にも副助詞 (から、さえ等) を取りうる
- 連体修飾は 0 から 2 個取りうる

ので、これらの場合に対しても規則を作成した。

ルール 9 連体修飾 (A)+名詞+サ変名詞+を+「する、図る、行う」→ (A を連用修飾に変換したもの)+名詞+を+サ変名詞+させる

ルール 10 連体修飾 (A)+名詞+サ変名詞+に+「努める」→ (A を連用修飾に変換したもの)+名詞+を+サ変名詞+させる

これらの規則では「サ変名詞+する」が自動詞になる場合に適用可能である。

例:組織存続を図る→組織を存続させる

なお

- 格助詞「を」の他にも副助詞 (から、さえ等) を取りうる
- 連体修飾は 0 から 2 個取りうる
- 名詞+「化」はサ変名詞に相当する。

ので、これらの場合に対しても規則を作成した。

ルール 11 連体修飾 (A)+名詞+サ変名詞+を+「する、図る、行う」→ (A を連用修飾に変換したもの)+名詞+を+サ変名詞+する

ルール 12 連体修飾 (A)+名詞+サ変名詞+に+「努める」→ (A を連用修飾に変換したもの)+名詞+を+サ変名詞+する

これらの規則では「サ変名詞+する」が他動詞になる場合に適用可能である。

例:シェア拡大を図る→シェアを拡大する

なお

- 格助詞「を」の他にも副助詞 (から、さえ等) を取りうる
- 連体修飾は 0 から 2 個取りうる
- 名詞+「化」はサ変名詞に相当する。

正しい言い換え	誤った言い換え	計
63(72%)	24(28%)	87

表 4.2: 言い換え結果

ので、これらの場合に対しても規則を作成した。

ルール 13 連体修飾 (A)+サ変名詞+を+連用修飾 (B)+「行う、図る、する」
→連用修飾 (B)+(A を連用修飾に変換したもの)+サ変名詞+する

ルール 14 連体修飾 (A)+サ変名詞+に+連用修飾 (B)+「努める」
→連用修飾 (B)+(A を連用修飾に変換したもの)+サ変名詞+する

例:中国についての研究を行う→中国について研究する
例:道路の整備に一層努める→一層道路を整備する

なお

- 格助詞「を」の他にも副助詞 (から、さえ等) を取りうる
- 連体修飾は 0 から 2 個取りうる
- 名詞+「化」はサ変名詞に相当する。

ので、これらの場合に対しても規則を作成した。
連体修飾を連用修飾に変換するルールは表 2.5で表される。

4.2 実験結果

実験対象として『第百三十六回国会における橋本内閣総理大臣施政方針演説』(14000 語強、28319 バイト) を用いた。総言い換え数は 87 件になった。これらの言い換えが正しいかどうかを人間が判定した結果を表 4.2に示す。87 件中、63 件 (72%) は言い換えが正しく行われた (元の意味を保ったまま、難解な表現をやさしく言い換えた)。残りの 24 件 (28%) は、言い換えの結果文の意味が変わったり、文法的に誤った文となったりした。これらの原因は以下のように分類できる。以下に問題がある言い換えの分析を行う。

1. 修飾句の係り受けが原因のもの (7 件)

   これは修飾句がどこの単語を修飾しているかが、用意した言い換えルールとは異なる場合に起こった。

   例:二十一世紀型経済社会の基盤の整備を行う→
   二十一世紀型経済社会を基盤を整備する

   この例では連体修飾が 2 個所現れているが最初の連体修飾は後ろの連体修飾に係っているので言い換えの際には最初の連体修飾は連用修飾に言い換える必要はない。しかし作成した規則では、連体修飾は全て連用修飾に変換するため文法的に誤った文を生成した。この問題を解決するためには、言い換える文をあらかじめ係り受け解析しておき、係り受け関係を考慮した言い換え規則を用意する必要がある。

2. 固有名詞および分離不可能な名詞句の言い換え (6 件)

   固有名詞や分離不可能な名詞句内の要素を言い換えてしまう例がみられた。

   例:公正取引委員会→公正な取引委員会

   この問題を解決するためには、固有名詞や分割不可能な複合語のリストを用意し、これらの語に対する言い換え規則の適用を抑制する機構を用意する必要がある。

3. JUMAN の解析ミス (4 件)

   本システムでは形態素解析システムとして JUMAN を利用しているが、JUMAN が正しい解析結果を出力しないため、誤った言い換え規則が適用されることがある。

   例:施策の一層の充実を図る→施策を一層を充実する

   「一層の」は「副詞+格助詞」と解析されるべきであるが、JUMAN では「名詞+格助詞」と解析されるため、連体修飾を連用修飾に変換する規則が誤って適用された。

4. おかしな表現 (2 件)

   文法的には間違っていないが、おかしな表現が生成される場合があった。

   例:必要な調査を行う→必要に調査する

   ナ形容詞「必要だ」は基本連用形となって述語を修飾することはない。しかし、それは文法的に不可能なのではなく意味的に不可能なのである。このような意味に関する

判定を本システムはすることができないのでシステム上では判断することができなかった。

5. 並列句の言い変え (2 件)

「名詞 A+の+サ変名詞 A+と+名詞 B+の+サ変名詞 B+を+行う」といった並列句を言い換える場合、「名詞 A+を+サ変名詞 A+し、名詞 B+を+サ変名詞 B+する」と複文に言い換えなければならないが、形態素解析のレベルでは並列句の解析ができなかった。

例:必要な機能の充実と防衛力の質的な向上を図る→
必要な機能の充実と防衛力を質的に向上する

6. その他 (3 件)

これ以外の原因と考えられるものが 3 件あった。

第 5 章

結論

本研究では、文章を言い換える手法について研究し、その成果を元に機械的に言い換えを行うシステムを作成した。

どのような言い換えを行うかは、言い換えを行う対象となる文書の種類や分野に強く依存すると考えられる。本研究では、堅苦しく難解な表現が多いと考えられる政府、自治体、公的機関などの報告文書を対象として選び、これらを読みやすいようにするための言い換えを研究対象とした。

WWW 上より調査対象として 2 文書を選び、その文書中の難解な表現をやさしく言い換える作業を行った。作成された言い換え例をまとめた上で、それらの言い換えを機械的に実現する手法を検討した結果、言い換えシステムの作成方針を次のように立てた。

1. 言い換えの対象となる文は、あらかじめ形態素解析し、その結果得られる形態素列を言い換え処理の対象とする。

2. どのような言い換えを行うかは言い換え規則によって定義できるものとする。1 つの言い換え規則は形態素列の置換規則とする。

3. 活用形の調整などを行う関数部を用意し、言い換え規則中でそれらを参照できるようにする。

作成されたシステムは以下の 6 つの部分から構成される。

1. 前処理
   入力文を形態素解析し、形態素列に変換する。

2. 言い換えルールセット

どのような言い換えを行うかを規則集合として記述したもの。言い換えシステムの動作を規定するプログラムに相当する。

3. ルールコンパイラ
   外部形式で記述された言い換えルールセットを内部形式に変換する。

4. 言い換えエンジン
   形態素列に対して内部形式に変換された言い換えルールセットを適用し、ルールに従った部分形態素列の言い換え (置換) を実行する。

5. 関数群
   言い換えルールで使用できるサブルーチン群。活用形の調整など行う関数が用意されている。

6. 辞書群
   サブルーチンで利用される辞書の集合。

また、システムは入力として一文を取り、言い換え処理を行った後一文を出力する。言い換え処理の内容は以下の通り。

1. 入力された文を形態素解析し、形態素列に変換する。この処理は前処理モジュールによって行われる。

2. 形態素列に対して言い換えルールセットを適用し、適用後の形態素列を得る。 言い換えルールセットはあらかじめルールコンパイラによって内部形式に変換されており、言い換えエンジンは内部形式に変換されたルールセットに従って実際の言い換えを行う。

3. 得られた形態素列を文字列に変換し、出力する。

政府自治体公的機関が発行する報告文書を対象とし、そこにみられる難解な表現を分かりやすい表現に言い換える規則集合を作成し、それを実際の文書に適用する実験を行った。実験対象として『第百三十六回国会における橋本内閣総理大臣施政方針演説』(14000 語強、28319 バイト) を用いた。実験の結果、総言い換え数 87 件のうち、63 件 (72%) は言い換えが正しく行われたが、残り 24 件 (28%) は、言い換えの結果文の意味が変わったり文法的に誤った文になったりした。以下に原因を示す。

- 係り受け解析が必要な言い換えがあった

- 言い換えルールの適用を抑制しなければならない言い換えでも抑制する機構がシステムになかった
- 形態素解析システムが誤った解析結果を返したため、誤った言い換えルールが適用された
- 並列句の複文への言い換えなど、構文解析が必要な言い換えがあった
- 意味的に不可能な言い換えがあったが、システムには判断不可能だった

実験の結果、以下のことが分かった。

1. 本システムは、単語列から単語列への置換で実現可能な言い換えを機械的に実行することができる。
2. 言い換えに、単語列中の語の係り受けの情報を必要とする場合、形態素解析のみの言い換えシステムでは誤った言い換え結果を出力する場合がある。
3. 言い換え結果が文法的に正しくても意味的に誤っている場合もある。

より多くの種類の言い換えを精度良く実現するためには、構文解析を利用し、構文解析木の一部を別の構文解析木に置き換える機能が必要となる。

謝辞

本研究を進めるにあたり、終始熱心なご指導を頂きました佐藤理史助教授に心から感謝致します。

中間審査などの折には諸先生方から貴重なご意見を頂きました。感謝致します。

また、日頃より研究において適切なアドバイスを頂いた知識工学講座の皆さんに感謝致します。ありがとうございました。

参考文献

[1] 益岡隆志, 田窪行則. 基礎日本語文法-改訂版- くろしお出版, 1994

[2] 近藤恵子, 佐藤理史, 奥村学「サ変名詞+する」の動詞への言い換え, 情報処理学会研究報告, 98-NL-127, pp179–186, 1998.

[3] 黒橋禎夫, 長尾眞, 日本語形態素解析システム JUMAN Version 3.5 使用説明書, 京都大学大学院工学研究科1998.

[4] イアン アーシーおえらがた 政官財の国語塾 中央公論社 1996.