修 士 論 文

質問応答集における質問文の標準形への自動変換

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

杉水流 英樹

1999 年 2 月 15 日

要 旨

本稿では、ネットニュースグループ fj.sys.sun の質問文を対象とし、様々な表現で言い表される質問文を標準形へ自動変換する手法について述べる。

目 次

第 1 章

序論

本論文では、ネットニュースグループ fj.sys.sun の質問文を対象とし、様々な表現で言い表される質問文を標準形へと変換する手法について述べる。標準形とは同じ内容を持つ質問文を 1 つの標準的な形で表したもので、本研究では、

1. 質問文をタイプごとに分類し
2. 目的語、述語の各重要部分を抽出し
3. タイプを元に語尾表現を変化させる

という手順で質問文の標準化を行なった。

1.1 研究の背景と目的

1.1.1 標準形

通常、自然言語で表現される文（例えば日本語文）には膨大な数の表現方法がある。しかし多くの表現法を持つ日本語文も、ある特定の目的について書かれた文に限定すれば、おおよそ似通った表現をされることが多いだろう。ここでは質問文に限定して考えてみよう。

質問文にも細かい表現の違いを含めれば膨大な数の表現方法があり、仮に複数の人間が同じ事柄について質問したとしても、それぞれの文の表記はかなり違ったものになるだろう。しかし文の表記ではなく、文の内容について考えてみてはどうだろうか。例えば、WWW(World Wide Web) 上の FAQ(Frequentry Asked Question) や電化製品の取扱説明書の Q&A のページなどは特定の分野における代表的な質問を集めたものであり、我々が

普段感じる疑問の多くに答えを与えてくれる。これらの存在は、特定の分野で見られる質問内容の多くはいくつかの代表的な質問に集約することが可能であることを示していると言えよう。

質問文の表現が異なっていたとしても、表現している内容が同じならば、質問文をある 1 つの標準的な形に統一して表現することが可能である。質問文の標準化とは「質問文を標準的な形で表すこと」であり、「同内容の質問文ならばそれは同じ形になる」と言えよう。そして、このときの標準的な形を**標準形**と呼ぶ。

質問文の標準化の例を図 1.1 に示す。どちらの質問文もほぼ同じ内容であるため、これらの質問を 1 つの標準形で表すことができる。

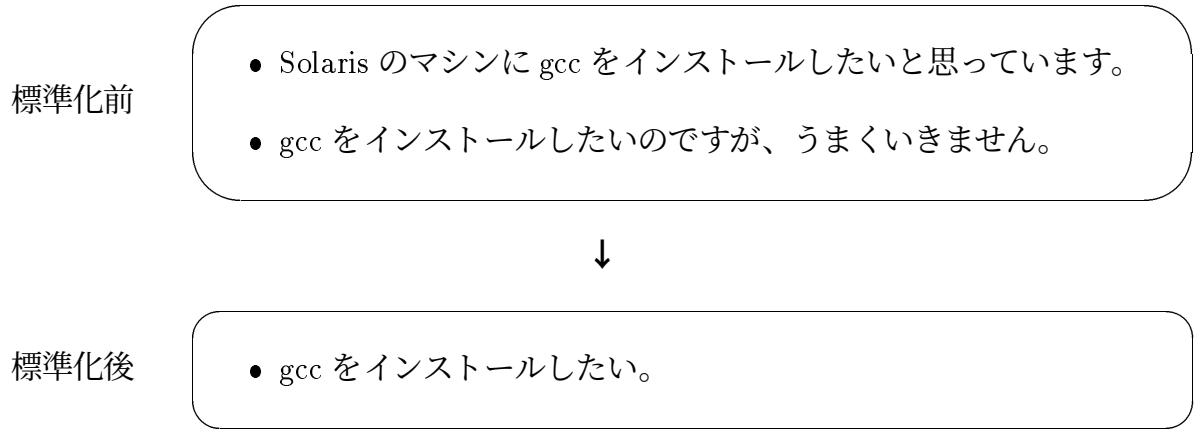

図 1.1: 質問文の標準化の例

1.1.2 質問応答集 Sun QA-Pack

現在、ネットワーク上では FAQ と呼ばれる、代表的な質問とその解答を集めた質問応答集を数多く見ることが出来る。このような FAQ は普通人間の手によってまとめられているのだが、本研究室では自動編集プロジェクトの一環として、この質問応答集をネットニュースグループ fj.sys.sun から自動的に作り出す研究を行なっており [1]、質問応答パッケージ (Sun QA-Pack) と名付けられたその成果が http://www-sato.jaist.ac.jp:8000/faq/ で公開されている。図 1.2 に Sun QA-Pack のスクリーンショットを示す。

図 1.2: Sun QA-Pack

1.1.3 研究の目的

本研究では、ネットニュースグループ fj.sys.sun の質問記事から質問に関する重要部分のみを抽出したサマリー文を標準形へ変換する手法について研究する。

fj.sys.sun は Sun ワークステーションに関する話題を扱うニュースグループであり、Sun ワークステーションに関する質問が数多く投稿され、Sun QA-Pack の対象となっている。fj.sys.sun の質問記事から抽出したサマリー文は現在 Sun QA-Pack のサマリーとして使われており、Sun QA-Pack の使いやすさを向上させている。しかし、現在のサマリーは元のテキストから文単位で重要度を判定し抽出したものであるため、文章の表現は元の記事を書いた人間に依存している。そのため、サマリーは全体として統一の取れた文章であるとは言えず、より使いやすいシステムとするため、テキストの編集が望まれている。標準形への変換が実現すると、サマリーの表示に一貫性が生まれ、より高いレベルでのテキストの編集が達成できる。

1.2　本論文の構成

本論文は、本章を含め 5 章から構成されている。第 2 章では、質問文の特徴について述べ、これを標準形へ変換するためにはどのような方法が考えられるかについて考察する。第 3 章では、本研究で作成した質問文の標準化システムと、その標準形への変換アルゴリズムについて述べる。第 4 章では、標準化システムの評価とその結果を述べる。第 5 章では、結論として本研究のまとめと今後の課題を述べる。

第 2 章

質問文の特徴とその標準化

本研究では、質問文を標準化する手法を求めるために、まず最初に本研究で標準化の対象とする fj.sys.sun の質問文を調査した。本章では質問文の特徴について述べ、その結果に基づいて質問文の標準化へのアプローチを検討する。

2.1 サマリー文のタイプ

本研究の対象となるのは、ネットニュースグループ fj.sys.sun の質問記事から質問に関する重要部分だけを抽出したサマリー文である。fj.sys.sun には Sun ワークステーションに関する質問記事が多数投稿され、その内容はソフトウェアからハードウェアまで多岐にわたる。本研究では、1997 年 4 月から 1997 年 9 月までの半年間に fj.sys.sun に投稿された質問記事から、サマリー文の抽出に成功した 319 件の記事についての調査を行ない、サマリー文を以下の 4 種類のタイプに分類した。

1. 「したい」型
   「〜したい」という自分の要求を述べているサマリー文

2. 「できない」型
   過去に自分が行なった結果、失敗した事柄に関して報告しているサマリー文

3. 「教えて下さい」型
   今後自分が行なう予定の事柄に関して知らないことを質問しているサマリー文

4. 状況説明型
   現在おかれているマイナスの状況について説明しているサマリー文

1. 「したい」型
   Win95 では CD-ROM を挿入するとその情報に応じて CD-ROM の名前が表示されたりするのですが、それと同じような事を（SunOS4.1.3 で）行いたいのです。

2. 「できない」型
   Sun の Ultra2 で OpenWindows 3.5.1 を使用しているのですが、ファンクションキーの F11、F12 が（MIT の X と）違う動作になっていて、それらのキーを押してもアプリケーションで認識してもらえません。

3. 「教えて下さい」型
   C++コンパイラがメソッドの名前を変更 (mangle) する際のアルゴリズムについてどなたかご存知の方いらっしゃいますか?

4. 状況説明型
   Sun IPX に RedHat 4.1 を入れ、Sparc/Linux を楽しんでおりますが、IPX のランチボックスの排気ファンの音がうるさくてしょうがありません。

図 2.1: サマリー文のタイプとその例

図 2.1 に 4 種類のタイプのサマリー文の例を示す。

1997 年 4 月から 1997 年 9 月までに fj.sys.sun に投稿された質問記事から抽出したサマリー文に対して、手作業でサマリー文のタイプを設定した結果を表 2.1 に示す。1 つのサマリー文には 1 つのタイプを設定し、タイプの重複は認めていない。「その他」の項目には、サマリー文として抽出されているものの、内容が不明瞭でサマリーとして機能していないもの、上記のタイプにはあてはまらないものが含まれる。

表 2.1: 各タイプの出現数

タイプ	**出現数**	**出現率**
「したい」型	42	13%
「できない」型	84	26%
「教えて下さい」型	89	28%
状況説明型	73	23%
その他	31	10%
サンプル総数	319	100%

2.2 行為のモデル化

前節で述べたサマリー文のタイプについて、人間が質問をするときの状況と照らし合わせて考えてみよう。

D.A. ノーマンは「行為の 7 段階理論」として行為の構造を近似的にモデル化している [2]。図 2.2 にノーマンのモデルを示す。ノーマンのモデルによると、行為は目的と実行と評価の 3 つの要素に分類することができる。人間が質問をする状況をこのモデルに当てはめると以下のようになると考えられる。

目的 ゴールには、最終的に自分が行ないたいこと、最終目的が当てはまる。この段階では「〜したい」という要求が出され、サマリー文のタイプで言うと「したい」型のサマリー文が該当する。

実行 目的を達成するため実行に移す際に、何をしたらいいのか分からない場合には「どのように実行したらよいのか？」という疑問が起こる。「教えて下さい」型のサマリー文はこの段階での質問となる。

評価 実際に実行しそれを評価した結果、自らの予想通りにはならなかった場合、「うまくいかなかった」という報告がされる。これには「できない」型のサマリー文が該当する。

また、外界の状態の知覚の段階でまだ解釈をしていない場合、状況説明型のサマリー文のように、現在の状況についての報告のみがされる。

以上のように、人間の行為とその時に出される質問のタイプについて関連付けることができるだろう。

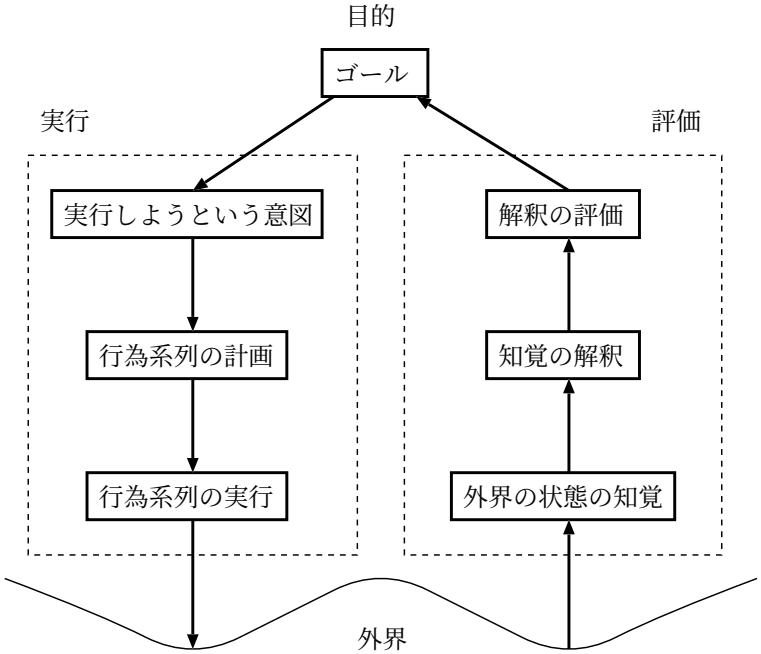

図 2.2: ノーマンの行為の 7 段階理論

2.3 標準形の設定

本研究では前節で説明した 3 つの要素において、それぞれの標準形を設定した。標準形の基本型は次の 2 つの形である。

- （名詞句）を（動詞）したい
- （名詞句）が（動詞）できない

状況説明型のサマリー文には明確な特徴がないため、便宜に応じて上記のどちらかの形で表すこととし、本研究での標準化の対象外とした。

「したい」に言い換え

「したい」型の質問の標準形は以下の形に設定した。

- （名詞句）を（動詞）したい

また、「教えて下さい」型の質問の標準形は「したい」型標準形の動詞部分を「知る」に固定する形で設定した。

- （名詞句）を 知り たい

「できない」に言い換え

「できない」型の質問の標準形は以下の形に設定した。

- （名詞句）が（動詞）できない

第 3 章

標準化システム

本章では、質問文の標準化システムについて詳しく説明する。まずシステムの概要について述べ、次にシステムの各モジュールについてアルゴリズムを交えながら説明する。

3.1 システム概要

標準化システムの構成を図 3.1 に示す。本システムは大きく分けて 3 つのモジュールから構成されている。

- 入力文整形モジュール
  入力は fj.sys.sun に投稿された質問記事から抽出したサマリー文である。サマリー文はそのままでは標準形に変換する際に不都合が生じるため、まず最初にサマリー文を整形する。具体的には、サマリー文に含まれる専門用語タグの削除と句点での文分割を行なう。

- 標準化モジュール
  次に整形されたサマリー文を標準形に変換する。本研究のメインルーチンである。

- 出力文選択モジュール
  最後に標準化された質問文を出力する。この時、1 つのサマリー文に対しては 1 つの標準形を出力することにする。そのため 1 つのサマリー文に対して標準形が複数存在する場合は、文の重要度を文のタイプから判定し、最も重要な標準形を選択する。

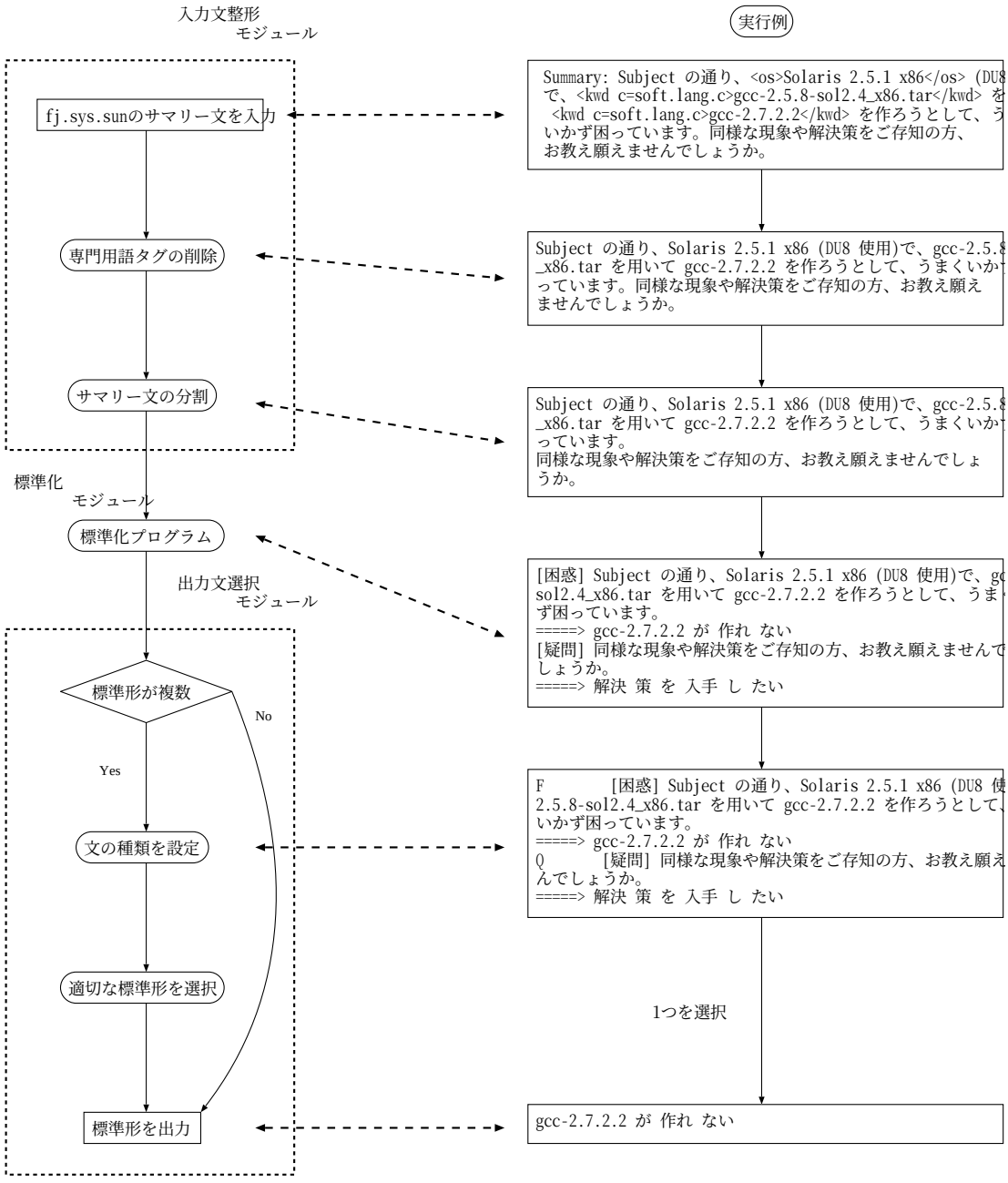

図 3.1: 標準化システムの構成

3.2 入力文整形モジュール

システムへの入力は、ネットニュースグループ fj.sys.sun に投稿された質問記事から、質問に重要だと思われる部分を抽出したサマリー文である。サマリー文はそのままでは標準化の際に不都合が生じるため、前処理としてサマリー文を整形する必要がある。本節では、これら前処理についての説明を行なう。

3.2.1 専門用語タグの削除

本システムの入力は、QA-Pack 自動生成システムによって質問記事から抽出されるサマリー文である。QA-Pack のデータベースから取得したサマリー文の例を図 3.2 に示す。

Summary: ＜ os ＞ Solaris 2.5.1 x86 ＜/os ＞上で動く＜ kwd c=soft.lang.c ＞ gcc-binaries ＜/kwd ＞を探しています。＜ kwd c=soft.lang.c ＞ gcc-binaries-2.5.8 ＜/kwd ＞があると聞いたのですが、archie で検索しても、＜ kwd c=soft.lang.c ＞ gcc-binaries-2.4.5 ＜/kwd ＞しか見つかりません。

図 3.2: サマリー文の例

サマリー文においては、専門用語、OS 名、マシン名の前後に QA-Pack の自動分類に用いられるタグが挿入されている [1]。これらのタグは本研究では用いないので、図 3.3 のアルゴリズムによって削除する。

図 3.2 のサマリー文を図 3.3 のアルゴリズムに従ってタグを削除すると、図 3.4 が出力される。

3.2.2 サマリー文の分割

サマリー文は 1 文である場合が多いが、複数の文から構成されていることもある。ここでは、以降の処理を 1 文単位で行なうために、サマリー文を文単位に分割する。図 3.5 にサマリー文を句点相当文字（「。」「.（半角）」「.（全角）」「?（半角）」「？(全角)」）で文分割するアルゴリズムを示す。

図 3.5 の文分割アルゴリズムでは、句点相当文字を発見して文末なのかどうかを判定し、文末でないならそのままで、文末なら改行してから残りの部分を再び文分割してい

1. 入力：QA-Pack のデータベースから Summary 行のみを取得
2. Summary の抽出に失敗している行を削除
3. 入力文の先頭から＜ x ＞を検索：x は＜＞を含まない任意の文字列
   (a) ＜ x ＞を発見
   (b) ＜ x ＞の後ろに＜/x ＞があれば専門用語タグとして削除
   (c) 行末まで繰り返し
4. 行頭の「Summary: 」を削除
5. 出力：専門用語タグを削除後のサマリー文

図 3.3: 専門用語タグ削除アルゴリズム

Solaris 2.5.1 x86 上で動く gcc-binaries を探しています。gcc-binaries-2.5.8 があると聞いたのですが、archie で検索しても、gcc-binaries-2.4.5 しか見つかりません。

図 3.4: 専門用語タグを削除後

1. 入力：専門用語タグを削除後のサマリー文
2. 入力文の先頭から句点相当文字を検索
3. 句点相当文字を発見
4. 文末であるかどうかを判定
    - if 以下の条件式の論理和が真なら
        - 句点相当文字の直前の文字が英数字
        - 句点相当文字の直後の文字が文頭禁則文字
        - 句点相当文字の直後の文字が同じ句点相当文字
    - then 文末ではない
      改行せずに、句点相当文字より後ろの文を入力として 2. へ
    - else 文末である
      句点相当文字の直後の文字が
        - 改行コードである
          出力へ
        - 改行コードではない
          改行コードを挿入して、句点相当文字より後ろの文を
          入力として 2. へ
5. 出力：文分割されたサマリー文

図 3.5: 文分割アルゴリズム

Solaris 2.5.1 x86 上で動く gcc-binaries を探しています。
gcc-binaries-2.5.8 があると聞いたのですが、archie で検索しても、gcc-binaries-2.4.5 しか見つかりません。

図 3.6: 文分割後

る。文末の判定は、以下の条件式の論理和が真なら文末ではない、偽なら文末であると判定している。

- 句点相当文字の直前の文字が英数字
  これは fj.sys.sun によく見られる「Solaris2.5.2」などの表現を文末と判定してしまわないための条件である。日本語文では最後に英数字で文が終わることはほとんどない。

- 句点相当文字の直後の文字が文頭禁則文字
  これは例えば「Solaris（2.5.2 です。）」のような表現で、句点相当文字の直後の文字が閉じ括弧のような普通文頭にはならない文字であるときは文末と判定しないための条件である。文頭禁則文字としては、”)”,”}”,”]”,”」” のそれぞれ半角と全角文字が設定してある。

- 句点相当文字の直後の文字が同じ句点相当文字
  これは例えば「しまいます...」のような表現で、最後のピリオドのみを文末として扱うための条件である。

図 3.4 の例を図 3.5 のアルゴリズムに従って文分割すると、図 3.6 が出力される。これで標準化プログラムに入力可能な形にサマリー文を整形できた。

3.3 標準化モジュール

3.3.1 標準化モジュールの構成

質問文を標準形に変換する、標準化モジュールの構成を図 3.7 に示す。

標準化モジュールは、整形された状態のサマリー文を入力とし、以下の順序で標準形への変換を行なう。

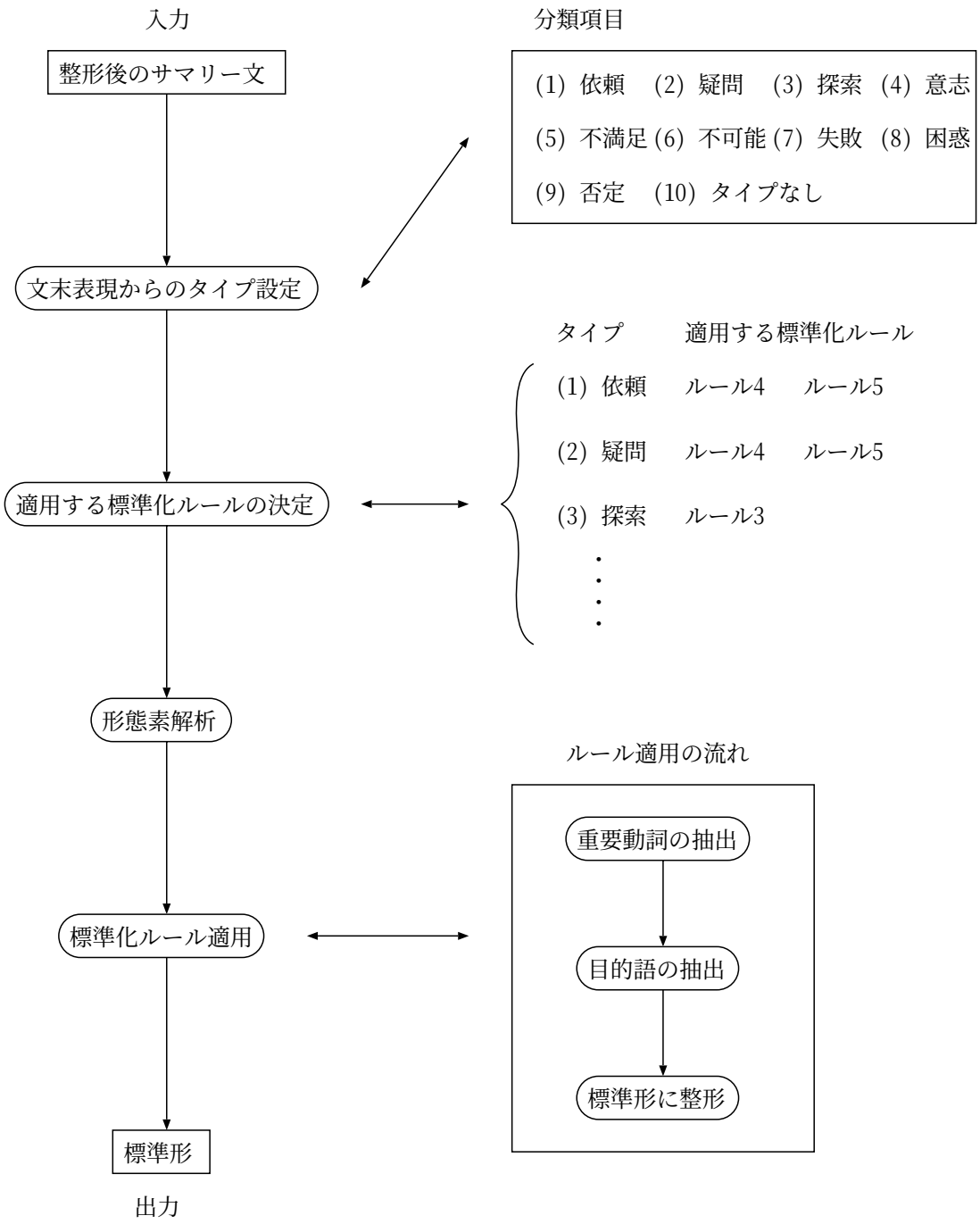

図 3.7: 標準化モジュールの構成

1. 文末表現からのタイプ設定
   サマリー文の文末表現から入力文にタイプを設定する。タイプには図 3.7 に示した 10 種類が存在する。

2. 適用する標準化ルールの決定
   タイプの情報を元に、入力文に適用する標準化ルールの種類を決定する。

3. 形態素解析
   入力文を形態素解析し、標準化ルールの適用に備える。

4. 標準化ルール適用
   標準化ルールに従って入力文を標準形へ変換する。標準化の流れは以下の通り。

   (a) 入力文から最も重要な動詞を抽出

   (b) 動詞に係る目的語を抽出

   (c) 標準形に整形

以下では、これらの詳細について説明する。

3.3.2 文末表現からのタイプ設定

まず最初に入力文の文末表現から、質問文のタイプを設定し分類する。標準化モジュールではこのタイプを元に、どの標準化ルールを適用するかを決定する。

本研究では、図 3.7 に示した 10 種類のタイプに入力文を分類する。文末表現と分類項目の対応を表 3.1 に perl の正規表現の形で示す。入力文を上の分類項目から順にパターンマッチングさせ、文のタイプを決定する。文末表現によるタイプ決定の例を図 3.8 に示す。

3.3.3 適用する標準化ルールの決定

質問文を標準形に変換するための標準化ルールは、全ての質問文に対応させることはできない。そのため文末表現で分類したタイプを元に、その入力文に適用する標準化ルールを選択する必要がある。

表 3.1: 文末表現での分類対応表

・分類 (1) ：依頼

m/((頂|いただ) け|もらえ)(ます|ません|ない)(でしょう|です)?か (\.|. |。|\?|？)?$/

・分類 (2) ：疑問

m/(でしょう|です) か (\.|. |。|\?|？)?$/

・分類 (1) ：依頼

m/(願います|幸いです|(下|くだ) さい|お願い (致|いた)?します) (\.|. |。|\?|？)?$/

・分類 (3) ：探索

m/(探して|捜して|さがして)(い)?ます (\.|. |。|\?|？)?$/

・分類 (4) ：意志

m/と (して|思|(思|おも)って|(考|かんが) えて)(い)?ます (\.|. |。|\?|？)?$/

・分類 (5) ：不満足

m/しまい (ます|ました)(\.|. |。|\?|？)?$/

・分類 (6) ：不可能

m/(出来|でき) ません (でした)?(\.|. |。|\?|？)?$/

・分類 (7) ：失敗

m/(失敗し|エラーにな (り|っ)|エラーが (出|で))(てしまい)?(ます|ました)(\.|. |。|\?|？)?$/

・分類 (8) ：困惑

m/((困|こま)って|はまって|(悩|なや) んで)(い|しまい)?(ます|ました)(\.|. |。|\?|？)?$/

・分類 (9) ：否定

m/ません (でした)?(\.|. |。|\?|？)?$/

・分類 (10)：タイプなし

その他

(1) [依頼] SUN solaris(2.5.1) で使用可能な CDplayer ソフトとその入手先を教えて**ください**。

(2) [疑問] C の入ってない SUN を使っているのですが gcc をインストールする方法はあるの**でしょうか**？

(3) [探索] Solaris 2.5.1 x86 上で動く gcc-binaries を**探しています**。

(4) [意志] SPARCstation5 に Solaris2.5 を載せて使おう**としています**。

(5) [不満足] キャンペーン中ということで Solaris for X86 を購入しましたのですが、インストールしようとすると途中でまったく反応しなくなって**しまいます**。

(6) [不可能] Solaris 2.5.1 (x86) に workman401 をインストールしましたが、ATAPI の CD-ROM drive で音楽 CD を再生**できません**。

(7) [失敗] solaris2.5.1(sparc) で gcc のバイナリーを入れたあと bision をインストールしようとすると　(–中略–)　という**エラーがでます**。

(8) [困惑] Subject の通り、 Solaris 2.5.1 x86 (DU8 使用) で、gcc-2.5.8-sol2.4_x86.tar を用いて gcc-2.7.2.2 を作ろうとして、うまくいかず**困っています**。

(9) [否定] Ｓファミリーのディスプレイなしのモデル５台に Solaris2.5.1 のインストールを予定しているのですが、どのような形態でインストールすればいいのかわかり**ません**。

(10) [] Solaris2.5.1(x86) をダイアルインサーバにしようと思い、dp-4.0 をインストールしましたが、dp という名のカーネルモジュールが以下のようなエラーを 30 秒に一度出力してきます。

図 3.8: 文末表現によるタイプ決定の例

標準化ルール

標準形の基本型は以下の 2 種類である。

- （名詞句）を（動詞）したい
- （名詞句）が（動詞）できない

本研究では、質問文を標準形に変換する際に用いる規則を、標準化ルールとして「したい」に言い換えるルールと「できない」に言い換えるルールの 2 種類に分類してまとめた。表 3.2 に、本研究でまとめた標準化ルールの一覧を示す。矢印の左側にある表現が質問文中に含まれていたら、動詞と目的語を抽出し、矢印の右側の標準形に変換する。

表 3.2 で「否定の表現」とは、「ません」や「ない」などの否定的な表現や、「ているのですが」のように、次に否定的な表現が現れると考えられる表現のことを指す。

適用する標準化ルールの決定

標準化ルールは一般的な形で書かれているため、全ての入力に対して全てのルールを適用することはできない。仮に全てのルールを適用するとしたら、例えば「〜をインストールできないのですが、使いたいと思います」のような入力があった場合、ルール 1 とルール 8 のどちらを適用するべきなのか判断できない。そのため、本システムでは文末表現から分類したタイプを元に入力文に適用する標準化ルールの種類を決定する。入力文に適用する標準化ルールは以下のように決定される。

- 分類 = 意志
  → 標準化：ルール 1 , ルール 2
- 分類 = 探索
  → 標準化：ルール 3
- 分類 = 疑問 or 依頼
  → 標準化：ルール 4 , ルール 5
- 分類 = 不満足 or 不可能 or 失敗 or 困惑 or 否定
  → 標準化：ルール 6 , ルール 7 , ルール 8
- 分類 = タイプなし
  → 標準化：ルール 1 , ルール 2 , ルール 5 , ルール 6 , ルール 7 , ルール 8

表 3.2: 標準化ルール一覧

ルールの書式

[質問文中に含まれる表現] → [標準形]

「したい」に言い換え

ルール 1

～ **動詞** + たい ～ → **目的語** を **動詞** + たい

ルール 2

～ **動詞** + ようと ～ → **目的語** を **動詞** + たい

ルール 3

～ 探しています → **目的語** を 入手したい

ルール 4

～ **動詞** + 方法 ～ → **目的語** を **動詞** + たい

ルール 5

～ 教えて ～ → **目的語** を 知りたい

「できない」に言い換え

ルール 6

～ **動詞** + たい ～ **否定の表現** → **目的語** が **動詞** + ない

ルール 7

～ **動詞** + ようと ～ **否定の表現** → **目的語** が **動詞** + ない

ルール 8

～ **動詞** + ない ～ → **目的語** が **動詞** + ない

標準化ルールは番号の小さいものから順に適用される。例えば「〜したいのですが、〜できません」のような例には、「ですが」という否定の表現があるため、ルール 6 とルール 8 の両方が適用できるが、この場合は番号の小さいルール 6 が適用される。図 3.8 の例のサマリー文に、適用する標準化ルールを選択した例を図 3.9 に示す。

3.3.4 形態素解析

標準化を実行する際に、日本語形態素解析システム Juman を用いて形態素解析を行なう。Juman では空白が間にあると正確に形態素解析できないため、形態素解析をする前に入力文のスペースとタブをアンダーバーに変換しておく。なお、このアンダーバーは最後に標準形を出力する際に元の空白に戻している。

Juman で形態素解析をした結果から「表層形」「読み」「標準形」「品詞」「品詞細分類」「活用型」「活用形」の 7 つの情報を配列に格納し、標準化の際に使用する [3][4]。例として「探しています」という文字列を形態素解析した結果を図 3.10 に示す。

3.3.5 重要動詞の抽出

入力文に選択された標準化ルールを適用する。標準化ルールを適用は、入力文中から (1) まず重要な動詞を発見し、(2) 次にその目的語を抽出、(3) 最後に標準形に整形する、という手順で行なう。

重要動詞の発見

サマリー文中からの重要動詞の発見は、適用されている標準化ルールに従って行なう。サマリー文の文頭から、発見した重要動詞の部分までを、次の目的語抽出ルーチンに渡す。

ルール 1 : 〜 **動詞** + たい 〜 → **目的語** を **動詞** + たい

ルール 1 を適用する場合、文末に重要動詞の位置を示す表現が存在する場合は、文末から「**動詞** + たい」の表現が存在するかを検索し、重要動詞の位置を示す表現がない場合は、文頭から「**動詞** + たい」の表現を検索して重要動詞を抽出する。なお、重要動詞の位置を示す表現とは、文末のタイプ判定の際に用いた次の表現のことである。

```
m/と(して|思|(思|おも)って|(考|かんが)えて)(い)?ます/
```

ルール 2 : 〜 **動詞** + ようと 〜 → **目的語** を **動詞** + たい

ルール 5 を適用

(1) [依頼] SUN solaris(2.5.1) で使用可能な CDplayer ソフトとその入手先を**教えて（教えて）**ください。

ルール 4 を適用

(2) [疑問] C の入ってない SUN を使っているのですが gcc を**インストールする（動詞）方法（方法）**はあるのでしょうか？

ルール 7 を適用

(5) [不満足] キャンペーン中ということで Solaris for X86 を購入しましたのですが、**インストールし（動詞）ようと（ようと）**すると途中でまったく反応しなく**なってしまいます（否定の表現）**。

ルール 8 を適用

(6) [不可能] Solaris 2.5.1 (x86) に workman401 をインストールしましたが、ATAPI の CD-ROM drive で音楽 CD を**再生でき（動詞）ません（ない）**。

ルール 2 を適用

(10) [] Solaris2.5.1(x86) をダイアルインサーバに**し（動詞）ようと（ようと）**思い、dp-4.0 をインストールしましたが、dp という名のカーネルモジュールが以下のようなエラーを 30 秒に一度出力してきます。

図 3.9: 適用する標準化ルールの決定例

入力：探しています

配列	表層形	読み	標準形	品詞	品詞細分類	活用型	活用形
$juman[0]	探して	さがして	探す	動詞	*	子音動詞サ行	タ形連用テ形
$juman[1]	い	い	いる	接尾辞	動詞性接尾辞	母音動詞	基本連用形
$juman[2]	ます	ます	ます	接尾辞	動詞性接尾辞	動詞性接尾辞ます型	基本形

図 3.10: 形態素解析結果例

ルール 2 を適用する場合も、ルール 1 の場合と同様に重要動詞の位置を示す表現が存在する場合は文末から、存在しない場合は文頭から「**動詞** + （よ or お or こ or ご or そ or と or の or ぼ or も or ろ）うと」の表現が存在するかを検索し、重要動詞を抽出する。重要動詞の位置を示す表現はルール 1 の場合と同様である。

ルール 3 : ～ 探しています → **目的語** を 入手したい

ルール 3 は「分類 (3):探索」に分類されたサマリー文に対してのみ適用されるので、文末表現に発見されている「探す」を「入手する」に置換して重要動詞とする。

ルール 4 : ～ **動詞** + 方法 ～ → **目的語** を **動詞** + たい

ルール 4 を適用する場合、サマリー文中から「方法」という文字列を検索し、その直前の語を調べて重要動詞を決定する。

- 直前の語が「動詞」ならばそれを重要動詞とする。
- 直前の語が「名詞 + (ため) + の」または「名詞」ならば、「名詞」に「する」を追加して「名詞 + する」を重要動詞とする。

ルール 5 : ～ 教えて ～ → **目的語** を 知りたい

ルール 5 を適用する場合、サマリー文中に以下の表現があるかを検索し、あれば「知りたい」に置換して重要動詞とする。

```
m/(教えて|お教え|(御|ご) 教示|(御|ご) 教授|お聞かせ|お聞きしたい|伺
いたい|(御|ご)(存知|存じ))/
```

ルール 6 : ～ **動詞** + たい ～ **否定の表現** → **目的語** が **動詞** + ない

ルール 6 の適用は、まず否定の表現を探すことから始める。否定の表現は、「分類:(5) 不満足,(6) 不可能,(7) 失敗,(8) 困惑,(9) 否定」に分類されたサマリー文に対して適用する際には文末表現にすでに現れている。「分類:(10) タイプなし」に適用する際には、以下の否定の表現のパターンがあるか調べる。

```
m/(((の|ん) です|ます|ました)(が|けど|けれど))/
```

否定の表現が発見できたならば、否定の表現より前に「**動詞** + たい」の表現が存在するかを検索して、重要動詞を抽出する。

ルール 7 : 〜 **動詞** + ようと 〜 **否定の表現** → **目的語** が **動詞** + ない

ルール 7 を適用する場合も、ルール 6 の場合と同様にまず否定の表現を探し、否定の表現より前に「**動詞** + （よ or お or こ or ご or そ or と or の or ぼ or も or ろ）うと」の表現が存在するかを検索する。

ルール 8 : 〜 **動詞** + ない 〜 → **目的語** が **動詞** + ない

ルール 8 を適用する場合、文頭から「**動詞** + ない」の表現を検索して、重要動詞を抽出する。ただし、動詞が以下の表現の場合は「できない」に置換して重要動詞とする。

```
m/((出来|でき)|うまくい(き|か))(ない|なく|ません)/
```

サ変名詞の扱い

抽出した動詞の標準形が「する」だった場合、「〜をしたい」の文章構造ならば上記アルゴリズムをそのまま用いるが、「〜をインストールしたい」の「インストール」のように、「する」の直前の語の品詞が未定義語、または品詞細分類がサ変名詞、もしくはカタカナであった場合は、「する」の直前の語も含めて動詞扱いとする。

- サ変名詞
  「名詞」+「する」で動詞的な働きをする名詞がサ変名詞である。そのため、「する」の直前の名詞も含めて動詞として扱う。

- 未定義語
  juman では fj.sys.sun によく見られる「ftp」などのアルファベットは品詞：未定義語として出力される。これらの語はサ変名詞と同様の働きをするため、動詞の一部として扱う。

- *カタカナ*
  fj.sys.sun によく見られる「インストール」などのカタカナ語もサ変名詞と同様の働きをするので、動詞の一部として扱う。

図 3.9 の例から重要動詞を抽出し、目的語抽出ルーチンへ出力する例を図 3.11 に示す。

3.3.6 目的語の抽出

次に、発見した動詞に係る目的語を抽出する。図 3.12 に目的語抽出アルゴリズムを示す。

目的語が係る動詞部分の抽出

目的語抽出アルゴリズムへの入力は元のサマリー文の重要であると思われる部分のみを抽出した文なので、アルゴリズムの 2 では単純に入力文の一番後ろの動詞に注目している。入力文の文末は動詞+接尾辞となっているため、文末より接尾辞をチェックして、最初に現れた接尾辞ではない語が動詞かどうかを判定している。

「を」の抽出

目的語の抽出は、まず動詞の直前の 10 語（形態素）以内に「を」があるかどうかを判定することから始まる。「を」の発見の際に、動詞の直前の 10 語以内から探すというのはヒューリスティックに決めている。この数が多すぎると全く関係ない別の動詞の目的語を取ってきてしまう場合があるし、逆に少なすぎると必要なはずの目的語を取れない可能性が高い。

10 語以内という曖昧な制限を外す方法としては、別の動詞を発見した時点で「を」の検索を止めるという方法が考えられるが、例えば「HDD を取り付けて使用したい」というような入力があった場合、「を」を発見する前に別の動詞を発見してしまい目的語の抽出に失敗してしまうため、現在のアルゴリズムになっている。

ただし「を」の前に「に」を発見した場合、ほとんど「〜を〜に」の表現がされており、「に」の前に「を」が見つかる可能性が高い。また、この場合は「〜に」の修飾部分が長いことが多いため、この場合に限り 10 語以内の制限を外し「を」が見つかるまで検索するようにしている。図 3.13 の例も「〜を〜に」の形になっており「〜に」の表現が長いため、動詞から 10 語以内には「を」を見つけることが出来ない。

ルール 5 を適用

(1) [依頼] SUN solaris(2.5.1) で使用可能な CDplayer ソフトとその入手先を**教えて（教えて）**ください。
→ SUN solaris(2.5.1) で使用可能な CDplayer ソフトとその入手先を知りたい

ルール 4 を適用

(2) [疑問] C の入ってない SUN を使っているのですが gcc を**インストールする方法（方法）**はあるのでしょうか？
→ C の入ってない SUN を使っているのですが gcc をインストールする

ルール 7 を適用

(5) [不満足] キャンペーン中ということで Solaris for X86 を購入しましたのですが、**インストールし（動詞）ようと（ようと）**すると途中でまったく反応しなく**なってしまいます（否定の表現）**。
→ キャンペーン中ということで Solaris for X86 を購入しましたのですが、インストールしよう

ルール 8 を適用

(6) [不可能] Solaris 2.5.1 (x86) に workman401 をインストールしましたが、ATAPI の CD-ROM drive で音楽 CD を**再生でき（動詞）ません（ない）**。
→ Solaris 2.5.1 (x86) に workman401 をインストールしましたが、ATAPI の CD-ROM drive で音楽 CD を再生できない

ルール 2 を適用

(10) [] Solaris2.5.1(x86) をダイアルインサーバに**し（動詞）ようと（ようと）**思い、dp-4.0 をインストールしましたが、dp という名のカーネルモジュールが以下のようなエラーを 30 秒に一度出力してきます。
→ Solaris2.5.1(x86) をダイアルインサーバにしよう

図 3.11: 重要動詞の抽出例

1. 入力：形態素解析後のサマリー文から重要な動詞部分までを抽出した文
2. 入力文を文末より後向きに検索し、最初に現れる動詞に注目
3. 動詞の直前の 10 語以内に「を」があるかどうかを検索
   (a) 「を」の前に「に」が見つかった場合
   → 10 語以内という制限を外す
4. 「を」が発見できない場合、同様に「について」を検索
5. 「を」or「について」に係る名詞句を抽出
   (a) 「を」or「について」より前にある語を後ろから順に選択
   - 「と（格助詞）」 or 「の（接続助詞）」は目的語である
   - 品詞細分類が名詞性名詞接尾辞ならば目的語である
   - 「方法」の直前に存在する「する（動詞）」は目的語である
   - サ変名詞の直前に存在する「を（格助詞）」は目的語である
   - 格助詞 or 接続助詞 or 読点 or 動詞 or 助動詞 or 接尾辞 or 副詞 or 判定詞ならば目的語ではない
     → 出力へ
   - 上記以外の語は目的語である

   (b) 次の語へ
6. 出力：目的語 + 動詞

図 3.12: 目的語抽出アルゴリズム

入力　dhcp server を、隣に転がっている Solaris2.5 のマシンに
インストールしようと思っています

目的語　dhcp server を

動詞　インストール する

図 3.13: 「に」の表現が長い例

「について」の抽出

動詞の中には、目的語として「を」以外に「について」をとる語がある。例えば「調べる」「知る」「聞く」などがこれに相当する。そのため「を」が発見できなかった場合は、「を」の時と同様に動詞の 10 語以内に「について」があるか検索する。「について」を発見した場合、以降の処理では「を」と全く同様の扱いをする。

なお、「を」も「について」も見つからなければ、目的語は存在しないとして終了する。

「を」(「について」) に係る名詞句の抽出

次に「を」(「について」) に係る名詞句を抽出する。手順としては「を」(「について」) より前に位置する語を後ろから順番に検索し、以下の条件により目的語であるか、そうではないかを判定している。

- 格助詞「と」は目的語である
  「x と y を」の表現の場合、「と」の前後の x と y は等価な関係の名詞句であるため、格助詞「と」は目的語として残す。

- 接続助詞「の」は目的語である
  「x の y を」の表現の場合、x は y を修飾する語であるため、接続助詞「の」は目的語として残す。

- 品詞細分類が名詞性名詞接尾辞ならば目的語である
  名詞性名詞接尾辞には「用」「製」「社」などが当てはまり、これらは「名詞」+「名詞性名詞接尾辞」で名詞と同様の働きをするため、他の接尾辞は目的語にはならないのだが、名詞性名詞接尾辞だけは目的語として残す。

- 「方法」の直前に出現する動詞「する」は目的語である
  「x する方法を」の表現の場合、「する」 は目的語として残す。この時、x はサ変名詞である。

- 品詞細分類がサ変名詞の語の直前に出現する「を」は目的語である
  「y を x する方法を」の表現の場合、サ変名詞 x の直前に出現する「を」は目的語として残す。

- 品詞細分類が格助詞 or 接続助詞 or 読点または、
  品詞が動詞 or 助動詞 or 接尾辞 or 副詞 or 判定詞ならば目的語ではない
  これらに関しては、対象としている動詞の目的語ではないと判断し、それまでに残した部分+「を」(「について」) を目的語として出力する。

- 上記以外の語は目的語である
  具体的には、形容詞、名詞、指示詞などが相当する。これらの語は目的語として残す。

図 3.14 に図 3.12 のアルゴリズムに従って目的語を抽出した例を示す。

特別な動詞に係る目的語

目的語の抽出は以上に述べた通りだが、目的語が係る動詞がいくつかの特別な動詞である場合は、図 3.12 のアルゴリズムと少し違った処理を行なう。

- 動詞の標準形が「できる」の場合
  → 目的語として、「〜を」の場合と同様に「〜が」を抽出する。

- 動詞の標準形が「する」で、サ変名詞を取らない場合
  → 目的語として、「〜を」だけでなく「〜に」も抽出する。

図 3.15 に例を示す。

3.3.7 標準形に整形

これまでに抽出した動詞と目的語を、標準化ルールに従って標準形に整形する。

[1] 目的語「を」の抽出例

入力　Solaris2.5.1 に sendmail をインストールしたい

目的語　sendmail を

[2] 目的語「について」の抽出例

入力　ftp の設定について調べたい

目的語　ftp の 設定 に ついて

[3] 格助詞「と」を含む例

入力　Solaris 2.5.1 上で kterm と atok8 を使いたい

目的語　kterm と atok8 を

[4] 接続助詞「の」を含む例

入力　ATAPI の CD-ROM drive で普通の音楽 CD を再生したい

目的語　普通 の 音楽 CD を

[5] 名詞性名詞接尾辞を含む例

入力　Ultra2 Enterprise Server に、市販の PC 用プリンターを接続したい

目的語　市販 の PC 用 プリンター を

[6] 「y を x する方法」の例

入力　gcc をインストールする方法を知りたい

目的語　gcc をインストールする方法を

図 3.14: 目的語の抽出例

[1] 動詞の標準形が「できる」の場合

入力　gcc-2.7.2.2 がインストールできません

目的語　gcc-2.7.2.2 が

[2] 動詞の標準形が「する」で、サ変名詞を取らない場合

入力　Solaris2.5.1(x86) をダイアルインサーバーにしたい

目的語　Solaris2.5.1(x86) を ダイアルインサーバー に

図 3.15: 特別な動詞に係る目的語の抽出例

「したい」に言い換えるルール

「したい」に言い換えるルールの場合、抽出した動詞を「たい」に繋ぐために、活用形を基本連用形に変換する。標準形への整形は以下の通り。

- 目的語 + 「を」 or 「について」 + 動詞（基本連用形） + たい

「できない」に言い換えるルール

「できない」に言い換えるルールの場合、抽出した動詞を「ない」に繋ぐために、動詞が「できる」の場合は活用形を未然形「でき」に、それ以外の動詞については命令形に変換する。

また、目的語の最後の「を」を標準化ルールに従って、「が」に置換する。標準形への整形は以下の通り。

- 目的語 + 「が」 + でき（未然形） or 動詞（命令形） + ない

図 3.16 にサマリー文を分類し、適用する標準化ルールを決定して動詞と目的語を抽出し、標準形へと整形して出力した例を示す。

標準化に失敗した場合

タイプは文末表現のみから決定しているため、常に正確であるとは限らない。そのため、設定したタイプで適用される全ての標準化ルールで標準形への変換に失敗した場合、

[1] 「したい」に言い換える例

入力　今まで使っていたSS1+の部品を流用しようと思っています。

分類　[意志]

ルール　ルール2

動詞　流用 する

目的語　SS1+ の 部品 を

標準形　SS1+ の 部品 を 流用 し たい

[2] 「できない」に言い換える例

入力　ジャンクでゲットしたSS2にSolaris1.1.2をインストールしようとしているのですが、うまくいきません。

分類　[否定]

ルール　ルール7

動詞　インストール する

目的語　Solaris1.1.2 を

標準形　Solaris1.1.2 が インストール でき ない

図 3.16: サマリー文の標準化例

設定したタイプが間違っていた可能性も考慮する必要がある。

本研究では、1 度目の標準化で失敗した場合、設定したタイプが間違っていた可能性を考慮し、タイプが「タイプなし」の場合と同様に標準化ルール 1,2,5,6,7,8 を再度適用して標準化成功率の上昇を図っている。

3.4 出力文選択モジュール

本システムでは、1 つのサマリー文につき 1 つの標準形を出力することとした。これまでに説明した標準化プログラムは、1 入力に対して 1 出力であるため、元のサマリー文が 1 つの文から構成されている場合は 1 つの標準形しか出力されず、特に問題はない。しかし、サマリー文が複数の文から構成されていた場合は、図 3.5 に示した文分割アルゴリズムによって複数の文に分けられるため、1 つのサマリー文に対して複数の標準形が出力される場合がある。

本研究では 1 つのサマリー文に対して複数の標準形が出力された場合、サマリー文を抽出する際に用いられる、質問記事中における文の役割を示したタイプから文分割後のサマリー文の重要度を判定し、最も適切な標準形を選択した。

3.4.1 文の種類を設定

質問記事からサマリー文を抽出する際、質問記事の各文に対して質問記事中に占める役割を示すタイプを設定し、このタイプを元に各文の重要度を判定する。その後、重要度の高い文を質問記事からサマリー文として抽出する。

本研究では、このタイプの中から「G(Goal)」「F(Fail)」「Q(Question)」の 3 つのタイプを流用し、本研究で文分割した後のサマリー文に対して文の種類を設定し、適切な標準形の選択に用いる。

Goal 自分の目的について述べている文
単文なら「したい」型のサマリー文に相当する

Fail 失敗した事柄について説明している文
単文なら「できない」型のサマリー文に相当する

Question 質問している文
単文なら「教えて下さい」型のサマリー文に相当する

表 3.3 にタイプと文の表現の対応を Perl の正規表現の形で示す。表 3.3 に示す表現をサマリー文とマッチングさせ、文の種類を設定する。

3.4.2 適切な標準形の選択

1 つのサマリーに対して標準形が複数出現した場合は、設定した文の種類から各文の重要度を判定して標準形を 1 つ選択する。1997 年 4 月から 9 月までに fj.sys.sun に投稿された質問記事から抽出されたサマリー文 319 件を調査した結果、複数の文からサマリー文が構成されていたのは 66 件あった。これらのサマリー文について、各文ごとに GFQ のタイプ判定を行ない調査した結果、以下のような特徴が見られた。

- G と F のタイプの組み合わせでは、より詳細度の高い内容が後の文で書かれている。
- G と Q のタイプの組み合わせでは、G が主題であり、Q は「どなたか教えて下さい」のように重要度が低い文章であることが多い。
- F と Q のタイプの組み合わせでも、F が主題であり、Q は重要度が低い場合が多い。

よって本研究では、1 つのサマリーに対して標準形が複数出力された場合、

1. 標準化の際に目的語を抽出できなかった標準形は除外
2. Q の文から出力された標準形よりも、G と F の文から出力された標準形を優先
3. G と F の標準形は、後の文から出力された標準形を優先

という手順で、適切な標準形の選択を行なった。

図 3.17 に文の種類を設定し、適切な標準形を選択した例を示す。

表 3.3: GFQ のタイプと文の表現の対応表

Goal

m/((行|おこな) おうと (思|おも|) (う|います|) |(行|おこな) いたい|しようと|(気持|きも) ちにな|する予定 (です|だ) |(検討|けんとう) して (いる|います) |(考|かんが) えて (いる|います|ます) |(させ|し|やり|やってみ|(知|し) り|(手|て) に (入|い) れ) た (い|くて) |(試|こころ) み (て|たい|よう|た))/

Fail

m/(((分|わ) か|判|解) (りません|らな (い|かった)) |うまく (い|行) (きません|かない) |表示されません|しろと (言|い) われます|(出来|でき) (な|ま) (い|く|せん) |してくれません|しない状態になって|(止ま|立ち上が) らな|なってしま (う|った|いました) |できない|されてしま (う|った|いま (す|した)) |おかし (い|く) |使え (ません|なく) |問題が生じて|(困|こま) っ (た|て) |(動|うご) きません|(機能|動作) し (て|) (おりません|いない|ない|ません) |(悩|なや) んでいます|を (起|お) こす|(落|お) ち (てしまい|) ます|(し|) てしま (っ(た|て) |いま (した|す)) |してもダメ|つまづいています|開かない|(エラー|(E|e)rror|ERROR|エラーメッセージ) (が|に) (で|出|な|なり) (る|た|ます|ました|る|った) |遅くなっている)/

Question

m/(でしょうか|ですか|いらしゃいますか|(教|おし) え (て|) (ください|ほしい|いただければ|) |ですよね|探しています|ありますか|ご教示下さい|ますか|あるのか|(御|ご) 教授 (いただければ|(下|くだ) さい) |お聞かせ下さい|お聞きしたい)/

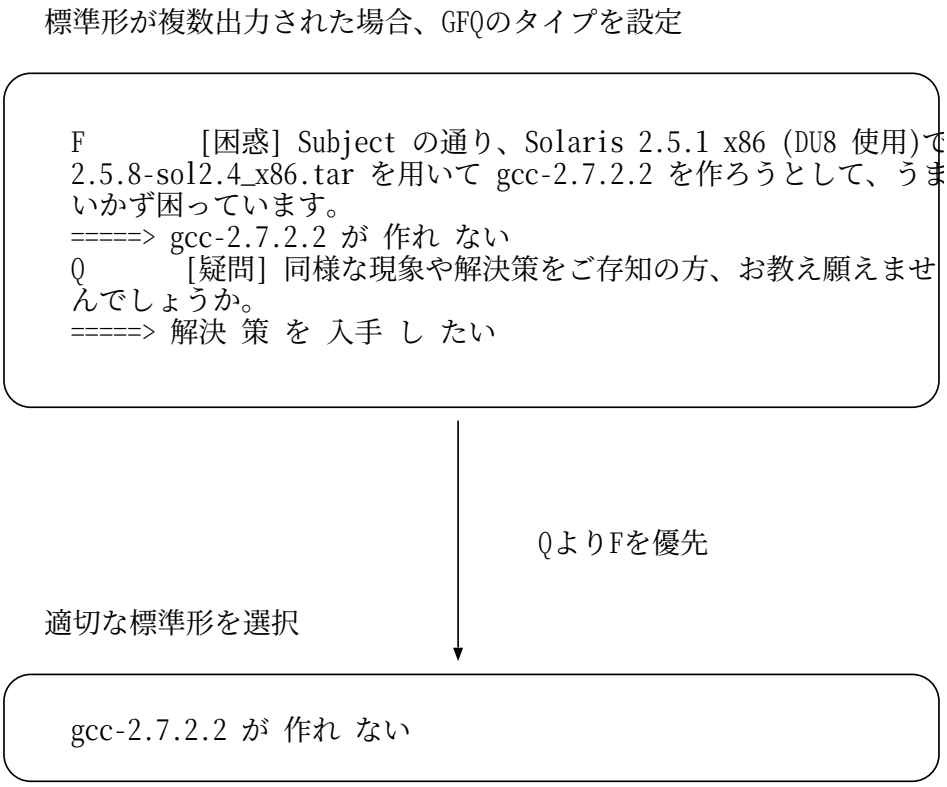

図 3.17: 適切な標準形の選択

第 4 章

実験と評価

前章までに、本研究で作成した質問文の標準化システムについて説明した。本章では作成したシステムについて実験を行なった結果を述べ、その結果について考察する

4.1 実験

本研究で作成した標準化システムについて実験を行なった。実験の対象としたデータは、ネットニュースグループ fj.sys.sun に 1997 年 4 月から 9 月までの半年間に投稿された質問記事の中で、サマリーの抽出に成功している 319 件のサマリー文である。

2 章で述べたように、本研究ではサマリー文を「したい」「できない」「教えて下さい」「状況説明」の 4 種類に分類した。このうち、状況説明型のサマリー文には明確な特徴が存在せず、本研究での標準化の対象外とした。

そこで実験の対象としたサマリー 319 件から、状況説明型のサマリー 73 件と、内容が不明瞭でサマリーとして機能していないその他に分類したサマリー 31 件の、計 104 件（2 章の表 2.1 参照）を除いた、215 件のサマリー文を対象に標準化モジュールの実験を行なった。

4.1.1 入力文整形モジュール

入力文整形モジュールでは、(1) 専門用語タグの削除 (2) サマリー文の分割 を行ない、入力文を整形している。これらの処理は比較的パターンが明確で、失敗する要因は少ない。

実験を行なった結果、専門用語タグは 100%削除することに成功した。また、複数の文から構成されているサマリーは 215 件中 49 件存在したが、これらも全て正しく文分割することに成功した。入力文整形モジュール全体では、正解率 100%である。

表 4.1: 標準化モジュールの出力

タイプ	標準形に変換			標準形に変換できず
	正解	不正解	計	
したい型	38 文 (78%)	11 文 (22%)	49 文 (100%)	0 文
できない型	50 文 (71%)	20 文 (29%)	70 文 (100%)	31 文
教えて下さい型	36 文 (62%)	22 文 (38%)	58 文 (100%)	56 文
全体	124 文 (70%)	53 文 (30%)	177 文 (100%)	87 文

4.1.2 標準化モジュール

実験データのサマリー 215 件の中には、2 文で構成されているサマリーが 49 件含まれている。これらを文分割後、標準化モジュールに入力するため、標準化モジュールに入力される質問文の数は 264 文（215+49）となる。

標準化モジュールを用いて、入力された 264 文を標準形へ変換した結果の評価を表 4.1 に示す。

264 文中、正しく標準形に変換できた文が 124 文、標準形に変換はしたものの文の主旨と違っていたり、日本語として誤っている文が 53 文、標準形に変換できなかった文が 87 文であった。標準化モジュールのカバレジは $177/264 = 67\%$ であり、標準化モジュールで標準形に変換した場合の正解率は $124/177 = 70\%$ となる。これは、元のニュース記事のテキストのクオリティが低いことを考慮すると、良い結果が出たと言える。

タイプ別に見てみると、それぞれのタイプで結果が異なることがわかる。最も正解率が高いのは「したい」型 (78%) である。「したい」型は変換規則が必ず適用されたため、変換できなかったものは 1 つもなかった。一方、「したい」型以外のタイプに関しては、正解率は「できない」型が 71%、「教えて下さい」型が 62% と良い。しかし、標準形に変換できなかったものがそれぞれ 31 文、56 文存在し、標準化モジュールのカバレジを低下させている。これらのタイプに対しては、より多くの標準化ルールを用意することで、標準化モジュールのカバレジを向上させることができると考えられる。

4.1.3 出力文選択モジュール

サマリーが 2 文以上から構成されているものが 49 件、その中で、2 つ以上の標準形が出力されている例は 16 件、標準形が 1 つしか出力されていない例が 25 件、標準形が 1 つ

(1) [] SPARK-LT に使うマウスは、SUN4 用でも使えますか？

(2) [疑問] Solaris for x86 の hostid はどのような管理をされているのでしょうか？

(3) [] しかし Solaris の DisplayPostscript を使うようにして configure すると（–with-ps=dps をつける）コンパイル時にエラーが出てしまいます（ちなみに DPS なしでやるとうまくいきます）.

図 4.1: 標準形に変換できなかった例

も出力されなかった例が 8 件あった。よって、出力文の選択が必要となったのは 16 件である。

対象となる 16 件を出力文選択モジュールに入力した結果、より重要であると思われる標準形を選択した例が 14 件、選択したものより別の標準形の方が重要だと思われる例が 2 件であった。これより、出力文選択モジュールの正解率は 88%である。

4.2 考察

ここでは、標準化モジュールが正しい結果を出力できなかった場合について考察する。

標準形に変換できなかった

まず、表 4.1 で標準形に変換できなかった 87 文について、その原因を考察してみる。

標準形への変換に失敗したのは、標準化モジュールにおいて重要動詞の抽出に失敗したためである。標準形への変換に失敗した例を図 4.1 に示す。

(1) の例は「教えて下さい」型に分類したものだが、「使えますか」という確認のための表現となっているため、本研究でまとめた標準化ルール「**動詞** + たい」「**動詞** + よう」「**動詞** + ない」では抽出することができなかった。(2) の例も同様で、「教えて下さい」型のサマリー文の変換率の悪い理由となっている。(3) の例は「できない」型に分類したが、やはり標準化ルールでは抽出できなかった。

この問題を解決し、標準化モジュールのカバレジを向上させるためには、標準化ルールのより一層の充実が必要である。

(1) [依頼] SparcStation でも Linux を CD から Install されてお使いの方が増えていますが、SunOS4.X や Solaris2.X ではメモリはパリティありでなければ boot のときにデッドロックして帰ってこなくなりますが、Linux では boot 出来るという噂を聞いたことがありますが現役でお使いの方、fj.sys.sun でお教えいただけませんでしょうか？
=====> 知り たい

(2) [不満足] Solaris2.5.1+openwin の環境で，xgraph を使いたいという要望があって，xgraph-11.3.2.tar.Z を install しようとしたのですが，st.c を gcc2.7 で compile するときに，strchr なんて知らないといってこけてしまいます.
=====> xgraph-11.3.2.tar.Z が install でき ない

図 4.2: 元の質問記事のクオリティが低い例

誤った標準形に変換した

次に、標準形に変換はしたものの標準形が適切ではなかった 53 文について、その原因を考察する。

失敗の要因の 1 つにはニュースグループに投稿される記事のテキストのクオリティの低さが挙げられる。本研究で構文解析を使用していないのはそのためであり、重要情報を抽出することで 70%の正解率は得られたのだが、記事のクオリティの低さが原因で不正解になったものも存在している。例えば図 4.2 の (1) では、どのような標準形が適切なのか人間でもよくわからない。

一方、図 4.2 の (2) は、本研究で用いた重要情報の抽出法でうまく標準形に変換できている例である。現在の構文解析システム（例えば KNP）は、このようなくだけた日本語文を正しく解析することができない。しかし、本システムは構文解析を用いていないため、このような文に対しても正しい標準形を出力することができる。

標準形の変換を失敗した理由のもう 1 つは、標準化ルールの誤りや目的語抽出ルーチンの不十分さに起因するものである。図 4.3 の (1) では重要動詞の抽出に失敗しており、図 4.3 の (2) では目的語の抽出に失敗している。

これらの問題を解決するためには、標準化ルールの見直しや目的語抽出アルゴリズムの

(1) [否定] Ｓファミリーのディスプレイなしのモデル５台に Solaris2.5.1 のインストールを予定しているのですが、どのような形態でインストールすればいいのかわかりません。
=====> Solaris2.5.1 の インストール が 予定 でき ない

(2) [] C++コンパイラがメソッドの名前を変更 (mangle) する際のアルゴリズムについてどなたかご存じの方いらっしゃいますか?
=====> メソッド の 名前 を 知り たい

図 4.3: 誤った標準形を出力した例

改良が必要である。

第 5 章

結論

本研究では、Sun QA-Pack のサマリー文を対象として、質問文を標準形へと変換する手法を研究した。本研究で作成した質問文の標準化システムは、入力文整形モジュール、標準化モジュール、出力文選択モジュールの 3 つのモジュールから構成されている。

入力文整形モジュールでは、Sun QA-Pack のサマリー文に対して専門用語タグの削除と文分割を行ない、標準化モジュールへサマリー文を整形して渡す。

標準化モジュールでは、入力文をその文末表現から分類し、本研究でまとめた 8 種類の標準化ルールに従って質問文を標準形に変換する。標準形への変換は、日本語形態素解析システム Juman を用いて入力文の形態素解析を行ない、重要な動詞とその動詞に係る目的語を抽出後、標準化ルールに従って整形することで実現した。

システム全体では、1 入力に対して 1 出力となるようにする。出力文整形モジュールでは、1 つのサマリーに対し複数の標準形が出力された場合、その中で最も適切な標準形を選択する。

本研究で作成したシステムについて、ネットニュースグループ fj.sys.sun に投稿された質問記事から抽出したサマリー文を対象として、評価実験を行なった。本研究での標準化の対象とはしなかった状況説明型のサマリー文と、内容が不明瞭なサマリー文を除外した 215 件に対して行なった実験では、なんらかの標準形が出力されたものが 177 文、標準形に変換できなかったものが 87 文であり、標準化モジュールのカバレジは 67% となった。うち、正しい標準形に変換できたものが 124 文 (70%)、誤って変換したものが 53 文 (30%) であり、正解率は 70% である。これは、元のニュース記事のテキストのクオリティが低いことを考慮すると良い結果であると言える。

今後の課題としては、カバレジを向上させるための標準化ルールの充実が挙げられる。

謝辞

本研究を進めるにあたり、終始御指導頂きました佐藤理史助教授に心よりお礼申し上げます。また、研究において日頃より適切なアドバイスを下さった知識工学講座の皆さんに感謝いたします。また、本論文を作成するにあたって協力して下さった佐藤円博士に感謝いたします。ありがとうございました。

参考文献

[1] 佐藤円, 佐藤理史: ネットニュース記事群の自動パッケージ化, 情報処理学会論文誌 vol.38 No.6, 1997.

[2] Donald A. Norman, 野島久雄（訳）: 誰のためのデザイン?, 新曜社, 1990.

[3] 益岡隆志, 田窪行則: 基礎日本語文法 -改定版-, くろしお出版, 1992.

[4] 黒橋禎夫, 長尾真: 日本語形態素解析システム JUMAN version 3.5. 京都大学大学院工学研究科, 1998.