富田一般化LRパーザを用いた情報抽出

江里口 善生† 木谷 強†

テキストから興味ある情報を抜き出す情報抽出の手法として，文字列の並びを認識するパターンマッチング処理が注目されている．パターンマッチング処理は構文解析に比べ一般的に処理時間が短く，全文の解析を必ずしも必要としない情報抽出に適している．これまでにARPAによる情報抽出コンテストMUCなどで，パターンマッチング処理を利用した情報抽出システムが開発されてきた．しかし，パターンマッチングの効率は，大量文書を処理する場合は重要であるにもかかわらず，ほとんど検討されていなかった．本論文では，スキップ機能を有する富田一般化LRパーザをパターンマッチングエンジンとして使用し，情報抽出のための効率的なパターンマッチング手法を検討する．まず，パーザへの入力単位を形態素と文節で比較し，企業の業務提携に関する新聞記事を使用した実験から，処理精度と速度ともに文節単位の入力が優れていることを示す．次に，マッチングする対象とならない語をパーザへ入力する前に取り除く不要語フィルタリング処理を提案する．実験により，文節単位の入力で不要語フィルタリングを使用する場合，形態素単位の入力で不要語フィルタリングを使用しない場合に比べ，処理速度が約23倍も向上することを示す．提案する手法により，富田一般化LRパーザを用いた情報抽出のための効率的なパターンマッチング処理が実現できることを明らかにする．

Information Extraction from Japanese Text Using Tomita's Generalized LR Parser

YOSHIO ERIGUCHI† and TSUYOSHI KITANI†

Pattern matching, which recognizes character sequences in a text, has been used for extracting information of user's interest. Pattern matching is suitable for information extraction, since it is generally fast by its nature and the extraction does not necessarily require full text analysis. Several information extraction systems such as ARPA-sponsored MUC systems were based on pattern matching. Efficiency in pattern matching for information extraction, however, has not been well investigated in spite of the importance in processing a large amount of text. This paper studies efficient pattern matching using Tomita's generalized LR parser known as one of the fastest practical parsers. Two different input formats to the parser, a morpheme (primitive word) format and bunsetsu format comprising a content word and following function words, are compared. From our experiments using newspaper articles of corporate joint ventures, the bunsetsu format is proved to be superior to the morpheme format in both processing speed and extraction accuracy. Furthermore, filtering out unnecessary words prior to pattern matching improves the parser's speed about twenty-three times faster compared to parsing the morpheme input without word filtering. Our proposed method applied to Tomita's generalized LR parser for information extraction raises pattern matching efficiency greatly.

1. はじめに

テキストから特定の情報を抽出する情報抽出は，データベースの自動構築や要約生成など応用範囲が広い．特に米国ではさかんに研究されており，ARPA (Advanced Research Projects Agency) が MUC (Message Understanding Conference) と呼ばれる一連の情報抽出コンテストを主催している．

MUCで発表された初期の情報抽出システムは，機械翻訳のアプローチと同様に，構文解析を主要な処理コンポーネントとしていたものが多かった．しかし，一般に構文解析は処理時間が長く，解析が途中で失敗した場合，解析を継続することが難しいという問題をかかえている[1]．これに対し，特定の情報は限られたパターンで表現されることが多いという考え方から，全文の構文要素を解析せず，表層の単語列の並びを認識するパターンマッチング処理が，情報抽出の新しいアプローチとして注目されている．パターンマッチング処理は構文解析に比べて処理が単純であるため，構

† NTTデータ通信株式会社情報科学研究所
Laboratory for Information Technology, NTT DATA Corporation

文解析より高速な実行が可能である．また，特定の単語の並びだけを解析対象とするパターンセットは，文中の全単語を解析対象とする構文解析ルールに比べて容易に作成できる．ただし，パターンは処理対象の分野に依存するため，あらかじめ分野を限定しておく必要がある．SRIのFASTUSシステム[2]とGE-CMU（General ElectricとCarnegie Mellon大学の共同研究チーム）のTEXTRACTシステム[3),4)]は，それぞれMUC-4とMUC-5でパターンマッチング処理を用いて高い処理精度を達成した．日本でも，松尾らによる新聞記事からの製品の内容を抽出する研究[5)]や，北陸科学技術大学院大学での佐藤らによるネットニュースからのダイジェスト自動作成システムの研究[6),7)]で，パターンマッチング処理の有効性が示された．

パターンマッチング手法は，構文解析を用いた手法と比較すると一般的に高速であるが，大量の文書を処理する場合は，さらに高速な処理が望まれる．ところが，従来の研究は処理精度の向上を目的としていたため，処理速度についてはほとんど議論されなかった．

TEXTRACTのパターンマッチング処理は，ある文にパターンがマッチするかをパターンを1つずつ順番に調べていく単純なもので，処理効率を向上させるための工夫はほとんどなく，500文字程度の新聞記事のパターンマッチング処理に平均で10秒程度要していた．MUC-5では参加システムの処理時間が公表されていないため，他のシステムの処理性能を正確に知ることはできないが，参加したシステムの中では，TEXTRACTはトップレベルの処理速度であったことが知られている．

日本における研究でも，処理時間については報告されていない．松尾らの研究では，パターンマッチング用の新しい言語を作成し，パターンの表現力とパターン開発の容易さの向上に成功しているが，処理時間については議論されていなかった．佐藤らの研究は，ネットニュースのダイジェストをパターンマッチング手法で作成することに成功しているが，パターンマッチングの手法やその効率については議論されていなかった．

本研究では，高速な構文解析エンジンとして知られる冨田一般化LRパーザをパターンマッチングエンジンとして使用する．Carnegie Mellon大学（CMU）では，冨田一般化LRパーザに対し，構文ルールに適合しない語（不要語）を読み飛ばしながら解析するスキップ機能を追加した（以降このパーザを拡張冨田パーザと呼ぶ）[8),9)]．パターンマッチング処理は，不要語が多く存在する文の構文解析と考えることができるので，スキップ機能を持つ拡張冨田パーザはパターンマッチングエンジンとして使用できる．本研究の目的は，パターンマッチングエンジンに拡張冨田パーザを使用し，日本語文書から情報を抽出するための効率的なパターンマッチング手法を提案することである．本論文では，企業の提携関係を扱った新聞記事から提携関係を抽出する実験を行い，提案するパターンマッチング手法の有効性を示す．

2. 拡張冨田パーザの特徴

本章では拡張冨田パーザの特徴の中から，パターンマッチングエンジンとして使用する場合に関係の深い特徴を説明する．

2.1 スキップ機能

スキップ機能とは，不要語が解析の途中に現れた場合，不要語を読み飛ばし解析を続ける機能である．スキップ機能は，1992年にCMUでオリジナルの冨田一般化LRパーザに付け加えられた[8),9)]．CMUでは音声認識をした会話文を解析するときに，文の途中に挿入される感動詞や言い直し語などを，不要語として読み飛ばして解析を続けるためのパーザとして使用している．不要語が文中に現れた場合，従来のパーザではその時点で解析が異常終了したが，拡張冨田パーザではスキップ機能により解析を続けることが可能である．拡張冨田パーザのスキップ機能は，パラメータの設定により，連続してスキップする不要語の数（スキップ幅）を任意に設定できる．初期の拡張冨田パーザでは，スキップ幅は全ルールに対して一律に設定され，ルール個別には設定できなかったが，最新のバージョンでは，ルールの中でスキップ幅を自由に変更できるようになった．

2.2 ユニフィケーション文法

拡張冨田パーザは，LFG文法に非常に近い文法スタイルを持つユニフィーケーション文法を採用している[10),11)]．ユニフィケーション文法は，一般的に句構造を規定する文脈自由文法と，構成要素の持つ属性構造を規定する付記から成る[12)]．属性構造とは属性とその値の対を要素とする集合である．

図1は拡張冨田パーザの文法の一例である．属性構造は，文脈自由文法の左辺の非終端記号をx0，右辺のn番目の非終端記号（終端記号）をxnという名前を用いて参照する．たとえば図1の右辺の <NP> の属性caseを参照するときは，(x1 case) というパス表現で指定する．

拡張冨田パーザには，フルユニフィケーションモードと疑似ユニフィケーションモードと呼ばれる2つの実行モードが用意されている．フルユニフィケーショ

```
(<S> <== (<NP> <VP> <PP>)
   (((x1 case) =c nominative)
    ((x1 agreement) = (x2 agreement))
    ((x0 pp-adjunct) > x3)
    ((x0 subj) = x1)
    (x0 = x2)))
```

図1 ルールの例
Fig. 1 Example of a rule.

ンモードは，標準的なユニフィケーション文法である．疑似ユニフィケーションモードは，属性構造の操作にユニフィケーション以外のものを加えたもので，正確にはユニフィケーション文法ではない．本実験では疑似ユニフィケーションモードを使用した．図1は，疑似ユニフィケーションモード用のルールである．

疑似ユニフィケーションモードで使用できる属性構造の操作で，本研究に特にかかわりの深い重要な操作を紹介する．

ユニフィケーション 通常のユニフィケーション．記号 '=' を使用する．

制約等式 左辺と右辺の値が等しいかチェックする式．記号 '=c' を使用する．

Multiple Value 左辺の属性構造の値に，右辺の値を追加する．したがって，左辺の属性構造の値は，同時に複数の値を持つことになる．記号 '>' を使用する．

Lisp 関数 式の右辺に LISP 関数を記述し，LISP 関数の評価値を左辺に代入する．記号 '<=' を使用する．

また，文脈自由文法の右辺に，すべての文字あるいはリストに適合するワイルドカードを記述できる．ワイルドカードは記号 '%' を使用する．

3. 拡張冨田パーザのパターンマッチング処理への適用

本章では拡張冨田パーザを利用したパターンマッチング手法について説明する．まず 3.1 節で，形態素を解析単位とするパターンマッチング手法を述べる．次に 3.2 節で，形態素単位の解析で発生する問題点を解消する手法として，文節を解析単位とするパターンマッチング手法を説明する．3.3 節では，一層の処理効率の向上を目指した不要語フィルタリングについて説明する．

3.1 形態素単位のパターンマッチング手法

3.1.1 入力データ

拡張冨田パーザはユニフィケーション文法が使え，入力データを属性構造で与えることが可能である．入力データとして形態素を単位とする場合，入力データは以下の手順で作成される．

表1 形態素の属性構造
Table 1 Feature names in the morpheme format.

項番	属性名	内容
1	root	形態素の表記
2	pos	品詞
3	roma	読みがな
4	subpos	助動詞の意味または，活用語尾の活用形
5	kind	固有名詞のカテゴリー（企業名，人名，地名）

1～3 はすべての形態素が有する属性である．
4～5 は，該当する形態素のみ有する属性である．

```
((root リンゴ)(pos N)(roma ringo))
((root を)(pos P)(roma wo))
((root 食べ)(pos V)(roma tabe))
((root た)(pos AUX)(subpos PAST)(roma ta))
```

図2 形態素の属性構造の例
Fig. 2 Example of the morpheme format.

(1) 形態素解析器 MAJESTY を使用し，新聞記事を基本的な単語に分割する[13]．MAJESTY が使う辞書サイズは約9万語で，その中には固有名詞（人名：約7,000語，地名：約7,000語，企業名：約2,000語）が含まれている．形態素解析の精度は，形態素分割の第一候補と品詞の全候補を対象とした場合 98.4%である☆．

(2) 企業名特定処理により，企業名に特有な接尾語（"～社" など）と企業名の直前によく現れる語（"大手の" など）から企業名を推定して，形態素解析で特定できなかった企業名を特定する[14]．この処理により，企業名の再現率は 84.3%，適合率は 81.4%になる．

(3) 以上の処理により得られた情報を，拡張冨田パーザが利用できるデータ形式（属性構造を有するリスト形式）に変換する．

本実験で使用した属性名と形態素情報の対応を表1に，入力データ例として "リンゴを食べた" の属性構造を図2に示す．

3.1.2 パターンセット

拡張冨田パーザが参照するパターンセットは，構文解析における文法ルールセットに相当する．本実験で

☆ 評価対象文書は企業の業務提携に関連する 31 の新聞記事で，全文字数は 16,428 文字であり，全形態素数は 9451 形態素である．

```
パタン1:
(<tie-up> <== (<company> <ga> <company>
                <to> <tie-up-word1>)
   (((x0 relation) = TIE-UP)
    ((x0 state) = (x5 state))
    ((x0 partner) = (x3 company))
    ((x0 partner) > (x1 company))))

パタン2:
(<company> <== (<company*> )
   (((x0 company) = (x1 company))))

パタン3:
(<company> <== (<company*> <to> <company>)
   (((x0 company) = (x3 company))
    ((x0 company) > (x1 company))))

パタン4:
(<company*> <== (%)
   (((x1 value kind) =c company)
    ((x0 company name) = (x1 root))))

パタン5:
(<tie-up-word1> <== (%)
   (((x1 value root) =c
     (*OR* 提携 タイアップ 契約 合意 交渉 協力
           共同 業務提携 技術協力 生産 販売))
    ((x0 state) = 現行)))

パタン6:
(<ga> <== (%)
      (((x1 value root) =c (*OR* が は も))
       ((x1 value POS) =c P)))
```

図3 提携関係を抽出するパターンの例
Fig. 3 Example of patterns for extracting joint venture relationships.

は，企業の提携関係を扱った新聞記事を対象とし，抽出する情報は企業名，企業の種類，所在地，役員名および，提携関係の種類（パートナー関係，合弁会社の関係，親子関係のいずれか）である．情報抽出用のパターンは，TEXTRACTで使用したパターンを元に作成した．使用パターンは80個で，45種の非終端記号と1種の終端記号で構成されている．終端記号はワイルドカードで表現し，付記の部分で終端記号の条件を記しているため，その数は1個である．図3は実験で使用したパターンセットの中の代表的なパターンの例である．パターン1は，“A社がB社と提携した”と書かれた文から提携関係を抽出する．パターン2，パターン3はト格を用いて表現される並列句まで考慮して企業名を抽出し，パターン4は企業名の終端記号を抽出する．パターン5は提携関係のキーワードを抽出し，パターン6は，助詞“が”，“は”，“も”を抽出する．45種の非終端記号は，図3のパターン1の左辺の <tie-up> や <company> のように複数の非終端記号のまとまりをシンボル化するタイプと，<ga>，<company*>，<tie-up-word1> のように，終端記号をシンボル化するタイプがある．

パターンは，1文単位でパターンマッチングを行うことを前提に作成した．文単位に抽出された情報は，文脈処理プログラムで文章全体の情報にまとめられる．文脈処理プログラムの主たる処理は，関連情報のリンクを張ることである．たとえば，“A社がB社と提携した．A社のX社長は，次のように述べている.”という文に対し，A社という企業はB社と提携関係にあり，X社長が社長であるというリンクを張り，3つの情報の関係を特定する．

3.1.3 スキップ幅に関する問題

初期の拡張冨田パーザでは，スキップ幅はユーザの指定により任意の値に変更できるが，全パターンに対して一律に作用する．したがって，隣接する終端記号（非終端記号）の間には他の終端記号が入ることが許されるため，隣接する形態素の並びをパターン化することはできない．その結果，本来適合すべきでない形態素の組合せにパターンが適合する可能性が高くなる．以下に，スキップ機能が原因となり，誤ったパターンマッチングを引き起こす例をあげる．

助詞に対するパターン記述

作成したパターンは，企業名と，企業名に付随する助詞に注目しているものが多い．たとえば，“A社 がB社と 提携 した”というパターンを抽出するときは，図3のパターンセットのように会社名と助詞を非終端記号にしたパターンセットを用いる．しかしスキップ機能があるため，図3のパターンは，次の文のように“B社”と関係のない助詞“と”を結び付けて望ましくない適合をすることがある．

“A社 が設立した新会社B社は，○○分野と
△△分野での提携先を探している.”

否定の助動詞のパターン記述

否定の助動詞は直前の用言の意味を打ち消すため，否定の助動詞を認識するパターンは重要である．しかし，否定の助動詞を認識するために，“用言＋否定の助動詞”をパターンとして定義した場合，スキップ機能のため本来関係のない用言と否定の助動詞が結び付く可能性がある．たとえば次の文には提携関係が存在するが，提携関係が否定されたものと誤認識される．

“A社とB社の提携 は，分野が競合しないので実現可能となった.”

処理時間への影響

スキップ幅の制約条件の弱さは，抽出内容だけでな

く処理時間にも影響する．拡張冨田パーザは，他の多くの一般化LRパーザと同様に，構文的に曖昧性が大きくなると処理時間は増大する．本実験で使用したパターンセットは，企業名と助詞の組合せに着目したパターンが多いが，前述したようにスキップ幅の制約条件が弱いため，解析過程で企業名と助詞の組合せを一意に特定できない．このため，企業名と助詞が多く含まれる文の解析では，その2つの組合せだけでも非常に多くなり，処理時間が増大する．

3.1.4 問題を解決する手法の考察

前節で述べた問題の第一の解決方法として，拡張冨田パーザのスキップ機能をさらに強化し，パターンごとにスキップ幅を指定できるようにすることが考えられる．本実験を開始した1993年には，パターン内部で，スキップ幅を変更することはできなかったが，1994年10月の改良で可能となった．これを利用し，パターンごとにきめ細かくスキップ幅を定義すると，パターンの曖昧性が減少するため，曖昧性が大きい一部の文に対しては処理時間が大幅に減少した．しかし，処理が複雑になったため，全体としては処理時間が数パーセント増加することが，実験により確かめられた．スキップ幅の設定は，前節で述べた問題を解決するためには有効な改良であったが，処理時間が増加するという欠点があり，今回は採用しなかった．

第二の解決方法として，スキップしてはならない連続する語を，1つの終端記号として扱う手法が考えられる．スキップ幅の制約条件が弱いために生じる問題は．構文的に関係がなく位置的にも離れた名詞と助詞，あるいは用言と助動詞を結び付けることであった．これは，自立語と付属語を別々の終端記号としてパターンマッチング処理していたことが直接の原因である．したがって，名詞と助詞，動詞と助動詞といった組合せを同じ終端記号で扱えるように，入力単位を形態素から文節に変更すれば，この問題を解決できる．次節では文節を入力単位とするパターンマッチング手法について説明する．

3.2 文節を単位とするパターンマッチング手法

3.2.1 文節の定義と文節の属性構造

本研究では文節を次のように定義した．

定義1 文節とは，1つの自立語と付随する付属語とする．

定義2 句読点，カッコ等の記号は単独で文節を作るものとする．

文節が持つ情報として**表2**の属性を用意した．入力文は表2中の属性を持つ属性構造に変換される．文節単位の属性構造は，C言語で作成した処理変換ツールを用いて形態素単位の属性構造から作成した．**図4**は，前述の例である“リンゴを食べた”を文節単位の属性構造に変換したものである．

表2 文節の属性構造
Table 2 Feature names in the bunsetsu format.

項番	属性名	内容
1	hyouki	文節全体の表記
2	root	自立語の表記
3	pos	自立語の品詞
4	kaku	文節の格（文節中の助詞）
5	subpos	助動詞の意味素性
6	kind	固有名詞のカテゴリー（企業名，人名，地名）
7	prefix	接頭語
8	suffix	接尾語

1～3はすべての文節が有する属性である．
4～8は，該当する文節のみ有する属性である．

```
((hyouki リンゴを)(root リンゴ)(pos N)(kaku を))
((hyouki 食べた)(root 食)(pos V)(subpos PAST))
```

図4 文節の属性構造の例
Fig. 4 Example of feature structures in the *bunsetsu*format.

3.2.2 文節単位のパターンセット

形態素単位と文節単位のパターンの相違点は，形態素単位のパターンでは，助詞が独立した非終端記号として句構造を規定する文脈自由文法で扱われていたのに対し，文節単位のパターンでは，助詞は文節の属性の1つとして扱われ，属性構造の関係を規定する付記の部分で定義される点である．

使用した文節単位のパターンは75個で，34種の非終端記号と1種の終端記号で構成されている．助詞に関する非終端記号が必要なくなったため，非終端記号の種類は，形態素単位のパターンセットに比べ11種少なくなっている．また，助詞に関するパターンを作成する必要がなくなり，パターンの数も減少している．**図5**に本実験で使用した文節単位のパターンセットの代表的なパターンを示す．図5のパターンセットは図3のパターンセットを文節単位の解析用に変換したもので，パターン1からパターン5までは，それぞれ対応している．形態素単位のパターン6に対応する文節単位のパターンはないが，その代わりにパターン1の付記に ((x1 kaku) = c (*OR* が は も)) という制約等式がある．

3.3 不要語処理

拡張冨田パーザでは，LRテーブルを参照するときに不要語を判別する．一般的には不要語であるか否かは文脈に依存するため，解析途中で生成される複数の解析木ごとにLRテーブルを参照する必要がある．とこ

```
パタン1:
(<tie-up> <== (<company> <company>
                <tie-up-word1>)
   (((x1 kaku) =c (*OR* が は も))
    ((x2 kaku) =c (*OR* と との とも))
    ((x0 relation) = TIE-UP)
    ((x0 state) = (x3 state))
    ((x0 partner) = (x2 company))
    ((x0 partner) > (x1 company))))

パタン2:
(<company> <== (<company*> )
   (((x0 company) = (x1 company))
    ((x0 kaku) = (x1 kaku))))

パタン3:
(<company> <== (<company*> <company>)
   (((x1 kaku) =c と)
    ((x0 company) = (x2 company))
    ((x0 company) > (x1 company))
    ((x0 kaku) = (x2 kaku))))

パタン4:
(<company*> <== (%)
   (((x1 value kind) =c company)
    ((x0 company name) = (x1 root))
    ((x0 kaku) = (x1 value kaku))))

パタン5:
(<tie-up-word1> <== (%)
   (((x1 value root) =c
     (*OR* 提携 タイアップ 契約 合意 交渉 協力
           共同 業務提携 技術協力 生産 販売))
    ((x0 state) = 現行)))
```

図5 提携関係を抽出する文節単位のパターンの例
Fig. 5 Example of patterns in the bunsetsu format.

ろが，本実験のように拡張冨田パーザをパターンマッチングエンジンとして使用した場合，パターンセットに終端記号として定義されていない語は不要語としてよい．終端記号になる可能性がない語は文脈に依存しないので，そのような語を生成された解析木ごとにLRテーブルを参照するのは無駄である．したがってパターンマッチング処理を行う前に，終端記号として定義されていない語を除去することで処理時間を短縮できる．終端記号に定義されていない語は，パターンセット中に存在する終端記号の集合から容易に判別できる．

3.3.1 不要語フィルタリング

終端記号に定義されていない不要語を除去するアルゴリズムを不要語フィルタリングと呼ぶ．不要語フィルタリングのために，まずパターンマッチング処理の実行前にパターンセット中に使われている終端記号の集合を作る．今回使用した情報抽出用のパターンでは，文脈自由文法の部分で終端記号をワイルドカード（%）で表現し，付記の部分でワイルドカードの条件を定義している．そこで文脈自由文法部にワイルドカードを持つ各パターンからワイルドカードの条件を抽出し，抽出した各条件の和集合を作ることで，終端記号の集合を作る．たとえば図5のパターン2では，付記に(x1 kind) = c COMPANYという制約等式，すなわち終端記号の属性kindの値がCOMPANYであるという条件が定義されているので，この制約等式を終端記号の1つとして集合に加える．不要語フィルタリングは，パターンマッチング処理で入力語を読み込むときに，入力語ごとに終端記号の集合に含まれない語を削除することで実現される．

本実験の形態素単位用のパターンの場合，終端記号の集合は，固有名詞（属性kindがCOMPANYまたはPLACEまたはPERSON），数詞，助詞，またはキーワード（“提携”，“子会社”，“法人”，“大手”などの単語や記号）で構成される．したがって，固有名詞でも数詞でも助詞でもなく，かつ，キーワードリストに含まれない語がすべて不要語である．

4. 評価結果と考察

拡張冨田パーザを使用した実験を実施し，文節単位の解析と不要語フィルタリングの有効性を抽出精度と処理時間の観点から評価した．

4.1 評価対象テキスト

実験では，情報抽出プロジェクトTIPSTERの研究のため，ARPAからCMUに提供された企業の提携関係を記載した148の日本語新聞記事を使用した☆．そのうち75記事をパターン作成用に使用し，残り73記事を評価用に使用した．

1記事は平均7.2文で構成され，1記事あたりの文字数は平均390文字である．1記事中には0～5件の企業の提携関係が記述されている．1文に含まれる形態素の数は5～123語で平均37語であり，文節の数は3～58文節で平均は17文節である．図6は使用した新聞記事の例である．記事は本文，日付，出典等がSGMLタグにより，タグ付けされている．

4.2 抽出精度の評価

4.2.1 抽出精度の実験結果

本実験では，企業の提携関係を記載した新聞記事から提携関係に関する情報を抽出している．正解データには，ARPAから提供された専門家が作成したデータを使用した．評価尺度は以下に示す再現率と適合率

☆ TIPSTERおよびMUC-5では，日本語と英語の企業提携分野と半導体製造分野を記載した新聞記事からの情報抽出を対象にしていた．

<doc>
< REFNO> 朝日新聞. 000023 </REFNO>
< DOCNO> 0023 </DOCNO>
< DD> 85.03.12 </DD>
< SO> 朝日新聞 朝刊 8頁 2経 写図無 (全175字) </SO>
< TXT>
大丸は四月四日から，住友クレジットサービス（本社・大阪市）などVISAカードグループ六社と提携した「大丸エクセルVISAカード」を発行する．銀行系のクレジット会社との提携は，昨年九月に発行を始めた「大丸エクセルDCカード」に続いて二番目．大丸で買い物をした金額の5%が割引になるほか，VISAカードとして，国内や世界百六十五カ国の加盟店で通用する．
< /TXT>
< /doc>

図6 新聞記事の例
Fig. 6 Example of an article.

表3 本実験の抽出精度（不要語フィルタリングなし）
Table 3 Extraction accuracy of this experiment without word filtering.

	再現率	適合率
形態素単位の解析	57%	62%
文節単位の解析	56%	76%

を用いる．

$$\text{再現率} = \frac{\text{抽出した正しい提携関係の個数}}{\text{正解データにある提携関係の個数}}$$

$$\text{適合率} = \frac{\text{抽出した正しい提携関係の個数}}{\text{抽出したすべての提携関係の個数}}$$

正解の基準は，提携に関係する企業が正しい名前で過不足なく抽出されていて，かつ提携関係の種類（提携，合弁，親子関係）が正しく認識されていることとする．

73記事には，提携関係の記述が107件存在した．73記事に対して形態素単位と文節単位の解析を行った結果，両者の提携関係に関する再現率と適合率は表3のようになった．表3から，再現率は形態素単位より文節単位の解析が1%低下したが，適合率については文節単位の解析が形態素単位の解析より14%上回っている．再現率がほぼ同じで適合率が大幅に向上しているので，抽出精度の面では形態素単位よりも文節単位の解析が優れているといえる．なおMUC-5でのTEXTRACTの抽出精度は，再現率が71%，適合率が78%という値が報告されている☆．本実験ではTEXTRACTと同様のパターンセットを使用したが，TEXTRACTには組み込まれている指示語の参照元の特定や，省略語の推定等，再現率を高めるための複雑な文脈処理を組み込んでいないため，TEXTRACTの処理結果より劣っている．

☆ 文献15) のJJV Summary Scoresの中から，rel-ent2-to-ent1の項目を参照した．

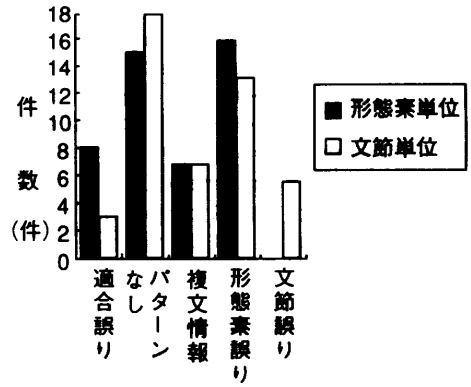

図7 情報を抽出できない要因
Fig. 7 Reason for unextracted information.

4.2.2 再現率についての考察

評価セット73記事に関し，抽出できなかった提携関係の原因を類別化し，その件数をグラフにしたものが図7である．まず，類別化した種類について説明する．

適合誤り 本来適合すべきでないパターンに適合し，誤った提携関係を抽出する誤り．再現率と適合率の両方に影響を与える．たとえば，“マツダはサーブ・スカニアとの提携交渉を事実上断念し，シトロエンと提携の交渉に入った.” に対しては，“<company> <ha> <company> <to> <提携> <断念>” というパターンに適合することが望まれる．しかし，“<company> <ha> <company> <to> <company> <to> <提携>” というパターンが存在し，本実験では優先順位の関係で誤ったパターンに適合し，正しい関係を抽出できなかった．

パターンなし 定義したパターン以外の形式で表現されているため抽出に失敗する提携関係．再現率に影響を与える．たとえば，“第一勧銀はATMをシティーバンクに解放することで合意した.” という文に対しては，提携関係を抽出するパターンを定義していなかったため，提携関係の抽出に失敗した．

複文情報 複数の文にまたがって提携関係が記述してあり，本実験で使っている文脈処理で情報を組み合わせることができない提携関係．たとえば，“日揮はマツオ産業と提携した．近く新会社「エム・

アンド・ジェイ」を共同で設立する."という文章では，2つの文にまたがって，合弁会社設立の記事が記述してある．本実験で使用した文脈処理では対応できない形式であったため，合弁会社の関係が特定できなかった．

形態素誤り 形態素解析誤り（固有名詞特定処理の誤りを含む）が原因となってパターンマッチングが正しく実施されず情報を抽出できない提携関係，または，誤った情報を抽出する誤り．再現率と適合率に影響を与える．たとえば，"中国汽車工業進出口公司"という企業名を，MAJESTYでは"中国（地名），汽車（名詞），工業進出口公司（企業名）と分割し，企業名を誤っている．そのため，パターンマッチング処理では誤った企業名を抽出する．

文節誤り 形態素から文節への変換が適切に実施されなかったことが原因で抽出できない提携関係．たとえば，"「ぴあ」と「ニチイ」が提携した."という文を文節に分割するときに，"ぴあ"と助詞の"と"を同一の文節に分割しなかったため，"ぴあ"に対する格助詞の情報が欠如し，解析に失敗した．

形態素単位の解析に比べ，文節単位の解析では"適合誤り"と"形態素誤り"が少なかった．形態素単位の解析では，名詞と助詞の組合せが一意に決定できないため，"適合誤り"の可能性が高かったが，文節単位の解析では名詞と助詞の組合せが一意に決まるので，"適合誤り"の可能性が低くなった．また，文節単位の解析では"パターンなし"のため抽出できない情報が，形態素単位の解析では，"形態素誤り"で抽出されているケースもいくつかあった．そのため，"形態素誤り"の数は，形態素単位の解析よりも文節単位の解析のほうが少なくなっている．一方，文節単位では制限が厳しくなったことにともない新たなパターンを定義する必要があったが，新規パターンを作成しなかったため，"パターンなし"による抽出もれが増加した．

全体的に見ると，文節単位の解析では形態素単位の解析に比べ"適合誤り"と"形態素誤り"が減少したが，"パターンなし"と"文節誤り"が増加したため，結果として形態素単位の解析と同程度の再現率となった．

4.2.3 適合率についての考察

誤った提携関係を抽出した原因を類別化し，その件数をグラフにしたものが図8である．抽出誤りの原因は，形態素解析時の誤り以外は"適合誤り"によるものであった．再現率で考察したのと同様に，名詞と助詞の組合せの制約が強くなった結果，形態素単位の解析に比べ文節単位の解析では"適合誤り"の可能性が低くなったため，適合誤りが減少した．この結果，文節単位の解析は形態素単位の解析に比べ，適合率が14%向上した．

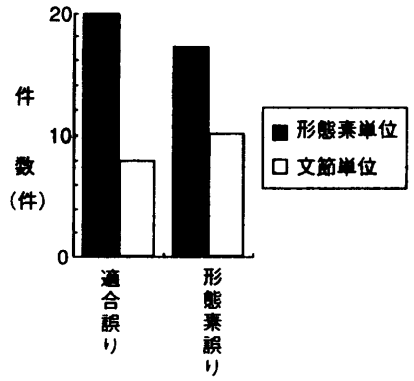

図8 誤った情報を抽出する要因
Fig. 8 Reason for incorrectly extracted information.

4.2.4 不要語処理と抽出結果の関係

不要語を削除すると，拡張冨田パーザが解析時にスキップする不要語の数が変化するため，スキップ幅の値によっては，抽出結果が異なる可能性がある．たとえば，"A社と，自動車の，部品を，作る，B社が，提携した."という文節に区切られた文を，スキップ幅2で，<company> <company> <tie-up-key>というパターンで抽出する場合は，A社とB社の間に3つの文節があるため，スキップ幅の制限により失敗する．しかし，もし不要語フィルタリングで"A社と，自動車の，B社が，提携した."のように，"部品を"と"作る"が不要語として削除された場合は，A社とB社の間の文節は1つなので，スキップ幅が2でも前述のパターンに適合する．本実験ではスキップ幅を無限大にしていたため，上記のようにスキップ幅の制限により失敗することがないため，不要語フィルタリングの有無にかかわらず抽出結果は等しかった．

4.3 処理速度の評価

処理時間の測定は，148記事中に含まれる1069文に対して行った．形態素単位と文節単位の解析に対して不要語フィルタリングの有無による差異を比較するため，4通りの処理方法で処理時間を測定した．実験環境としてSun Sparc 10（メモリ96MB）を使用し，Sun Common Lisp言語により記述された拡張冨田パーザを動作させた．計測時間はパターンマッチング処理の開始から終了までの時間であり，形態素解析，固有名詞特定処理，形態素から文節へのデータ変換の処理時間を含まない．

表4は各条件での1文あたりの平均処理時間と，最

表4 平均処理時間（文単位）
Table 4 Average processing time per sentence.

	不要語フィルタリングなし		不要語フィルタリングあり	
	形態素	文節	形態素	文節
平均処理時間（秒）	9.2	1.0	2.5	0.40
最長処理時間（秒）	2541	132	657	73

表5 1文あたりの入力語数
Table 5 Number of input words per sentence.

	不要語フィルタリングなし		不要語フィルタリングあり	
	形態素	文節	形態素	文節
平均語数	37.5	16.1	17.0	5.1
最長語数	123	55	58	28

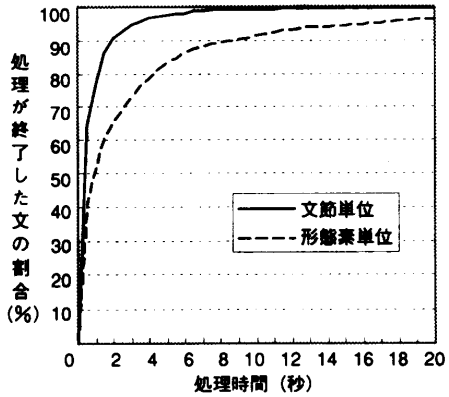

図9 処理時間と処理が終了した文の割合（形態素単位の解析と文節単位の解析の比較：不要語フィルタリングなし）
Fig. 9 Relationship between parsed sentences and processing time (The comparison of the morpheme format and bunsetsu format without word filtering).

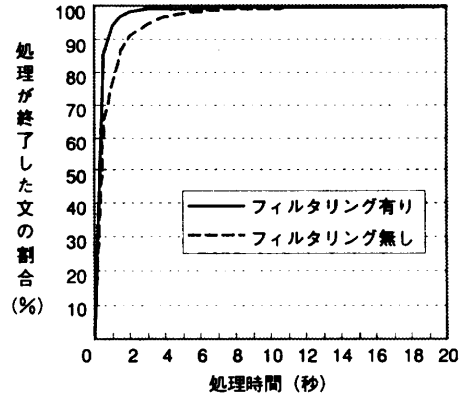

図10 不要語フィルタリングの有無による処理時間と処理が終了した文の割合（文節単位）
Fig. 10 Relationship between parsed sentences and processing time with or without a filtering program of unnecessary words (The *bunsetsu* format).

も処理時間を要した文の処理時間を示す．不要語フィルタリングがない場合，文節単位の解析は形態素単位の解析に比べ，平均処理時間が約9分の1になった．形態素単位のデータを文節単位のデータに変換する時間は1記事あたり約0.1秒（1文あたり0.01秒）であり，パターンマッチング処理の時間に対しきわめて短い．したがって全体の処理時間が大幅に減少したといえる．表4の最長処理時間が示すように，処理時間は文によって大きく異なり，その実態は平均値だけではつかみにくい．そこで，図9には，不要語フィルタリングを実施しないときの形態素単位と文節単位の処理時間に対する，時間内に処理できた文の割合を示す．同じ時間内に処理できる文の割合は，文節単位の解析の方が形態素単位の解析に比べ多い．特に処理時間が短いほどその差は大きい．たとえば2秒で解析が終了する文の割合は，形態素単位と文節単位の解析でそれぞれ66%と91%と大きく異なる．以上の分析結果から文節単位の解析が形態素単位の解析よりも処理速度の面で優れているといえる．

次に不要語フィルタリングの有無で処理時間の差を比較する．表5は不要語フィルタリングの有無による入力データの語数の変化を示している．不要語フィルタリングの使用により入力語数が減少し，表4に示すように1文あたりの平均処理時間が形態素単位の解析で9.2秒から2.5秒に，文節単位の解析で1.0秒から0.40秒に短縮できた．

図10は文節単位の解析における不要語フィルタリングの有無による処理時間の違いを示したグラフである．不要語フィルタリングを実施した方が，同じ時間内に処理できる文の割合が増加していることが分かる．以上の分析結果から不要語フィルタリングは処理速度の高速化の観点から有効であるといえる．

文節単位の解析で不要語フィルタリングを行った場合，処理時間は平均で1文あたり0.40秒（1記事に換算すると約3秒）であった．処理速度の向上に成功した．本実験と同じ環境で測定したTEXTRACT（日本語GNU AWK言語で記述）のパターンマッチング処理の処理時間は1文あたり1.3秒（1記事に換算すると約9秒）で，本方式はTEXTRACTに比べ約3倍速い．実験したデータ量は少ないが，処理時間の傾向をつかむには十分な量であった．処理時間は個々の文ではバラツキがあるものの，全体的に見ると解析する文の量に比例する傾向がある．

本方式での処理時間は文によりばらつきがあり，図10に示すように，5%程度の文が処理時間が1秒を超えている．中には極端に処理時間がかかる文も存在し，本実験での最長処理時間は73秒であった．処理時間の長い文は，ほとんどが多数の企業名をト格や読

点を用いた並列句を含む文であり，企業名の並列句の組合せが複数あるために処理時間が増大したと考えられる．したがって，企業名の並列関係を前処理で特定し，文節よりもさらに大きな単位でパターンマッチング処理をすることができれば処理時間を短縮することができる．また，本質的な改善ではないが，処理時間が特定の秒数を超えるか，構文解析木の数がある値を超えると処理を中断し，次の文の処理に移行する方式も考えられる．

5. おわりに

情報抽出におけるパターンマッチング処理の有効性が注目されているが，パターンマッチングエンジンの処理効率に関する研究は少なかった．そこで，不要語をスキップする機能を持ち，高速な一般化LRパーザである拡張冨田パーザを，パターンマッチングエンジンとして使用した．しかし，拡張冨田パーザを情報抽出のためのパターンマッチングエンジンとして使用する場合，不要語のスキップ機能が適合パターンの多様な組合せを発生させ，処理時間を長くする原因となっていた．この問題点を解決するため，文節単位のパターンマッチング手法を提案した．実験では，文節単位のパターンマッチング手法を適用することにより，形態素単位のパターンマッチング手法と再現率を同程度に保ちつつ，適合率を62%から76%に向上させ，処理時間を平均11%に短縮することができた．さらにパターンマッチング処理の処理時間を短縮する手法として不要語フィルタリングを提案し，文節単位の入力で不要語フィルタリングを実施した場合，処理時間を40%に短縮できた．文節単位のパターンマッチング手法と不要語フィルタリングを組み合わせることにより，形態素単位のパターンマッチング手法で不要語フィルタリングを用いない場合と比較して，適合率を62%から76%に14%向上させると同時に約23倍もの処理速度を達成した．この結果，新聞記事1文あたり平均0.40秒，1記事あたり約3秒で処理できるようになった．

実験の結果から，文節単位のパターンマッチング手法と不要語フィルタリングを用いることで，拡張冨田パーザを情報抽出のための効率的なパターンマッチングエンジンとして使用できることを示した．

謝辞 本研究の機会を与えてくださった，CMUのCenter for Machine Translation所長であるJaime Carbonell教授，研究実施にあたり有益なコメントをいただいた，慶應大学の冨田勝助教授，拡張冨田一般化LRパーザについての情報を提供していただいたCMUのAlon Lavie氏に感謝します．

参考文献

1) Hobbs, J., Appelt, D., Bear, J., Israel, D. and Tyson, B.: FASTUS: A System for Extracting Information from Natural-language Text, Technical Report 519, SRI International (1992).
2) Appelt, D., Hobbs, J., Israel, D., Kameyama, M. and Tyson, B.: SRI: Description of the JV-FASTUS System Used for MUC-5, *Proc. Fifth Message Understanding Conference (MUC-5)*, pp.221–235, Advanced Research Projects Agency (1993).
3) Kitani, T., Eriguchi, Y. and Hara, M.: Pattern Matching and Discourse Processing in Information Extraction from Japanese Text, *Journal of Artificial Intelligence Research*, Vol.2, pp.89–110 (1994).
4) Jacobs, P.: GE-CMU: Description of the Shogun System Used for MUC-5, *Proc. Fifth Message Understanding Conference (MUC-5)*, pp.109–120, Advanced Research Projects Agency (1993).
5) 松尾比呂志，木本晴夫：抽出パターンの階層的照合に基づく日本語テキストからの内容抽出法，情報処理学会論文誌，Vol.36, No.8, pp.1838–1844 (1995).
6) 佐藤 円，佐藤理史，篠田陽一：電子ニュースダイジェストの自動生成，情報処理学会論文誌，Vol.36, No.10, pp.2371–2378 (1995).
7) 佐藤理史，佐藤 円：ネットニュースダイジェストの自動生成，言語処理学会第1回年次大会論文集，pp.297–300 (1995).
8) Bates, J. and Lavie, A.: Recognizing Substrings of LR(k) Languages in Linear Time, Technical Report CMU-CS-91-188, Carnegie Mellon University (1991).
9) Lavie, A. and Tomita, M.: GLR*-An Efficient Noise-skipping Parsing Algorithm for Context-free Grammars, *Proc. Third International Workshop on Parsing Technologies*, pp.123–134 (1993).
10) Tomita, M., Mitamura, T. and Kee, M.: The Generalized LR Parser/Compiler Version 8.1: Users Guide, Technical Report CMU-CMT-88-MEMO, Carnegie Mellon University (1988).
11) Tomita, M. and Nyberg, E.: Generation Kit and Transformation Kit Version 3.2 User's Manual, Technical Report CMU-CMT-88-MEMO, Carnegie Mellon University (1988).
12) 安川秀樹：自然言語の基礎理論，第4章，昭晃堂，pp.109–143 (1986).
13) Kitani, T. and Mitamura, T.: An Accurate Morphological Analysis and Proper Name Identification for Japanese Text Processing,

Journal of Information Processing Society of Japan, Vol.35, No.3, pp.404–413 (1994).
14) 木谷　強：固有名詞の特定機能を有する形態素解析処理，情報処理学会研究報告，Vol.92, No.55, pp.73–80 (1992).
15) APPENDIX B: MUC-5 TEST SCORES, *Proc. Fifth Message Understanding Conference (MUC-5)*, pp.351–413, Advanced Research Projects Agency (1993).

(平成8年 2 月28日受付)
(平成8年10月 1 日採録)

江里口善生（正会員）

1970年生．1992年東京工業大学工学部情報工学科卒業．同年NTTデータ通信（株）入社．現在，同社情報科学研究所において，情報抽出を中心とする自然言語処理の研究に従事．

木谷　強（正会員）

1960年生．1983年慶應義塾大学工学部電気工学科卒業．同年日本電信電話公社入社．1991～1993年までカーネギーメロン大学Center for Machine Translation研究員．形態素解析，情報抽出，情報検索などの自然言語処理の研究に従事．現在，NTTデータ通信（株）情報科学研究所主任技師．工学博士．言語処理学会，ACM各会員．