人文科学とコンピュータ　38－7
（１９９８．５．２９）

構文解析木を対象とするデータ解析法の研究 －構文木からの知識発見システムSYKDの開発と応用－

雄山　真弓　　岡田　孝
関西学院大学　情報処理研究センター

文を対象としたデータ解析を考える場合、構文構造や意味構造を含めて解析する必要がある。著者らは、文の構文解析木中の特定節点を *viewpoint* として設定し、その周辺節点の属性から ID3 法により知識を探索的に発見する方法論を提案してきた。本報告では、クラスラベルの与えられた構文木を入力データとして、会話的にデータ解析を遂行するシステム SYKD について解説する。このソフトウェアを用いることにより、木構造自体をデータ解析の対象として直接取り扱えるため、通常の統計的手法では見いだせなかった知識を容易に発見できる。EDR コーパスから採った例文を対象として、日本語助詞の使用法に対する簡単な SYKD システムによる解析を行ったので、その結果も示す。

Data analysis of syntactic trees development and applications of knowledge discovery system: SYKD

Mayumi OYAMA　Takashi OKADA

Information Processing Research Center, Kwansei Gakuin University

Knowledge discovery software: SYKD has been developed to analyze sentences based on their syntactic structures. Some node in a syntactic tree is specified as *viewpoint*, and the features on its surrounding nodes allow us to extract knowledge about the characteristic usage of the word at the *viewpoint*. Since this software handles the syntactic structure directly, we can expect results, which cannot be attained by other conventional methods of data analysis. Some results are reported on the analysis of Japanese particles in EDR corpus.

1　はじめに

文のデータ解析においては、文を文字列、単語の並び、構文解析木、意味構造などの各レベルから多面的に解析することが必要である。しかしながらこれまでの研究では、文字列以外のデータを準備することが困難であったこと、構造を有するデータに対する適当な分析法がなかったこと、等の理由により、構文解析木や意味構造の解析はほとんど行われてこなかった。しかし、近年多数のコーパスが利用可能となってきており [1]、なかでも EDR コーパスは意味構造までを含めたすべてのレベルでの情報を備えている [2]。このようなコーパスを利用して、文のデータ解析を実行することにより、統計的なものも含めた新たな文法的知見が得られ言語学の発展に寄与することが期待できる。

我々はこれまで文中にある特定の単語または構文木の中間節点を *viewpoint* として定めれば、構文木のトポロジー的な性質および各節点に付随する通常の属性を使用して、拡張された ID3 法による構文木のデータ解析が可能であることを示してきた [3, 4, 5]。また、この方法を EDR 日本語コーパス中の「が」と「は」を含む文例に適用したところ、これら助詞の使用法に関するいくつかの既知の文法的知見を再確認できたばかりでなく、新たな統計的な知見をも得ることができた [5]。

本報告では以下の各節において、（1）これまでに提案してきた方法論の概説と、今回新たに SYKD システムの開発において導入した枠組みの解説、（2）SYKD システムの概要と実例に沿った動作の説明、（3）本システムの応用により得られた助詞の用法に関する知見、について順に述べる。更に、「読点」を *viewpoint* とした場合に、文例の出典による構造上の違いが見られるかどうかについても調べた。

2　構文木からの知識発見

2－1 これまでの理論的枠組み

助詞「が」と「は」の使用法を対象とした構文木からの特徴抽出を例として、著者らがこれまで採用してきた解析法の概要を以下に述べる。

この場合、使われた助詞の種類自体をクラスラベルとし、これらクラスを識別する何らかの特徴的なパターンを発見することが目的である。

まず、構文木中で当該助詞の存在する節点を *viewpoint* とする。*viewpoint* を含むいくつかの連結した節点の集合を *field-of-view* と呼ぶ。*field-of-view* 内の各節点に付随した属性を対象として、ID3 法による解析を実行すれば、クラス毎の構文上の特徴が得られる。ここで *field-of-view* の設定法が問題となる。

これまでの発表 [5] においては、最初 *viewpoint* 節点のみからなる *field-of-view* を、有効な知識発見の可能性が高い方向へ順次自動的に拡張するという方法を提案し、システムの一部を作成した。

2－2　SYKD システムにおける変更点

システムの開発を進める過程で、各種の実験を行ったところ、異なった方向へ *field-of-view* を展開した場合に、それぞれの *field-of-view* を対象とした解析から、互いに独立した有効な知識の得られる場合が存在した。このことは自動的な *field-of-view* の展開を指向するならば、見落とされる知識の存在を意味する。

そこで、今回の SYKD システムの開発においては、よりインタラクティヴなシステムとする方向性を採った。すなわち、*field-of-view* の展開はすべて利用者の指定により行うこと、また利用者が複数の方向へ *field-of-view* を展開し、そのそれぞれにおいて解析を遂行できるように設計した。

さらにこれまでの解析では、ある *field-of-*

view の解析から特徴的な構文パターンが発見された場合、そのパターンを支持するインスタンスはすでに解析済みと考え、以降の解析の対象事例からは除外していた。

しかし、同一のインスタンス群に対し、より広い *field-of-view* で高精度のパターンが発見される可能性が存在する。SYKD では与えられた *field-of-view* のトポロジーに合致するすべてのインスタンスを、常に解析対象とすることとした。

3　SYKDシステムの概要

3—1　システムの構成

システムは Fig. 1 に示すような構成をとっている。ここで学習用構文木ファイルは解析の対象となる構文木インスタンスを保持するファイルであり、Fig. 2 にインスタンスの例を示す。各インスタンスは、クラスラベル、括弧で表した木構造と各節点毎に埋め込まれた各種の属性値、[ ] で表される *viewpoint* 位置などの情報を持つ。

テスト用の構文木ファイルは、学習されたパターンの正確さを検定するために用い、そのフォーマットは学習用のものと全く同じである。

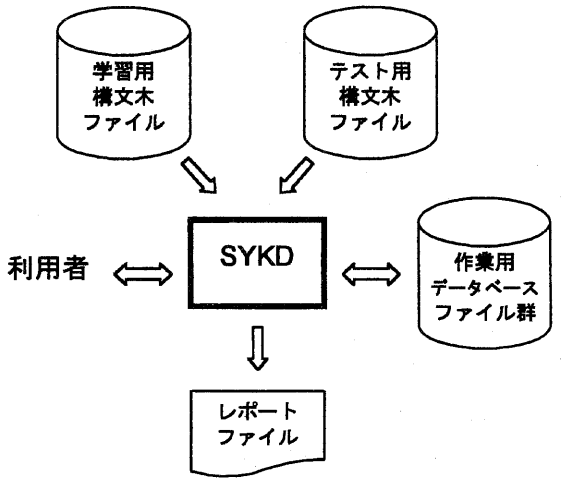

Fig.1 SYKD システム構成

システムはこれらの入力ファイルと学習用のパラメータを指定する場合以外は、タスクと呼ばれる単位を基本として利用者とのインターフェースを取る。ここでタスクとは、利用者にとって特定の *field-of-view* に関連する一連の解析を行う単位である。

利用者の作業は、*field-of-view* の展開による新しいタスクの生成、タスクの選択、およびタスク毎に見出されたパターンやサポートするインスタンス等の結果のブラウジングからなる。

次節以下で、EDR コーパスから抽出した助詞「が」と「は」を含む各 200 文を対象とし、それらの構文上の特徴抽出を行う場合のシステムの動きを例として、SYKD ソフトウェアの機能を解説する。

```
Wa
確率' の意味については，これまでさまざ
まな解釈がなされている.
(T S(t M(T S(t S(t M(T S(t S
(T 記号 " '") (t 名詞 "確率")
(T 記号 "' ")) (T 助詞 "の"))
(t 名詞 "意味")) (T 助詞 "に")
(T 動詞 "つ") (T 語尾 "い")
(T 助詞 "て") [T 助詞 "は"])
(T 記号 ", "))
(t M(T S(t 名詞 "これ")
(T 助詞 "まで")) (t M(T S(t M
(T S(t 形容動詞 "さまざま")
(T 語尾 "な")) (t 名詞 "解釈"))
(T 助詞 "が"))
(t S(t 動詞 "な") (T 語尾 "さ")
(T 助動詞 "れ") (T 助詞 "て")
(T 動詞 "い") (T 語尾 "る")))))
(T 記号 ". "))
```

Fig.2 入力インスタンスの例

3－2　初期化作業

SYKDを起動するとFig. 3に示すメインwindowが現れる。ソフトウェアの各種機能はメニューバーまたはツールバー上のボタンで指定する。

解析を始めるに当たって最初になすべきことは、学習時に有効な構文木パターンか否かを判定するためのパラメータの指定である。このパラメータとしては、有効なパターンであるための最少のサポートインスタンス数と、クラス判別に必要な最低の確信度を［File］メニューから与える。

次に、学習用構文木ファイルをツールバー上の［S］ボタンにより指定し、必要に応じてテスト用構文木ファイルも［TD］ボタンで設定する。

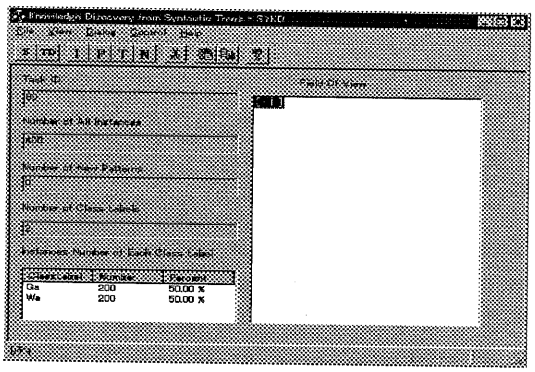

Fig 3　メイン window

学習用のファイルを指定すると、システムは *viewpoint* のみを *field-of-view* とするタスク（ID: 00）を生成し、そのタスクに関する情報がメイン window に表示される。Fig. 3 から、このタスクのインスタンス数が 400 であり、このタスクのもとで有効なパターンの数が 0 であること、クラスラベルには Ga, Wa の２種類が存在して、それぞれのクラスに属するインスタンス数が等しいことが判る。

Window 内で右側の領域には、このタスクに対応する構文木中の *field-of-view* が表示される。Fig. 3 に示す最初の段階では、[-1]で表される *viewpoint* 節点のみが表示されている。

ツールバー上の［I］ボタンのクリックにより、Fig. 4 に示すインスタンス表示の window が現れる。ここでは、右上側のボタン操作により、タスク内のインスタンスをブラウジングすることができる。また構文木表示中の［－］、［＋］ボタンは構文木構造の折り畳みと再展開を指示する機能を持つ。さらに、[AddToReport] ボタンは表示されているインスタンスをレポートファイルに出力する。

3－3　Field-of-view の展開とタスク選択

初期化作業の終了後、*field-of-view* を順次拡大し有効なパターンを探索する作業を行う必要がある。以下、すでに作業が進行した段階を想定した説明を行う。

ツールバー上の［T］ボタンのクリックにより、Fig. 5 のタスク選択画面が現れる。
ここでは、それまでの作業により生成されたタスクの一覧が、展開過程に従い木として表示されている。木中の［－］、［＋］ボタンは表示の折り畳みと展開を指示する。なお、各節に付された ID 自体は意味のあるものではない。

ここで例えば、上から１行目にあるタスク ID（000000）の節点をクリックした後に［OK］すれば、メイン window は Fig. 6 に示したように変化する。このタスクのインスタンス数は、Ga:191, Wa:184 の計 375 であること、この中に識別力の高いパターンが５種類存在することが判る。右側に示す *field-of-view* の表示は、

[-1]の節点で表示されている *viewpoint* 周辺の部分木の構造を示す。ここで、例えば先頭の節点に付された（00 00 -1）等の表示は、

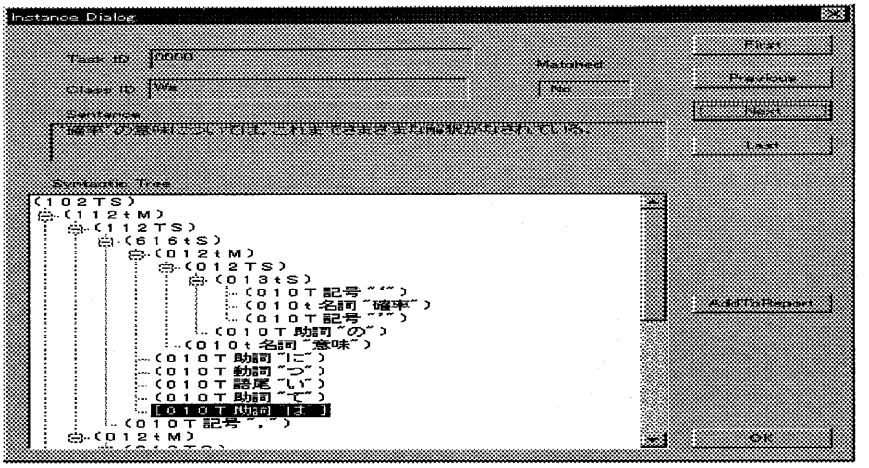

Fig. 4　インスタンス window

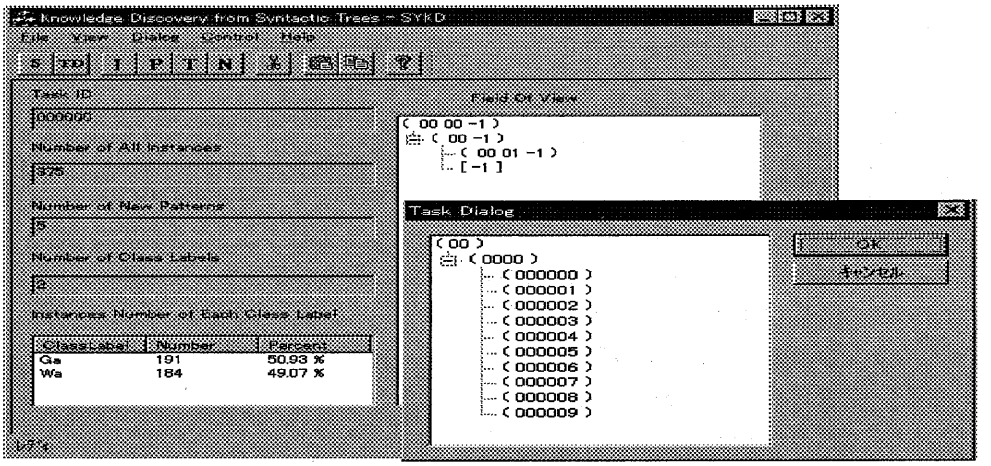

Fig. 6　展開後のメイン window　　Fig. 5　タスク選択 window

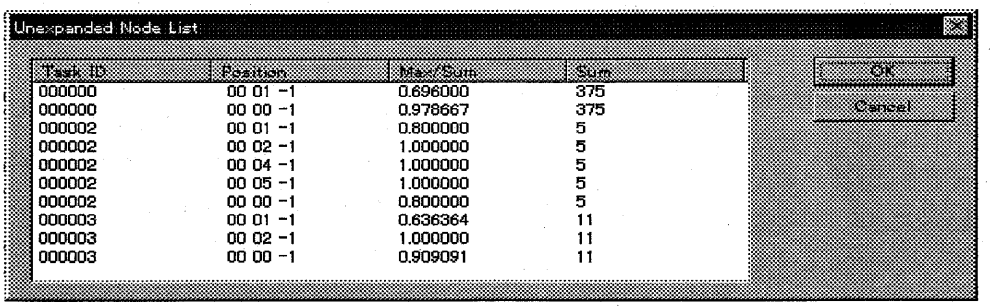

Fig. 7　未展開節点リスト

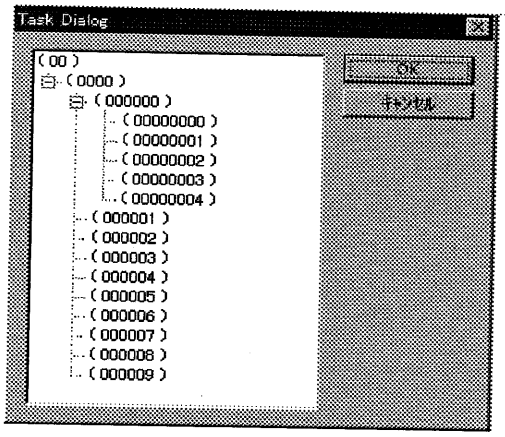

Fig. 8　展開後のタスク選択 window

viewpoint からその節点へ至る path を表示している。すなわち、00 は親節点への移動を、また 01 は１番目の子節点への移動を表し、-1 は path の終結を表す。この例の *field-of-view* 内には、*viewpoint* の他にその親、祖父および兄に当たる節点が含まれていることが判る。

この段階でツールバーの［N］ボタンのクリックにより、Fig. 7 に示すような未展開節点の一覧 window が表示される。この window の最左欄にはタスクの ID が、また最右欄にはタスク内のインスタンス数が表示されている。

Fig. 6 のタスクからは、第１，２行目の２種の展開が可能であることが判る。ここで第２欄の position が *field-of-view* 中で展開されるべき節点を表示している。左記と同様の path 表示から、１行目が *viewpoint* の兄節点の展開を、２行目が祖父節点の展開を指示していることがわかる。

ここで第１行目を選択して［OK］することにより兄節点の展開が行われ、次にタスク選択 window を表示すると、その内容は Fig. 8 のように変化している。なお、兄節点の展開後、新たな未展開節点が Fig. 7 の window に付加されるとともに、選択されなかった祖父節点の展開は一覧上から消去される。

上記の過程を繰り返すことにより、利用者の指定する任意の方向へ *field-of-view* を展開して、解析を進めることができる。

また、タスクを指定してメイン window ツールバーの［ハサミ］ボタンをクリックすれば、当該タスクから展開して生成されたタスク群を消去することができる。この機能を用いると、例えば Fig. 7 の２行目に表示されている祖父節点の展開を実行することができ、結果としてすべての *field-of-view* を対象とした解析が可能となる。

３－４　パターンの表示

Fig. 6 のタスクを選択した状態で、ツールバー［P］ボタンをクリックすれば、Fig. 9 に示すようなパターン表示 window が現れる。ここでは５種のパターンを window 右上側のボタンにより選択して表示することができる。この例のパターンでは、サポートするインスタンスが 17 個存在し、クラス Wa に対する精度が 82%であることが判る。

また、これらのサポートインスタンス群を、[Instances]のクリックにより Fig. 4 のように表示できる。

さらに、[Test]のクリックにより、テスト用構文木ファイルに対して、このパターンに対するサポート数と精度を確認することが可能である。また、[AddToReport]ボタンでレポートファイルにパターンを書き込むこともできる。

Fig. 9 の左下部分は、ID3 法により検出されたパターン本体を表示している。タスクの *field-of-view* に対応した構文木中の各節点毎にパターンの特徴が示されている。この例では、節点に表示される（ ）内に５種の情報が記されている。それらは左から順に、*viewpoint* の方向（0: 親節点、i: i 番目の子節点）、親節

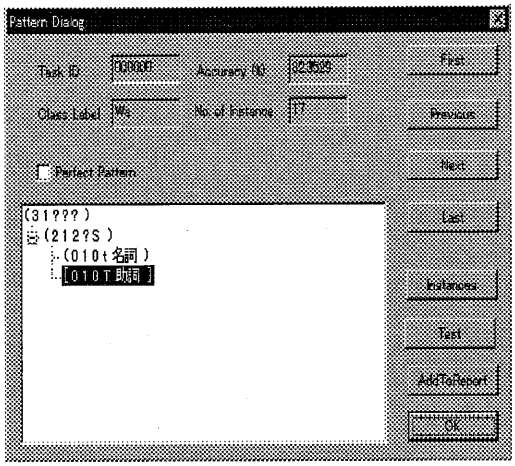

Fig. 9　パターン表示 window

点の数、子節点の数、親節点下でこの節点が主要な意味を担っているか否か（t と T）、品詞や節に関する文法的属性である。?はパターンに含まれない属性である。この例では、EDR コーパス中に記載されている情報のみを用いているが、構文木入力ファイルに記載すればどのような属性でも利用できる。

３－５　入力構文木ファイルの作成

Fig. 2 に示した入力ファイルのフォーマットは、SYKD システムに固有のものである。EDR コーパスから抽出した文に、*viewpoint* やクラスラベルを付加し、このフォーマットに適合させるため、前処理用のシステムを作成した。

４　SYKD システムによる助詞の分析

SYKD システムを使って EDR 日本語コーパスデータに含まれる助詞の周辺構造について分析を行った。

４－１　助詞「が」と「は」の上位節パターン

表１は、ＥＤＲ日本語コーパスデータから助詞「が」と「は」を含む構文木データを、各 1000, 5000, 10000 文例抽出し、これら助詞の上位節でパターンの分析をおこなった結果である。表中最左欄の（　）は、これら助詞の親節の属性を表している。

表１から、「は」と「が」の出現頻度の比率が、文例数によりどのように変化するかが分かる。この結果から、構文構造の特徴を議論するためには、5,000 文例程度が必要と考えられる。

また、それぞれの文例で使われやすい構文の特徴を捉えることができた。例えば、表１の中で「が」と「は」の識別率がもっとも高い構造は（３１３Ｔｓ）である。この構造を持つ文例は、「は」が「が」に比べて 10 倍以上の文例がある。少なくとも、EDR コーパスでは、以下のような構造をもつ場合、「は」が「が」よりも圧倒的に多いことを示している。

４－２　６種の助詞に対する比較分析

EDR コーパスデータから、助詞「が」、「は」、「と」、「て」、「に」、「で」を含む全ての文例を抜き出した。表２は、それぞれの助詞を *viewpoint* として、SYKD システムで見つかった代表的な 18 の上位節パターンについて、システムの検索機能を使って出現頻度を求めたものである。比較を行いやすくするため、10,000 文例を基準とした換算値で出現頻度を表示している。助詞間で大きな差の存在することが分かる。

４－３　助詞「が」と「は」の周辺パターン

viewpoint を基点として *field-of-view* を段階的に広げ、それぞれの *field-of-view* 構造をもつ文例数をクラスラベル毎に分類する作業を SYKD システムでおこなった。データとしては、助詞「が」、「は」をクラスラベルとする 2001 件の文例を用いた。

助詞「が」と「は」を含む文例の比較では *field-of-view* を拡大しても構造的に顕著な差

表１．助詞「が」と「は」における上位節パターン分布の文例数による変化

上位節パターン	1000 / 1000	5000 / 5000	10000 / 10000	識別比
(？１４？ｓ)	13 / 2	46 / 25	78 / 47	1.66
(４１４？ｓ)	7 / 1	30 / 19	49 / 19	2.57
(４１４ｔｓ)	6 / 1	28 / 16	46 / 12	3.83
(２１４？ｓ)	6 / 0	15 / 4	27 / 9	3.00
(？１８？ｓ)	1 / 4	8 / 24	15 / 22	1.47
(？１６？ｓ)	10 / 4	47 / 38	80 / 81	1.01
(？１６ｔｓ)	10 / 3	36 / 38	73 / 56	1.30
(４１６ｔｓ)	5 / 0	15 / 13	33 / 11	3.00
(６１６ｔｓ)	5 / 2	19 / 9	29 / 22	1.32
(５１５？ｓ)	4 / 2	17 / 14	37 / 33	1.12
(５１５ｔｓ)	4 / 1	11 / 5	36 / 17	2.12
(３１５？ｓ)	2 / 3	9 / 52	15 / 75	5.00
(？１３？ｓ)	17 / 82	71 / 379	131 / 716	5.47
(？１３ｔｓ)	10 / 27	35 / 119	78 / 243	3.12
(３１３ｔｓ)	9 / 27	33 / 119	76 / 242	3.18
(？１３Ｔｓ)	7 / 55	36 / 260	53 / 473	8.92
(３１３Ｔｓ)	7 / 52	31 / 247	44 / 442	10.05
(？１２ｔｓ)	129 / 299	720 / 1479	1436 / 2827	1.97

x/y は、(「が」の文例数) / (「は」の文例数) をあらわす。
識別比は 10000 件データもとづき計算した値である。

異は見当たらないように思われる。これらが「が」と「は」についての論争を呼んでいる所以とも 考えられる。しかしながら、一部に特徴的と考えられるパターンが見出された。Fig. 10 と Fig. 11 は、「が」と「は」それぞれに特徴的なパターンを表示した。Fig. 10 の場合、「が」の比率がタスク内で 100%であり、Fig. 11 では「は」の比率が 82.29%である。

４－４　助詞「で」の周辺パターン

すべてが同一のクラスラベルを持つ助詞「で」を含む 7436 文例から、その周辺構造を分類した。Fig. 12 に、7 文例しかない特徴的なパターンを表示した。

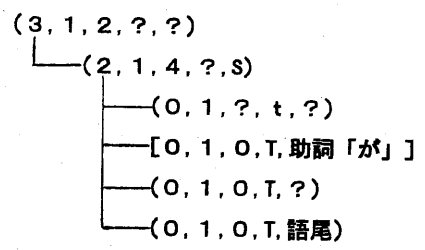

Fig.10　助詞「が」の特徴的な構文パターン

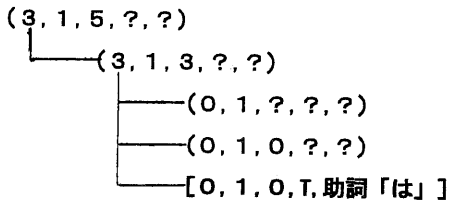

Fig.11　助詞「は」の特徴的な構文パターン

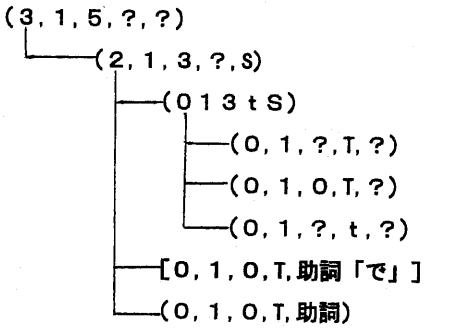

Fig.12　助詞「で」の特徴的な構文パターン

(注) (V,P,C,M,N) の意味は V＝viewpoint の方向 (0：親節点，ｉ：ｉ番目の子節点)，Ｐ＝親節点の数，Ｃ＝子節点の数，Ｍ＝親節点下で、この節点が主要な意味を担っている（Ｔ）か否（ｔ）か，Ｎ＝品詞や節に関する文法的属性，？はﾊﾟﾀｰﾝに含まれない場合

表２．　５種の助詞における上位節のパターン分布

上位節パターン (構文木数)	「が」 (12476)	「は」 (15263)	「と」 (8604)	「て」 (10001)	「に」 (16042)	「で」 (7436)
(？１４？ｓ)	79	54	1885	1982	489	278
(４１４？ｓ)	50	29	24	658	48	28
(４１４ｔｓ)	46	20	21	263	19	22
(２１４？ｓ)	28	9	1846	1216	430	223
(？１８？ｓ)	14	32	84	122	14	23
(？１６？ｓ)	72	91	203	1167	97	78
(？１６ｔｓ)	67	66	159	1099	46	74
(４１６ｔｓ)	29	18	71	648	7	9
(６１６ｔｓ)	28	23	17	9	3	5
(５１５？ｓ)	34	35	113	666	5	11
(５１５ｔｓ)	34	15	95	313	3	8
(３１５？ｓ)	14	97	70	2648	17	305
(？１３？ｓ)	131	708	1474	1174	507	843
(？１３ｔｓ)	75	237	1176	1052	164	317
(３１３ｔｓ)	72	236	88	995	21	108
(？１３Ｔｓ)	56	471	298	122	343	526
(３１３Ｔｓ)	46	445	33	111	61	51
(？１２ｔｓ)	1432	2824	1341	972	1442	2800

各助詞の構文木数を 10,000 として換算した上位節パターンの出現頻度を示す。

５　おわりに

構文木を入力として会話的にデータ解析をおこなうシステム SYKD の概要と助詞の分析に適用した結果を報告した。本システムは、助詞にかぎらず構文木に付随するさまざまな属性や文体をクラスラベルにして分析を行うことができる。今後は本システムを用いて、より多角的な構文木の解析を行っていきたい。

参考文献：

[1] 竹沢寿幸、末松博：音声・テキストコーパスとその構築技術、標準化動向、人工知能学会誌、Vol.10, 168-180 (1995).

[2] EDR: EDR 電子化辞書 1.5 判仕様説明書、EDR TR2-006、日本電子化辞書研究所 (1996).

[3] 雄山真弓、岡田孝、李貴峰：構文解析木を対象とするデータ解析法の研究（１）方法論についての一考察、シンポジウム：人文科学における数量的分析、東京 (1996).

[4] Guifeng Li, Mayumi Oyama and Takashi Okada : Knowledge Discovery from Syntactic Trees, 1996 年度人工知能学会全国大会（第 10 回）, 08-01, 東京 (1996).

[5] 雄山真弓、岡田孝、李貴峰：構文解析木を対象とするデータ解析法の研究 (2) EDR コーパス文例を用いた格助詞の分析、シンポジウム：人文科学における数量的分析、東京 (1997).