関係データベースを用いた構文構造検索ツール
Management Tool of Syntactically Annotated Corpus

吉田 恭祐 † **橋本 泰一** † **徳永 健伸** † **田中 穂積** †
Yoshida Kyosuke Hashimoto Taiichi Tokunaga Takenobu Tanaka Hozumi

1. 背景と目的

近年，自然言語処理の分野では，大規模な言語資源に基づく統計的手法が研究の中心となっている．大規模な言語資源を人手により作成することは，非常に困難である．そのため，言語資源の作成を支援するシステムに関する研究も盛んに行われている．

本論文では，構文構造付きコーパスからユーザが指定した構文構造を含む文を検索するシステムについて述べる．まず，構文付きコーパスを関係データベースに変換する方法について述べ，構築したデータベースの検索手法について述べる．検索にはSQLを用いるが，SQLの知識がなくても簡単に最適なクエリを作成するGUIツールを構築した．ここで，最適なクエリとは，検索時間が最も短いという意味で，検索条件の絞り込みを工夫することにより実現している．

2. コーパスのデータベース化

我々は木構造をデータベース化する方法として，XMLに着目した．吉川らはXML文書をオブジェクト関係データベースを用いて効率良く格納，検索する手法を提案している [1]．この手法では「出現位置」というデータを用いて，ノードが木の構造のどの部分に位置するかをうまく表現している．我々は吉川らの手法を参考に，コーパスのデータベース化を以下のように実現した．

本論文で扱うコーパス‡中の文は，図1のような構文木で表現される．データベース化は，各ノード毎に以下のような情報をテーブルに格納することにより行なう§

sentence_id : その構文木にユニークな名前 (JCO99)
path : ルートからのパス．ただし，ノードの間は!で区切る．(⟨動詞句⟩!⟨動詞句⟩!⟨動詞⟩!⟨動詞語幹⟩)
name : そのノードの名前 (⟨動詞語幹⟩)
id : ノード毎にユニークな数値．先行順で与える．(12)
parent_id : そのノードの親のid (11)
depth : そのノードの深さ．ルートの深さは1．(4)

さらに,「出現位置」を以下の4つのフィールドで表す
l_pos_1，**l_pos_2**，**r_pos_1**，**r_pos_2** (2,6,3,1)

「出現位置」は図1のように各ノードに与える．このような与え方により図2のようなノード間の位置関係が表現できる．包含関係は，⟨Xの子孫⟩ノードが⟨X⟩ノードをルートとする木の中のノードであることを特定する．親戚関係は，⟨i⟩ノードと⟨j⟩ノードの左右に関する位置を特定する．

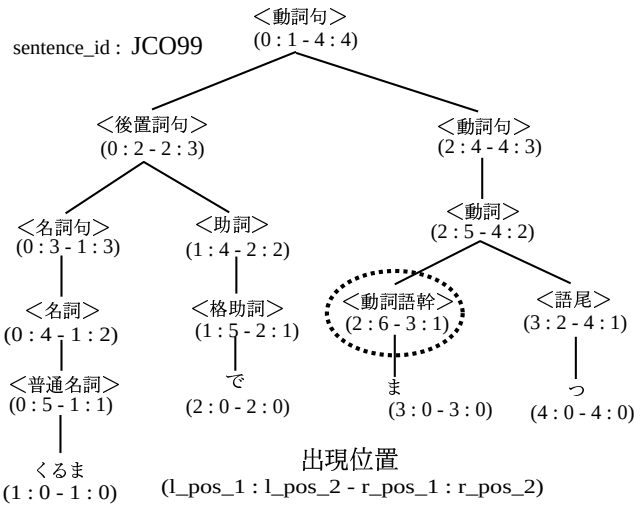

図 1: 構文木の例

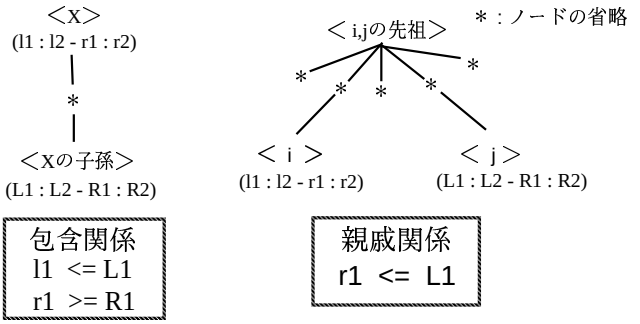

図 2: 包含関係と親戚関係

3. 検索手法

3.1 検索単位と優先度

クエリは図3のように木構造で与える．⟨＊⟩は任意ノード，＊3はそのパスの間に3つ以下のノードの省略を許すことを表す．

クエリは，図3のように，部分木 (これを検索単位と呼ぶことにする) に分解し，それぞれをSQLで検索する．こうするのは，大きな木構造を1度で検索するには，1つのSQL文内で定義するノード数を多くしなくてはならなく，SQL検索は，ノード数が多くなればなるほど検索時間が増加するからである．

検索をするにあたり，各検索単位の検索順序は検索効率に大きく影響する．図3の各検索単位の下に示す数字 (rows) は，実験で用いたデータベース¶中にその検索単

†東京工業大学大学院情報理工学研究科
‡EDR コーパス中の 9,721 文を野呂らの文法に基づいて構文情報を付与したもの [2]
§各フィールドの説明の後に，図1のノード「⟨動詞語幹⟩」に対する具体例をかっこ内に示す．
¶EDR コーパス中の 9,721 文を野呂らの文法に基づいて構文情報

位がいくつあるかを示している．データベース中での数が少ない検索単位から検索すれば，候補の数を早く絞り込むことができるので検索時間は短くなる．各検索単位の検索順序を決めるために以下のように優先度を与え，優先度の大きい検索単位を先に検索する．図 3 の各検索単位の下の実数は優先度を表す．

優先度の与え方

以下の 4 つの情報より算出した数値の和を，各検索単位の優先度とする．

子の数

1,2 文木に比べて，3 分木以上のものはデータベース内に 303 個とかなり少ない．そこで，n 分木には優先度 n を与える．ただし，1 分木と 2 分木の検索候補の差は，2 文木と 3 分木の差ほどはないので，1 分木には優先度 1.5 を与える．

＊の数 (特定ノード率)

特定ノード率とは，検索単位のノードのうち，ノード名がはっきりしている（⟨ ＊ ⟩ ではない）ノードが，全体のどれだけを占めるかを表す割合である．この値を 1.85 倍し優先度として与える．この比率は，実験で用いたデータベースでうまくいくように調整して得た値である．以下に示す係りうけ，＊ (省略ノード数) においても同じように人手で与えた比率を用いているが，これについては 3.2 で述べる．

係り受け

係り受け先条件に出てくるノード数 (ただし形態素は，品詞ノードよりもユニークなので，1.2 と数える) を総ノード数で割ったものに 1.5 をかけたものを優先度に加える．

＊ (省略ノード数)

＊の引き数 (指定されてないものは 1) の総和に 0.1 をかけたものを優先度から引く．つまり，省略ノードが多いほど優先度は低くなる．

3.2 優先度の精度

図 3 に各検索単位の優先度を示しているが，この場合はデータベース中での数が少ないものほど優先度が高くなっている．これは，うまくいった例であるが，先ほど述べたように優先度を与える際に人手で調整した比率を用いているので，この比率は常に最良の結果を与えるわけではない．3 分木を含まず，任意ノード (⟨ ＊ ⟩) の多い検索単位のみで構成されたクエリの場合，うまくいかない可能性が高くなる．より精度の高い優先度の与え方は，今後の課題である．

4. GUI ツールと今後の予定

クエリの木構造は，図 4 のような GUI を用いて作成し，検索結果は図 5 のように表示される．現段階では，クエリを作成し，検索を行ない，クエリとマッチした部分を赤く表示して出力する機能しかないが，将来的には，コーパスの修正機能，コーパスの一貫性を管理する機能などをつけ加える予定である．

を付与したもの [2] のうちの 1,000 文をデータベース化したもの

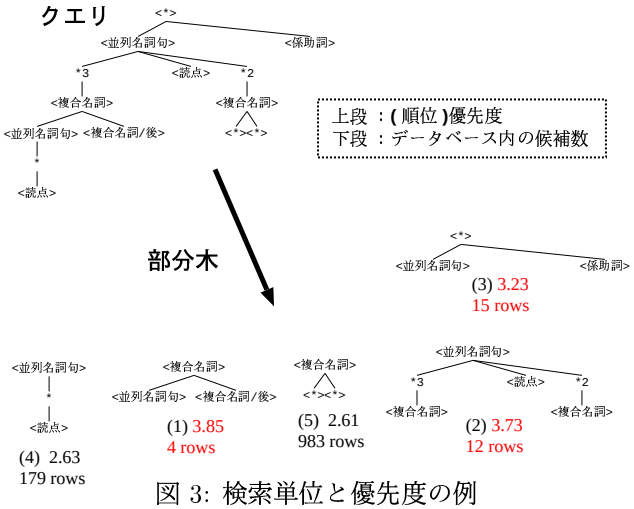

図 3: 検索単位と優先度の例

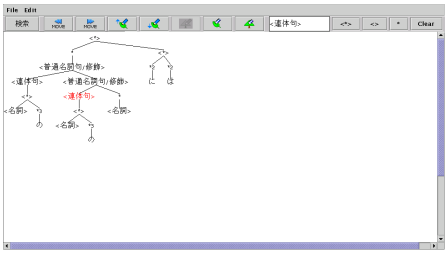

図 4: GUI ツール

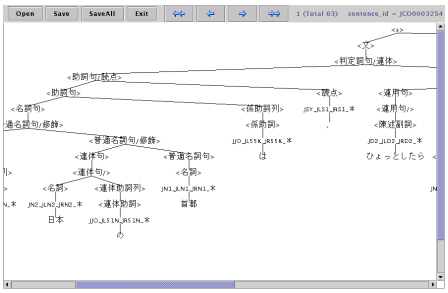

図 5: 検索結果の表示

また，クエリと類似した部分木を持つ文を検索する「あいまい検索」のような検索オプションをつけ加え，より柔軟な検索システムの構築をする予定である．また，優先度の与え方の修正も重要な課題である．

参考文献

[1] 吉川正俊，志村壮是，植村俊亮．オブジェクト関係データベースを用いた XML 文書の格納と検索．情報処理学会論文誌 Vol.40 No.0 *1999*

[2] 野呂智哉，白井清昭，徳永健伸，田中穂積．大規模日本語文法の開発ー事例研究．情報処理学会研究報告 NL-150 pp.149-156 *2002*