依存構造に基づく英語用例検索システム

加藤 芳秀† 松原 茂樹†† 稲垣 康善†††

† 名古屋大学大学院国際開発研究科
†† 名古屋大学情報連携基盤センター
††† 愛知県立大学情報科学部
E-Mail: yosihide@gsid.nagoya-u.ac.jp

1 はじめに

近年，大規模コーパスの重要性はますます高まっており，言語現象の調査，外国語学習，自然言語処理システムの開発など様々な場面で言語資源として活用されている．コーパスを効果的に活用するために，様々なコーパス検索システムが提案されている．

多くのシステムでは，いくつかのキーワードをクエリとして受け取り，それらを含むような用例を検出する．直感的で，ユーザビリティが高いという利点があるが，構文構造などの言語的構造を活用した検索は実現されていない．

これに対して，難波らは，構文木に基づく 2 単語間の距離を定義し，クエリ中のキーワードが，ある距離以下で出現する文のみを検出する手法を提案している [4]．この手法は，従来の手法に比べれば言語的な情報を活用しているといえるものの，構文木の情報を距離に変換した時点で，構文木のもつ階層的な情報は失われてしまう．

一方，構文構造の階層的な情報を用いたコーパス検索システムとして Gsearch[2] や Linguist's Search Engine (LSE)[5] といったシステムが提案されている．これらのシステムでは，句構造パターンを活用してコーパスを検索する．Gsearch では，入力として，句構造パターンを生成するための文法と検出したい句構造パターンを受け取る．システムは入力された文法を用いてコーパス中の文を構文解析し，与えられた句構造パターンを持つ文を検出する．LSE では，ユーザはまず，検出したい文の一例を入力する．システムは，入力された例文を統計的構文解析により解析し，その解析結果をユーザに提示する．ユーザは提示された構文解析結果を編集し，構造的なパターンを作成する．最終的にシステムは，編集により得られたパターンを用いてコーパス内を検索し，パターンにマッチする句構造を持つ文を検出する．これらのシステムにより，構文情報を利用したコーパス検索が実現できる．しかし，キーワードに基づく検索システムのような，簡単で，直感的な検索を実現しているとは言いがたい．

そこで本稿では，キーワードの系列から構造的なクエリを自動生成するコーパス検索システムを提案する．本システムは，依存構造に基づくコーパス検索を実現する．まず，ユーザからのクエリとしてキーワードの系列を受け取り，受け取ったクエリに対して，クエリ中のキーワードを含むような依存構造パターンを，依存構造付きコーパスを参照しながら自動生成し，それにマッチする依存構造を持つ文を検出する．依存構造を活用したコーパス検索が実現できる一方で，ユーザは Gsearch にように事前に文法を設計する必要もなければ，LSE のように構造的なクエリを編集する必要もなく，キーワードに基づくシステムと同じように簡単にシステムを利用できる．

2 依存構造に基づく用例文検索

本節では，依存構造に基づく用例文検索システムを提案する．

提案システムでは，検索対象のコーパスには，依存構造解析器などにより予め依存構造が付与されていることを前提とする．ユーザのクエリは，キーワード（単語，または品詞[1] ）の系列である．システムは，ユーザのクエリを受け取ると，コーパス中の各文に対して以下で述べる依存構造パターン生成アルゴリズムを適用し，依存構造パターンの生成を試みる．アルゴリズムによりパターンが生成された文は，システムの検索結果としてユーザに提示される．生成される依存構造パターンは，文の依存構造にマッチするパターンのみであるため，検索結果中の文はすべて，生成されたパターンにマッチする依存構造をもつ文であることが保証される．

2.1 依存構造パターン生成アルゴリズム

本節では，キーワードの系列，文，及び文の依存構造から依存構造パターンを生成するアルゴリズムを提案する．本アルゴリズムは，入力されたキーワードをその順番で含む依存構造パターンを生成する．文に付与された依存構造を参照しながら，それにマッチするパターンのみを生成する．

[1] 単語や品詞の文字列パターンを正規表現により記述することもできる．

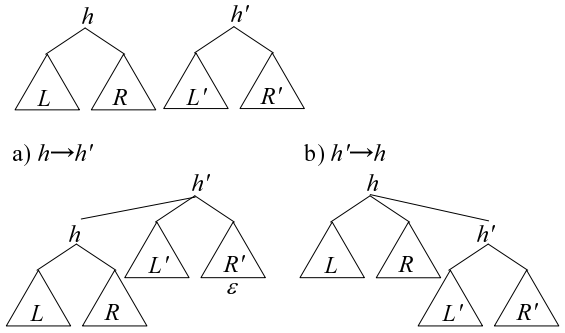

図 1: 結合操作

入力として，以下を受け取り，依存構造パターンの集合を出力する．

クエリ $q_1 \cdots q_m$ ($q_1, \ldots, q_m$ はキーワード)

文 $s = w_1 \cdots w_n$ ($w_1, \ldots, w_n$ は単語と品詞のペア)

依存構造（= 依存関係の集合） D

ここで，D は，文 s に出現する単語間の依存関係の集合である．w_i が w_j に依存するとき，その単語位置の対 (i, j) は D の要素である．以下では，(i, j) を $i \to j$ と書くことにする．

依存構造パターンは 3 項組 $d = (h, L, R)$ である．h は単語位置であり，これを d の主辞と呼ぶ．L, および R は，依存構造パターンのリストである．L 中の依存構造パターンの主辞が左から h に依存することを意味する．R の場合は，右から依存することを意味する．

本アルゴリズムは，クエリ $q_1 \cdots q_m$ に対して以下の操作をボトムアップに適用することにより，クエリ中のキーワードから構成される依存構造パターンを生成する．クエリに対して依存構造パターン d が生成されるとき，s の依存構造が d とマッチすることを意味する．

初期化 各 $q_i (1 \leq i \leq m)$, $w_j (1 \leq j \leq n)$ に対して，$q_i = w_j$ ならば，q_i に対する依存構造パターンとして $(j, \varepsilon, \varepsilon)$ を生成する．

結合操作 $d = (h, L, R)$, および $d' = (h', L', R')$ をそれぞれ，$q_i \cdots q_j$，および $q_{j+1} \cdots q_k$ に対する依存構造パターンとする．$h \to h'$ かつ $R' = \varepsilon$ ならば，$q_i \cdots q_j q_{j+1} \cdots q_k$ に対して，依存構造パターン (h', dL', R') を生成する（図 1a）参照）．$h' \to h$ ならば，依存構造パターン (h, L, Rd') を生成する（図 1b）参照）．

結合操作により，クエリ中の各キーワードが，クエリ中の別のキーワードに直接依存するようなすべてのパターンを生成できる．これらの依存構造パターンを持つ文，すなわちパターンが生成された文を検索結果とすることにより，クエリ中のキーワードが依存関係を持つ形で現れるような文のみをユーザに提示できる．

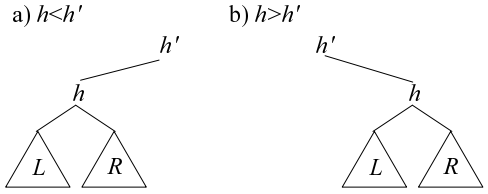

図 2: 補間操作

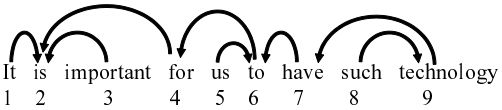

図 3: It is important fo us to have such technology に対する依存関係

クエリ中のキーワードが，クエリ中の他のキーワードと直接依存関係をもたないような文を検出したい場合も考えられる．しかし，上で述べた結合操作をいくら適用しても，そのような依存構造パターンは生成できない．そこで，このようなクエリを頑健に処理するために，次の補完操作を導入する．

補完操作 d を $q_i \cdots q_j$ に対する依存構造パターンとし，その主辞を h とする．このとき，$h \to h'$ である h に関して，$h < h'$ ならば，$q_i \cdots q_j$ に対して，依存構造パターン (h'^*, d, ε) を生成する（図 2a）参照）．$h > h'$ ならば，$q_i \cdots q_j$ に対して，依存構造パターン (h'^*, ε, d) を生成する（図 2b）参照）．

主辞 h' に付与された記号 $*$ は，h' が補完操作により導入されたものであることを意味する．

この操作により，クエリ中の単語間に直接の依存関係がないような依存構造パターンを生成することができる．本手法では，補間操作に対してコストを与える．補完操作により生成された依存構造パターンのコストを，その生成に用いられた補完操作の回数と定義する．コストに対する閾値を設定し，閾値以下の依存構造パターンを生成することにより，補完操作の少ない依存構造パターンのみ生成できる．

2.1.1 依存構造パターン生成例（結合操作の例）

依存構造パターン生成の例として，クエリ

$$\text{it is for to} \tag{1}$$

と文

$$\text{It is important for us to have such technology} \tag{2}$$

に対する生成を考える．依存関係は，図 3 のように与えられるものとする．

まず，クエリの 1 番目の単語 “it” は，文 (2) の 1 番目の単語にマッチするので，依存構造パターン

$$(1, \varepsilon, \varepsilon) \tag{3}$$

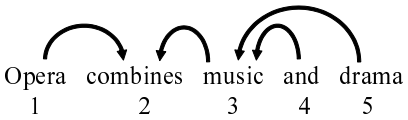

図 4: Opera combines music and drama に対する依存関係

が生成される．同様にして，“is”, “for”, 及び “to” に対してそれぞれ，

$$(2, \varepsilon, \varepsilon) \tag{4}$$

$$(4, \varepsilon, \varepsilon) \tag{5}$$

$$(6, \varepsilon, \varepsilon) \tag{6}$$

が生成される．

隣接する依存構造パターン (3) と (4) の主辞に注目すると，依存関係 $1 \to 2$ が成り立つ．したがって，“it is” に対して，依存構造パターン

$$(2, (1, \varepsilon, \varepsilon), \varepsilon) \tag{7}$$

が生成される．同様にして，“for to” に対して，

$$(4, \varepsilon, (6, \varepsilon, \varepsilon)) \tag{8}$$

が生成される．さらに，$4 \to 2$ であるので，依存構造パターン (7) と (8) が結合され，“it is for to” に対する依存構造パターンとして，

$$(2, (1, \varepsilon, \varepsilon), (4, \varepsilon, (6, \varepsilon, \varepsilon))) \tag{9}$$

が生成される．したがって，文 (2) には，依存構造パターン (9) の形で，“it is for to” が出現していることがわかる．一方，依存関係が成り立つ形でクエリ中のキーワードが出現していない文（例えば，“It is clear whether support for the proposal will be broad enough to a serious challenge”）に対しては，結合操作によって依存構造パターンは生成されない．

2.1.2 依存構造パターン生成例（補完操作の例）

依存構造パターン生成の別の例として，クエリ

$$\text{combines and} \tag{10}$$

と文

$$\text{Opera combines music and drama} \tag{11}$$

に対する生成を考える．依存関係は，図 4 のように与えられるものとする．

“combines” と “and” に対しては，初期化により，

$$(2, \varepsilon, \varepsilon) \tag{12}$$

$$(4, \varepsilon, \varepsilon) \tag{13}$$

表 1: クエリ “it is for to” に対する検索精度

コストの閾値	精度	再現率
0	100.0% (17/17)	81.0% (17/21)
1	90.5% (19/21)	90.5% (19/21)
2	70.0% (21/30)	100.0% (21/21)
3	60.0% (21/35)	100.0% (21/21)
単純な方法	27.3% (21/77)	100.0% (21/21)

が生成される．依存構造パターン (12) と (13) は，隣接するが，$2 \to 4$ でも $4 \to 2$ でもないので，結合操作によっては，“combines and” に対する依存構造パターンは生成されない．

依存構造パターン (13) に対して補完操作を適用すると，“and” に対して，

$$(3^*, \varepsilon, (4, \varepsilon, \varepsilon)) \tag{14}$$

が生成される．$3 \to 2$ であるので，(12) と (13) に対して結合操作を適用すると，

$$(2, \varepsilon, (3^*, \varepsilon, (4, \varepsilon, \varepsilon))) \tag{15}$$

が生成される．

このように，補完操作を適用すれば，クエリに出現しない単語を介して間接的に依存関係が成り立つような依存構造パターンも生成できる．一方で，生成された依存構造パターンにはコストが割り当てられるため，コストに対して閾値を設定し，閾値以下のパターンのみ生成すれば，クエリに出現しない単語を多く含むような依存構造パターンの生成を回避できる．

3 評価

本節では，いくつかのクエリに対するシステムの具体的な動作について考察する．検索対象コーパスとして，英語新聞記事に句構造を付与したコーパスである Penn Treebank [3] を使用した．コーパス中の句構造は，文献 [1] の方法に従って依存構造に変換した．

クエリに対する正解データを人手により与え，検索の精度，及び再現率を以下の定義に従い評価した．

$$\text{精度} = \frac{\text{検出された正解の数}}{\text{検出結果の数}}$$

$$\text{再現率} = \frac{\text{検出された正解の数}}{\text{正解の数}}$$

コストの閾値を 0〜3 に設定し，閾値以下のコストをもつ依存構造パターンを生成しないようにして，それぞれの場合についてその検索精度を測定した．参考として，クエリ中のキーワードを含む文を取り出す単純な方法の精度も測定した．

表 2: クエリ “combine and” に対する検索精度

コストの閾値	精度	再現率
0	0.0% (0/ 2)	0.0% (0/11)
1	55.0% (11/20)	100.0% (11/11)
2	42.3% (11/26)	100.0% (11/11)
3	32.4% (11/24)	100.0% (11/11)
単純な方法	24.5% (11/45)	100.0% (11/11)

表 3: クエリ “have 前置詞 mind” に対する検索精度

コストの閾値	精度	再現率
0	100.0% (10/10)	90.9% (10/11)
1	73.3% (11/15)	100.0% (11/11)
2	61.1% (11/18)	100.0% (11/11)
3	61.1% (11/18)	100.0% (11/11)
単純な方法	50.0% (11/22)	100.0% (11/11)

動作例 1 : “it is for to” に対する検索

クエリ “it is for to” に対する精度を表 1 に示す. コストの閾値が 0 のとき，精度は 100%，再現率は 81.0%と良好な結果が得られた．コスト 0 の依存構造パターンはすべて，2.1.1 節の例と同様の形のパターンであった．コスト 1 以上の依存構造パターンにマッチした正解データを調べたところ，それらに対する依存構造では，“for” は be 動詞ではなく，補語に依存していた．このことは，コーパス中の依存構造に揺れがあると検索精度が低下することを意味する．一方で，補完操作を導入することにより，これらの正解も検出できることが確認できた.

動作例 2 : “combine and” に対する検索

次に，クエリ “combine and” に対する検索について考える．検索精度を表 2 に示す．コスト 0 の依存構造パターンを用いても正解を検出できなかったが，コスト 1 の依存構造パターンにより，正解がすべて検出されている．この結果は，ある種のクエリについては，結合操作のみでは頑健な検索ができないことを意味する．検索の精度を見ると，コストの閾値が 1 のとき精度が最大となったが，その値は 55.0%と低い．コスト 1 以下の依存構造パターンを調べたところ，合計 5 つのパターンが生成されていた．依存構造パターンの種類で検索結果を分類すると，正解データは，2 つの依存構造パターンのいずれかにマッチしており，その他のパターンとはマッチしていなかった．したがって，マッチした依存構造パターンによって検索結果を分類することにより，より精度の高い検索が実現できると考えられる.

動作例 3 : “have 前置詞 mind” に対する検索

クエリ “have 前置詞 mind” に対する検索について考える．検索精度を表 3 に示す.

このクエリに対しても高い精度で用例文を検出できた．前置詞にマッチした 11 個の単語のうち，10 個は “in” であり，1 個は，“on” であった．キーワードに品詞を利用することにより，それが具体的にはどのような単語であるかを調べることができる.

4 おわりに

本稿では，キーワードの系列から依存構造パターンを自動生成し，生成されたパターンとマッチする依存構造をもつ文を検出するコーパス検索システムを提案した.

いくつかのクエリについてその動作を確認したが，今後の課題として，提案システムの定量的な評価が挙げられる．十分な量のテストセットを作成し検索実験を実施するとともに，ユーザビリティの観点から提案システムを評価したい.

謝辞

本研究の一部は，名古屋大学学術振興基金，ならびに文部科学省科学研究費若手研究 (B)(課題番号: 17700145) の助成を受けています.

参考文献

[1] Collins, M.: *Head-Driven Statistical Models for Natural Language Parsing*, PhD Dissertation, University of Pennsylvania, 1999.

[2] Corley, S., Corley, M., Keller, F., Crocker, M. and Trewin, S.: Finding Syntactic Structure in Unparsed Corpora: The Gsearch Corpus Query System, *Computers and the Humanities*, 35:2, 81–94, 2001.

[3] Marcus, M. P. and Santorini, B. and Marcinkiewicz, M. A.: Building a Large Annotated Corpus of English: the Penn Treebank, *Computational Linguistics*, Vol.19, No.2, pp.310–330, 1993.

[4] 難波 英嗣，森下 智史，相沢 輝昭：論文データベースからのイディオム用例検索，情報処理学会研究報告，NL-170，pp.53–59, 2005.

[5] Resnik, P. and Elkiss, A.: The Linguist's Search Engine: An Overview, in *Proceedings of the ACL Interactive Poseter and Demonstration Sessions*, pp.33–36, 2005.