トップ k 頻出係り受けパターンのリアルタイム検索
Searching Top-k Most Frequent Dependency Patterns

宅間大介[*1]
Daisuke Takuma

[*1] 日本アイ・ビー・エム株式会社 東京基礎研究所
IBM Japan Tokyo Research Laboratory

Phrase information such as subject-verb-object plays an important role in text analysis. The phrase extraction is usually implemented as tree pattern matching on parse trees of text data. This paper proposes a new approach to pattern creation that dynamically enumerates the top-k most frequent expanded patterns consisting of a given pattern and one additional word. In this approach, users or programs can incrementally create a pattern by adding words from those which are frequent in the context of a sub-pattern.

1. イントロダクション

1.1 木構造パターンの有用性と作成時の問題

企業の持つテキストデータが増加する中で，テキストデータからの情報抽出技術は，問題分析や FAQ 抽出等，実務に役立つレベルまで発展した．実務で要求される高精度な情報抽出では，テキスト中の単語だけでなく，係り受けパターン(以後「パターン」)が重要になる．一般的に，パターンによる情報抽出は文や文書内の単語共起よりも，precision が高いと考えられており，化合物の相互作用の抽出における比較例[Ding 2002]等の裏付けもある．本稿において，パターンは，図 1に示すような単語でラベル付けされたノードからなる木構造を指すものとする．

パターンによる情報抽出は，高精度である一方，抽出対象のテキスト中での表現のされ方が分からないと有用なパターン作成ができない問題がある．これに対し，「製品名－動詞」といった一般化されたパターンにマッチする表現を全て抽出する方法もあるが[Nasukawa 2001]，パターンのノード数に限界がある．また，tree mining による頻出木構造の列挙も考えられるが，頻出パターンの大半は「お客様－仰る」，「商品－購入する」といった自明な表現であり，有用性の評価が別途必要となる．

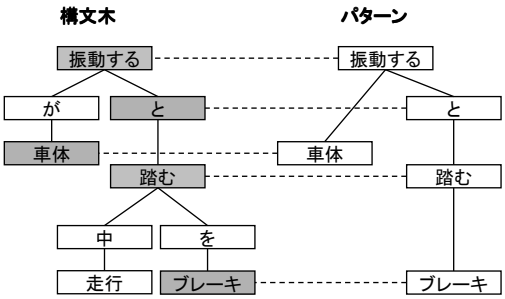

図 1 構文木とそれにマッチするパターン

1.2 パターン拡張によるアプローチ

前述のパターン作成の問題に対する本研究のアプローチは，パターンの入力に対し，そのパターンの指定箇所に一つノードを追加した形の拡張パターンで，テキストデータ中に頻出するものトップ k 個を高速に列挙する方法を取る(図 2参照)．処理の高速化は，テキストデータをパースした構文木に対する上記のトップ k 処理専用の索引を作成することで実現する．ここで，本研究の貢献は以下の二点であると考える．

- トップ k 処理を用いたパターン拡張によるパターン作成方法の提案
- 上記に特化した索引とアルゴリズムの提案

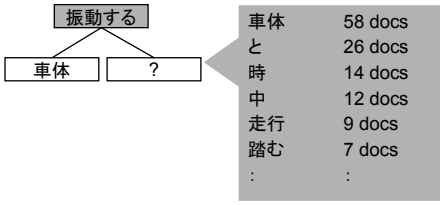

図 2 拡張パターンのトップ k

2. 分析とパターン作成のイメージ

ここでは，提案手法が実現しようとしている分析を実例とともに紹介する．使用しているテキストデータは国土交通省が公開している自動車のリコール・不具合情報である(表 1参照)．

表 1 シナリオ用サンプルデータ

ソース	国土交通省
データ名称	自動車のリコール・不具合情報
文書数	18338
構文木サイズ	77MB
索引サイズ	26MB

自動車の特定のパーツに関する問題の検出のためのパターンを作成する想定で，「車体の問題」に関するシナリオを説明する．まず初期ワードとして「車体」を与え，「車体」の係り先(パターン木の親ノード)の頻出ワードを検索する．図 3の一番上の吹き出しには提案手法で取得した単語が示されており，図中の「341 docs」は「車体」を含む文書数，「振動する 58 docs」は「車体－振動する」というパターンを含む文書数を表す．頻出係り先から「振動する」を選択してパターン「車体－振動する」を作成し，「振動する」の係り元の頻出ワードを，再び検索する．この操作

連絡先：宅間大介，日本アイ・ビー・エム株式会社，〒242-8502 神奈川県大和市下鶴間 1623-14，046-215-4415，ta9ma@jp.ibm.com

は，用言の係り元に状況や原因を表す表現が多く存在するという経験に基づく．この操作を繰り返すことで「ブレーキを踏むと車体が振動する」という事象に対するパターン(図中最下)を得る．

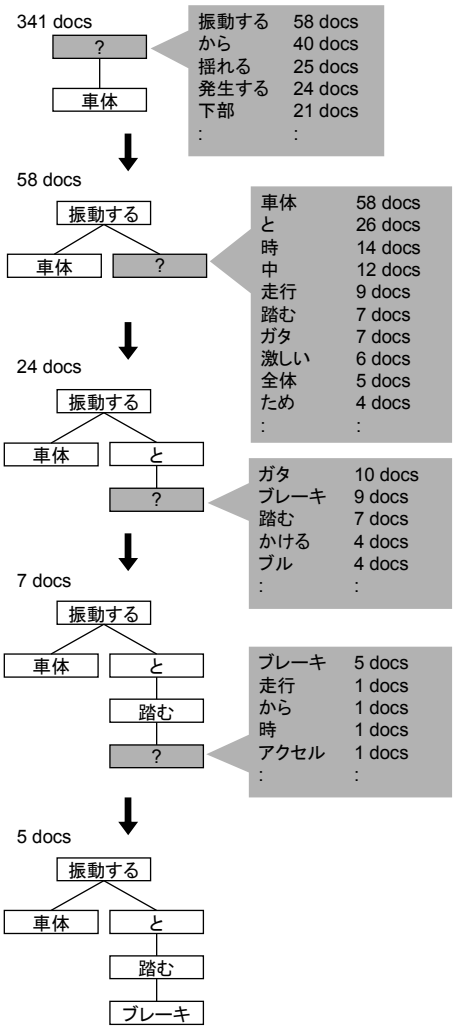

図 3 パターン作成シナリオ

このシナリオで重要なことは，パーツ情報「車体」のみから問題発見，状況分析も含めたパターン作成が実現している点である．また，検索対象中で高頻度な単語を含むようにパターンを拡張することで，単語数が多くても，比較的マッチ箇所の多いパターンを作成できている(表 2参照)．更に，トップ k の結果に検索対象中で同種の意味を持つ単語が多く含まれるため，実際には，「振動する」の他に「揺れる」を，「と」の他に「時」，「際」を，「踏む」の他に「かける」を，「ブレーキ」の他に「ブレーキペダル」を追加して，15 文書をヒットさせることも可能であった．

表 2 検索例におけるパターン中の単語数と該当文書数

問題の現象	単語数	文書数
ブレーキを踏むと車体が振動する	5	5
ABS警告灯が点灯しブレーキが効かなくなる	7	4
走行中アクセルを踏んでも加速しない	6	11
走行中エンジンが停止して再始動できなくなる	7	20

3. 関連研究

3.1 構文木を用いた情報抽出

構文木からの動的な情報抽出としては，固有表現のテーブルを作成する[Sekine 2006]の研究が挙げられる．これは，本研究と同様に動的に取得した文書集合に注目しているが，本研究が数秒程度のインターラクションを繰り返していくのに対し，完全自動で数分程度の時間をかけて情報抽出を行う点，言語処理を動的に使用する点にアプローチの差異が見られる．

3.2 木構造の検索

木構造の検索は XML の分野で多くの研究がある．先祖-子孫ノードの判定に preorder/postorder を用いる手法が知られており[Bellahsene 2003]，(木の ID, preorder, postorder, level)等の形式で情報を保持するのが一般的である[Zhang 2001]，[Al-Khalifa 2002]．ノードの中から先祖-子孫の関係を調べる演算の効率化の研究が盛んだが[Bruno 2002]，[Chien 2002]，本研究では演算のケースが限られており，索引の更新も仮定しないため，そこに困難は無い．

3.3 頻出木構造のマイニング

本研究のパターン拡張に相当する木構造の拡張は，tree mining [Zaki 1999]のアルゴリズムに現れる．tree mining が出現頻度の閾値処理で全テキストデータ中の頻出木を探すのに対し，提案手法のアプローチでは，人手を介して木を選別しながら検索のコンテキストにおける頻出木を探していることになる．

4. トップ k 処理

4.1 定式化

構文木へのパターンマッチは図 1に示すように，ノード間のギャップを許す部分木の包含で定義する．木構造を，ノード集合 N, 枝の集合 $B = \{(n, m) | n \text{ は } m \text{ の親ノード}\}$，ノードからラベル(単語)への写像 L の 3 つ組(N, B, L)で表す．パターンを，木構造に枝からギャップ(正整数)への関数 g を追加した 4 つ組 $p = (N_p, B_p, L_p, g_p)$で表す時，「構文木 $t = (N_t, B_t, L_t)$ に p がマッチする」とは，$N_p = \{n_1,\cdots,n_l\}$の各 n_i に対し，N_tのノード m_iで $L_p(n_i) = L_t(m_i)$となるものが存在し，$\forall(n_i, n_j) \in B_p$ $\{m_i$ が m_j *の先祖ノード, かつ,* $level(m_j)\text{-}level(m_i) \leqq g_p((n_i, n_j))\}$ であることと定義する．

トップ k 処理のクエリ $q = (N, B, L, g, n_p, g_p)$ は，上記のパターン(N, B, L, g) に，ピボット n_pと呼ぶパターン拡張の基点となるノード，ギャップ g_pと呼ぶ追加ノードとピボットの間の最大 level 差を付加したものとする(図 4参照)．ルートに親ノードを追加する拡張はこれでは表現できないが，親ノード側への拡張も本質的な処理は変わらないため割愛する．

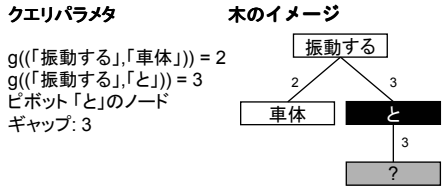

図 4 トップ k クエリの例

次にクエリの処理結果としてのパターンの拡張を定義する．$q=(N, B, L, g, n_p, g_p)$ をクエリ，w を単語とする．パターン p が $p=(N', B', L', g')$, $N'=N \cup \{n_{new}\}$, $B'=B \cup \{(n_p, n_{new})\}$, L':$N' \rightarrow$ *単*

語集合, $L'|_N=L$, $L'(n_{new})=w$, g': $B'\rightarrow$正整数, $g'|_N=g$, $g'((n_p, n_{new}))=g_p$ と書ける時, p を「q の w-拡張パターン」と呼ぶことにする.

以上に基づいて,「クエリ q による構文木集合 T 上のトップ k パターンの検索」を q の w-拡張パターンがマッチする T の構文木の数が k 番目以内に多い単語 w を取得する処理と定める.

4.2 索引

図 5にトップ k パターン検索で使用する索引の論理構造を示す. ランクファイルは入力となる構文木の中に出現した全単語について, 累積出現頻度, 文書頻度(出現した構文木の数), 単語を文書頻度降順でソートして保持する. Quad ファイルは, 各単語の出現について, 文書 ID(構文木 ID), 構文木上の preorder, postorder, level の 4 つ組(以下 quad と記す)を文書頻度降順でソートして保持する. 累積頻度は Quad ファイル上の特定の単語の quad へのポインタである. quad 配列を圧縮する場合は, 圧縮した箇所へのポインタとして良い.

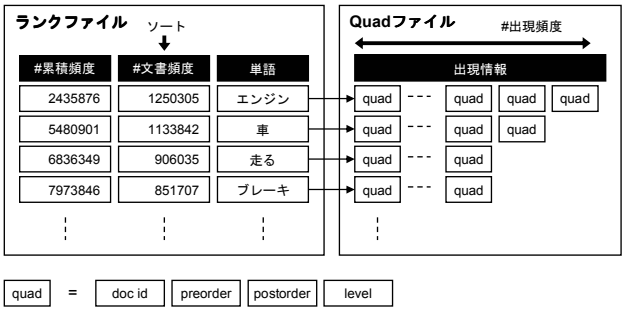

図 5 トップ k パターン検索のための索引

4.3 トップ k パターン検索

トップ k パターン検索は, クエリ q のパターン p の出現箇所を求める検索ステップと, p の各出現のピボット n_p からギャップ g_p 以内の子孫ノードを集計するマイニングステップの二段階で行う.

検索ステップでは, p の各ノードの出現情報を quad 配列で取得し, 非ピボットのリーフノード n とその親ノード m について, m の quad を, 指定ギャップ以内に n の quad を子孫に持つものに絞り込む. m の quad u を残すかどうかの具体的な判定方法は, $\exists v$: n の quad $\{doc\_id(u) = doc\_id(v), preorder(u) < preorder(v), postorder(v) > postorder(u), level(v)\text{-}level(u) \leqq g(m, n)\}$ となる. この絞り込みとリーフノード n を p から削除する操作を, p から非ピボットのリーフノードがなくなるまで繰り返す(図 6の<1>). その後, 非ピボットのルートノードとその子ノードについて, 指定ギャップ以内に存在する quad の絞り込み, ルートノードの削除を非ピボットのノードが無くなるまで繰り返す(図 6の<2>).

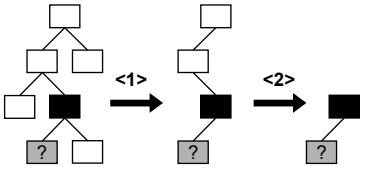

図 6 検索ステップでのクエリのパターン(黒はピボット)

マイニングステップでは, 索引の Quad ファイルから文書頻度の高い順に単語 w の quad 配列 u_w を取得し, 検索ステップで絞り込んだ quad u_s と指定ギャップ g_p 以内の先祖子孫関係にあるものに絞り込み, その中のユニーク文書 ID を調べることで w-拡張パターンの文書頻度を計算する. 拡張パターンの文書頻度トップ k 個を保持しながらこの処理を回し, 次に読み込む単語 w の文書頻度が k 番目の拡張パターンの文書頻度以下になったところで処理を打ち切る. 打ち切りできない場合は, Quad ファイルを全て読み込むことになるが, 頻繁に使用される高頻度なパターンを用いたトップ k パターン検索は, キャッシュを利用して高速に処理できるメリットがある. また文書頻度でのソートにより, 処理はファイルごとに 1 回の連続アクセスで終わる.

5. パフォーマンススタディー

大規模なデータにおけるトップ k パターン検索のパフォーマンスを測定するために, MEDLINE([MEDLINE 2008])の論文アブストラクトをテキストデータとして用いた. クエリは図 7に示す 2 種類を使用した. クエリ 1 は 1 つの単語から係り先を探すクエリで, 最もよく使うものである. クエリ 2 はノード数 3 のパターンの拡張を行うもので, より複雑な問い合わせに対応する. 各ノードの単語には高頻度語からランダムに選択したものを用いた.

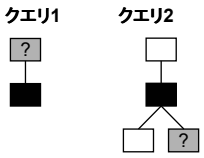

図 7 テストに用いたクエリの形式

図 8に各クエリ 100 回の処理における平均レスポンスタイムを示す. HW は Xeon 2GHz(dual), メモリ 2GB, HD 10,000rpm を用いている. いずれのケースでも 3,000,000 万件の文書までは 10 秒以内で処理できており, クエリ間隔が取れる用途であれば, インターラクティブに使うことも可能と考えられる. 各クエリの比較においては, クエリ 1, k=10 のケースで他よりも高速に動作しているが, これは打ち切り地点の違いによるものと考えられる.

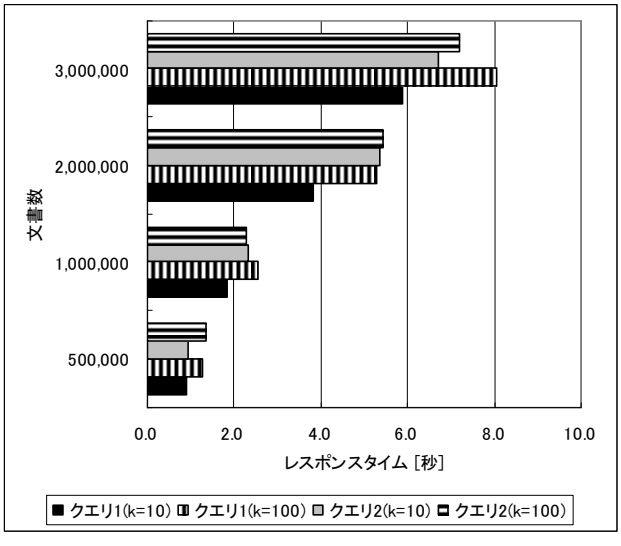

図 8 トップ k パターン検索レスポンスタイム

上記の実験では, Quad ファイルから取得した quad 配列を 750MB 分メモリにキャッシュしている. 図 9にキャッシュサイズを変えた場合のレスポンスタイムへのインパクトを示す. 550MB 周辺で効果が頭打ちになっているのは, Quad ファイルの平均的な打ち切り位置までがキャッシュに乗っているためと考えられる.

索引全体のサイズは 3,000,000 文書(2,991MB)に対し 1,997MB であったため，索引サイズの約 1/4 のキャッシュで効果が現れていることは，索引構造と打ち切りアルゴリズムによって I/O を局所化できていることを意味していると言える．

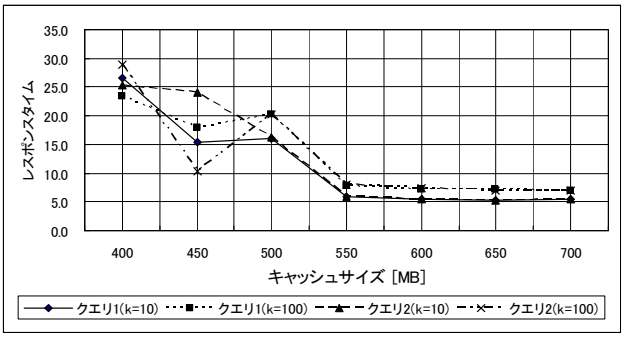

図 9 キャッシュ効果(3,000,000 文書)

6. 結論

6.1 提案内容

本研究では，テキストデータから情報を抽出するための有用な係り受けパターンを作成するために，所与のパターンに一語を追加したものの頻度順トップ k 個をユーザーやプログラムに提示するアプローチを提案した．また，そのアプローチを実現する方法として，トップ k パターン検索に特化した索引構造，アルゴリズムを提示し，パフォーマンススタディーによりコールセンター等，企業内の多くの文書をカバーできると考えられる 3,000,000 文書(2,991MB)の規模までのスケーラビリティーを確認した．

6.2 将来への課題

本研究で提案したトップ k パターン検索によるパターン作成のアプローチは，そのままのインターフェースではユーザーに自然言語の構文構造の知識を要求することになる．実際には，車のパーツの現象を調べる場合はパーツ名の係り先を検索する，原因／状況分析の場合は用言の係り元を検索する，といったように，実務に役立つシナリオとクエリ作成方法とのマッピングについても研究開発が必要になる．

また，より理論的な側面からのチャレンジとしては，ツリー構造の拡張タスクをグラフ構造の拡張タスクに一般化することが考えられる．グラフの場合は，ツリーの場合の preorder, postorder, level のような格納方式が確立されていないため，索引構造は自明ではない．

6.3 テキスト解析技術へのインパクト

係り受け構造は，評判分析，question answering，同義語抽出をはじめとして，テキスト解析の多くの分野で中心的な役割を果たしている．しかし，単語同士の係り受けの偏り，逆接／順接を挟んでの表現の出現傾向等，係り受け構造から二次的に得られる情報は，殆どの場合，予め抽出しておく必要があった．これに対し，提案手法が実現する処理は，プログラムが必要とした段階で動的に係り受けの傾向を調べられる仕組みを提供する．この特性は，ノード数の多いパターン等，予め抽出するにはバリエーションが多すぎる情報を用いるタイプのアルゴリズムには非常に都合が良いため，自然言語処理に新たな可能性をもたらすものと期待される．

参考文献

[Ding 2002] J. Ding,D. Berleant,D. Nettleton,E. Wurtele: Mining Medline: Abstracts, Sentences, or Phrases ?, PSB, 2002.

[Nasukawa 2001] T. Nasukawa, T. Nagano: Text analysis and knowledge mining system，IBM Systems Journal 40(4)，1999.

[Sekine 2006] S. Sekine: On-Demand Information Extraction, COLING/ACL, 2006.

[Bellahsene 2003] Z. Bellahsene: Database and Xml Technologies: First International Xml Database Symposium, Springer, 2003.

[Zhang 2001] C. Zhang，J.F. Naughton，D. J. DeWitt，Q. Luo,G. M. Lohman: On Supporting Containment Queries in Relational Database, SIGMOD, 2001.

[Al-Khalifa 2002] S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel , D. Srivastava , Y. Wu: Structural Joins: A Primitive for Efficient XML Query Pattern Matching, ICDE, 2002.

[Bruno 2002] N. Bruno，N. Koudas，D. Srivastava: Holistic Twig Joins: Optimal XML Pattern Matching，SIGMOD，2002.

[Chien 2002] S.-Y. Chien, Z. Vagena, D. Zhang, V. J. Tsotras: Efficient Structural Joins on Indexed XML Documents，PAKDD, 2002.

[Zaki 1999] M. J. Zaki: Efficiently Mining Frequent Trees in a Forest, SIGKDD, 2002.

[MEDLINE 2008] U.S. National Library of Medicine , http://www.ncbi.nlm.nih.gov/.