大規模データベースからの頻出構造化パターンの抽出

松 澤 裕 史†

大規模なデータベースから構造化された相関パターンを発見するというデータマイニングの問題を取り上げる．構造化された相関パターンというのは，アイテムの集合の集合である．これは，対象データ中に存在する特定の集合における 2 段階の構造を表現することができる．抽出される構造そのものは，非常に簡単であるが，しかし，従来のパターン発見アルゴリズムでは抽出することはできない．本論文では，すべての頻出する構造化相関パターンを発見するアルゴリズムを示す．この構造化相関パターンの抽出問題は，テキストマイニングのあるアプリケーションに対するソリューションとして取り組んだ問題である．人工データおよび実データに対して実験を行い，本論文のアルゴリズムが大規模データに対して効率良く構造化相関パターンを抽出することを確認した．

Mining Structured Association Patterns from Large Databases

Hirofumi Matsuzawa†

We consider the data-mining problem of discovering structured association patterns from large databases. A structured association pattern is a set of sets of items that can represent a two level structure in some specified set of target data. Although the structure is very simple, it cannot be extracted by conventional pattern discovery algorithms. We present an algorithm that discovers all frequent structured association patterns. We were motivated to consider the problem by a specific text mining application. Experiments with synthetic and real data show that our algorithm efficiently discovers structured association patterns in a large volume of data.

1. はじめに

データベース中のパターンを見つけることは，データマイニングにとっては基本的な問題であり，これまでに相関ルール[1)~7)] やシーケンシャルパターン[8),9)] のマイニングアルゴリズムについて多くの研究がなされてきた．これらのアルゴリズムの多くは，アイテムの集合やシーケンスなど簡単なパターンの発見を目的としている．しかし，このような簡単なパターンではユーザが知見を得るのが難しい場合がある．

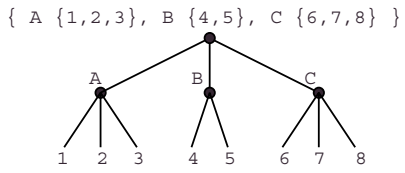
実際の興味の対象となるデータが複雑な構造を持っている場合でも，マイニングアルゴリズムが生成するパターンが単純であるため，これらのアルゴリズムを適用するには，ユーザは複雑な構造のデータに対して，興味ある特定の部分に着目してフラットなテーブルに変換し，残りを無視するという手段を講じている．しかしながら，Web 上のデータや自然言語で記述された文書などの半構造データ[10),11)] は，事前に構造が固定されていないが何らかの構造を持っており，その構造自体が知識発見の興味深い対象となっている．もし，同じ構造を持つ特定のアイテム集合をパターンとして取り出すことができれば，このようなパターンは直感的で有益なパターンとなりうる．

複雑なパターンのマイニングを行う第 1 段階として，本論文が対象とするパターンの形式をアイテムの集合の集合とした．パターンは，内部の集合をラベル付き集合として 2 段階の木構造（**図 1**）として記述できる．以下，このようなパターンを**構造化相関パターン**，または単に**構造化パターン**と呼ぶ．本論文では，データベース中に含まれるすべての頻出する構造化相関パターンを発見するアルゴリズムについて述べる．

1.1 例 題

本研究はテキストマイニングの 1 つのアプリケーションに取り組むために行った研究であり，ここに，その例題として企業のコールセンタ業務を紹介する．企業のコールセンタでは，顧客からの電話に応対したコールテーカが顧客の質問内容などについて要約し，自然言語で記述し，そのままデータベースに保存していく．コールテーカによって蓄積された大量の文書

† 日本アイ・ビー・エム株式会社東京基礎研究所
Tokyo Research Laboratory, IBM Japan, Ltd.



A, B, and C are labels of inner sets.

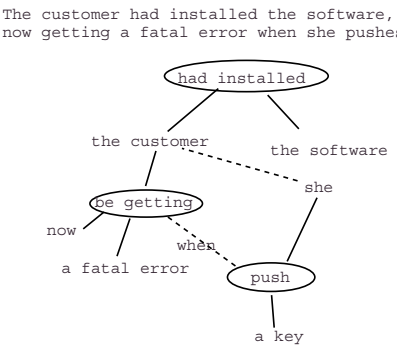
図 1 パターンの例
Fig. 1 An example of a pattern.

データから典型的な，あるいは頻繁に尋ねられる質問，要求，不満，賞賛などを知識として抽出することができれば，企業はこれらの知識を業務や商品の改善などに有効に活用することができる．しかし，自然言語で記述された大量の文書データからそのような知識を取り出すためには，すべての文書に目を通さなければならず，人間にとっては非常に困難であり，マイニングツールの支援が必要となる．

従来のマイニング手法を活用する単純な方法として，文書中の単語をアイテムとして抽出し，相関ルールやシーケンシャルパターンを従来のマイニングアルゴリズムを用いて発見するという方法が考えられる．この方法により，多くのパターンを生成することができる．しかしながら，発見されたパターンから元の文の意味を再現することはたいてい困難である．なぜなら，本来，元の文中に含まれている各単語の役割，単語間の係り受け関係などの重要な情報が失われてしまっているからである．たとえば，仮に，「音」「映像」「再生」「出ない」という単語の共起パターンが発見できたとしても，「音を再生する」のか，「映像を再生する」のか，「音が出ない」のか，「映像が出ない」のか，判断できないのである．

コールセンタに蓄積されるテキストデータは，その企業の製品を購入した顧客，あるいは，購入を検討している顧客からの問合せである．その内容は，発生したトラブルなどの細かな状況や詳細な質問内容であり，製品名やパーツの名称だけでカウントしても，製品ごとやパーツごとの問合せ件数が分かるだけで，具体的に，どのような問合せが多いのか，どのようなトラブルが発生しているかなどを判断することは不可能である．ここで，取り出したいと考えているのは，「何がどうした」という文書で頻出するもの，「何をどうしたら，何がどうなった」という文章で頻出するものである．

自然言語処理技術の進歩により，個々の単語の意味の曖昧性を取り除いて，自動的に単語間の関係を取り出すことが可能になってきている．その技術は完全ではないが，非常に有益である．実際に，典型的な機械翻訳技術では，文章から単語間の意味的な関係に基づくネットワーク構造を構築し，ネットワーク上で何らかの操作を行ったうえで，そのネットワーク構造を他の言語の文章へ変換を行っている[12),13)]．**図 2** に，文章から構築されたネットワーク構造例を示す．ネットワーク上では，個々の単語の意味が決定されるので，もし，ネットワーク構造から重要なパターンが発見できれば，典型的な文章を取り出すという目的を実現するのに役立つと思われる．したがって，そのようなネットワークの集合が与えられたとき，パターンとして頻出するサブグラフを発見する問題は，非常に興味深い．しかしながら，この問題は複雑で計算コストが非常に高い．したがって，以下のような問題の簡略化を行う．



円で囲まれた単語は動詞

図 2 単語のネットワーク例
Fig. 2 An example of a network.

表 1 述語と引数の組
Table 1 Predicate-argument tuples.

predicate	arguments
install	{customer, software}
get	{customer, fatal error, now}
push	{she, key}

本論文では，文章中で最も重要な意味的役割を持つ“述語”（動詞）に着目した．各述語は，直接あるいは間接的に係り受けの関係となるような単語を“引数”として持つ．そのような単語のグループを“述語–引数”のタプルとして扱う．たとえば，図 2 の英文例は，**表 1** に示すような 3 つのグループを持つ．これらのグループは元の文章に含まれている情報のすべてを保持しているわけではないが，個々の単語からなる 1 つの集合として扱うよりも多くの情報を保持しており，ユーザが元の文書の意味を再現することがより容易になる．

述語と引数のタプルは，述語をラベルとし，引数を集合の要素とすることで，ラベル付きの集合として扱うことができる．そして，文はラベル付き集合の集合と見なすことができる．したがって，解くべき問題は，文書データベースから { (install {software A, driver B}), (get {fatal error}) } というようなラベル付き集合の集合で表現されるパターンで，かつ頻出するパターンをすべて発見する問題である．

1.2 関連研究

Web 上で扱われるデータは，一般に，半構造データと呼ばれている．半構造データとは，何らかの構造は持つけれども，構造が固定されていないデータである．近年，半構造データを効率的に扱うこと自体が研究対象になっている[14)~16)]．ラベル付き有向グラフで表現された半構造データからスキーマを抽出する問題についても研究がなされている[17),18)]．彼らの対象データは，本論文の動機となった例題に似ているが，適用することはできない．

文献 1) によりデータベースからの相関ルールのマイニングが提案され，文献 2) で改善されたアルゴリズム Apriori が示された．Apriori には，多くのバリエーション[5)~7)] が存在しており，これらは，入力データが候補パターンにマッチしているかどうかを調べる方法が異なっている．しかし，簡単なアイテム集合を発見するという点では同じである．

文献 8) においてシーケンシャルパターンを発見する問題が提案され，文献 9) において効率的なアルゴリズム GSP が示された．シーケンシャルパターンとはアイテムの集合のシーケンスであり，その構造は本論文で扱う構造化相関パターンと類似する．しかしながら，シーケンシャルパターンに対するこれらのアルゴリズムはアイテムの出現順序に依存しているため，例題の問題に適用させることができない．

テキストデータを対象とするアプリケーションについては，様々なアプローチがある．これらのアプローチの中で，自然言語処理技術を適用することの意義については議論があり，いまだ結論が出ていない[19)]．たとえば，情報検索の分野において文献 20)，21) など自然言語処理技術を用いない研究が広く行われている一方で，自然言語処理技術を用いて単語の係り受け関係を利用した研究[22)] も行われている．

自然言語処理技術を用いることなしに，テキストを単語のシーケンスとして扱うことで，大量のテキストデータから頻出するパターンを高速に発見する方法が提案されている[23)]．文献 23) が発見の対象とするパターンとは，(<phrase 1>, <phrase 2>, ..., <phrase n>; k) で表される．これは，各フレーズ (<phrase i>) 間の距離がたかだか k 個以内であるような n 個のフレーズの並びであり，各フレーズとは <natural language processing> というような連続する単語のシーケンスである．このパターンは，単語の並びにのみ着目しているという点で，我々が目的とするような単語間の係り受け構造を考慮した構造化パターンとは異なっている．

一方で，テキストデータから「何がどうした」という関係を取り出して，出現頻度の月別の推移や，カテゴリとそのカテゴリに特徴的に現れる単語との関係を視覚的に表示するシステム[24)] が開発されており，大量のテキストデータから自然言語技術を用いた情報抽出を行う研究も行われている．

メッセージ理解の文献において，文書データからの情報抽出を目的とした多くのアルゴリズムが提案されている (たとえば，文献 25) など)．多くの場合，これらのアルゴリズムは事前に定義されたパターンを必要としており，ユーザは，あらかじめどのような種類のパターンが発見されるか知らなければならない．事前に定義されたパターンが与えられるならば，既存のパターンマッチングアルゴリズムを使って，データベース中の各パターンの頻出度を確認すればよい．しかし，データマイニングのアプリケーションでは，事前にパターンを定義することは想定していない．

本論文の構成は，以下のとおりである．2 章において問題の形式的な定義と必要な記法について述べる．3 章においてすべての頻出する構造化パターンを効率良く発見するアルゴリズムの概略を述べる．4 章においてアルゴリズムの詳細について述べる．5 章において人工的データおよび実データを使った実験結果からアルゴリズムの評価を行う．評価に用いた人工的データの生成方法については，付録 A.1 で述べる．最後に，6 章において，まとめと今後の方針について述べる．

2. 準備

本章では，文献 1)，2) に沿って問題を形式的に定義する．

定義

$\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ をリテラルの集合とし，各リテラルをアイテムと呼ぶ．ラベル付き集合 $g = (l, A)$ は，ラベル $l \in \mathcal{I}$ とアイテムの集合 $A \subseteq \mathcal{I}$ のペアである．ラベル付き集合をグループと呼ぶ．$\mathcal{G} = \mathcal{I} \times \mathrm{set}(\mathcal{I})$ はグループのドメインを示す．トランザクションの集合を $\mathcal{D}$ とし，各トランザクション T は，グループの集合 ($T \subseteq \mathcal{G}$) である．各トランザクションは *TID* と呼

ばれる固有の識別子を持ち，トランザクション中の各グループもまた，*GID* と呼ばれる固有の識別子を持つ．“構造化相関パターン” もグループの集合である．

パターンマッチング

グループ $g = (l_g, A_g)$ が他のグループ $h = (l_h, A_h)$ にマッチするとは，両者のラベルが等しく $(l_g = l_h)$，かつ，A_g が A_h の部分集合である $(A_g \subseteq A_h)$ ことをいう．パターン $P = \{g_1, g_2, \ldots g_k\}$ がトランザクション $T = \{h_1, h_2, \ldots h_l\}$ にマッチするとは，パターン P 中の各要素 $g_i \in P$ に対して，それぞれマッチするグループ $h_j \in T$ が，別々に 存在していることをいう．ここで，複数のグループどうしのマッチングでは，それぞれ別々のグループどうしがマッチしなければならないことに注意する．たとえば，パターン $P_1 = \{(1, \{2\}), (1, \{4\})\}$ は，トランザクション $T = \{(1, \{2, 3, 4\}), (1, \{4, 5\})\}$ にマッチするが，$P_2 = \{(1, \{2\}), (1, \{3\})\}$ は T にマッチしない．なぜならば，P_2 の 2 つのグループは，どちらもトランザクション T の最初のグループにしかマッチしないからである．もし，パターン P' が P にマッチするならば，P' は P のサブパターンであるという．

サポート

パターン P が，トランザクションセット $\mathcal{D}$ に対してサポート s を持つとは，パターン P がトランザクションセット $\mathcal{D}$ のうちの s%のトランザクションにマッチすることをいう．ただし，あるパターンがトランザクションに複数の方法でマッチする場合でも，そのトランザクションに対するサポートのカウントは 1 である．

問題定義

すべての構造化相関パターンを発見するという問題は，トランザクションの集合 $\mathcal{D}$ が与えられたとき，ユーザが与えた最小サポート値（*minsup*）よりも多くのトランザクションとマッチするようなすべての構造化相関パターンを生成することであると一般化することができる．もし，パターンのサポートが *minsup* よりも大きければ，そのパターンを “頻出（構造化相関）パターン” と呼ぶ．文献 2) に記載の方法と同じ方法を用いて，$X \cup Y = P$ かつ $X \cap Y = \phi$ のときに，パターン P から $X \Rightarrow Y$ という形式をしたルールを導出することができる．したがって，ルール導出に関しては本論文では議論しない．

グループ中のラベルを含めたアイテム数をグループの**サイズ**と呼ぶ．同様に，パターン中のグループサイズの総和をパターンの**サイズ**と呼ぶ．また，サイズ k のパターンを k-パターンと呼ぶ．

各グループ内のアイテムは，一般性を失うことなく辞書順にソートされていると仮定する．ここでは，集合のみを考えており，シーケンスについては考えていないからである．同様に，各トランザクションやパターンのグループは，各グループのラベルの辞書順にソートされていると仮定する．もし，グループ $g_1 = (l_1, A_1)$ とグループ $g_2 = (l_2, A_2)$ が同じラベルを持つ（すなわち，$l_1 = l_2$ である）ならば，アイテム集合 A_1 と A_2 を最初の要素から辞書順に比較していき，最初に違う要素が現れた時点で順位付けを行うことができる．トランザクション中に複数のグループが同じラベルを持つことを許しているので，上記のようなケースが出現する．このようなケースを想定することで，問題が複雑になり，興味深いものになっている．また，各グループに対して，*NULL* という特別なラベルを割り当てることにより，ラベルを持たないような単なる集合の集合を扱うことも可能である．

1.1 節の例題への適用を考えた場合，各アイテムを動詞以外の単語と考えることができる．また，データや自然言語ツールの性質にもよるが，fatal error など複合名詞を 1 つのアイテムとしてとらえることも可能である．各グループのラベルは，述語（動詞）であり，トランザクションが 1 つの文に相当する．

3. アルゴリズムの概略

トランザクションやパターンを単純なアイテムのシーケンスになるようにし，ラベルを特別なアイテムとして扱うことができれば，本論文で想定している問題は制約付きの相関ルールのマイニングの問題になると考えられるかもしれない．実際，この方法を適用してみたが，この章の後で示すように，このアイデアは同じラベルを持つ複数個のグループが 1 つのトランザクションに含まれる場合には十分ではない．動機となった例題がテキストマイニングであり，1 つの文章中に述語である動詞が複数使われることが珍しくないのである．この章では，この問題の特徴について記述し，アルゴリズムの概略を示す．

我々のアルゴリズムの概略は，Apriori のアルゴリズムに類似している．はじめに，データベース中の候補パターンの出現数を集計し，頻出パターンから 1 つ大きなサイズの候補パターンを生成する．これを候補パターンが生成できなくなるまで繰り返す．

パターンのサポートの集計

グループの集合 p が与えられると，これをシンボルのシーケンス $s(p)$ に変換する．たとえば，

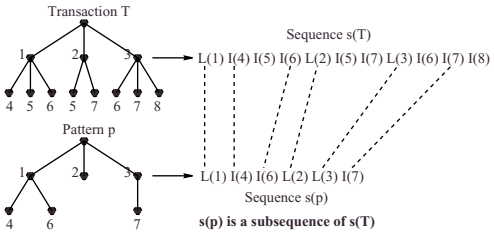


図 3 シーケンスマッチング（1）
Fig. 3 Sequence matching (1).

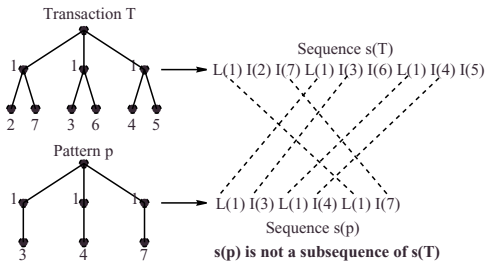


図 4 シーケンスマッチング（2）
Fig. 4 Sequence matching (2).

$p = \{(1, \{4, 6\}), (2, \{\}), (3, \{7\})\}$ が与えられたとき，$s(p) = \langle L(1)\ I(4)\ I(6)\ L(2)\ L(3)\ I(7)\rangle$ に変換される．ここで，$L(*)$ はグループラベルを表すシンボルであり，$I(*)$ は，グループの要素を表すシンボルである．4.1 節にて，この変換方法の詳細を記す．ここで，同じラベルを持つ複数のグループがトランザクション中に含まれないと仮定する．パターン p がトランザクション T にマッチするかどうかを調べるためには，パターン p のシーケンス $s(p)$ がトランザクション T のシーケンス $s(T)$ のサブシーケンスであるかどうかを調べればよい．ここで，S_1，S_2 を 1 個以上の要素を持つシーケンスとする．S_1 から，ある要素をいくつか取り除くことで S_1 と S_2 が等しくなるとき，S_2 を S_1 のサブシーケンスと呼ぶ（ただし，$s(T)$ 中の要素 $L(x)$ を取り除くときは，x をラベルに持つグループ中のアイテムも同時に取り除かなければならないことに注意する）．たとえば，**図 3** では，パターン p のシーケンス $s(p)$ は，トランザクション T のシーケンス $s(T)$ のサブシーケンスとなっているため，パターン p は，トランザクション T のサブパターンである．

Apriori は，与えられたトランザクションのサブシーケンスとなっているようなパターンのすべてを効率良く発見するために，ハッシュ木を用いており，我々の問題に対して類似のアルゴリズムが適用可能と考えられるかもしれない．しかしながら，トランザクション中の複数のグループが同じラベルを有する場合には，上記の方法はうまく機能しない．たとえば，あるトランザクション T からいくつかのアイテムを取り除いてパターン p を作ったとする．p に同じラベルを有するグループが複数あるとき，パターンは辞書順にソートされているという制約からグループをソートするので，その順序が入れ替わることがある．このため，p に対応するシーケンス $s(p)$ は，T のシーケンス $s(T)$ のサブシーケンスに必ずしもならない．たとえば，**図 4** の場合，パターン p はトランザクション T にマッチするにもかかわらず，$s(p)$ は $s(T)$ のサブシーケンスにはなっていない．

したがって，パターン p がトランザクション T にマッチするかどうかを決めるためには，$s(p)$ のすべての要素をカバーするような $s(p)$ と $s(T)$ の二部グラフのマッチングが存在するかどうかを調べる必要がある．候補パターンが多く存在し，与えられたトランザクションにマッチするようなパターンのすべてを候補パターンから見つけ出したいので，同時に複数の二部グラフのマッチングを求める必要がある．我々は，この問題を解決するために別のハッシュ木を用いる．4.1 節でその詳細について述べる．

候補パターンの生成

Apriori や GSP の候補生成手法では，パターン a のある決められた位置のアイテムを除去して得られたサブパターンが，他のパターン b のある決められた位置のアイテムを除去して得られたサブパターンと同じならば，a と b を join することで，候補パターンの生成を行う．Apriori では，最後の要素を除去する．すなわち，パターン $a = \{a_1, a_2, \ldots, a_k\}$ とパターン $b = \{b_1, b_2, \ldots, b_k\}$ があるとき，最後の要素だけが違う場合（すなわち，$a_1 = b_1, \ldots, a_{k-1} = b_{k-1}$，$a_k < b_k$ のとき），a と b を join して，新しい候補パターン $c = \{a_1, \ldots, a_{k-1}, a_k, b_k\}$ を生成する．この方法は，パターンの数が多くても，非常に効率良く機能する．

しかし，構造化されたパターンに対しては，その制約に違反することなく，最後の要素，あるいは，決められた位置の要素を除去することができない．たとえば，パターン $p = \{(1, \{3\}), (1, \{4\})\}$ を考える．join してパターン p を生成するようなパターンで，最後の要素だけが違うようなパターン a，b は存在しない．そのようなパターンを生成するためには，$p_1 = \{(1, \{\}), (1, \{4\})\}$ と $p_2 = \{(1, \{3\}), (1, \{\})\} = \{(1, \{\}), (1, \{3\})\}$ を join する必要がある．しかしながら，p_1 を p_2 と join すると，別のパターン $p' = \{(1, \{\}), (1, \{3, 4\})\}$ が生成される．したがって，構造化されたパターンの候補

生成は，従来の単純な候補生成手法を適用することができない．

2 つの $(k-1)$-サイズのパターンを join したとき，どんな k-サイズのパターンが生成されるかを考える代わりに，k-サイズのパターンが，どのような 2 つの $(k-1)$-サイズのサブパターンに分割できるかについて考える．$k \geq 3$ のサイズのパターンを，最後の 2 つの要素に基づいて，次の 4 つのタイプに分類した．それぞれのタイプにおいて，2 つの $(k-1)$-サイズのサブパターンを join して k-サイズのパターンが生成できるように，k-サイズのパターンを 2 つの $(k-1)$-サイズのサブパターンに分割する．

- Type A：パターンの最後の 2 つの要素が同じグループに属するアイテムである．
- Type B：パターンの最後の要素がラベルで，最後のグループはラベルだけからなる．さらに，最後から 2 つ目の要素が，最後から 2 番目のグループの要素である．
- Type C：パターンの最後の要素が最後のグループのアイテムで，かつ，最後から 2 番目の要素がラベルである（最後のグループは，このラベルとアイテムからなる）．さらに，このタイプは最後から 3 番目の要素に基づき，Type C1 と Type C2 に分類される．
- Type D：最後の 2 つの要素がラベルである（すなわち，最後の 2 つのグループは，どちらも，ラベルだけのグループである）．

図 5 は，これらの分類と k-サイズパターンを生成するために join する $(k-1)$-サイズのサブパターンを示している．この図において，三角形はサブパターンおよびサブグループを示している．各要素はラベルかアイテムのいずれかなので，すべてのパターンは上記の Type A から Type D のいずれかに必ず分類される．したがって，どのようなパターンを生成する場合であっても，図のいずれかの式を使ってパターンを生成することができる．最後と最後から 2 番目のグループが同じラベルを持つような場合には，グループの順序が変わるため注意する必要がある．

4. アルゴリズム

図 6 に，我々のアルゴリズムの概略を示す．このアルゴリズムは，Apriori を基本とした他のパターンマイニングアルゴリズムと類似のものである．はじめに，単一アイテムからなる候補パターンを生成する（4 行目）．ここで，$(*, \{i\})$ という形式のパターンは，すべてのラベルにマッチする特別なラベル “*” を持つ．

各パス（パス k と呼ぶ）は，2 つのフェーズからなる．

最初のフェーズではデータベースがスキャンされ，“Count” という関数が呼ばれて C_k 中の各候補のサポートがカウントされる（7 行目）．Count のアルゴリズムで用いるデータ構造は，Apriori [2] が一部で用いているデータ構造に似ている．しかし，本アルゴリズムでは，パターン中の複数のグループが同じラベルを持つ場合を処理する必要があるため，同じデータ構造を使うことができない．4.1 節において，Count のアルゴリズムが，どのようにして効率的に与えられたトランザクション t にマッチする C_k 中の候補を決定するのかについて記述する．

次に “Generate-Candidates” ファンクション（9 行目）において，L_k 中の頻出パターンからパス k+1 のための候補パターンの集合 $C_{(k+1)}$ を生成する．4.1.2 項において，このファンクションの詳細について述べる．

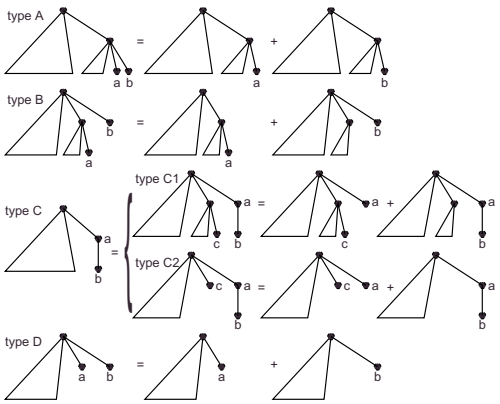


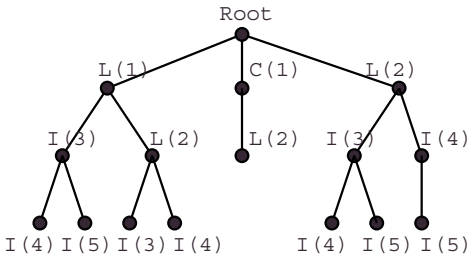
図 5 パターンの分類
Fig. 5 Classification of patterns.

```
0)  Main(DataSet 𝒟) {
1)      // 𝒟 は，入力トランザクション集合.
2)      // C_k は，サイズ k の候補パターン集合.
3)      // L_k は，サイズ k の頻出パターン集合.
4)      C_1 ←{(l, ∅) | l ∈ ℐ} ∪ {(*, {i}) | i ∈ ℐ};
5)      k ← 1;
6)      while (C_k ≠ ∅) {
7)          Count(𝒟, C_k);
8)          L_k ←{p ∈ C_k | p は頻出する };
9)          C_{k+1} ← Generate-Candidates(L_k);
10)         k ← k + 1;
11)     }
12)     return ⋃_k L_k;
13) }
```

図 6 アルゴリズムの概略
Fig. 6 Outline of the algorithm.

```
0) COUNT(DataSet 𝒟, CandidateSet C_k) {
1)     for each transaction t ∈ 𝒟 {
2)         C_t ←{p ∈ C_k | p matches t};
3)         for each candidate c ∈ C_t {
4)             c.count ← c.count + 1;
5)         }
6)     }
7) }
```

図 7 COUNT ファンクション
Fig. 7 COUNT function.



A hash-tree storing $\langle L(1)I(3)I(4)\rangle$, $\langle L(1)I(3)I(5)\rangle$, $\langle L(1)L(2)I(3)\rangle$, $\langle L(1)L(2)I(4)\rangle$, $\langle C(1)L(2)\rangle$, $\langle L(2)I(3)I(4)\rangle$, $\langle L(2)I(3)I(5)\rangle$, and $\langle L(2)I(4)I(5)\rangle$.

図 8 ハッシュ木
Fig. 8 A hash-tree.

4.1 パターンのサポートの集計

図 7 に COUNT ファンクションの概略を示す．

与えられたトランザクション t にマッチする C_k 中の候補パターンを効率良く見つけるために，候補パターンをハッシュ木に格納する．候補パターンが同じラベルを持つ複数のグループを持つ場合を扱うため，各パターン p を次の手続きによりシーケンス $s(p)$ に変換する．

- $s(p)$ を空にする．
- p 内のラベルを含むすべてのアイテム i をスキャンし，次のステップを適用する．
  - i がグループラベルで，次に続くグループが同じラベルを持つならば，$s(p)$ に要素 $C(i)$ を付け加える．$C(i)$ は，i というラベルを持つすべてのグループを表している．これらのグループを**クラスタ**と呼ぶ．クラスタ内のすべての要素をスキップし，違うラベルを持つ次のグループへ移動する．
  - i がグループラベルで，そのグループが最後のグループ，あるいは，次のグループが違うラベルを持つならば，ラベルを表す要素 $L(i)$ を $s(p)$ に付け加え，次のアイテムへ移動する．
  - i がグループラベルでないならば，アイテムを表す要素 $I(i)$ を $s(p)$ に付け加え，次のアイテムへ移動する．

上記のとおり，3 種類の要素，C（**クラスタ**），L（**ラベル**），および I（**アイテム**）がある．たとえば，パターン $\{(1,\{2,3\}),\ (2,\{1,3\}),\ (2,\{2,4\}),\ (3,\{2\})\}$ は，シーケンス $\langle L(1),\ I(2),\ I(3),\ C(2),\ L(3),\ I(2)\rangle$ に変換される．ここで，パターン中の 2 番目と 3 番目のグループがクラスタを形成している．パターンがクラスタを持つとき，クラスタを処理するための特別な処理が必要である．これ以外は，我々のアルゴリズムは，Apriori とほぼ同様に振る舞う．

ハッシュ木へのパターンの格納

本手法では，C_k 中のすべてのパターンのシーケンスをハッシュ木に格納する．ハッシュ木の内部ノードはハッシュ表になっている．シーケンス $s(p)$ をハッシュ木に付け加えるとき，ルートノードから始めてシーケンスをスキャンしながら木を下に向かって降りていく．深さ d のノードにおいて，$s(p)$ の d 番目の要素をハッシュ関数に適用しながらどの枝をたどっていくべきかを決める．このハッシュ木は suffix-tree [26] の一種であるので，全パターンに対する共通の接頭辞は，ルートノードからの経路で共有されており，ルートノードからリーフノードへのパスでパターンのシーケンスを識別することができる．**図 8** に例を示す．たとえば，パターン $\langle L(1),\ I(3),\ I(4)\rangle$ と $\langle L(1),\ I(3),\ I(5)\rangle$ は，図 8 において，ルートからノード $L(1)$，$I(3)$ と経由してそれぞれリーフ $I(4)$，$I(5)$ へたどり着く．シーケンス中で共通の $L(1)$，$I(3)$ は経路が共有されている．

パターンのサポートの集計

トランザクション t が与えられると，はじめに，次の手続きを用いて t をシーケンス $s(t)$ に変換する．なお，この手続きは，上述のパターンをシーケンスに変換する手続きと一部違う個所がある．

- $s(t)$ を空にする．
- t 内のラベルを含むすべてのアイテムをスキャンする．
  - i がグループラベルで，かつ，次に続くグループが同じラベルを持つならば，これらすべてのラベル i を持つグループのクラスタを表す $C(i)$ を $s(t)$ に加える．$s(t)$ を加えると同時に，$L(i)$ も $s(t)$ に付け加える．
  - i がグループラベルで，次のグループが違うラベルを持つか，グループラベル i が以前のステップですでに処理されている，あるいは，そのグループが最後のグループである場合には，ラベルを表す要素 $L(i)$ を $s(t)$ に付け加

え，次のアイテムへ移動する．
– i がグループラベルでないならば，アイテムを表す要素 $I(i)$ を $s(t)$ に付け加え，次の要素へ移動する．

同じラベルを持つグループは，1 つはクラスタとして，もう 1 つは個別の要素として 2 度処理されている．たとえば，トランザクション $\{(1,\{2,3\}),(2,\{1,3\}),(2,\{2,4\}),(3,\{2\})\}$ は，シーケンス $\langle L(1),I(2),I(3),C(2),L(2),I(1),I(3),L(2),I(2),I(4),L(3),I(2)\rangle$ に変換される．

トランザクション t にマッチするすべての候補パターンを発見するために，次の手続きを適用する．適用する手続きはハッシュ木をルートからたどっていき，現在どのノードにいるかにより決まる．

- **ルートノード**：$s(t)$ の各要素をハッシュし，対応するバケット中のノードに，この手続きを再帰的に適用する．トランザクション t にマッチするどんなパターン p に対しても，$s(p)$ の最初の要素は $s(t)$ に存在しなければならない．$s(t)$ 内のすべての要素をハッシュすることで，$s(t)$ 内に存在しない要素から始まるパターンを無視することができる．
- ***I* または *L* のラベルを持つ内部ノード**：$s(t)$ 中の要素 e をハッシュして，このノードにたどり着いたとする．$s(t)$ の e の次に来る要素をハッシュし，対応するバケット中のノードにこの手続きを再帰的に適用する．
- ***C* のラベルを持つ内部ノード**：二部グラフを構築し，トランザクション中のクラスタにマッチする候補パターンを発見する．詳細については，4.1.1 項で述べる．$s(t)$ の要素 e をハッシュして，このノードにたどり着いたとする．t にマッチするようなクラスタを持つすべてのパターン p に対して，この手続きを p の接尾辞の木に対して，再帰的に適用していく．
- **リーフノード**：リーフノードにたどり着けば，トランザクションにマッチするパターンを見つけたことになる．

4.1.1 クラスタのマッチング

ハッシュ木の C のラベルを持つノードに着目する．ノードは複数のクラスタを持ち，クラスタはパターンの一部分でその接頭辞が共通になっている．$C=\{c_1,c_2,\ldots,c_k\}$ は，クラスタの集合を表す．このノードにたどり着いたとき，トランザクション t はクラスタ $c(t)$ を含み，$c(t)$ は，c_i $(i=1,\ldots,k)$ と同じグループラベルを持っている．我々の目的は，$c(t)$ にマッチする C に含まれるすべてのクラスタを発見することである．

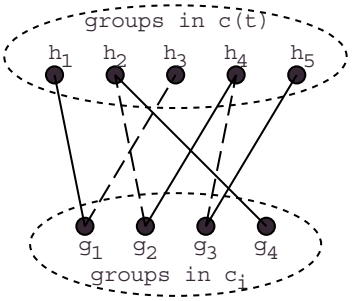


図 9 二部グラフのマッチング
Fig. 9 Bipartite matching.

ここでは，単一のクラスタ c_i が，$c(t)$ にマッチするかどうかを決める問題について考える．まず，c_i が n 個のグループを持っているとし，$c(t)$ が m 個のグループを持っているとする $(n\le m)$．$c_i=\{g_1,g_2,\ldots,g_n\}$ および，$c(t)=\{h_1,h_2,\ldots,h_m\}$ とする．$g_x\in c_i$ が $h_y\in c(t)$ にマッチすれば，かつ，マッチするときのみ，エッジ $(g_x,h_y)\in E_i$ が存在するような二部グラフ $G_i=(c_i,c(t);E_i)$ を考える．**図 9** に二部グラフの例を示す．c_i が $c(t)$ にマッチするのは $i=1,\ldots,n$ に対してすべての g_i をカバーするような二部グラフが G_i に存在しているときである．c_i が $c(t)$ にマッチするかどうかという問題は，G_i の二部マッチングの最大値のサイズが n であるかどうかをチェックすればよいことになる．これは，古典的なアルゴリズム[27] を使って効率良く解くことができる．図 9 において，直線は二部マッチングの最大エッジを示している．マッチングにより c_i のすべてのグループがカバーされていれば，c_i は $c(t)$ とマッチする．

効率良く，すべての $c_i\in C$ に対して，すべてのグラフ $G_i=(c_i,c(t);E_i)$ を構築するため，c_i $(i=1,\ldots,k)$ のすべての候補グループを格納するハッシュ木を事前に構築しておく．各グループ $h_j\in c(t)$ に対して，ハッシュ木を使って h_j にマッチするすべてのグループを識別しておき，対応するエッジをグラフに加えておく．上記の手続きをしてしまえば，すべてのグラフが事前に分かってしまうので，トランザクション t にマッチするすべてのクラスタを発見することができる．

いくつかの場合において，二部グラフ G_i を完全に構築する前に，c_i が $c(t)$ とマッチするかどうかを見ることができる．したがって，すべてのグラフを構築する前にアルゴリズムを停止させるような何らかの

表 2 候補生成の例
Table 2 An example of candidate generation.

Frequent 3-patterns (L_3)		Candidate 4-patterns (C_4) after join		Candidate 4-patterns (C_4) after pruning	
p_1	{(1 {}) (1 {3})}	c_1	{(1 {}) (1 {3,5})}	c_2	{(1 {3}) (1 {5})}
p_2	{(1 {}) (1 {5})}	c_2	{(1 {3}) (1 {5})}	c_4	{(1 {3}) (2 {4})}
p_3	{(1 {}) (2 {4})}	c_3	{(1 {3}) (2 {}) (2 {})}		
p_4	{(1 {3}) (2 {})}	c_4	{(1 {3}) (2 {4})}		
p_5	{(2 {}) (2 {4})}				

ヒューリスティックスがあれば，パフォーマンスは改善されるが，現在そのような技術は知られていない．

4.1.2 候補生成

GENERATE-CANDIDATES ファンクションは，引数としてすべて頻出する k-パターンの集合 L_k をとり，サイズ $(k+1)$ の候補パターンの集合 $C_{(k+1)}$ を返す．このファンクションは，次の 2 つのフェーズからなる．

Join フェーズ 頻出パターン集合 L_k を L_k と join してサイズ $k+1$ の候補パターンの集合 $C_{(k+1)}$ を生成する．もし，パターン $p_1 \in L_k$ が，図 5 の右側のサブパターンのどれかと一致するならば，パターン $p_2 \in L_k$ を探し，p_1 と p_2 を join して，新しい候補パターンを生成する．L_k を事前に辞書順にソートしておくことで，p_1 の join の相手は，その範囲が限定されており，p_1 の相手を探すために，L_k 全部を探す必要はない．

Prune フェーズ サイズ $k+1$ の候補パターンから生成されうるサイズ k のサブパターンのサポート値が最小サポート値以下の場合には，この候補パターンを $C_{(k+1)}$ から除去する．効率良くすべての k-サブパターンが L_k の要素であるかどうかを効率的に見るために，L_k を格納するハッシュ木を用いる．

上記の方法は単純な相関ルールの候補生成手法を連想させるが，細部で大きく異なっている．

候補生成例

表 2 は，L_3 と join と prune が終わった後の C_4 の例を示している．join フェーズにおいて，パターン p_1 は p_2 と join して，c_1 と c_2 が生成される（それぞれ図 5 の Type A と Type C1）．パターン p_3 は p_4 と join して c_4 が生成される（Type C1）．また，パターン p_4 は p_4 と join して c_3 が生成される（Type D）．prune フェーズにおいて，c_1 のサブパターン {(1, {3,5})} と c_3 のサブパターン {(1, {}) (1, {}) (2, {})} が L_3 にないので，c_1 と c_3 は除去される．

5. 実験とその評価

本手法のアルゴリズムの評価を行うため，C++で実装し，いくつかの実験を行った．実験には，Windows NT 4.0 が導入された IBM PC Server 330 を用いた．このマシンには，333 MHz の Intel Pentium II プロセッサと 320 MB のメインメモリが搭載されいている．データは，3.5 インチの 9 GB の SCSI ディスク上で NT ファイルシステム上に格納した．シーケンシャルリードのスループットは約 8 MB/sec であった．

表 3 人工データのパラメータ値
Table 3 Parameter settings of synthetic datasets.

ラベルの総数	1,000
アイテムの総数	1,000
1 トランザクション中のグループ数の平均	5
1 グループ中のアイテム数の平均	5

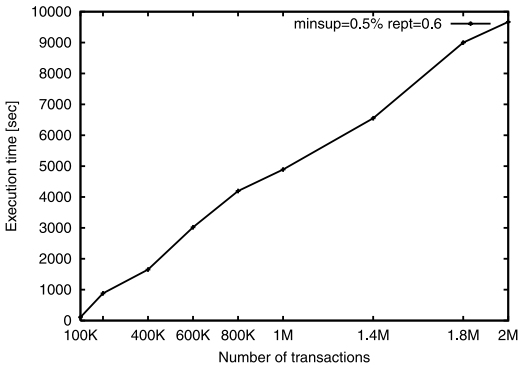
はじめに，人工的データセットを使って得られた結果を示す．続いて，日本アイ・ビー・エム（株）のコールセンタから得た実データ（文書データ）を用いた実験の結果を示す．

5.1 人工的データ

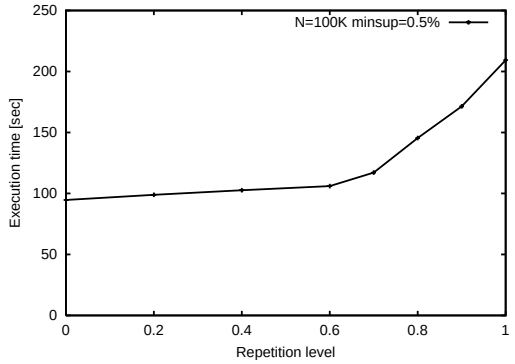
文献 2) に示された人工的データ生成方法は，パターンマイニングアルゴリズムの評価に広く使われいる．この方法を本手法のアルゴリズム用に改造し，構造化されたトランザクションデータを生成できるようにした．詳細は付録 A.1 に示す．**表 3** に主なパラメータを示す．

入力サイズに対するアルゴリズムのスケーラビリティを知るために，トランザクションの数を 100,000（14.4 MB）から 2,000,000（288 MB）までのいくつかのデータセットを用意した．最小サポートを 0.5%に，後述する repetition level を 0.6 に設定した．**図 10** (a) は，入力サイズとアルゴリズムの実行時間の関係を示している．実行時間は，トランザクション数に比例して，リニアに増えている．トランザクションの数が 2,000,000 のとき，6,236 個のパターンを発見した．

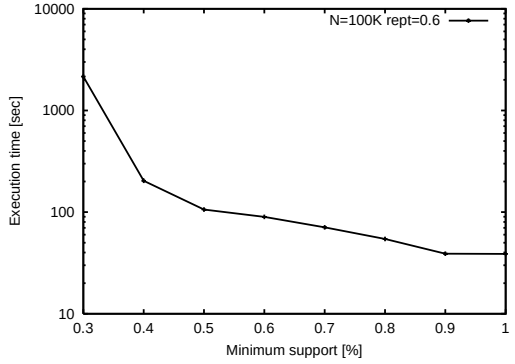
グループラベルの重複がどれだけパフォーマンスに影響を与えるかを調べるため，*repetition level* と呼ばれるパラメータを変えて，いくつかのデータセットを生成した．repetition level は，1 トランザクション中で，あるグループが他のグループと同じラベルを持つ



(a) Data volume vs. execution time.



(b) Repetition level vs. execution time.



(c) Minsup level vs. execution time.

図 10 人工データを使った実験結果
Fig. 10 The experimental results on the synthetic datasets.

確率の平均を制御するパラメータである．repetition level を 0，0.2，0.4，0.6，0.8 および 1.0 と変えてデータセットを作成した．トランザクション中の平均グループ数が 5 である一方で，トランザクション中の異なるグループラベル数（重複したものは 1 回しか数えない）の平均は，repetition level を上げると減少する（すなわち，それぞれ，5.0，4.3，3.6，2.9，2.2 および 1.6 となった）．トランザクション数を 100,000 とし，最小サポートを 500 個（0.5%）に設定した．図 10 (b) に，その結果を示す．実行時間は，repetition level が比較的低いときにはリニアに増加しているが，0.7 を超えたあたりから急激に増加し始める．

さらに，最小サポートの閾値がパフォーマンスに与える影響についても調べた．トランザクションの数を 100,000 に，repetition level を 0.6 に固定した．図 10 (c) にその結果を示す．注意すべきは，縦軸がログスケールであることである．すなわち，最小サポートの閾値がパフォーマンスに与える影響が大きいことを示している．また，これは，Apriori ベースの他のアルゴリズムと類似した傾向である．最小サポートが 0.3，0.5，0.7，0.9，1.0%のとき，頻出パターンの数は，それぞれ 93,074，1,264，750，602，565 であった．

5.2 実データ

メッセージ理解の分野において，文献 25) など文書データからの情報抽出を目的とした多くのアルゴリズムが提案されているが，これらの多くは事前にパターンを定義付ける必要がある．1.1 節の例題で示したコールセンタ業務のように，事前に知られていないようなパターンを抽出する目的には不適である．従来の相関ルールのアルゴリズムでは，事前にパターンを規定せずに頻出するすべてのパターンの発見を行っている．そこで，従来の相関ルールのアルゴリズムを用いて，共起する単語のパターンを抽出し，本手法と比較実験を行った．

コールセンタの実際のデータから日本語のテキストデータを取り出して，本手法のマイニングアルゴリズムと従来の相関ルールのマイニングアルゴリズムに適用した．従来の相関ルールの発見手法で用いられる Apriori を基本とするアルゴリズムでは，頻出するアイテム集合 Z を発見した後，Z の部分集合 X と Y（$X \cup Y = Z$，$X \cap Y = \phi$）からなるルール $X \rightarrow Y$ を生成している[1),2)] が，この実験の比較では，相関ルールを用いるわけではなく，発見された頻出するアイテム集合を単語の共起パターンとして用いる．したがって，係り受けの有無に関係なく，キーワードの共起関係のみをパターンとして取り出している．

実験に使われたデータは，ある企業のコールセンタにお客様から寄せられた質問などの応答の記録を保存，蓄積した文書であり，質問は年間数十万件に上る．文書データは，電話に応対するたびに記録されており，次々に短時間で入力しているため，新聞記事のように推敲されているわけではない．また，1 件の電話に対して，1 件の文書データが蓄積される．1 件の文書は，平均 5〜6 個の文（sentence）からなっている．文は，

表 4 コールセンタのデータ
Table 4 Call center data.

文の数	109,451
述語になる単語（動詞）の数	12,754
引数になる単語の数	54,453
1 文中の平均単語数	4.23
1 文中の平均グループ数	1.48
データサイズ	3.6 MB

読点（.）までを区切りとする単位である．

本手法のアルゴリズム用に，日本語の文を 1.1 節で示した述語と引数のタプルの集合へ変換するような自然言語パーザを用いて，構造化データを生成するという前処理を行った．

新聞記事のように，推敲を重ねた文章では，公開されている自然言語パーザなどを用いることで，約 90%の精度で構文解析を行うことができる[28]．一方で，この実験で用いられたコールセンタの文章は，コールテーカがわずかな時間で急いで入力するため，文章は，必ずしも文法的に正しいわけではない．したがって，完全な構文解析を行うことが難しい．また，文献 28) のような高精度を指向するパーザでは，数万件の文を処理するには，時間がかかりすぎてしまうという問題がある．那須川らの研究[24),29)] では，同じコールセンタの文書データから完全な構文解析を目指すのではなく，「何がどうした」という係り受け関係となる可能性のあるすべての単語の組合せを取り出すことで頻出する概念の発見を行っている．文献 24)，29) で用いられたパーザは，それほど精度が良いわけではないが，係り受け関係を取り出すのに十分な性能を持っており，本研究の目的に沿った動詞と名詞からなるグループを生成することが可能である．そこで，このパーザを本実験で用いることにした．また，すべての文を単なる単語の集合とするようなデータの変換を行い，従来の相関ルールのマイニングアルゴリズム（以下では，従来のアルゴリズムと呼ぶ）が，パターンとして，頻出する単語の集合を求められるようにした．**表 4** にデータの特徴を示す．このデータは，1997 年のある 1 カ月分のデータに相当する．

最小サポートを 5 個（0.005%）にしたときでさえ，本手法のアルゴリズムが，すべてのパターンを見つけるのに要した時間は 2 分以下であった．ここで用いた従来のアルゴリズムは違うプラットフォーム上で動いているので，実行時間の比較はここでは行わない．構造化されたパターンは，単なるアイテム集合よりも組合せ的には多く存在するにもかかわらず，**図 11** に示すように，従来のアルゴリズムが発見するパターンの総数は本手法のアルゴリズムが発見するパターンの総数の 2 倍になっている．

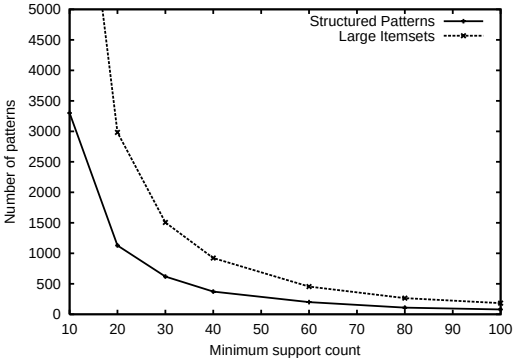


図 11 コールセンタデータを使った実験結果
Fig. 11 Experimental results on the call center data.

このパターンの総数に差が出る理由として次のことが原因として考えられる．たとえば，“メモリを購入するために，パソコンショップに行ったが，どの種類がいいのか分からなかったので教えてほしい” などの文章の場合，本手法では，{ 購入する (メモリ)} というパターンにはマッチするが，{ 行く (メモリ)} というパターンにはマッチしない．しかし，一方で，従来手法では，「メモリ」と「行く」からなるアイテム集合にマッチするため，直接，係り受け関係にない単語からなる共起パターンであっても発見されてしまう．例題で述べた「何がどうした」という情報を取り出したいという目的に対しては，従来の方法が発見するパターンの半分以上が，役に立たないパターンであることを示している．

コールセンタにおいて，実際にコール内容の分析を行っている担当者に依頼して，発見されたパターンやパターンにマッチする元の文書について考察を行った結果，分析者が興味を持ったパターンというのは，サポートが 10 から 20 ぐらいのパターンであった．最小サポートがそのような低水準のときには，従来の方法があまりにも多くのパターンを生成してしまい，半分以上が役立たないのである．したがって，従来のパターンマイニングアルゴリズムを使ったテキストの分析に比べ，本手法は分析の負荷を大きく低減することができる．

動機付けとなったアプリケーションの観点から本手法を評価するため，情報検索の分野で一般的な精度を測る尺度である再現率と適合率を調べた．

再現率と適合率を次のように定義する．たとえば，「PC のメモリを増設したい」という文を考える．この文には，3 つの単語があって，2 つの単語 “メモリ” と

表 5 再現率と適合率
Table 5 Recall and precision.

本手法			従来手法	
パターン	再現率 (%)	適合率 (%)	再現率 (%)	適合率 (%)
{ (認識しない { }) (増設する { ハードディスク }) }	75.0	81.8	100	75.0
{ (教える { }) (増設する { メモリー }) }	91.7	84.6	100	85.7
{ (出る { 画面, 文字 }) }	76.9	96.7	100	90.5
{ (増やす { ハードディスク, 容量 }) }	87.1	95.0	100	87.5
{ (出ない { スピーカ, 音 }) }	95.9	100	100	82.2
{ (取り込む { 画像 }) }	100	100	100	50.0
{ (接続する { プリンタ }) }	93.6	95.2	100	76.9
{ (使用する { インターネット }) }	100	60.0	100	24.0
50 パターンの平均	90.3	94.1	100	83.2

“PC” が，動詞 “増設する” に直接，または間接的に係り受け関係を持っていると考えることができる．上記の文は構造化パターンとして，{ 増設する (メモリ, PC)} と記述できる．また，従来手法では，{PC, メモリ, 増設する } と記述する．ツールを利用することにより各パターンがマッチしている元の文書をそれぞれ抽出することができる．これを検索文書と呼ぶ．もし，文書が「PC のメモリを増設したい」という意味を含んでいるならば，その文書を関連文書と呼ぶ．ここで，X を検索文書集合の文書数，Y を関連文書集合の文書数，そして，検索文書で，かつ，関連文書である文書集合の文書数を Z とする．そして，再現率を Z/Y，適合率を Z/X と定義する．再現率は，本来抽出すべき関連文書のうち，何%が検索できたかを示し，適合率は，抽出した検索文書のうち，何%が関連文書であったかを示している．

発見されたパターンに対して，再現率と適合率を調べた．発見されたすべてのパターンの中で，サイズ 1 のパターン，および，ラベルだけのグループからなるパターンは，従来手法との差が出ないため除外し，残りのパターンの中からランダムに，50 個のパターンを選択した．選択されたパターンはすべて，サイズ 2 または 3 のパターンであった．これは，表 4 にもあるように，1 文章中の平均単語数が 4.23 と少ないため，このような結果になったと思われる．**表 5** には，そのうちの数個のパターンに関する再現率と適合率，および 50 個の平均についての再現率と適合率をそれぞれ示す．従来手法では，明らかに，意図する文に含まれる単語を含む文書がすべて検索できるので，すべての関連文書をカバーし，再現率は必ず 100%となるが，適合率は，それほど高くない．

情報検索の分野で用いられる F-measure という指標[30] を用いて従来手法と本手法の比較を行った．F-measure は，適合率 P と再現率 R を用いて，以下の式で表される．

$$\text{F-measure} = \frac{(\beta + 1.0) \times P \times R}{\beta^2 \times P + R}$$

ここで，β は相関係数で 1.0 に設定する．実験の結果から，50 パターンのそれぞれについて F-measure を計算し，その平均を求めたところ，従来手法は 0.900 であり，本手法は 0.917 であった．本手法の方がわずかながら優位であった．

再現率と適合率の比較に用いたパターンは，従来手法と本手法のどちらでも発見できるパターンであり，従来手法での適合率が比較的高いパターンである．図 11 に示すように，従来手法は大量のパターンを抽出し，その多くは，係り受け関係を持たない共起関係があるだけの単語のパターンである．共起関係があっても係り受け関係を持たない単語からなるパターンは，適合率が低く，本手法では発見できないパターンであるため，比較対象外であリ，評価からも漏れている．これらのことを考慮すると，自然言語のパーザが完璧でないにもかかわらず，本手法の有効性を示しており，パーザの精度が向上すれば，本手法の再現率，適合率の向上も期待できる．

6. ま と め

本論文では，データベースから構造化相関パターンのマイニング問題を取り上げた．構造化相関パターンはアイテムの集合の集合であり，2 階層の構造を表現することができる．データベースから頻出する構造化相関パターンを効率良く発見するアルゴリズムを示した．また，このアルゴリズムを実装し，人工データおよび実データを用いて評価を行った．本論文で記述さ

れた方法は，個々の要素はそれほど重要でないが，要素の組合せが重要な意味を持ち，要素の組合せの集合が発見すべき対象として興味深い問題を解くのに利用可能である．

現在，コールセンタのためのテキストマイニングシステムのプロトタイプを開発しており，実際のテキストデータを使っていくつかの興味深いパターンを発見し，FAQ の生成にも役立てることができた．テキストマイニングにとって，複合語やシソーラスといった事前の知識を活用することは，本質的なことである．事前の知識を使って，構造化パターンを効率良く発見する問題についても検討する予定である．自然言語処理の精度が，例題であげたアプリケーションに対する有用性に，どの程度，影響を及ぼすかについて調べることも興味深い．さらに，文書や半構造データからより複雑なパターンを発見する問題についても調べていきたいと考えている．

参 考 文 献

1) Agrawal, R., Imielinski, T. and Swami, A.: Mining association rules between sets of items in large databases, *Proc. ACM SIGMOD Conference on Management of Data*, pp.207–216 (1993).
2) Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules, *Proc. International Conference on Very Large Data Bases*, pp.487–499 (1994).
3) Srikant, R. and Agrawal, R.: Mining Generalized Association Rules, *Proc. International Conference on Very Large Data Bases*, pp.407–419 (1995).
4) Han, J. and Fu, Y.: Discovery of Multiple-Level Association Rules from Large Databases, *Proc. International Conference on Very Large Data Bases*, pp.420–431 (1995).
5) Park, J.S., Chen, M.-S. and Yu, P.S.: An Effective Hash-Based Algorithm for Mining Association Rules, *Proc. ACM SIGMOD Conference on Management of Data*, pp.175–186 (1995).
6) Brin, S., Motowani, R., Ullman, J. and Tsur, S.: Dynamic Itemset Counting and Implication Rules for Market Basket Data, *Proc. ACM SIGMOD Conference on Management of Data*, pp.255–264 (1997).
7) Bayardo, R.J.: Efficiently Mining Long Patterns from Databases, *Proc. ACM SIGMOD Conference on Management of Data* (1998).
8) Agrawal, R. and Srikant, R.: Mining Sequential Patterns, *Proc. International Conference on Data Engineering* (1995).
9) Srikant, R. and Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements, *Proc. International Conference on Extending Database Technology* (1996).
10) Abiteboul, S.: Querying semi-structured data, *Proc. International Conference on Database Theory*, pp.1–18 (1997).
11) Buneman, P.: Semistructured Data, *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (1997).
12) Nirenburg, S. (Ed.): *Machine Translation*, Cambridge University Press, Cambridge (1987).
13) Nagao, M., Tsujii, J. and Nakamura, J.: The Japanese Government Project for Machine Translation, *Machine Translation Systems*, Slocum, J. (Ed.), pp.141–186, Cambridge University Press, Cambridge (1988).
14) Buneman, P., Davidson, S., Hillebrand, G. and Suciu, D.: A Query Language and Optimization Techniques for Unstructured Data, *Proc. ACM SIGMOD Conference on Management of Data* (1996).
15) McHugh, J., Abiteboul, S., Goldman, R., Quass, D. and Widom, J.: LORE: A Database Management System for Semistructured Data, *Proc. ACM SIGMOD Conference on Management of Data* (1997).
16) Nestorov, S., Abiteboul, S. and Motwani, R.: Inferring structure in semistructured data, *Proc. Workshop on Management of Semistructured Data* (1997).
17) Wang, K. and Liu, H.: Schema Discovery for Semistructured Data, *Proc. International Conference on Knowledge Discovery and Data Mining*, pp.271–274 (1997).
18) Nestorov, S., Abiteboul, S. and Motwani, R.: Extracting Schema From Semistructured Data, *Proc. ACM SIGMOD Conference on Management of Data*, pp.295–306 (1998).
19) Lewis, D.D. and Jones, K.S.: Natural Language Processing for Information Retrieval, *Comm. ACM*, Vol.39, No.1, pp.92–101 (1996).
20) Walker, S.: *The Okapi online catalogue research projects*, pp.424–435 (1997).
21) Salton, G. and Buckley, C.: *Improving Retrieval Performance by Relevance Feedback*, pp.353–364 (1997).
22) Strzalkowski, T. and Vauthey, B.: Information Retrieval Using Robust Natural Language Processing, *Proc. Association for Computational Liguistics* (1992).
23) Arimura, H. and Shimozono, S.: Efficient discovery of optimal word association patterns in large text databases, *New Generation Comput-*

ing, Vol.18, pp.49–60 (2000).
24) 那須川哲哉，諸橋正幸，長野　徹：テキストマイニング—膨大な文書データの自動分析による知識発見，情報処理学会誌，Vol.40, No.4, pp.358–364 (1999).
25) Muraki, K., Doi, S. and Ando, S.: Description of the Veniex System as used for MUC-R, *Proc. MUC5*, pp.147–159 (1993).
26) McCreight, E.M.: A Space-Economical Suffix Tree Construction Algorithm, *J. ACM*, Vol.23, No.2, pp.262–272 (1976).
27) Hopcroft, J.E. and Karp, R.M.: An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs, *SIAM Journal of Computing*, Vol.2, No.4, pp.225–231 (1973).
28) 黒橋禎夫：“結構やるな，KNP”，情報処理学会誌，Vol.41, No.11, pp.1215–1220 (2000).
29) Nasukawa, T., Morohashi, M. and Nagano, T.: Customer Claim Mining: Discovering knowledge in vast amounts of textual data, Technical Report RT0319, IBM Tokyo Research Laboratory (1999).
30) van Rijsbergen, C.J.: *Information Retrieval*, 2nd edition, Butterworhts, London (1979).

付　　録

A.1 人工的データの生成

実験のために，構築した人工的データの生成ツールは，文献 2) のデータ生成手法を基本としている．そのパラメータを**表 6** に示す．

データの生成方法は次のとおりである．はじめに，次に生成するトランザクションのグループ数を決定する．その数は，平均が $|T_g|$ のポアソン分布から取り出す．トランザクションは一連の潜在的な頻出パターンである．パターンは，潜在的に頻出するパターン P の集合から選ばれる．各パターンは，corruption レベル c を持つ．パターンをトランザクションに付加するとき，0 から 1 までの一様分布乱数を発生させ，c より小さければパターンからグループを削除し，これを c よりも大きな数字が発生するまで繰り返す．グループがトランザクションに割り当てられるとき，グループのいくつかのアイテムは同じ方法で削除される．パターンがトランザクションのサイズに収まらない場合があるときトランザクションに入れるか，次のトランザクションへ割り当てる．どちらにするかは，2 値のサイコロでランダムに決める．

パターンは，P より選ばれる．P のパターンの総数は，$|P|$ である．パターン中のグループ数は，平均が P_g のポアソン分布より選択する．最初のパターンを生成するために，グループのラベルはラベルの集合 L から選択される．各グループのアイテム集合は，アイテム集合 G から選択する．

表 6 人工データのパラメータ
Table 6 Parameters for synthetic data.

$\|T\|$	100,000	トランザクションの数
$\|T_g\|$	5	平均グループ数
$\|P\|$	1,000	P 中のパターンの数
$\|P_g\|$	5	パターン中の平均グループ数
$\|P_i\|$	5	グループ中の平均アイテム数
rept_level	変数	繰返しのための分布の平均
rept_var	0.1	繰返しのための分布の分散
$\|G\|$	1,000	G 中のアイテム集合の数
$\|L\|$	1,000	L 中のラベルの数
$\|I\|$	1,000	I 中のアイテムの数

次のパターンと直前のパターンを関連付けるため，直前のパターンをコピーし，0 から 1 までの一様分布の乱数を発生させ，c よりも小さければ 1 つのグループを削除する．これを c よりも大きな数が出るまで繰り返す．グループ中のアイテムも同様にして削除される．削除するグループとアイテムは，ランダムに選ばれる．残りのグループは，G よりランダムに選ばれる．グループのラベルを決めるために，乱数 r を用いる．r は，平均が *rept_level* で，分散が *rept_var* である正規分布から選ばれた数値である．0 から 1 の一様分布から取り出した乱数が r よりも小さければ，パターン中で直前に用いたラベルを割り当てる．これを r よりも大きい数値が出るまで繰り返す．r よりも大きな数値が出たら，L から選んだ新しいラベルを割り当てる．各パターンは，corruption レベル c を持っており，これは正規分布（平均 0.5，分散 0.1）から選ばれる．

L はサイズ $|L|$ の集合であり，各ラベルには，そのラベルが選択される確率に相当する重みがそれぞれ付けられている．その重みは，指数分布（unit mean）から選ばれ，L 内のすべてのラベルに対する重みの総和が 1 になるように正規化される．L 個の側面を持つ重み付きのコインを振ることで，次に選ばれるべきラベルは L から選ばれる．ここで，コインの各側面の重みとは各側面に相当するラベルの選ばれる確率である．

G はサイズ $|G|$ のアイテム集合の集合である．各アイテム集合の数は，平均 P_i のポアソン分布から選ばれる．最初のアイテム集合を生成するために，アイテムはアイテム集合 I からランダムに選ばれる．次のアイテム集合を直前のアイテム集合と関連付けるために，直前のアイテム集合をコピーし，0 から 1 の一様

分布乱数を発生させ，c より小さければ 1 つのアイテムを削除していく．これを c より大きな数値が出るまで繰り返す．c より大きな数値が出た場合には，I より選ぶ．アイテム集合 I は，L と同様にして生成される．

(平成 12 年 9 月 20 日受付)
(平成 12 年 12 月 27 日採録)

（担当編集委員 河野 浩之）

松澤 裕史（正会員）

1991 年早稲田大学理工学部電気工学科卒業．1993 年同大学院理工学研究科電気工学専攻修士課程修了．同年日本アイ・ビー・エム（株）入社．東京基礎研究所に所属．ソフトウェアの部品化・再利用，3D ソリッドモデリング，データマイニング，テキストマイニング等の研究に従事．人工知能学会，日本ソフトウェア科学会各会員．