ニュース文書からの頻出パターン抽出による知識獲得に関する研究

清水, 一宏 / SHIMIZU, Kazuhiro

(発行年 / Year)
2008-03-24
(学位授与年月日 / Date of Granted)
2008-03-24
(学位名 / Degree Name)
修士(工学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

2007年度　修 士 論 文

ニュース文書からの頻出パターン抽出による知識獲得に関する研究

STUDIES ON KNOWLEDGE DISCOVERY BY FREQUENT PATTERN EXTRACTION FROM NEWS DOCUMENTS

指導教官　三浦 孝夫 教授

法政大学大学院工学研究科
電気工学専攻修士課程

06R3112　清水 一宏
Kazuhiro SHIMIZU

目 次

第1章　序論

1.1　問題の背景

近年のインターネットや Web 技術の発達，計算機の高性能化により，文章や画像，映像，音楽などの多種多様なデータを扱う機会が増え，誰もが容易に入手し利用できるようになった．しかし，これらのデータは日々爆発的に増加しており，その多くが，手つかずで未整理のまま蓄積されている．利用者が，この膨大なデータの山から，一つ一つのデータ内容を吟味し，重要な情報が何かを把握するのは到底不可能であり，計算機による支援が急務となっている．

データマイニング (data mining) は，このような大量のデータの中から，データ間の隠れた相関関係や頻繁に出現するパターンを効率よく見つけ出す，知識発見技術の一つである．これまで，データマイニング手法は，明示的で整った構造を持つ関係データベースを対象の中心とし，統計解析手法や人工知能分野での機械学習等を応用した研究が盛んに行われてきた．

また最近では，電子メールや電子掲示板，Blog 記事等に代表されるように，Web 上を中心として，文書データを介したコミュニケーション，情報交換が積極的に行われている．そのため，これらを構成する文章など，文字列テキストの特徴を数量化し，データマイニング手法を適用することで効率よく解析を行うことを目指す，**テキストマイニング** (text mining) に関する研究に注目が集まっている．

テキストとは語の並びで構成される，意味を有する系列データ (sequence data) である．テキストマイニング手法の特徴は，頻度 (重要な語は何度も生じる) や共起性 (類似した語は同時に近くに表れる) にある．出現頻度の高いパターン (語の並び，フレーズ) を抽出し，内容の要約 (抽象化) やラベル付けを行うことができる．

ニュース文書は文書データの特性を端的に示す，その典型的な例の一つである．新聞や TV などで流れるニュースは，それぞれ発行 (配信) 時間を持ち，無限に続く文書の流れとして表現することができる．このように，時系列順に生成・入力されるデータの流れを (時系列) データストリームといい，ニュース記事や放送内容のアノテーション・トランスクリプションを扱うものを，**ニュースストリーム**と呼ぶ．このデータに対し，テキストマイニング手法を適用することで,「全体でどのような話題が語られているのか」といったことや,「今，最も注目されている話題は何か」といった，ニュース全体の傾向を理解する上で手がかりとなる情報を，頻出パターンとして得ることができる．また，これらをニュース内容の自動要約や，自動分類に応用することで，利用者の内容把握に関する負担を一層軽減することが期待できる．ニュースストリーム

は他と際立って異なる特性を有する．短時間の間に集中して同一トピックの記事が配信され，それらが連続して関連しあう．頻出する語よりも語同士の関連や共起性が重要な意味を有することが多い．また，過去に生じた事件との関連を繰り返し報じられるが，それらも急速に興味が失われていく．

しかし，文書データのような非定型データを計算機で扱うのは難しい．決められた構造を持たないために，著者や用途によってデータの表現方法が異なり，個別内容を示すのか話題の枠組みを考慮しているのかを区別することが難しい．このことが，データの定型性を利用して効率よく処理を行うデータマイニング手法の，テキストマイニングへの適用を困難にしている．テキストの特徴を考慮し，計算機上で上手に処理を行うためのアイデアが必要である．

また，ニュースストリームのようにデータストリームを扱う問題では，“時間”をどのように扱うかが重要となる．例えば，ニュース文書中で語られる話題はその発行時間に対応することが知られている．しかし，その発行時間は一定ではなく，さらに時間経過とともに傾向が次々と変化をするという特性を持つ．このため，限られた計算機資源のもとで，時間に対して柔軟でかつ即応性を持つ手法を提案することが課題となる．

一方，テキストマイニングにおいて，時間をかけ計算し得られた結果のほとんどが，利用者にとって自明なパターンとなることが少なくない．これは，現在提案されている手法の多くが，主に頻度をパターンの重要度として扱うことが原因であると考える．確かに頻繁に繰り返し出現するパターンは，データ中の特徴を捉えたものになると考えられるが，それが必ずしも利用者にとって興味の対象になるとは限らない．このようなパターンが結果に含まれることで，それがノイズとして振る舞い，利用者の内容解析・把握の妨げとなる可能性がある．従って，価値のあるものだけにフィルタすることが望ましいが，これは利用者の目的や考え方によって変化するものであり，柔軟性をもって対応しなければならない問題である．

1.2 扱う問題

本研究では，ニュースストリームようなニュース文書集合から，利用者が内容分析・把握をする際の負担軽減を目指し，重要な話題を頻出パターンとして効率よく抽出するための手法の提案を行う．

まずはじめに，静的なデータ環境下における頻出パターン抽出を考える．テキストのような系列データを扱う問題に対しての一般的な問題を解決し，データマイニング手法適用することで，効率のよい手法の提案を行う．一般に利用者は頻出基準となるパラメタを変更し結果を比較するが，このように再計算を伴う環境についても考慮する．次に，上記の手法を一般化し，動的なデータに適用することを考える．時間の経過を考慮し，ニュースストリームに対して効果的な頻出パターン抽出手法を提案する．最後に，利用者の好みを考慮することで，利用者が興味を持つ，有用な頻出パターンの抽出手法を提案する．

1.2.1 静的なデータ環境下における頻出パターン抽出

テキストのような系列データから，出現頻度の高いパターンをすべて抽出することは容易ではない．これは，通常，テキストが非常に疎なデータであるためで，パターンの出現頻度が全体的に小さくなってしまうことも珍しくない．また，探索空間が巨大となり，組合わせ的な探索問題となることが知られている．

この問題を解決するアイデアとして，頻出パターン抽出を高い抽象レベルで行うことが考えられる．例えば，次の例を考える．

“サッカーとフットサルの試合をした．明日，フットサルとサッカーの試合をする”

上記の例では，パターンとして “サッカー, フットサル, 試合” と “フットサル, サッカー, 試合” がそれぞれ 1 回ずつ出現する．これを，“サッカー” と “フットサル” の順序を考えず，まとめて “足を使った球技” と一つ上の抽象レベルで解釈する．こうすることで，“[サッカー, フットサル] 試合” というパターン (**選言パターン**と呼ぶ) が 2 回出現するものとして検出することができ，出現頻度の小さなパターンに関しても抽出が期待できる．

しかし，選言パターンを抽出する際には膨大な数の候補が生成され，候補を絞って収束させるには何度もデータベースを走査する必要があり，記憶装置上に格納されたデータに対し，数多くのアクセスが生じる．対処法として，サンプリング手法や特殊なデータ構造の利用による性能改善が考えられるが，データの分布特性に依存してしまうという欠点がある．

これまでの主要な研究の多くは APRIORI に基づいている [12, 17]．APRIORI アプローチでは，パターンの出現頻度が逆単調性 (anti-monotonicity) を満たすという性質を利用し，これにより大幅に探索量を下げることができる．ところがテキストのような系列データでは逆単調性が成り立たず，APRIORI アプローチを利用することができない．

そこで本研究では，まず，高い抽象度を表すことが可能な選言パターンを導入し，既に提案されている手法 [37] に基づき，単一系列データ上で逆単調性を満たす出現尺度を考える．その上でデータマイニング手法である APRIORI を適用し，探索量を削減することで，静的なデータ環境下で効率よく頻出パターン抽出を行う手法の提案を目指す．

1.2.2 頻出パラメタ変更に起因する再計算問題

先に述べた APRIORI を用いたテキストマイニングは探索問題となる．逆単調性の性質を用いても，繰り返し計算に耐えるためには多くの計算機資源を必要とし，結果を得るのに数時間，数日を要することも珍しくない．そのため，実際に適用する場合には条件を変更し繰り返して探索を行い，小規模な問題あるいはサンプリング手法などによ

り，これを回避する必要がある [28]．実務などで利用される OLAP(Online Analytical Processing) では，多種大量のデータを扱うため，(例えば APRIORI では支持度，確信度などの) 種々のパラメタを予め設定することは困難な作業となる．

一般に，条件を充足するパターン抽出問題では，データを繰り返し走査する必要がある．各走査で多くの読み込みが発生する場合，それがボトルネックとなる．また，計算結果の再利用ができないため，利用者がパラメタの設定を変更するたびに再計算が必要となり，効率が悪い．

これらの問題を回避するため，オンライン分析手法が提案されている [1]．ここでは，予めデータを前処理してその結果を保持し，これを繰り返して利用することにより負荷を軽減する．考え方自身は新しいものではないが，更新頻度が少ない状況，特に OLAP では相性が良い．この考えをテキストマイニングに適用することで，同様の効果が期待できる．

本研究では，テキストマイニングにオンライン分析手法 [1] の考え方を取り入れ，以前の計算結果を再利用することで，負荷の軽減を目指す．すなわち，特定の形式でデータを表現し，繰り返して生じる問い合わせに対して，実用的な時間内で頻出パターン抽出を行う方法の提案をする．

1.2.3 データストリーム環境下における頻出パターン抽出

上記で述べた手法は，静的なデータ環境下を対象としている．すなわち，完全に完成されたデータが与えられ，過去に遡って解析をするという立場をとる．しかし，この手法を動的で大規模なデータある，データストリームに対して適用するのは難しい．

静的なデータ環境下では，記憶装置上に蓄えられているデータを対象とする．しかしデータストリームは，新しいデータが絶え間なく生成される大量かつ潜在的に無限なデータの流れである．このため，全てのデータを記憶装置に蓄積し，それから解析を行うというプロセスをたどるのは，現実的に不可能である．

一般に動的データストリームは，(1) 大量，(2) 高速，(3) データ分布の動的変化，(4) 連続的という特徴を持つ．これを対象とする手法には，限られた計算資源で高速かつ長期間の解析が要求される．さらに，任意の時点において蓄積された個々のオブジェクトを問い合わせるのではなく，データ全体の傾向や動向を正しく反映した解を出力する手法であることが望ましいとされている．

これに加え，本研究で扱うニュースストリーム環境下では，絶えず新しい記事が発生しており，新しい記事と過去の記事が混在している．利用者は，一般に新しい記事の内容に興味を持っていることが多く，両データを同等の重みで扱うことはできない．また，その発行間隔は不定であり，記事の発行時刻をどのように扱うかが問題となる．

この問題を解決する方法として，例えば，出現頻度に対して記事の発行時刻を基に重み付けをすることが考えられる．新しい記事に出現するものほど大きな重みを与え，過去のものに対しては減衰させるようなモデルを導入することで，計算結果に時間変化の影響を反映することができる．

本研究では，ニュースストリームから，時間変化の影響を考慮した上で，過去から現在までの期間で頻出となるパターンを，近似解として高速に抽出する手法の提案を目指す．すなわち，出現頻度を確率的に誤差保証しながら，特定の形式のデータ構造を構築し，過去の頻度に対して重みを与えることで，任意の時点における頻出パターン抽出を行う手法を提案する．

1.2.4 利用者の好みの考慮

これまで述べてきた手法では，効率面での問題は解決できるが，パラメタを与えて計算した予備計算結果中に，利用者が興味を持つパターンが含まれているか否かは，実際の抽出を行うまでわからない．つまり，時間をかけ得られた結果が全く役に立たない可能性がある．また，抽出を行った結果，膨大な数のパターンが生成されることも珍しくない．利用者はパラメタを大きくすることでその数を絞り込もうとするが，すると今度は中程度の頻度で興味のあるパターンが失われてしまう．

一般に，高頻度なパターンの多くは利用者にとって自明であることが多い．例えば，ニュース記事などでは `said` というパターンが高頻出となるが，これは，専門家の発言や意見等を引用するために “He said ...”, “She said...” という表現が良く使われるからであり，利用者には自明なパターンとなる．このように，頻度のみで利用者が興味を持つパターンであると評価するのは困難であり，別の基準が検討されている [16].

しかし，利用者がどのようなパターンに興味を持つのかは，個々の利用者の目的や考え方によって変化するものであり，システム側が勝手に判断することはできない．そこで，利用者側から事前に好みに関する情報を提供してもらい，それを参考に出現頻度に対して重み付けを行うことを考える．こうすることで，パラメタを大きく設定した場合でも，中頻度で興味のあるパターンが強調され，抽出の期待ができる．また同時に，自明なパターンに関しても，負の重みを与えることでノイズフィルタのような役割を果たし，高頻度なパターンとしての抽出を抑制できるだろう．

そこで，本研究では利用者の好み情報をキーワードとして与え，これらをパターンの抽出結果に反映し，頻度だけに依らない選言パターン抽出方法を提案する．具体的には，利用者が興味を持つパターンの分布と対象パターンの分布の類似度に基づき重み付けを行い，パラメタの動的推定や，利用者の好みを反映したパターンを含むデータ構造を構築する．

1.3 論文の構成

本研究では，以上の問題について以下の構成で論じる．第 2 章では，単一系列データ上の選言パターンとその逆単調性について論じる．第 3 章では，選言パターン抽出のオンライン分析手法について論じる．第 4 章では，選言パターンを木パターンに拡張し，半構造データからの頻出パターン抽出を行う．第 5 章では，ニュースストリーム

からの選言パターン抽出のためのオンライン型単一パスアルゴリズムを提案する．第6章では，ニュースストリームから利用者の好み情報に基づいた選言パターン抽出手法を論じる．第7章で結論とする．

1.4 発表論文

1. 清水一宏, 三浦孝夫: “単一系列データ上の系列選択パターンと逆単調性”，データ工学ワークショップ (*DEWS*), 2005.
   単一系列データ上で，系列選択の頻出パターン発見手法を論じる．このため，高速な自動抽出方法を提案して逆単調性を満たす出現尺度を与え，APRIORI 流の計算により有用性を検証する．

2. 清水一宏, 三浦孝夫: “Disjunctive Sequential Patterns on Single Data Sequence and its Anti-Monotonicity”, *International Conference on Machine Learning and Data Mining* (*MLDM*), pp.376-383, 2005.
   単一系列データ上で，系列選択の頻出パターン発見手法を論じる．このため，高速な自動抽出方法を提案して逆単調性を満たす出現尺度を与え，APRIORI 流の計算により有用性を検証する．

3. 清水一宏，三浦孝夫: “選言パターン抽出のオンライン分析”, 日本データベース学会 Letters (*DBSJ Letters*) Vol.4, No.3, pp.9-12, 2005.
   オンライン分析手法に基づいて，頻出な選言パターンの実用的発見の為のデータ構築法を提案する．また，実験によりその有用性を検証する．

4. 清水一宏, 三浦孝夫: “選言的木パターンマイニング”, データ工学ワークショップ (*DEWS*), 2006.
   選言パターンの考え方を木パターンに拡張し，半構造データ上で逆単調性を満たす出現尺度を用いることで，高速な選言的木パターンマイニング手法を提案する．また実験によりその有用性を検証する．

5. 清水一宏, 三浦孝夫: “Online Analysis for Disjunctive Sequential Patterns”, *ADBIS Workshop on Data Mining and Knowledge Discovery* (*ADMKD*), pp.61-72, 2006.
   オンライン分析手法に基づいて，頻出な選言パターンの実用的発見の為のデータ構築法を提案する．また，実験によりその有用性を検証する．

6. 清水一宏, 三浦孝夫: “Mining Single Sequence At Once Using Disjunctive Tree Patterns”, *Hybrid Intelligent Systems* (*HIS*), (*CD-ROM*), 2006.
   複数文書からの頻出選言パターンの高速抽出手法を論じる．選言パターンの考え方を木パターンに拡張し，半構造データ上で逆単調性を満たす出現尺度を用いることで，高速な選言的木パターンマイニング手法を提案する．

7. 清水一宏, 三浦孝夫: “ニュースストリームからの選言パターン抽出”, データ工学ワークショップ (*DEWS*), 2007.
   ストリームに対する選言パターン抽出方法を提案する，ストリームデータは時間に対する重みを有しており，精度を確率的に保証しながらオンライン分析手法の構築とその性能を評価している．

8. 清水一宏, 塩谷勇, 三浦孝夫: “Mining Disjunctive Sequential Patterns from News Stream”, *8th Intn'l Conf. on Intelligent Data Engineering and Automated Learning* (*IDEAL*), pp.630-642, 2007.
   ストリームに対する選言パターン抽出方法を提案する．ストリームデータは時間に対する重みを有しており，精度を確率的に保証しながらオンライン分析手法の構築とその性能を評価している．

9. 清水一宏, 三浦孝夫: “利用者の好み基づくニュースストリームマイニング”, データ工学ワークショップ (*DEWS*), 2008.
   パターン頻度だけでなく利用者の好み情報を用いることで，対象となる選言パターンに対して重み付けを行い，それらのパターンを含むデータ構造構築のための動的なパラメタを推定する手法を提案する．

第2章　単一系列データ上の系列選択パターンと逆単調性

本論文では，単一系列データ上での系列選択の頻出パターン発見手法を論じる．このため，高速な自動抽出方法を提案して逆単調性を満たす出現尺度を与え，APRIORI 流の計算によりその有用性を検証する．

2.1　前書き

近年, 文書データ分析にデータ発見手法を使用することが多くなってきた．従来，文書データは定性的であり量的に判断することが無かった．このため，統計的推定やデータ発見手法など定量的な分析に基づく分析は難しい [12, 17]．しかし，テキストの特徴を数量化し解析しようとする研究（テキスト発見）が注目されるにつれ，従来提案されてきた手法についても適用可能であると考え出されている．

データ発見手法の特徴は，頻度 (重要な項目は何度も生じる) や共起性 (複数の項目が同時に生じる) にある [39, 40]．例えば，書店の購買情報を分析することで,「“蛇にピアス”を購入する客は同時に “蹴りたい背中”も購入する」といった相関性[1]の検出することができ，書籍の配置やキャンペーン方法を工夫するきっかけとなる．

この考え方は多方面に拡張できる．系列データ (sequence data) とは情報を時系列などのある順序で並べたものであり，購買パターンを動作として検出することができる．例えば，書店の購買情報を分析することで,「“蛇にピアス”を購入する客は 1 ヶ月以内に “蹴りたい背中”も購入する」といった相関性検出により，DM など販売促進の方法を示唆することができる．この他，Web アクセスパターンの解析，医療診断，DNA 系列の解析など幅広い応用が考えられる [18, 41, 21].

このうちテキスト発見は，近年注目を浴びている研究分野である．テキストは語の並びで構成された意味を表し，系列データの一種である．テキスト発見手法の特徴は, 頻度 (重要な語は何度も生じる) や共起性 (類似した語は同時に近くに表れる) にある．頻度の高いパターン (語の並び，フレーズ) を抽出し，テキストの要約 (抽象化) や重要な事象のラベル付けを行うことができる．

[1] “蛇にピアス”金原 (かねはら) ひとみ著と “蹴りたい背中” 綿矢 (わたや) りさ著は 2003 年第 130 回芥川賞を同時に受賞した文学作品であり，両著者が 20 才前後であったことから注目を浴びた．

長大なテキストから発生頻度の高い語句すべてを発見することは容易ではない．探索空間が巨大であり組合わせ的な探索問題となっている．例えば，次の例を考える．

(1) “友人の兄の母に会った．今日，兄の友人の母に会った”
(2) “赤くて大きい旗を見た．今日，大きくて赤い旗を見た”

明らかに，(1) で論じているのは異なる人物であるが，(2) で論じているのは同じ旗である．頻出パターンを発見するとき，後者の場合では複数回カウントすべきである．テキスト発見技法ではこれを “[赤い，大きい] 旗” というパターンとしてを検出したい．すなわち，“赤い旗” でも “大きい旗” ではなく，“赤い” と “大きい” の語順 (permutation) を無視したパターンとして捕らえたい．本研究では，これを**選択** *(disjunctive)* **パターン**という．Kleene 閉包を加え正規表現を考察できるが，計算処理量が増大し，本稿では議論しない．選択パターンを発見するとき，膨大な数の候補を生成すべきであり，そのひとつが “蛇にピアス　蹴りたい背中”である．候補を絞って収束させるには何度もデータベースを走査する必要があり，ハードディスク上に格納されたデータに数多くのアクセスが生じる．しかし，サンプリング手法や特殊なデータ構造の利用ではデータの分布特性によって大きく性能に差が生じる．

これまで主要な研究は APRIORI に基づいている [12, 17]．このことで探索量を大幅に下げることができるが，系列パターンについては十分ではない．本来，APRIORI アプローチでは，パターン q がパターン p の “部分”であるとき，p に適合する系列データは必ず q にも適合する，という性質 (パターンの逆単調性) により探索量を削減する．ところがテキストでは逆単調性が成り立たず，もはや APRIORI アプローチを利用できない．
例えば テキスト `"aabbba"` においてパターン `"ab"` の適合回数は 6 回, `"a"`，`"b"` は共に 3 回, `"[ab]"` は 9 回である．

単一の長大な系列データ S に対して，パターン p でその出現尺度 $\mathcal{M}_S(p)$ がある整数値 m 以上のものをすべて検出することを論じる．従って，発生回数を意識したカウント方法 $\mathcal{M}$ が問題である．本稿では，既に提案されている手法 [37] を拡張し，選択パターンを APRIORI で取り扱う枠組みの提案を行う．2 章で問題を定式化し，3 章で逆単調性を満たす出現尺度を導入する．続く 4 章でこれを用いたテキスト発見アルゴリズムを述べ，5 章でいくつかの実験結果を示す．

2.2　系列データからのパターン発見

本稿では，語 (word) を基本単位 (item) として扱う．アイテム集合 $I = \{i_1, .., i_L\}, L > 0$ の要素をアルファベット，**系列データ** (テキスト) $S = s_1...s_m, m > 0$ とはアイテムの順序リストである．S には同じアイテムが複数回生じてよく，S に含まれる (重複を含めた) 語の数 m に対し，S を m-系列データという．

選択パターン (あるいは単にパターン) p とは $t_1t_2...t_n$ で表され，各 t_i はアルファベット a または選択 $[a_1a_2...a_m], m > 0$ (各 a_j は相異なるアルファベット) である．パ

ターン $p = t_1 t_2 ... t_n$，$q = v_1 v_2 ... v_m, m \leq n$ があるとき，q が p の**部分パターン** である ($q \sqsubseteq p$ と記す) とは，$1 \leq j_1 < ... < j_m \leq n$ があり，各 v_k は t_{j_k} に対応して次を満たす (混乱の無い限り $v_k \sqsubseteq t_{j_k}$ と記す)：

v_k がアルファベット a のとき， $t_{j_k} = a$ もしくは a を含む選択
v_k が 選択 $[a_1 a_2 ... a_m]$ のとき，$t_{j_k} = [b_1 b_2 ... b_l]$ でかつ $\{a_1, .., a_m\} \subseteq \{b_1, .., b_l\}$

例 2.1 “ac” は “abcd” の部分パターンである．同様に，“[ac]” は “[abcd]” の，“bd” は “[ab]b[cd]de” の，“b” は “[ab]” の，そして “ac”は “[ab][cd]” の部分パターンである．
しかし，“ab” は “[ab]” の部分パターンではない．また “[ab]” は “ab” の部分パターンではない．

系列データ $S = c_1 c_2 ... c_m$ にパターン $p = t_1 t_2 ... t_n$ が**適合する** (match) とは，t_1 がアルファベット a_1 のとき，$t_1 = a_1 = c_{i_1}, 1 \leq i_1 \leq m$ でありかつ 部分パターン $t_2 ... t_n$ が系列データ $c_{i_1+1} ... c_m$ に適合するときを言う．t_1 が選択 $[a_1 a_2 ... a_m]$ のとき $a_1, ..., a_m$ のある並びからなるパターン $a_{j_1} ... a_{j_m}$ が系列データ $c_1 ... c_{i_1}$ に適合し，かつ 部分パターン $t_2 ... t_n$ が系列データ $c_{i_1+1} ... c_m$ に適合するときを言う．

例 2.2 系列データ S が aabbba であるとき，パターン a は S に 3 回適合する．また，パターン ab は 6 回適合する．一方 [ab] は 9 回適合する．

系列データ S に関してパターンから非負整数への関数 $\mathcal{M}$ が逆単調性 (Anti Monotonicity, AM) を満たすとは，パターン p, q に対して $q \sqsubseteq p$ のとき $\mathcal{M}_S(q) \geq \mathcal{M}_S(p)$ が成り立つときを言う．以下で考慮する系列データは単一であり S を略す．
逆単調な $\mathcal{M}$ が与えられ，また最小頻度 m が与えられているとする．このとき，$\mathcal{M}(q) < m$ ならば $q \sqsubseteq p$ となる p は $\mathcal{M}(p) \geq m$ ではない．この性質を用いれば，部分パターンの探索範囲を減少させることができる．これが APRIORI に基づく探索アルゴリズムであり [2]，以下の手順で頻出パターンを計算することができる．

(1) 「最小のパターン」で頻出のものを探す
(2) 大きいパターン p で，そのすべての部分パターンが頻出である　ものを探す (候補パターン集合を得る)．候補がなくなれば終了する
(3) S をスキャンして頻出なものだけを求める．(2) へ

単一の長大な系列データ S に対して，選択パターン p でその出現尺度 $\mathcal{M}(p)$ がある整数値 m 以上のものをすべて検出することが，本論文の目的である．しかし，$\mathcal{M}$ で逆単調なものを見つけるのは簡単ではない．例えばパターン p に対して系列データ S に適合する回数を $\mathcal{M}(p)$ とする．このとき p の部分パターン q で $\mathcal{M}(q) \geq \mathcal{M}(p)$ を満たさないものがあるのは例 2.2 に示した．

2.3 逆単調出現尺度

単純な適合頻度では逆単調性が得られないため，系列の先頭要素の適合頻度を考察する．

系列データ $S = s_1 s_2 ... s_r$，パターン p を $t_1 t_2 ... t_n$ とする．このとき **系列先頭頻度** $H(S,p)$ を次のように定義する．

$$H(S,p) = \sum_{i=1}^{r} Val(S,i,p) \tag{2.1}$$

ただし $Val(S,i,p)$ は次を満たすとき 1, さもなければ 0 であるとする:

> $S(i)$ を S の i 番目からの接尾辞 $s_i ... s_r$ とする．t_1 がアルファベット a のとき，$s_i = a$ かつ $t_2 t_3 ... t_n$ が $S(i+1)$ に適合する．t_1 が選択 $[a_1 a_2 ... a_m]$ のとき，$s_i = a_j$ となる j があり (例えば $j = 1$)，$[a_2 a_3 ... a_m] t_2 ... t_n$ が $S(i+1)$ に適合する．

$H(S,p)$ は S またはその接尾辞において p が先頭から適合する回数を表している．

例 2.3 (1) S を bbba とするとき，$p = ba$ のとき $H(S,p) = 3$，実際の適合回数は 3 回．$p = a$ のとき $H(S,p) = 1$，実際の適合回数は 1 回である．定義から $a \sqsubseteq ba$ であるが $H(S,a) > H(S,ba)$ ではない．
(2) S を aabbba とする．$p = ab$ のとき $H(S,p) = 2$，実際の適合回数は 6 回．$p = ba$ のとき $H(S,p) = 3$，実際の適合回数は 3 回．$p = [ab]$ のとき $H(S,p) = 5$，実際の適合回数は 9 回 $p = a$ のとき $H(S,p) = 3$，実際の適合回数は 3 回である．定義から $a \sqsubseteq [ab]$ であるが $H(S,a) > H(S,[ab])$ ではない．

この例が示すように，系列先頭頻度 $H(S,p)$ は逆単調性を満たさない．p の先頭での適合回数は，後続系列での適合した回数を無視している (そうでなければ逆単調にならない)．そこで p のすべての部分パターン q を調べ，その系列先頭頻度 $H(S,q)$ のうちの最小の値を**全体出現頻度** $D(S,p)$ と呼ぶ．

$$D(S,p) = \mathrm{MIN}\{H(S,q) | q \sqsubseteq p\} \tag{2.2}$$

定理 2.1 $D(S,p)$ は逆単調性を満たす．

証明 パターン p, q が $q \sqsubseteq p$ を満たすとき，定義より $D(S,p) = MIN\{H(S,r) | r \sqsubseteq p\}$, $D(S,q) = MIN\{H(S,r) | r \sqsubseteq q\}$ であるから，$q \sqsubseteq p$ より $D(S,q) \geq D(S,p)$ となる．*Q.E.D.*

例 2.4 (1) S を bbba とするとき，$p = ba$ のとき $D(S,p) = 1$，$p = a$ のとき $D(S,p) = 1$ である．

(2) S を aabbba とする．$p = ab$ のとき $D(S,p) = 2$，$p = ba$ のとき $D(S,p) = 3$，$p = [ab]$ のとき $D(S,p) = 3$，$p = a$ のとき $D(S,p) = 3$ である．
(3) S を caabbbc, $p = ab$ とすれば $H(S,p) = 2, D(S,p) = 2$ である．$p = ac$ とすれば $H(S,p) = 2, D(S,p) = 2$ である．また $p = [ac]$ とすれば $H(S,p) = 3, D(S,p) = 2$ である (実際の適合頻度は 4 回)．ac や $[ac]$ の部分系列 a, c が分離して生じていても適合するとみなすため，系列先頭頻度や適合回数とは合わない．

定理 2.1 から，長さ n のパターン p の全体出現頻度を得るには，p のすべての部分パターンの系列先頭頻度を調べればよい．次の定理から，この計算は p の接尾辞 (n 個存在) のうち，長さの小さいものから順にその最小値を決定すればよいことがわかる．

定理 2.2 系列データ S, パターンを p とすると，$D(S,p) = \mathrm{MIN}\{H(S,p), D(S,p(2))\}$

証明 $S = s_1 s_2 ... s_m$, $p = t_1 t_2 ... t_n$, $p(i) = t_i ... t_n$ とする．$D(S,p) = MIN\{H(S,p(i)) | i = 1, ..., n\}$ を言えばよい．

p の任意の部分パターンを $q = u_1 u_2 ... u_k$ とする．仮定より $1 \leq j_1 < ... < j_k \leq n$ で $u_i \sqsubseteq t_{j_i}, i = 1, ..., k$ *(*アルファベットなら同一，選択なら部分集合*)* となるものがある．

q の位置 i での右拡張 q' を次のように定義する *($i > 0$):*

> q' は次のいずれかの形をしている
> *(i)* $u_1 u_2 ... u_i v u_{i+1} ... u_k$, つまり u_i の直後にパターン v が挿入されている
> *(ii)* $u_1 u_2 ... u'_i ... u_k$, つまり u_i が u'_i に変更され $u_i \sqsubseteq u'_i$ となっている

このとき $H(S,q) \leq H(S,q')$ が成り立つ．すなわち，$Val(S,i,q') = 1$ ならば必ず $Val(S,i,q)$ をいう．実際，*(i)* による右拡張なら，適合検査時にこれを無視すれば $Val(S,i,q) = 1$ となる．*(ii)* による右拡張なら，適合検査時に新たに拡張されたパターンを無視すれば $H(Val,i,q) = 1$ となる．

t_{j_1} は q を何度か右拡張したものなので $H(S,q) \geq H(S,t_{j_1})$ となる．これより，どの部分パターン q でも系列先頭頻度値でそれより小さい $t_{j'}$ が存在する．*Q.E.D.*

2.4 実験

2.4.1 準備

実験データとして，NTCIR-3 特許検索タスクコレクション中の「日本国英語特許出願抄録データ PAJ」を使用する．これは 1995 年から 1999 年までの JAPIO 出願抄録を英語に翻訳したコーパスであり，コーパス中の抄録はそれぞれ特許出願日時順に並んでいる．今回は，1995 年の特許出願抄録データから 1000 件分の特許内容要約文を用い，各要約文について不要語 (stop word) の削除および単語のステミング [36] を行った

ものを実験データとして使用する．以下に実験データの一部を示す.

```
...control adjust speed thresh depth regul grain culm state oper load
combin shape initi puls power high load devic regul control thresh depth
grain culm detect thresh depth sensor...
```

2.4.2 実験の手順と評価

本実験は [2] の APRIORI 法を転用し，実験データから頻出パターンの抽出を試みる．この結果と，[37] の系列全体頻度 $T\text{-}freq(S,p)$ の手法で得られる頻出パターンとを比較し，どちらの手法がより多くの頻出パターンを抽出できるかを調べる．生成される候補パターンはそれぞれ，系列全体頻度 (a,ab,ba,abc,bac...)，全体出現頻度 (a,[ab],[ab]c,[abc]d....) とする．なお，全体出現頻度において [ab]c[de] 等の 1 つのパターン中に複数の選択が出現するものについては対象としない.

2.4.3 実験結果

最小出現頻度は両手法とも 2%(20 回) 以上とし，全体出現頻度，系列全体頻度それぞれの手法を用いて実験データから頻出パターンを抽出する実験を行う．実験結果を以下の表 2.1，図 2.1 に示す．表 2.1 では，全体出現頻度と系列全体頻度のそれぞれが抽出に成功した頻出パターンの数を示しており，図 2.1 ではその分布を示している．括弧内の数値は全体出現頻度でのみ抽出が成功したパターン数を示す．また，それぞれの手法において実際に抽出に成功した頻出パターンの一部を以下に示す.

[全体出現頻度]

・([medicin,patient]),([prepar,medicin]),([surfac, sheet])...
・([prepar,medicin]patient)...

[系列全体頻度]

・(medicin,prepar),(medicin,patient),(devic,capabl)...
・(provid,capabl,devic)...

2.4.4 考察

表 2.1 からも明らかなように，$n = 2$ および $n = 3$ において全体出現頻度は系列全体頻度と比べ，より多くのパターンの抽出に成功している．また，全体出現頻度のみでしか抽出されない頻出パターンが $n = 2$ の時 76 組，$n = 3$ の時 1 組あり，これは系列全体頻度では抽出されない，最小出現頻度より少ない出現頻度を持つパターンが，全体出現頻度では選択パターンとして抽出されるからだと考えられる．例えば，具体例

表 2.1: 特許出願抄録データから抽出された頻出パターンの数

n-頻出パターン	全体出現頻度	系列出現頻度
$n = 1$	207(0)	207
$n = 2$	121(76)	45
$n = 3$	3(1)	2
$n = 4$	0(0)	0

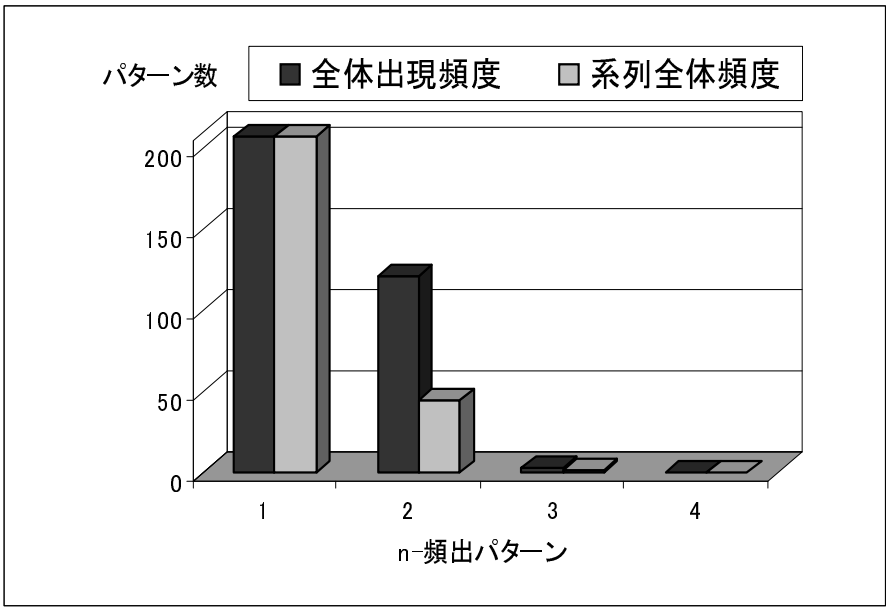

図 2.1: 特許出願抄録データから抽出された頻出パターンの数

として上記にも示した ([surfac, sheet] 22 回) や ([prepar,medicin]patient 21 回) は全体出現頻度のみでしか抽出されていない．実験で与えた最小出現頻度は両手法共に 2% と同条件であったので，この点において全体出現頻度は系列全体頻度と比べ優れていると言える．

2.5 関連研究

系列パターンを発見する問題 [3, 27] は，これまで相関性を検出する APRIORI アルゴリズム [2] を利用して研究されている．系列データ集合があり，各系列は項目集合のリストから成るとする．このとき最小頻度個以上の系列に出現する系列パターン (これも項目集合のリストとして定義される) をすべて発見する，というのが扱うべき問題である．素直に考えれば組み合わせ的問題であるが，高速化アルゴリズムについては APRIORI に基づく FreeSpan, PrefixSpan の提案 [11, 23] や，束 (Lattice) を用いた形式化による SPADE が知られている [29]．正規表現を直接扱う SPIRIT アルゴリズム [5] や，これを制約充足問題とみなす提案もある [13]．しかし，最小頻度条件を満たすパターンで，しかも正規表現パターンに属するものを発見しようとしており，本研究のように選択系列パターンを発見するものではない．さらに，複数系列を対象としており，出現回数とは系列パターンを含む系列数と定義されているため，テキスト発見に直接利用できない．

(本稿で論じているような) 単一系列データ上で頻出パターンを発見する問題[2]は，事象列に関するアプローチ (エピソード)[18] が知られているが，テキスト発見に特化したものではない．本研究と直接関連を有する提案がある [37]．しかし選択系列を扱わず，本稿で扱う問題に十分ではない．

2.6 結論

本論文では，単一系列データ上での系列選択の頻出パターン発見を目的として，高速な自動抽出方法を提案して逆単調性をみたす出現尺度である全体出現頻度を提案した．また系列全体頻度との頻出パターン抽出数の比較実験を行い，同条件の最小出現頻度で，より多くの頻出パターンを抽出することができ，本手法の有用性を確認した．今後は，抽出するパターンを正規表現に広げ考察していく予定である．

[2]従って出現する回数も考慮している．

第3章　選言パターン抽出のオンライン分析

頻出な選言パターンは単一系列データ上で，APRIORI に基づいた逆単調性を満たす効率的なアルゴリズムを論じることができ，精巧なテキストマイニング技法として知られている．本稿では，オンライン分析手法に基づいて，頻出な選言パターンの実用的発見の為のデータ構築法を提案する．また，実験によりその有用性を検証する．

3.1　前書き

これまで，文書データから重要な語句を抽出するために，統計的推定やデータマイニング手法などの定量的な分析を適用することは容易ではないと言われてきた．しかし，テキストの特徴を数量化し解析しようとする研究 (テキストマイニング) が注目されるにつれ，文書データ分析にデータマイニング手法を適用する研究が開始されている [31].

データマイニング手法の特徴は，頻度 (重要な項目は何度も生じるという考え方) や共起性 (複数の項目が同時に生じる) にある．このデータマイニング手法を，系列データ (sequence data) に適用するアプローチをテキストマイニングと呼ぶ．系列データは語の並びで構成された意味を有し，テキストマイニング手法により，高頻度パターン (語の並びや句) を抽出し，内容の要約 (抽象化) やラベル付けを行うことができる．

パターンは多様な解釈を持ち，出現頻度は各系列における興味の度合いを示す．しかし，系列データを扱う問題は，単一の系列データを扱う場合と複数系列データを扱う場合で大きく異なる．系列データの集合を単位として捉え，系列パターンを含む系列数を出現頻度とする問題を複数系列データ問題と呼ぶ.

一方，単一系列データでは，出現頻度とは系列におけるパターンの重要度と考えることができる．複数系列データとは異なって，単一の著者によって執筆されていることが期待できるため，ひとつのパターンが異なる解釈を有することは少ないと考えられる．従って，単一系列データではパターンの出現頻度がそのまま重要性を表すと考えられるが，既存のデータマイニング手法の多くは複数系列データを対象としており，単一系列中におけるパターン出現頻度については考慮されないため，テキストマイニングには直接利用できない.

一般的に，長大なテキストから発生頻度の高い語句すべてを発見することは，探索

空間が巨大であるため，組合わせ的な探索問題となっている [37]．このため，何らかの工夫を要する必要がある．データマイニングの主要な研究のほとんどでは，探索空間の削減にパターンの逆単調性が用いられてきた．例えば, APRIORI では探索時間を大幅に下げることができる [2]．しかし，系列パターンについてはこの逆単調性が成り立たず，根源にさかのぼった検討が必要である．系列パターンに対して正規表現を適用する方法が提案されているが NP 完全な問題を含み実用的計算量が得られない [13]．著者らは，**選言** (disjunctive) パターンを導入し，この上で逆単調性を満たす出現尺度を提案した [24]．

APRIORI を用いたテキストマイニング操作は，探索問題であるため，一般的にかなりの計算資源を用い，簡単な問題でも数時間を要することは珍しくない．実際に適用する場合，様々な条件により繰り返して探索することから，小規模な問題あるいはサンプリング手法などにより問題を回避する [28]．実務などで利用される OLAP(*Online Analytical Processing*) では，多種大量のデータを扱うため，(例えば APRIORI では支持度，確信度などの) 種々のパラメタを予め設定することは容易ではなく，かなり時間を要する作業となっている．

一般的に，条件を充足するパターンの発見問題では，繰り返しデータを走査する必要がある．しかし，各データ走査で非常に多くの読み込みが発生する場合，それがボトルネックとなってしまう．また，計算結果の再利用ができないため，パラメタの設定を変更するたびに再計算が必要となる．

これらの問題を回避するため，オンライン分析手法が提案されている [1]．ここでは，予めデータを前処理してその結果を保持し，これを繰り返して利用することにより負荷を軽減する．考え方自身は新しいものではないが，更新頻度が少ない状況，特に OLAP では相性が良い．

本稿では，テキストマイニングに対するオンライン分析手法 [1] を論じる．即ち，特定の形式でデータを表現し，繰り返して生じる問い合わせに対し効率よく選言パターンを抽出する方法を提案する．[1] では，最小支持度を満たす頻出アイテム集合を高速に得ることが可能だが，相関ルールのオンライン生成を扱うため選言パターンは扱わない．SPADE[29] は，複数系列データを対象としているため，テキストマイニングには直接利用できない．また，すべての頻出パターンを列挙するのに，1-頻出パターン，2-頻出パターン，それ以上の頻出パターンで抽出し，それぞれでデータ走査を行う必要がある．2 章では選言パターンとその逆単調性を満たす出現尺度を示し，3 章でデータの構築法及びそのデータからの選言パターン抽出法を提案する．4 章でいくつかの実験結果を示す．5 章で結びとする．

3.2 選言パターン (Disjunctive Pattern)

与えられたアルファベット (あるいはアイテムとも言う) の集合 $I = \{i_1, .., i_L\}, L > 0$ に対し，系列データ S とはアルファベットの順序リスト $S = s_1...s_m, m > 0$ である．S に含まれる (重複を含めた) 語の数 m に対し，S を m-系列データという．

選言パターン(あるいは単にパターン)p とは $t_1t_2...t_n$ で表され，各 t_i はアルファベット a または $[a_1a_2...a_m], m > 0$ (各 a_j は相異なるアルファベット) の形である．この $[a_1a_2...a_m]$ を選択と呼ぶ．パターン $p = t_1t_2...t_n$，$q = v_1v_2...v_m, m \leq n$ があるとき，q が p の**部分パターン** である ($q \sqsubseteq p$ と記す) とは，$1 \leq j_1 < ... < j_m \leq n$ があり，各 v_k は t_{j_k} に対応して次を満たす (混乱の無い限り $v_k \sqsubseteq t_{j_k}$ と記す)：

v_k がアルファベット a のとき， $t_{j_k} = a$ もしくは a を含む選択
v_k が 選択 $[a_1a_2...a_m]$ のとき，$t_{j_k} = [b_1b_2...b_l]$ でかつ $\{a_1, .., a_m\} \subseteq \{b_1, .., b_l\}$

例 3.1 “ac” は “abcd” の部分パターンである．同様に，“[ac]” は “[abcd]” の，“bd” は “[ab]b[cd]de” の，“b” は “[ab]” の，そして “ac”は “[ab][cd]” の部分パターンである．
しかし，“ab” は “[ab]” の部分パターンでも，“[ab]” は “ab” の部分パターンでもない．実際，選択 “[ab]” は選択でない “ab” と一致せず，“a” は “[ab]”に生じるが “b”が対応しない．

系列データ $S = c_1c_2...c_m$ にパターン $p = t_1t_2...t_n$ が**適合する** (match) とは，t_1 がアルファベット a_1 のとき，$t_1 = a_1 = c_{i_1}, 1 \leq i_1 \leq m$ でありかつ 部分パターン $t_2...t_n$ が系列データ $c_{i_1+1}...c_m$ に適合するときを言う．t_1 が選択 $[a_1a_2...a_m]$ のとき $a_1, ..., a_m$ のある並びからなるパターン $a_{j_1}...a_{j_m}$ が系列データ $c_1...c_{i_1}$ に適合し，かつ 部分パターン $t_2...t_n$ が系列データ $c_{i_1+1}...c_m$ に適合するときを言う．

例 3.2 系列データ S が $\mathtt{a_1a_2b_3b_4b_5a_6}$ (各アルファベットの添え字は S 中における出現位置) であるとき，パターン $\mathtt{a}$ は S に 3 回適合 ($\mathtt{a_1, a_2, a_6}$) し，パターン $\mathtt{ab}$ は 6 回適合 ($\mathtt{a_1b_3, a_1b_4, a_1b_5, a_2b_3, a_2b_4, a_2b_5}$) する．一方 $\mathtt{[ab]}$ は 9 回適合する．

系列データ S に関してパターンから非負整数への関数 $\mathcal{M}$ が**逆単調性** (Anti Monotonicity, AM) を満たすとは，パターン p, q に対して $q \sqsubseteq p$ のとき $\mathcal{M}_S(q) \geq \mathcal{M}_S(p)$ が成り立つときを言う．以下で考慮する系列データは単一であり S を略す．
逆単調な $\mathcal{M}$ が与えられ，また最小支持度 m が与えられているとする．このとき，$\mathcal{M}(q) < m$ ならば $q \sqsubseteq p$ となる p は $\mathcal{M}(p) \geq m$ ではない．この性質を用いれば，部分パターンの探索範囲を減少させることができる．整数パラメタ m は探索範囲を表すため，特に最小支持度 (minimum support) と呼ぶ．
$\mathcal{M}$ は出現頻度を表すことが多い．例えば APRIORI ではパターンが生じるトランザクションレコード数であると定義される．しかし，本稿で論じるような系列データには “トランザクションレコード” のような明確な区別が無いため，出現頻度の定義自体を論じる必要がある．複数系列データを扱う場合，出現頻度 (適合頻度) は “当該パターンを含む系列”の数であると定義されることが多い．この場合，逆単調性は自明に成り立つ．しかし，単一系列データの場合，例 3.2 のように単純な適合頻度では逆単調性が得られず，効率よい探索を期待することができない．これが本稿で逆単調性を有する出現尺度を提案する理由である [24].

系列データ $S = s_1 s_2 ... s_r$，パターン p を $t_1 t_2 ... t_n$ とすると系列先頭頻度は以下のように表される．

$$H(S,p) = \sum_{i=1}^{r} Val(S,i,p) \tag{3.1}$$

ただし $Val(S,i,p)$ は次を満たすとき 1, さもなければ 0 であるとする:

> $S(i)$ を S の i 番目からの接尾辞 $s_i ... s_r$ とする．t_1 がアルファベット a のとき，$s_i = a$ かつ $t_2 t_3 ... t_n$ が $S(i+1)$ に適合する．t_1 が選択 $[a_1 a_2 ... a_m]$ のとき，$s_i = a_j$ となる j があり (例えば $j = 1$)，$[a_2 a_3 ... a_m] t_2 ... t_n$ が $S(i+1)$ に適合する．

$H(S,p)$ は S またはその接尾辞において p が先頭から適合する回数を表している．しかし，系列先頭頻度は逆単調性を満たさない．そこで， p のすべての部分パターン q を調べ，その $H(S,q)$ のうちの最小の値を全体出現頻度とする．

$$D(S,p) = \mathrm{MIN}\{H(S,q) | q \sqsubseteq p\} \tag{3.2}$$

また，この計算は p の接尾辞 (n 個存在) のうち，長さの小さいものから順にその最小値を決定するとすれば以下のように表せる．系列データ S, パターンを p とすると，

$$D(S,p) = \mathrm{MIN}\{H(S,p), D(S,p(2))\} \tag{3.3}$$

ただし，$p(2)$ は p より 1 つ長さの短い部分パターンである．このとき，$D(S,p)$ は逆単調性を示すことが知られている [24].

例 3.3 S を caabbbc, $p = ab$ とすれば $H(S,p) = 2$, $D(S,p) = 2$ である．$p = ac$ とすれば $H(S,p) = 2$, $D(S,p) = 2$ である．また $p = [ac]$ とすれば $H(S,p) = 3$, $D(S,p) = 2$ である (実際の適合頻度は 4 回)．ac や $[ac]$ の部分系列 a,c が分離して生じていても適合するとみなすため，系列先頭頻度や適合回数とは合わない．

3.3 選言パターン束

以下，本稿ではしきい値 m 以上の全体出現頻度を持つパターンを**頻出パターン** (frequent pattern) という．選言パターンに対するオンライン分析を行うため，アイテム集合 I，単一系列データ S に対して，図 3.1 のような，あるしきい値を持つアイテムに関するべき集合 2^I 上の**束** (lattice) を構築する．この束を**選言パターン束**（Disjunctive Pattern Lattice, 以下 DPL）と呼ぶ．DPL とは，閉路の無いラベル付き根付き有向グラフ (V,E) であり，V は節点の有限集合, E は有向辺の集合 ($E \subseteq V \times V$) を表す．DPL にはただ一つの **根節** (root) が含まれ，ラベルを有さない．根からの距離が n の **節点** $v \in V$ は 要素数 n の頻出パターン (n-頻出パターン) を表し，$D(S,v)$ を**ラベル**として持つ．$(n+1)$-頻出パターン v' が v を部分パターンにもつとき，**有向辺** $(v,v') \in E$ が存在する．

例 3.4 S を cabcbba，$m = 1$ とした時，DPL は図 3.1 のように構築される．例えば節点 $v_{[a,b]}$ は，ラベルとして全体出現頻度 2 を持つ．また，有向辺 $(v_a, v_{[a,b]})$, $(v_b, v_{[a,b]})$ から，節点 v_a, v_b をそれぞれ部分パターンとして持つ．

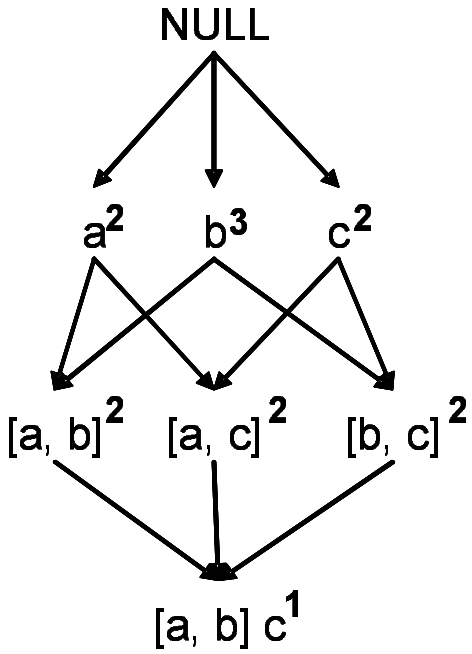

図 3.1: $m = 1$ で構築された DPL の例

3.3.1 系列データからの DPL 構築

系列データ S から DPL を構築するため，頻出パターン p の S 中での出現開始位置 $head$ と終了位置 $tail$ を保持した**出現位置リスト**を定義する．すなわち，p の S 中での出現位置を $P_p(head, tail)$ とするとき，p に関する出現位置リスト $list_p$ を次のように定義する．

$$list_p = \{P_{p_i}(head, tail), ..., P_{p_{max}}(head, tail)\} \tag{3.4}$$

ここで max は系列先頭頻度を表し，各 $head$ は同じリスト中では重複しないようにとる．

与えられた頻出規準値 (しきい値 m) に対して，出現位置リストを用いて，系列データ S から DPL を構築する手順を以下に示す.

入力:系列データ S
出力: DPL $\mathcal{L}$

1. S を 1 回走査し，全ての 1-頻出パターン p の出現位置リスト $list_p$ を生成し，$\mathcal{L}$ に節点と辺として追加する．n を 1 とする.
2. n-頻出パターンの出現位置リストから，順次 $(n+1)$-頻出パターン p' とその出現位置リスト $list_{p'}$ を取得し，$\mathcal{L}$ に節点と辺として追加する．n を $n+1$ とする.
3. 頻出パターンが生成できなくなるまで (2) を繰り返し，最後に $\mathcal{L}$ を出力する.

上述の手順 (2) で，n-頻出パターン p ,q の出現位置リスト $list_p$, $list_q$ から $(n+1)$-頻出パターン pq の出現位置リスト $list_{pq}$ を得る手順を精密に示す.

入力:出現位置リスト $list_p$, $list_q$
出力:出現位置リスト $list_{pq}$

1. $list_p$, $list_q$ を突き合わせ，$P_{p_i}(tail) < P_{q_j}(head)$ を満たす $P_{p_i}(head)$, $P_{q_j}(tail)$ をすべて見つけ，それぞれを P_{pq_k}のhead, tail として $list_{pq}$ に追加する.
2. $list_{pq}$ の要素数を $kmax$ とする．すべての出現位置を追加し終わったとき，$kmax \geq m$ ならば $list_{pq}$ を出力する.

$$
\begin{aligned}
list_p &= \{P_{p_1}(head, tail), ..., P_{p_{imax}}(head, tail)\} \\
list_q &= \{P_{q_1}(head, tail), ..., P_{q_{jmax}}(head, tail)\} \\
list_{pq} &= \{P_{pq_1}(P_{p_i}(head), P_{q_j}(tail)) | P_{p_i}(tail) < P_{q_j}(head)\}
\end{aligned}
$$

最大パターンサイズ N とするとき，APRIORI では N+1 回の S の走査が必要であるのに対し，上述の手順では S の走査は N に関係せず，S から 1-頻出アイテムの出現位置リストを生成する時に 1 回だけ必要となる.

例 3.5 S を `caabbbc` とすると，パターン a,b,c の出現位置リスト $list_a$, $list_b$, $list_c$ はそれぞれ,

$$
\begin{aligned}
list_a &= \{P_{a_1}(2,2), P_{a_2}(3,3)\} \\
list_b &= \{P_{b_1}(4,4), P_{b_2}(5,5), P_{b_3}(6,6)\} \\
list_c &= \{P_{c_1}(1,1), P_{c_2}(7,7)\}
\end{aligned}
$$

また，パターン ab と $[ac]$ の出現位置リスト $list_{ab}$, $list_{[ac]}$ は，

$$
\begin{aligned}
list_{ab} &= \{P_{ab_1}(2,4), P_{ab_2}(3,4)\} \\
list_{[ac]} &= \{P_{[ac]_1}(1,2), P_{[ac]_2}(2,7), P_{[ac]_3}(3,7)\}
\end{aligned}
$$

となる．

3.3.2 希望語を含む DPL の構築

系列データ S から DPL を構築する際に特定の語句 (希望語) を指定し，これを含むパターンとその部分パターンのみで DPL を構築する方法を考えることができる．実際，n-頻出パターン中に希望語を含むパターンが生成されなければ，(出現尺度が逆単調性を満たすので) 以降希望語を含むパターン生成はありえず，この時点で構築を終了すれば良い．

3.3.3 DPL からのパターン抽出

構築された DPL からのパターン抽出は，**深さ優先探索**で行う．以下にその手順を示す．

入力:DPL $\mathcal{L}$, 最小支持度 m
出力:出力リスト $\mathcal{L}_o$

1. NULL 節点から開始して，m 以上の度数を満たす 1-頻出パターン節点を 1 つ選択し，その節点をスタック $\mathcal{L}_o$ へ出力する．
2. その節点の持つ 1 辺を選択し辿る．
3. 次の節点が，過去に訪れたことがなく，かつ m 以上の度数を有すれば，節点を $\mathcal{L}_o$ へ出力し (2) へ戻る．
4. 過去に訪れていれば，1 つ前の節点に戻り (2) から開始する．
5. すべての訪問可能な節点の探索が終わったら $\mathcal{L}_o$ を出力する．

例 3.6 図 3.1 において，$m = 2$ 以上の度数を有するすべてのパターンを抽出する．この探索は図 3.2 のような順路に従い，出力リスト $\mathcal{L}_o$ を得る．図中の丸囲い数字は，探索の順序を示す．

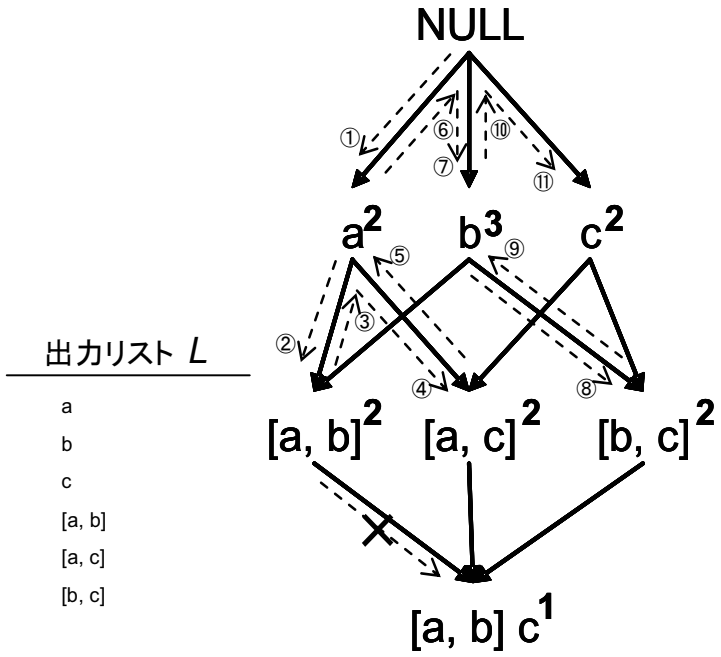

図 3.2: DPL からの頻出パターン抽出

3.4 実験

3.4.1 実験の方法

本稿では，3 種類の実験を行う．実験 1 では，DPL を用いたオンライン分析と APRIORI による方法を用いて，選言パターン抽出に要する実行時間の比較を行う．実験 2 では，DPL 構築に要する負荷を調べるため，出現位置リストと APRIORI を用いて実行時間の比較を行う．APRIORI を用いての DPL 構築とは，出現位置リストを用いずに，データ走査を行うことで頻出パターンを取得する方法である．実験 3 では，希望語を指定した DPL 構築の負荷を調べる．このため，頻出パターンを制限する場合としない場合の DPL 構築の実行時間の比較を行う．

なお，以下の実験で生成される候補パターンは，(a, [ab], [ab]c, [abc]d 等のように) 選言の深さを 1 レベルに限り，また，1 つのパターン中に選択パターンが 1 回だけ出現するものに限定する．

実験データとして，Reuters-21578 text categorization test collection を使用する．これは 1987 年のロイター社の発信記事を日時順に並べたコーパスであり，全記事数は 21,578 件である．今回は，この記事から 925 件分の記事を用い，各記事について不要語 (*stop word*) の削除および単語のステミングを行ったものを使用する．

3.4.2 実験結果の評価方法

一般に長大な記事ほど単語の出現数が増加し，その頻度値は記事内容の重要性に対応しなくなる可能性が高い．このため，本実験では，記事の長さに対して適切な重み付けを行う．ここでは，各頻出語の出現頻度として対数化索引語頻度 (logarithmic TF) を用いる．実験データ中の記事 j における，パターン p に対する出現頻度 $LTF_{j,p}$ は，次で定義される．

$$LTF_{j,p} = \frac{\log(H(j,p)+1)}{\log(N_j+1)} \tag{3.5}$$

ただし，j 中の総単語数を N_j とする．従って，重みを考慮した先頭系列頻度 $Hfreq(S,p)$ は，

$$Hfreq(S,p) = \sum_{j=1}^{jmax} LTF_{j,p} \tag{3.6}$$

となる．$Hfreq(S,p)$ は重み付けられた頻度であるので，$H(S,p)$ とは異なる．全体出現頻度についての変更はない．

3.4.3 実験結果

実験 1 では，頻度しきい値 m を 20 から 80 まで 10 刻みで変化させ，各手法でのパターン抽出にかかる実行時間の比較を行う．しきい値とそれに対する抽出パターン数を表 3.1 に，実行時間を表 3.2 に示す．表 3.2 は，パターン数と実行時間の関係を示す．実際に抽出した選言パターンの一部を以下に示す．

```
[mln, oper, shr, net] note
[loss, profit, shr, ct, net] note
[mln, dlr, 1, shr, ct, net] year
```

表 3.1: しきい値とパターン数の関係

しきい値	8	10	20	30	40	50	60	70	80
パターン数	7766	4060	517	145	58	25	14	9	6

実験 2 では，生成されるパターン数を固定し，データ長のみを変えて各手法での DPL 構築にかかる実行時間の比較を行う．ここでデータ 1 は実験データ全体，データ 2 はデータ 1 を連結したもの (従って 2 倍の長さになる) である．実験 2 の結果を表 3.3 に示す．この表では，各データと実行時間の関係を示す．

実験 3 では，しきい値を 8 に固定し，希望語として `fell` および `analyst` を用い，指定した場合と指定しない場合の DPL 構築にかかる実行時間の比較を行う．実験 3 の結果を表 3.4 にまとめ，各希望語と実行時間の関係を示す．

表 3.2: 各手法における選言パターン抽出の実行時間比較

しきい値		20	30	40	50	60	70	80
実行時間	DPL	4.70	3.82	3.05	2.19	1.77	1.15	0.47
$[s]$	APRIORI	26020	4655	1729	487	242	155	127

表 3.3: 各データサイズにおける DPL 構築の実行時間比較

	実行時間 $[s]$	
データサイズ	DPL	APRIORI
データ 1	473	26020
データ 2	950	128471

表 3.4: 希望語の有無による DPL 構築の実行時間比較

	実行時間 $[s]$	
希望語	希望語有	希望語無
`fell`	5245	7734
`analyst`	7345	7734

また，実験 1 においてしきい値が 20 における DPL と APRIORI による，リスト (パターン) 数，1 リストあたりの平均ディスク使用量及び総ディスク使用量の比較を表 3.5 に示す．

表 3.5: 各手法におけるディスク使用量の比較 (しきい値 20)

手法	リスト数 (パターン数)	ディスク使用量 $[byte]$	
		平均使用量	総使用量
DPL	517	3954	2043969
APRIORI	517	27	14037

3.4.4 考察

実験 1 のパターン抽出において，一旦 DPL を構築すれば，これを用いて選言パターン抽出を輪無し有向グラフの探索問題に帰着させることができる．逆単調性を有する頻度定義であることは，いったん対象外であると判明したノードから先はたどらなくて良いことに対応し，探索範囲が削減できることを意味する．これは APRIORI のような全件データ走査と比較して本質的に効率が良い．さらに，選言パターンが含まれれば組み合わせ的に探索空間が広がる．しかし，本方式で提案するように，選言パターンをそのまま残した DPL を構築することで空間の拡大を抑え，空間自体の走査を 1 回だけ実施することで，効率よく探索することが可能になる．他方，APRIORI のような全件データ走査方式では選言パターンは選択肢ごとの探索を個別に実行して結果を重ねることに対応し，重複した走査を要するため，本質的に効率が悪い．実際，本実験結果からもわかるように，APRIORI を用いた方法に比べおよそ 270 倍〜5500 倍の高い性能を得る．

しきい値 (最小支持度) が低いほど探索空間は増加する．しかし，上記 2 つの違いは，重複した計算部分の肥大化となって表れるため，本方式との差が拡大する原因となる．実際，本実験結果ではその割合はしきい値 4 倍に対して DPL 法で 10 倍なのに対し，APRIORI では 200 倍以上に低下する．

実験 2 では DPL 構築の負荷を調べている．2 種類のデータとも，出現位置リストを用いた方法は APRIORI を用いた方法に比べおよそ 50〜130 倍高速となる．これは，出現位置リスト法は実験データの局地的走査のみの計算で構築できることによる．データサイズの増加に伴い実行時間は増加する．出現位置リスト法では実験データの全走査が 1 度実行され，ちょうど 2 倍悪化するのに対し，APRIORI 法では，N+1 回のデータ走査を行うことから，データサイズの影響が 5 倍の悪化を生んだと考えられる．

実験 3 では，希望語の影響を調査している．2 つの語句 `fell`, `analyst` それぞれを指定した場合は，希望語を指定しない場合に比べおよそ 47%及び 5%の高速化となってい

るに過ぎない．本実験では，DPL 構築に際し，“希望語を含む頻出パターン”の探索を各記事内に閉じて探索している．つまり複数の記事にわたる出現を考慮していないため，全記事に対する探索を避けることができる．しかし，希望語に適合するテキストの検索対象範囲は当該記事全体に及ぶため，希望語の有無が大幅な改善を生むことは無い．

DPL 構築のためのディスク使用量を考察する．DPL を用いた手法では，APRIORI を用いた手法に比べ，およそ 146 倍に悪化する．実際，DPL 法では計算の途中で出現位置リストを保持するため，このような結果を生む．本実験では，実験を簡単にするため計算中すべての出現位置リストを保持したが，実際には $(n+1)$-頻出アイテムを計算するのには n-頻出アイテムの出現位置リストだけで良く，計算中のディスク使用量は減らすことが可能である．

3.5 結論

本稿ではオンライン分析の考え方を用い，単一系列データ上からの高速な DPL 構築法および DPL からの頻出パターン抽出法を提案した．また，APRIORI を用いた手法とのいくつかの比較実験を行い，同じ条件下でより高速に頻出パターンを抽出および DPL 構築できることを示し，本手法の有用性を確認した．また本稿では，テキストデータを用いたが，これは (MIDI などのような) 一般的な系列データにも拡張可能だと考えられる．

第4章　選言的木パターンマイニング

頻出な選言パターンは，単一系列データ上で APRIORI に基づいた逆単調性を満たす効率的なアルゴリズムを論じることができ，精巧なテキストマイニング技法として知られている．本稿では，この選言パターンの考え方を木パターンに拡張し，半構造データ上で逆単調性を満たす出現尺度を用いることで，高速な選言的木パターンマイニング手法を提案する．また実験によりその有用性を検証する．

4.1　前書き

これまで，文書データから重要な語句を抽出するために，統計的推定やデータマイニング手法などの定量的な分析を適用することは容易ではないと言われてきた [12, 17]．しかし，テキストを対象とした研究が注目されるにつれ，この手法を適用する研究が開始されている [31, 10]．

データマイニング手法の特徴は，頻度や共起性にある [39, 40]．この手法を，系列データ (sequence data)，特に文字列テキストに適用するアプローチをテキストマイニングと呼ぶ．系列データを扱う問題には，単一系列と複数系列を扱うものがあるが，既存のデータマイニング手法の多くは後者を対象としており，個々のパターン出現の探索による空間量の増大を，パターンの出現系列数を重要度と定義することで抑える．しかし，各パターンの出現頻度を調べることは，データの偏りや分布，TF*IDF といった重要度の算出や検索に有効である．複数の単一系列データを分析することは，各系列でのパターン頻度分析を多重化するのと同時に，複数系列の系列毎の頻度分析を行うことになり，より精密な解析が可能となる．従って，これまでに提案されている手法は適用できない．

APRIORI などデータマイニングの主要な研究の多くでは，探索空間の削減にパターンの逆単調性が用いられているが，系列パターンでは成り立たない [2, 9]．また，APRIORI を用いたテキストマイニング操作は組み合わせ探索問題であり，一般的に多くの計算資源を要するため，実際には小規模な問題やサンプリング手法などによりこれを回避する [28]．著者らは，**選言** (disjunctive) パターンを導入し，この上で逆単調性をみたす出現尺度を提案した [24]．

一般的に利用者が興味あるパターンを発見する場合，検出結果を見ながら設定条件を変化させるというプロセスを繰り返すことが多い．各ステップは効率よく処理できても，全体としてデータを繰り返し走査するため，膨大な読込みが生じ，最終結果を

得るまでに多くの時間を要する．また，各処理が独立であるため，設定条件を変更するたびに再計算が必要となる．これらの問題を回避するため，オンライン分析手法が提案されている [1]．また筆者らは，[24] に対するオンライン分析を導入し，効率的な選言パターン抽出法を提案した [25].

本稿では，半構造データ (semi structured data) を単一系列データの集合と見ることで，この単一系列データの集合を同時実行的に効率よく解析することを考える．複数系列データとは違い，パターンの頻度計算量が膨大となる単一系列データでは，処理を同時実行することで大幅な計算量削減が期待できる．ここでは単一系列データの構文的特長を利用し，(トライ構造などのような) 先頭から一致する複数の系列を木構造として捉え，重複計算を回避する方式を論じる．また，各単一系列データをニュース記事などのように互いに独立であると考え，タグ値とテキスト値を同等に木の経路上に配置する．従って解析されるパターンは深さ方向にのみ展開されるため，構造抽出に特化した木パターンマイニング手法 [31] などは単純には適用できない.

ここで提案する解析方式の特徴は，木の探索時に子孫節点が上位節の情報を利用し解析結果を継承する点にある．実際には，上位節における各パターンの出現位置と頻度を継承し，下位節での頻度計算に利用することで上位節の重複した計算を避ける．このようにして得られた解析結果を特定の形式でデータを表現し，オンライン分析の考え方を用いて，繰り返して生じる問い合わせに対し，効率よく選言的木パターンを抽出する方法を提案する．2 章では選言的木パターンとその逆単調性を満たす出現尺度を示し，3 章でデータの構築法及びそのデータからの選言的木パターン抽出法を提案する．4 章でいくつかの実験結果を示す．5 章で関連研究をまとめ，6 章で結びとする．

4.2 選言的木パターン (Disjunctive Tree Pattern)

与えられたアルファベット (あるいはアイテムとも言う) の集合 $I = \{i_1, .., i_L\}, L > 0$ に対し，**系列データ** S とはアルファベットの順序リスト $s_1...s_m, m > 0$ である．S には同じアルファベットが複数回生じてよく，S に含まれる (重複を含めた) 語の数 m に対し，S を m-系列データという.

ここで，S の i 番目に出現するアルファベットが v であるとき，節 i でこれをラベルにもつ有向グラフを考える．例えば図 4.1(a) に $S = a_1b_2c_3$ に対応する有向グラフを示す．2 つの系列 $S = a_1a_2...a_n$, $T = b_1b_2...b_m$ に対してそのアルファベットが先頭から第 i 番目まで一致するとき ($i \geq 0$)，先頭からの長さ i の経路が一致する有向グラフに対応させる．i 番目で初めて異なれば位置 i で分岐し，$i = 0$ であれば非連結となる．一般に，系列データの集合 $G = \{S_1...S_n\}, n > 0$ があたえられたとき，2 つの系列 S_i, S_j に対してアルファベットが先頭から第 i 番目まで一致するとき，先頭からの長さ i の経路を共有する森グラフ (木構造グラフの集合) に対応させれば，すべての系列を経路と対応させることができる．例えば $G = \{aca, abc, abb, aa\}$ であるとき図 4.1(b) G を得る。なお $|G| = n$ とする．**選言的木パターン** (あるいは単にパターン) p とは

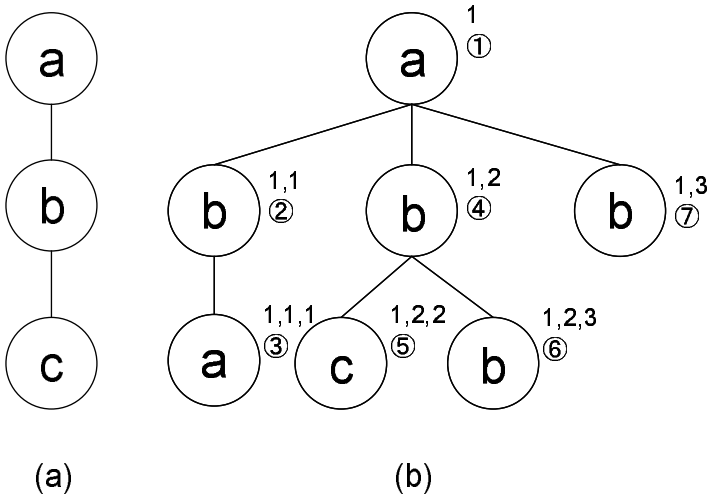

図 4.1: (a) 系列データの例　(b) 半構造データの例

$t_1t_2...t_n$ で表され，各 t_i はアルファベット a または $[a_1a_2...a_m], m > 0$ (各 a_j は相異なるアルファベット) の形である．この $[a_1a_2...a_m]$ を選択と呼ぶ．パターン $p = t_1t_2...t_n$, $q = v_1v_2...v_m, m \leq n$ があるとき，q が p の**部分パターン** である ($q \sqsubseteq p$ と記す) とは，$1 \leq j_1 < ... < j_m \leq n$ があり，各 v_k は t_{j_k} に対応して次を満たす (混乱の無い限り $v_k \sqsubseteq t_{j_k}$ と記す)：

v_k がアルファベット a のとき，$t_{j_k} = a$ もしくは a を含む選択
v_k が 選択 $[a_1a_2...a_m]$ のとき，$t_{j_k} = [b_1b_2...b_l]$ でかつ $\{a_1, .., a_m\} \subseteq \{b_1, .., b_l\}$

例 4.1 “ac” は “abcd” の部分パターンである．同様に，“[ac]” は “[abcd]” の，“bd” は “[ab]b[cd]de” の，“b” は “[ab]” の，そして “ac”は “[ab][cd]” の部分パターンである．
しかし，“ab” は “[ab]” の部分パターンでも，“[ab]” は “ab” の部分パターンでもない．実際，選択 “[ab]” は選択でない “ab” と一致せず，“a” は “ [ab]”に生じるが “ b” が対応しない．

系列データ $S = c_1c_2...c_m$ にパターン $p = t_1t_2...t_n$ が**適合する** (match) とは，t_1 がアルファベット a_1 のとき，$t_1 = a_1 = c_{i_1}, 1 \leq i_1 \leq m$ でありかつ 部分パターン $t_2...t_n$ が系列データ $c_{i_1+1}...c_m$ に適合するときを言う．t_1 が選択 $[a_1a_2...a_m]$ のとき $a_1, ..., a_m$ のある並びからなるパターン $a_{j_1}...a_{j_m}$ が系列データ $c_1...c_{i_1}$ に適合し，かつ 部分パターン $t_2...t_n$ が系列データ $c_{i_1+1}...c_m$ に適合するときを言う．

例 4.2 系列データ S が $\mathtt{a_1a_2b_3b_4b_5a_6}$ (各アルファベットの添え字は S 中における出現位置) であるとき，パターン a は S に 3 回適合 $(\mathtt{a_1}, \mathtt{a_2}, \mathtt{a_6})$ する．また，パターン ab は 6 回適合 $(\mathtt{a_1b_3}, \mathtt{a_1b_4}, \mathtt{a_1b_5}, \mathtt{a_2b_3}, \mathtt{a_2b_4}, \mathtt{a_2b_5})$ する．一方 [ab] は 9 回適合する．

半構造データ G に関してパターンから非負整数への関数 $\mathcal{M}$ が**逆単調性** (Anti Monotonicity, AM) を満たすとは，パターン p, q に対して $q \sqsubseteq p$ のとき $\mathcal{M}_G(q) \geq \mathcal{M}_G(p)$ が成り立つときを言う．以下で考慮する半構造データは単一であり G を略す．

逆単調な $\mathcal{M}$ が与えられ，また最小支持度 m が与えられているとする．このとき，$\mathcal{M}(q) < m$ ならば $q \sqsubseteq p$ となる p は $\mathcal{M}(p) \geq m$ ではない．この性質を用いれば，部分パターンの探索範囲を減少させることができる．整数パラメタ m は探索範囲を表すとも見れるため，特に最小支持度 (minimum support) と呼ぶ．

$\mathcal{M}$ は出現頻度を表すことが多い．例えば APRIORI ではパターンが生じるトランザクションレコード数であると定義される [2]．しかし，本稿で論じるような半構造データには “トランザクションレコード” のような明確な区別が無いため，出現頻度の定義自体を論じる必要がある．複数の半構造 (系列) データを扱う場合，出現頻度 (適合頻度) は “当該パターンを含むデータ” の数であると定義されることが多い．この場合，逆単調性は自明に成り立つ．しかし，単一な半構造 (系列) データの場合，例 4.2 のように単純な適合頻度では逆単調性が得られず，効率よい探索を期待することができない．これが本稿で逆単調性を有する出現尺度を提案する理由である [24]．

系列データ $S = s_1 s_2 ... s_r$，パターン p を $t_1 t_2 ... t_n$ とすると系列先頭頻度は以下のように表される．

$$H(S, p) = \sum_{i=1}^{r} Val(S, i, p) \tag{4.1}$$

ただし $Val(S, i, p)$ は次を満たすとき 1, さもなければ 0 であるとする:

> $S(i)$ を S の i 番目からの接尾辞 $s_i ... s_r$ とする．t_1 がアルファベット a のとき，$s_i = a$ かつ $t_2 t_3 ... t_n$ が $S(i+1)$ に適合する．t_1 が選択 $[a_1 a_2 ... a_m]$ のとき，$s_i = a_j$ となる j があり (例えば $j = 1$)，$[a_2 a_3 ... a_m] t_2 ... t_n$ が $S(i+1)$ に適合する．

$H(S, p)$ は S またはその接尾辞において p が先頭から適合する回数を表している．また，G に対する系列先頭頻度 $H(G, p)$ は以下のように表される．

$$H(G, p) = \sum^{|G|} H(S, p) \tag{4.2}$$

しかし，系列先頭頻度は逆単調性を満たさない．そこで， p のすべての部分パターン q を調べ，その $H(S, q)$ のうちの最小の値を全体出現頻度とする．

$$D(S, p) = \mathrm{MIN}\{H(S, q) | q \sqsubseteq p\} \tag{4.3}$$

また，この計算は p の接尾辞 (n 個存在) のうち，長さの小さいものから順にその最小値を決定するとすれば以下のように書き直すことができる．系列データ S, パターンを

p とすると，

$$D(S,p) = \mathrm{MIN}\{H(S,p), D(S,p(2))\} \tag{4.4}$$

ただし，$p(2)$ は p より 1 つ長さの短い部分パターンである．このとき，$D(S,p)$ は逆単調性を示すことが知られている [24]．また，G に対する全体出現頻度 $D(G,p)$ は以下のように表される．

$$D(G,p) = \mathrm{MIN}\{H(G,p), D(G,p(2))\} \tag{4.5}$$

例 4.3 S を caabbbc, $p = ab$ とすれば $H(S,p) = 2$, $D(S,p) = 2$ である．$p = ac$ とすれば $H(S,p) = 2$, $D(S,p) = 2$ である．また $p = [ac]$ とすれば $H(S,p) = 3$, $D(S,p) = 2$ である (実際の適合頻度は 4 回)．ac や $[ac]$ の部分系列 a,c が分離して生じていても適合するとみなすため，系列先頭頻度や適合回数とは合わない．

4.3 選言的木パターン束

以下，本稿ではしきい値 m 以上の全体出現頻度を持つパターンを**頻出パターン** (frequent pattern) という．選言的木パターンに対するオンライン分析を行うため，アイテム集合 I，単一半構造データ G に対して，図 4.2 のような，あるしきい値を持つアイテムに関するべき集合 2^I 上の**束** (lattice) を構築する．この束を**選言的木パターン束**（Disjunctive Tree Pattern Lattice, 以下 DTPL）と呼ぶ．DTPL とは，閉路の無いラベル付き根付き有向グラフ (V,E) であり，V は節点の有限集合, E は有向辺の集合 ($E \subseteq V \times V$) を表す．DTPL にはただ一つの **根節** (root) が含まれ，ラベルを有さない．根からの距離が n の**節点** $v \in V$ は 要素数 n の頻出パターン (n-頻出パターン) を表し，$D(G,v)$ を**ラベル**として持つ．$(n+1)$-頻出パターン v' が v を部分パターンにもつとき，**有向辺** $(v,v') \in E$ が存在する．

例 4.4 G を 図 4.1(b)，しきい値を $m = 1$ とした時，DTPL は図 4.2 のように構築される．

4.3.1 半構造データからの DTPL 構築

半構造データ G から DTPL を構築するため，頻出パターン p の G 中での前置順序で表される出現開始位置 $head$ と終了位置 $tail$，同一親での兄弟順序を表す深さ符号 d の順序リストである**深さ符号列** $depth = \{d_1, ..., d_n\}$ をそれぞれ保持した**出現位置リスト**を定義する．深さ n は G の根 r の深さを 1 とした時，r から $tail$ までの経路上に存在するアイテム数である．例えば，図 4.1(b) において，図中の数字列は深さ符号列を，丸囲い数字は前置順序による出現位置をそれぞれ表している．p の G 中での出現位置を $P_p(head, tail, depth)$ とするとき，p に関する出現位置リスト $list_p$ を次のように定義する．

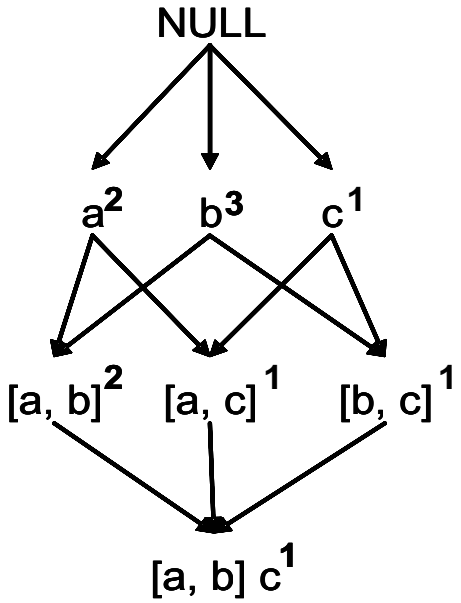

図 4.2: $m = 1$ で構築された DTPL の例

$$list_p = \{P_{p_i}(head, tail, depth), ..., P_{p_{max}}(head, tail, depth)\}$$

ここで max は系列先頭頻度を表す．与えられた頻出規準値 (しきい値 m) に対して，出現位置リストを用いて，半構造データ G から DTPL を構築する手順を以下に示す．

入力:半構造データ G
出力: DTPL L

1. G をプリオーダー順に 1 回走査し，全ての 1-頻出パターン p の出現位置リスト $list_p$ を生成し，L に節点と辺として追加する．n を 1 とする．
2. n-頻出パターンの出現位置リストから，順次 $(n+1)$-頻出パターン p' とその出現位置リスト $list_{p'}$ を取得し，L に節点と辺として追加する．n を $n+1$ とする．
3. 頻出パターンが生成できなくなるまで (2) を繰り返し，最後に L を出力する．

上述の手順 (2) で，n-頻出パターン p,q の出現位置リスト $list_p$, $list_q$ から $(n+1)$-頻出パターン pq の出現位置リスト $list_{pq}$ を得る手順を精密に示す．

入力:出現位置リスト $list_p$, $list_q$
出力:出現位置リスト $list_{pq}$

1. $list_p$, $list_q$ を突き合わせ，$P_{p_i}(tail) < P_{q_j}(head)$ かつ $P_{p_i}(depth) \subseteq P_{q_j}(depth)$ を満たす $P_{p_i}(head)$, $P_{q_j}(tail)$, $P_{q_j}(depth)$ をすべて見つけ，それぞれを P_{pq_k} の $head, tail, depth$ として $list_{pq}$ に追加する．
2. $list_{pq}$ の要素数を $kmax$ とする．すべての出現位置を追加し終わったとき，$kmax \geq m$ ならば $list_{pq}$ を出力する．

$$
\begin{aligned}
list_p &= \{P_{p_1}(head, tail, depth), ..., P_{p_{imax}}(head, tail, depth)\} \\
list_q &= \{P_{q_1}(head, tail, depth), ..., P_{q_{jmax}}(head, tail, depth)\} \\
list_{pq} &= \{P_{pq_1}(P_{p_i}(head), P_{q_j}(tail), P_{q_j}(depth))) \\
&\qquad |P_{p_i}(tail) < P_{q_j}(head), P_{p_i}(depth)) \subseteq P_{q_j}(depth))\}
\end{aligned}
$$

最大パターンサイズを N とするとき，上述の手順では G の走査は N に関係せず，G から 1-頻出アイテムの出現位置リストを生成する時に 1 回だけ必要となる．

例 4.5 図 4.1(b) におけるパターン a,b,c の出現位置リスト $list_a$, $list_b$, $list_c$ はそれぞれ，

$$
\begin{aligned}
list_a &= \{P_{a_1}(1, 1, \{1\}), P_{a_2}(3, 3, \{1, 1, 1\}), P_{a_3}(7, 7, \{1, 3\})\} \\
list_b &= \{P_{b_1}(4, 4, \{1, 2\}), P_{b_2}(6, 6, \{1, 2, 3\})\} \\
list_c &= \{P_{c_1}(2, 2, \{1, 1\}), P_{c_2}(5, 5, \{1, 2, 2\})\}
\end{aligned}
$$

またパターン bc と $[ac]$ の出現位置リスト $list_{bc}$, $list_{[ac]}$ は，

$$
\begin{aligned}
list_{bc} &= \{P_{bc_1}(4, 5, \{1, 2, 2\})\} \\
list_{[ac]} &= \{P_{[ac]_1}(1, 2, \{1, 1\}), P_{[ac]_2}(1, 5, \{1, 2, 2\}), P_{[ac]_3}(2, 3, \{1, 1, 1\})\}
\end{aligned}
$$

となる．

4.3.2 DTPL からのパターン抽出

構築された DTPL からのパターン抽出は，**深さ優先探索** (*Depth-First Search*) で行う．以下にその手順を示す．

入力:DTPL L, 最小支持度 m
出力:出力リスト $\mathcal{L}$

1. NULL 節点から開始して，m 以上の度数を満たす 1-頻出パターン節点を 1 つ選択し，その節点をスタック $\mathcal{L}$ へ出力する．

2. その節点の持つ 1 辺を選択し辿る．
3. 次の節点が，過去に訪れたことがなく，かつ m 以上の度数を有すれば，節点を $\mathcal{L}$ へ出力し (2) へ戻る．
4. 過去に訪れていれば，1 つ前の節点に戻り (2) から開始する．
5. すべての訪問可能な節点の探索が終わったら $\mathcal{L}$ を出力する．

例 4.6 図 4.2 において，$m = 2$ の最小支持度を有するすべてのパターンを抽出する．この探索は図 4.3 のような順路に従い，出力リスト $\mathcal{L}$ を得る．図中の丸囲い数字は，探索の順序を示す．

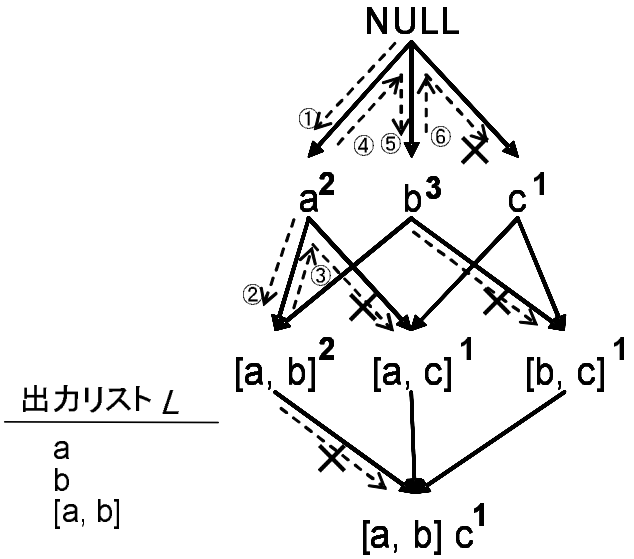

図 4.3: DTPL からの頻出パターン抽出

4.4 実験

4.4.1 実験の方法

本稿では，2 種類の実験を行う．実験 1 では，DTPL を用いたオンライン分析手法と，毎回しきい値を指定し出現位置リストを利用する手法を用いて，選言的木パター

ン抽出に要する実行時間の比較を行う．実験 2 では，DTPL 構築に要する負荷を調べるため，使用する実験データのサイズを変えて，構築にかかる実行時間の比較を行う．

なお，以下の実験で生成される候補パターンは，(a, [ab], [ab]c, [abc]d 等のように) 選言の深さを 1 レベルに限り，また，1 つのパターン中に選択が 1 回だけ出現するものに限定する．

実験データとして，オンライン論文目録データベース *DBLP* で公開されている XML データ [7] を用いる．全データサイズは 296MB であるが，その中から 500 件分のデータを用い，各記事について不要語 (*stop word*) の削除および単語のステミング [36] を行ったものを実験データとして使用する．

今回の実験では，タグ値の他にテキスト値 (*value*) もアイテムとして扱う．両者の扱いは基本的に変わらないが，テキスト値の深さは直前のタグ値によって決められるものとする．また，タグ値内の属性情報については，予め除去する．以下に実験データの一部を示す.

⟨ dblp ⟩
  ⟨ articl ⟩
  ⟨ author ⟩ abraham kandel ⟨ /author ⟩
  ⟨ author ⟩ mordechai schneider ⟨ /author ⟩
  ⟨ titl ⟩ fuzzi set applic artifici intellig ⟨ /titl ⟩
…
  ⟨ /articl ⟩
…
⟨ /dblp ⟩

4.4.2 実験結果

実験 1 では，頻度しきい値 m を 5 から 50 まで変化させ，各手法でのパターン抽出にかかる実行時間の比較を行う．しきい値とそれに対する抽出パターン数を表 4.1 に，実行時間を表 4.2 に示す．表 4.2 は，パターン数と実行時間の関係を示す．実際に抽出した選言的木パターンの一部を以下に示す.

```
[<articl>, <journal>]advanc
[<titl>, softwar]engin
[<journal>, advanc]comput
```

実験 2 では，生成されるパターン数を固定し，データ長のみを変えて DTPL 構築にかかる実行時間の比較を行う．ここでデータ 1 は実験データ全体，データ 2 はデータ 1 を 2 つ連結したもの (従って 2 倍の長さになる)，同様にデータ 3 はデータ 1 を 3 つ連結したものである．実験 2 の結果を表 4.3 に示す．この表では，各データと実行時間の関係を示す.

表 4.1: しきい値とパターン数の関係

しきい値	5	10	15	20	25	30	50
パターン数	1266	394	223	169	123	97	81

表 4.2: 各手法における選言的木パターン抽出の実行時間比較

しきい値		5	10	15	20	25	30	50
実行時間	オンライン分析	0.93	0.90	0.73	0.63	0.60	0.57	0.55
$[s]$	しきい値毎回指定	833	293	203	177	150	138	128

表 4.3: 各データサイズにおける DTPL 構築の実行時間比較

データサイズ	実行時間 $[s]$
データ 1	293
データ 2	726
データ 3	1170

4.4.3 考察

実験 1 のパターン抽出において，いったん DTPL を構築すれば，これを用いて選言的木パターン抽出を閉路無し有向グラフの探索問題に帰着させることができる．逆単調性を有する頻度定義であることは，いったん対象外であると判明したノードから先はたどらなくて良いことに対応し，探索範囲が削減できることを意味する．さらに，選言的木パターンが含まれれば組み合わせ的に探索空間が広がる．しかし，本方式で提案するように，選言的木パターンをそのまま残した DTPL を構築することで空間の拡大を抑え，空間自体の走査を 1 回だけ実施することで，効率よく探索することが可能になる．実際，本実験結果からもわかるように，毎回しきい値を指定して抽出する方法に比べおよそ 230 倍〜900 倍の高い性能を得る．

しきい値（最小支持度）が低いほど探索空間は増加する．しかし，上記 2 つの違いは，重複した計算部分の肥大化となって表れるため，本方式との差が拡大する原因となる．実際，本実験結果ではその割合はしきい値 10 倍に対して DTPL を用いたオンライン分析を利用した方法でおよそ 1.7 倍なのに対し，オンライン分析を行わず，毎回しきい値を指定して抽出する方法ではおよそ 6.5 倍に低下する．

実験 2 では DTPL 構築の負荷を調べている．データサイズが 2 倍増加に対して実行時間はおよそ 2.5 倍，データサイズが 3 倍増加に対して実行時間はおよそ 4.0 倍となっており，データサイズの増加に対して，実行時間の急激な増加は見られない．これは，出現位置リストを用いた DTPL 構築が実験データの局地的走査のみの計算で構築できることによるからだと考えられる．

4.5 関連研究

半構造データ上で頻出順序木を高速に発見するアプローチとしては，TreeMiner[30] や FREQT[4]，左右木結合を利用した手法 [43] などがある．また頻出無順序木に対しては UNOT[44] や WTIMiner[8] などがある．しかし，いずれの手法も複雑な頻出木パターンの発見を目的とし，計算効率が悪い．一方，本研究で扱う選言的木パターンは，各データが独立しており，深さ方向にのみ展開されるという性質上，深さ方向へのパターン生成にのみ注目すれば良いため，その分効率良く頻出パターンが発見可能である．またこれらの研究では，構造の抽出に特化しているため，タグ値のみを対象としているものが多いが，本手法では，テキストマイニング手法である [24] を拡張しているため，タグ値とテキスト値を同時に扱うことができ，より多くの情報をパターンとして抽出することが期待できる．

単一系列データ上で頻出パターンを発見するアプローチとしては，これまで事象列に関するエピソード手法 [18] が知られている．またテキストマイニング問題として [37] があるが，いずれも半構造データマイニングを前提には論じられていない．

本稿で扱うような，オンライン分析の考え方を利用した高速化手法として，隣接束 (Adjacency Lattice) が知られている [1]．隣接束とは，しきい値以上の頻度を有する頻

出アイテムの集合から構成されており，この集合に対する問い合わせにより，最小支持度を満たす頻出アイテム集合を高速に得ることが可能である．しかし，相関ルールのオンライン生成が前提となっており，本研究のような選言的木パターンは扱わない．

また，束を用いた形式化によるパターン列挙法として SPADE が知られている [29]．この手法は複数系列データを対象としているため，本稿のような半構造データには直接利用できない．SPADE はすべての頻出パターンを列挙するのに，1-頻出パターン，2-頻出パターン，それ以上の頻出パターンの各抽出ステップを要し，それぞれでデータ走査を行う．一方，本提案手法では DTPL 構築に要するデータ走査は，先に述べた通り 1-頻出パターンの出現位置リストを生成する際の 1 回のみで良い．

4.6 結論

本稿ではオンライン分析の考え方を用い，半構造データ上からの高速な DTPL 構築法および DTPL からの頻出な選言的木パターン抽出法を提案した．また，実データを利用したいくつかの比較実験を行い，本手法の有用性を確認した．選言的木パターンは，実用的には辞書のような構造を持つ半構造データに有効であると考えられるため，引き続きその有効性について検証していきたい．

第5章　ニュースストリームからの選言パターン抽出

頻出な選言パターンは，単一系列データ上で APRIORI に基づいた逆単調性を満たす効率的なアルゴリズムを論じることができ，精巧なテキストマイニング技法として知られている．本稿では，この考え方をニュースストリームに適用し，過去の頻度に対して忘却関数を用いた重み付け法を用いることで時間変化に適切に追従し，任意の時間において確率的にエラー保証された近似解を提供するオンライン型単一パスアルゴリズムを提案する．また実験によりその有用性を検証する．

5.1　前書き

これまで，文書データから重要な語句や規則を抽出するために，統計的推定やデータマイニング手法などの定量的な分析を適用することは容易ではないと言われてきた [12, 17]．しかし，テキストを対象とした研究が注目されるにつれ，この手法を適用する研究が開始されている [37].

データマイニング手法の特徴は，頻度の数え上げや共起性の検出にあり [39, 40]，この手法を，系列データ (sequence data)，特に文字列テキストに適用するアプローチをテキストマイニングと呼ぶ.

一方，近年のネットワークや Web 技術の発達により，ネットワーク上を流れる膨大な量の動的なデータであるデータストリーム (data stream) に対するデータマイニング研究が注目されている [14]．一般にデータストリームは，(1) 大量，(2) 高速，(3) データ分布の動的変化，(4) 連続的という特徴を持っており，これを対象とする手法には，限られた計算資源で高速かつ長期間の解析が要求され，任意の時点において正しい解が出力されるような手法であることが望ましく，過去のデータマイニング手法を適用するのは困難であると言われている [33, 14].

データストリームの中でも特に，時系列順にニュース記事や放送内容のアノテーション・トランスクリプションが配信されるものを，ニュースストリーム (news stream) と呼ぶ．本稿ではこのニュースストリームを対象とする．一般に，ニュース記事の発行間隔は不定であるが，発行される記事は速報性が高いため，本稿では記事の発行時間と記事の内容時間は一致するものと仮定する.

ニュースストリーム環境下では，絶えず記事が発生しており，新しい記事と過去の

記事が混在している．ユーザは，一般に新しい記事の内容に興味を持っていることが多く，両データを同等の重みで扱うことはできない．従って本稿では，各記事内に発生する語句については平等にオブジェクトとして扱うが，新しい記事に生じるものほど重要であるとする．

APRIORI など静的なデータを対象としたデータマイニングの主要な研究の多くでは，探索空間の削減にパターンの逆単調性を用いる．しかし，系列パターンでは成り立たない [2, 9]．また，APRIORI を用いたテキストマイニング操作は組み合わせ探索問題であり，一般的に多くの計算資源を要するため，実際には小規模な問題やサンプリング手法などによりこれを回避する [28]．著者らは，**選言** (disjunctive) パターンを導入し，この上で逆単調性をみたす出現尺度を提案した [24]．

一般的に利用者が興味あるパターンを発見する場合，検出結果を見ながら設定条件を変化させるというプロセスを繰り返すことが多い．各ステップは効率よく処理できても，全体としてデータを繰り返し走査するため，膨大な読込みが生じ，最終結果を得るまでに多くの時間を要する．また，各処理が独立であるため，設定条件を変更するたびに再計算が必要となる．

これらの問題を回避するため，オンライン分析手法が提案されている [1]．特にデータストリーム環境下では，その性質上，過去に遡った再計算が困難であるので，オンライン分析の考え方がより重要となる．筆者らは，オンライン分析を導入し，静的なデータ環境下で，効率的な選言パターン抽出法を提案した [25]．

本稿では，ニュースストリーム環境下で，時間変化に適切に追従し，任意の時点において効率的に近似解を提供するオンライン型単一パスアルゴリズムを論じる．即ち，特定の形式のデータ構造を構築し，過去の頻度に対して適切な重み付けを行い，確率的にエラー保証された頻出な選言パターンを抽出する方法を提案する．2 章で関連研究を紹介し，3 章では選言パターンとその逆単調性を満たす出現尺度を示す．4 章でデータの構築法及びそのデータからの選言パターン抽出法の提案を行い，5 章でいくつかの実験結果を示し，6 章で結びとする．

5.2 関連研究

データストリームに対するデータマイニング手法としては，損失カウント (Lossy Counting) が提案されている [19]．これは，トランザクションデータストリームを対象としたオンライン型単一パスアルゴリズムであり，確率的にエラー保証された頻出アイテム集合を高速に生成する手法である．また，この考えを応用した研究も提案されており，例えば，センサーネットワークへの応用がある [15]．反面，時間変化への適応性が考慮されていない．

一方，StreamT [33] では忘却型減衰モデルを導入することで時間変化問題を解決している．しかし，半構造データストリームを対象としており，本稿で扱うようなニュースストリームへ応用することは困難である．

データストリーム環境下での研究としては，忘却関数による重み付け [22, 34, 35] や，Tilted-Time Window による重み付け [6] などが提案されているが，いずれの研究もニュースストリームに対するデータマイニングを想定していない．

厳密解を求めるアルゴリズムとしては，[20] が提案されており，上位 k 個のパターンを対象とすることで実行効率を改善しているが，本手法のように全ての頻出パターンの抽出を対象とするアルゴリズムではない．

5.3 選言パターン (Disjunctive Pattern)

与えられたアルファベット (あるいはアイテムとも言う) の集合 $I = \{i_1, .., i_L\}, L > 0$ に対し，**系列データ** (記事)S とはアルファベットの順序リスト $s_1...s_m, \infty > m > 0$ をいう．S には同じアルファベットが複数回生じてよく，S に含まれる (重複を含めた) 語の数 m に対し，S を m-系列データという．

選言パターン (あるいは単にパターン) p とは $t_1t_2...t_n$ で表され，各 t_i はアルファベット a または選択 $[a_1a_2...a_m], m > 0$ (各 a_j は相異なるアルファベット) である．パターン $p = t_1t_2...t_n$，$q = v_1v_2...v_m, m \leq n$ があるとき，q が p の**部分パターン** である ($q \sqsubseteq p$ と記す) とは，$1 \leq j_1 < ... < j_m \leq n$ があり，各 v_k は t_{j_k} に対応して次を満たす (混乱の無い限り $v_k \sqsubseteq t_{j_k}$ と記す)：

v_k がアルファベット a のとき， $t_{j_k} = a$ もしくは a を含む選択
v_k が 選択 $[a_1a_2...a_m]$ のとき，$t_{j_k} = [b_1b_2...b_l]$ でかつ $\{a_1, .., a_m\} \subseteq \{b_1, .., b_l\}$

例 5.1 “ac” は “abcd” の部分パターンである．同様に，“[ac]” は “[abcd]” の，“bd” は “[ab]b[cd]de” の，“b” は “[ab]” の，そして “ac”は “[ab][cd]” の部分パターンである．

しかし，“ab” は “[ab]” の部分パターンではない．また “[ab]” は “ab” の部分パターンではない．

系列データ $S = c_1c_2...c_m$ にパターン $p = t_1t_2...t_n$ が**適合する** (match) とは，t_1 がアルファベット a_1 のとき，$t_1 = a_1 = c_{i_1}, 1 \leq i_1 \leq m$ でありかつ 部分パターン $t_2...t_n$ が系列データ $c_{i_1+1}...c_m$ に適合するときを言う．t_1 が選択 $[a_1a_2...a_m]$ のとき $a_1, ..., a_m$ のある並びからなるパターン $a_{j_1}...a_{j_m}$ が系列データ $c_1...c_{i_1}$ に適合し，かつ 部分パターン $t_2...t_n$ が系列データ $c_{i_1+1}...c_m$ に適合するときを言う．

例 5.2 系列データ S が $\mathtt{a_1a_2b_3b_4b_5a_6}$ (各アルファベットの添え字は S 中における出現位置) であるとき，パターン a は S に 3 回適合 $(\mathtt{a_1}, \mathtt{a_2}, \mathtt{a_6})$ する．また，パターン ab は 6 回適合 $(\mathtt{a_1b_3}, \mathtt{a_1b_4}, \mathtt{a_1b_5}, \mathtt{a_2b_3}, \mathtt{a_2b_4}, \mathtt{a_2b_5})$ する．一方 [ab] は 9 回適合する．

系列データ S に関してパターンから非負整数への関数 $\mathcal{M}$ が**逆単調性** (Anti Monotonicity, AM) を満たすとは，パターン p, q に対して $q \sqsubseteq p$ のとき $\mathcal{M}_S(q) \geq \mathcal{M}_S(p)$ が成り立つときを言う．以下で考慮する系列データは単一であり S を略す．

逆単調な $\mathcal{M}$ が与えられ，また最小頻度を与えるしきい値 σ (最小支持度) が与えられているとする．このとき，$\mathcal{M}(q) < \sigma$ ならば $q \sqsubseteq p$ となる p は $\mathcal{M}(p) \geq \sigma$ ではない．この性質を用いれば，部分パターンの探索範囲を減少させることができる．しかし，例 5.2 のような単純な適合頻度では逆単調性が得られない．そこで本稿では逆単調性を満たす出現尺度を提案する [24].

系列データ $S = s_1 s_2 ... s_r$，パターン p を $t_1 t_2 ... t_n$ とすると系列先頭頻度は以下のように表される．

$$H(S,p) = \sum_{i=1}^{r} Val(S,i,p) \tag{5.1}$$

ただし $Val(S,i,p)$ は次を満たすとき 1, さもなければ 0 であるとする:

> $S(i)$ を S の i 番目からの接尾辞 $s_i ... s_r$ とする．t_1 がアルファベット a のとき，$s_i = a$ かつ $t_2 t_3 ... t_n$ が $S(i+1)$ に適合する．t_1 が選択 $[a_1 a_2 ... a_m]$ のとき，$s_i = a_j$ となる j があり (例えば $j = 1$)，$[a_2 a_3 ... a_m] t_2 ... t_n$ が $S(i+1)$ に適合する．

$H(S,p)$ は S またはその接尾辞において p が先頭から適合する回数を表している．しかし，系列先頭頻度は逆単調性を満たさない．そこで， p のすべての部分パターン q を調べ，その $H(S,q)$ のうちの最小の値を全体出現頻度とする．

$$D(S,p) = \mathrm{MIN}\{H(S,q) | q \sqsubseteq p\} \tag{5.2}$$

また，この計算は p の接尾辞 (n 個存在) のうち，長さの小さいものから順にその最小値を決定するとすれば以下のように書き直すことができる．系列データ S, パターンを p とすると，

$$D(S,p) = \mathrm{MIN}\{H(S,p), D(S,p(2))\} \tag{5.3}$$

ただし，$p(2)$ は p より 1 つ長さの短い部分パターンである．このとき，$D(S,p)$ は逆単調性を示すことが知られている [24].

例 5.3 S を `caabbbc`, $p = ab$ とすれば $H(S,p) = 2$, $D(S,p) = 2$ である．$p = ac$ とすれば $H(S,p) = 2$, $D(S,p) = 2$ である．また $p = [ac]$ とすれば $H(S,p) = 3$, $D(S,p) = 2$ である (実際の適合頻度は 4 回)．ac や $[ac]$ の部分系列 a,c が分離して生じていても適合するとみなすため，系列先頭頻度や適合回数とは合わない．

ここで，系列先頭頻度について S の長さに対して適切な重み付けを与える．これは，一般に長大な系列ほど単語の出現数が増加し，その頻度値が系列内容の重要性に対応しなくなる可能性が高いためである．ここでは，各頻出語の出現頻度として索引語頻度 (term frequency) を用いる [36]．すなわち，系列データ S における，要素数 n のパターン p に対する，重みを考慮した系列先頭頻度 $H_w(S,p)$ は，次で定義される．

$$H_w(S,p) = \frac{H(S,p)}{|S| - (n-1)} \tag{5.4}$$

ただし，$|S|$ は S 中の総単語数を表す．従って，全体出現頻度も以下のように書き直すことができる．

$$D_w(S,p) = \mathrm{MIN}\{H_w(S,p), D_w(S,p(2))\} \tag{5.5}$$

本稿では以降，この重み付けられた系列先頭頻度 $H_w(S,p)$ および全体出現頻度 $D_w(S,p)$ を用いる．

5.4 ニュースストリームからの選言パターン抽出

ニュースストリームからの選言パターン抽出問題とは以下で定義される．

入力: ニュースストリーム NS, 最小支持度 σ , 許容誤差 ϵ

問題: 現在時刻 $time_{now}$ において，出現頻度が $(\sigma - \epsilon)$ 以上となるような全てのパターンを出力せよ．

ニュースストリーム NS とは，系列データを要素とする無限長順序リスト $S_1...S_i...S_j...$ であり，S_i とは，S_1 を基準として NS 中に現れる i 番目の系列データであることを示す．また，NS 中の各 S_i はそれぞれ発行時刻 $time_i$ を持ち，$i \leq j$ ならば $time_i \leq time_j$ である．本研究では，発行時刻 $time_i$ が等しい複数の系列データ $S_{i_1}...S_{i_{wsize}}$ は，単一の系列データ S_i とみなす．ここで，$wsize$ をウインドウサイズと呼び，何らかの上限値を有する，一度の計算で同時に処理できる系列データ数とも考えられる．現在時刻 $time_{now}$ は問い合わせが行われた時刻を表すが，各時点で解釈が若干異なる．以下，本稿では最小支持度 σ 以上の頻度を持つパターンを**頻出パターン** (frequent pattern) と呼ぶ．データストリームを扱う問題の多くでは，データの規模が巨大でかつ長期的な解析を想定しており，一般に解として出力される頻出アイテム集合を近似解とすることで計算コストを抑える．本稿で提案する手法は，近似解を出力する手法の一つである**損失カウント** [19] の手法に基づく．損失カウントでは，各アイテムの推定頻度とその頻度の最大誤差の情報を用いて，次の規則に基づきデータ構造を構築する．

規則 1: あるアイテムがデータ構造中に存在する時，その正確な頻度は推定頻度以上でかつ推定頻度と最大誤差の合計より小さい．

規則 2: あるアイテムがデータ構造中に存在しない時，その正確な頻度は許容誤差以下である．

このようにして構築されたデータ構造から任意の時点で出力される解は，許容誤差 ϵ によって確率的に保証された頻出アイテム集合，すなわち $\sigma - \epsilon$ 以上の頻出パターン集合となる．

本稿では，上記の問題に対して以下の 3 つのフェーズからなるオンライン型単一パスアルゴリズムを提案する．

1. 系列データからの候補パターン集合生成

2. 候補パターン集合からの選言パターン束構築

3. 選言パターン束からの頻出パターン集合抽出

アルゴリズムは，ある時点でニュースストリームから系列データが入力されるたびに (1),(2) を繰り返し実行し，ユーザからの任意の問合せに対して (3) を実行することで，その時点での結果を出力する．フェーズ (1) では，**出現位置リスト** と呼ぶ情報を用いることで効率的に候補パターン集合を生成する．フェーズ (2) では，**選言パターン束** と呼ぶ特殊なデータ構造を上記の規則に従い構築する．フェーズ (3) では，(2) で構築したデータ構造から，任意の時点において高速に頻出パターン抽出を行う．

また，本手法では，発行時刻に基づく重み ω を用い，発行時刻が新しい系列データ上に生じる頻度ほどより大きな重みを与える．ここではフェーズ (2) と (3) において，選言パターン束中の各節点へのアクセス時に**忘却関数** (decay function) による重みを計算し，発行時刻の古い系列データ上で生じる頻度を指数的に減衰させる．

$$\omega = u\lambda^{(time_{now} - time_{last})} \tag{5.6}$$

ここで $time_{last}$ とは，選言パターン束中の各節点の最終更新時刻である．λ は忘却定数 $0 \leq \lambda \leq 1$ を表し，この値が小さくなるほど ω も小さくなり，頻度の忘却の度合いが大きくなる．また u は，

$$u = \begin{cases} 1 \ (time_{now} - time_{last} \leq \tau) \\ 0 \ (time_{now} - time_{last} > \tau) \end{cases} \tag{5.7}$$

で表される単位ステップ関数 (unit step function) であり，τ は有効期間を表す．従って，節点の最終更新時刻からの経過期間が有効期間外であれば，λ がどんな値をとっても，$\omega = 0$ となる．次章では各フェーズについて示す．

5.5 提案手法

5.5.1 系列データからの候補パターン集合生成

NS 中の系列データ S が入力されると，**候補パターン** (candidate pattern) の生成を行う．生成には，候補パターン p の S 中での出現開始位置 $head$ と終了位置 $tail$ を保持した**出現位置リスト**を用いる．すなわち，p の S 中での出現位置を $P_p(head, tail)$ とするとき，p に関する出現位置リスト $list_p$ を次のように定義する．

$$list_p = \{P_{p_i}(head, tail), ..., P_{p_{max}}(head, tail)\} \tag{5.8}$$

ここで max は系列先頭頻度 $H(S, p)$ を表し，各 $head$ は同じリスト中では重複しない．

出現位置リストを用いて，系列データ S からパターンの要素数が n であるような n-候補パターン集合 C_n を生成する手順を以下に示す．

入力: 系列データ S
出力: n -候補パターン集合 C_n
手順:

$n = 1$ **の時:** S を 1 回走査して 1-候補パターンとその出現位置リストを生成し，C_1 に追加する．

$n > 1$ **の時:** $(n-1)$-候補パターンの出現位置リストから，順次 n -候補パターンとその出現位置リストを取得し，C_n に追加する．

上述の手順で，$(n-1)$-候補パターン p,q の出現位置リスト $list_p$, $list_q$ から n-候補パターン pq の出現位置リスト $list_{pq}$ を得る手順を示す．

入力: 出現位置リスト $list_p$, $list_q$
出力: 出現位置リスト $list_{pq}$
手順:

1. $list_p$, $list_q$ を突き合わせ，$P_{p_i}(tail) < P_{q_j}(head)$ を満たす $P_{p_i}(head)$, $P_{q_j}(tail)$ をすべて見つけ，それぞれを P_{pq_k}のhead, tail として $list_{pq}$ に追加する．
2. すべての出現位置を追加し終わったら $list_{pq}$ を出力する．

$$
\begin{aligned}
list_p &= \{P_{p_i}(head, tail), ..., P_{p_{imax}}(head, tail)\} \\
list_q &= \{P_{q_j}(head, tail), ..., P_{q_{jmax}}(head, tail)\} \\
list_{pq} &= \{P_{pq_k}(P_{p_i}(head), P_{q_j}(tail)) | P_{p_i}(tail) < P_{q_j}(head)\}
\end{aligned}
$$

例 5.4 S を caabbbc とすると，パターン a,b,c の出現位置リスト $list_a$, $list_b$, $list_c$ はそれぞれ,

$$
\begin{aligned}
list_a &= \{P_{a_1}(2,2), P_{a_2}(3,3)\} \\
list_b &= \{P_{b_1}(4,4), P_{b_2}(5,5), P_{b_3}(6,6)\} \\
list_c &= \{P_{c_1}(1,1), P_{c_2}(7,7)\}
\end{aligned}
$$

また，パターン ab と $[ac]$ の出現位置リスト $list_{ab}$, $list_{[ac]}$ は,

$$
\begin{aligned}
list_{ab} &= \{P_{ab_1}(2,4), P_{ab_2}(3,4)\} \\
list_{[ac]} &= \{P_{[ac]_1}(1,2), P_{[ac]_2}(2,7), P_{[ac]_3}(3,7)\}
\end{aligned}
$$

となる．

5.5.2 候補パターン集合からの選言パターン束構築

生成された候補パターン集合を用い，選言パターンに対するオンライン分析を行うため，アイテム集合 I ，系列データ S に対して，図 5.1 のような，あるしきい値を持つアイテムに関するべき集合 2^I 上の**束** (lattice) を構築する．この束を**選言パターン束** (Disjunctive Pattern Lattice, 以下 DPL) と呼ぶ．DPL とは，閉路の無いラベル付き根付き有向グラフ (V, E) であり，V は節点の有限集合, E は有向辺の集合 $(E \subseteq V \times V)$ を表す．DPL にはただ一つの **根節** (root) が含まれ，ラベルを有さない．根からの距離が n の**節点** $v \in V$ は n -候補パターンを表し，三項組 (推定頻度:f, 最大誤差:Δ, 節点最終更新時刻:$time_{last}$) を**ラベル**として持つ．ここで，f とは $time_{last}$ の時点までに計算されたパターンの推定頻度値であり，Δ とは DPL に節点が追加された時点までに生じた可能性のある，パターンの最大の頻度誤差値を示しており，[19] の手法に基づき計算される．$(n+1)$-候補パターン v' が v を部分パターンに持つとき，**有向辺** $e(v, v') \in E$ が存在する．

例 5.5 $S \in NS$ を cabcbba，$time = 1$, $\epsilon = 0.1$ とした時，DPL は図 5.1 のように構築される．例えば $v_{[a,\ b]}$ は，ラベルとして $(f : 0.29, \Delta : 0.0, time_{last} : 1)$ を持つ．また，$e(v_a, v_{[a,\ b]})$, $e(v_b, v_{[a,\ b]})$ から，v_a, v_b をそれぞれ部分パターンとして持つことがわかる．

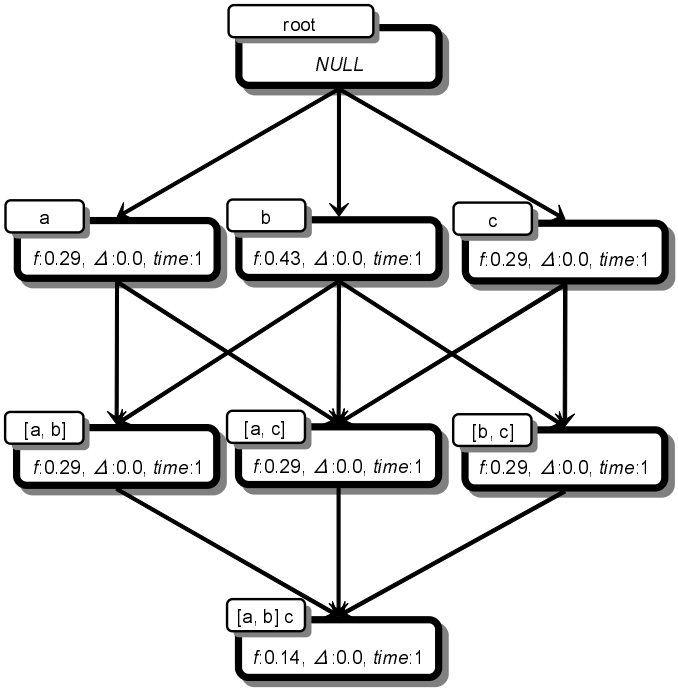

図 5.1: $\epsilon = 0.1$ で構築された DPL

系列データ S_i から生成された n -候補パターン集合中のパターン $p \in C_n$ について，DPL $\mathcal{D}$ への節点としての追加，更新，削除の各操作を次に示す．

入力: 候補パターン集合 C, 許容誤差 ϵ
出力: DPL $\mathcal{D}$
操作:

追加: $\exists v_p \notin \mathcal{D}$ かつ $\sum D_w(S_i, p) \geq \epsilon(N_{all} - N_{last})$ であれば，ラベル $(\sum D_w(S_i, p), \epsilon N_{last}, time_i)$ を持つ節点として $\mathcal{D}$ へ v_p を追加する．

更新: $\exists v_p \in \mathcal{D}$ かつ $(\sum D_w(S_i, p) + \omega f + \Delta \geq \epsilon N_{all})$ であれば，v_p のラベルを $(\sum D_w(S_i, p) + \omega f, \Delta, time_i)$ として v_p を更新する．

削除: $\exists v_p \in \mathcal{D}$ かつ $(\sum D_w(S_i, p) + \omega f + \Delta < \epsilon N_{all})$ であれば，v_p およびその下位節点 ($n+1$ 以降) を全て削除する．

N_{all} とは S_i も含め，いままでに処理された全系列データ数を表し，N_{last} は S_i を除く，いままでに処理された系列データ数 $N_{all} - wsize$ を表す．

$\sum D_w(S_i, p)$ とは $N_{last}+1$ から N_{all} までの系列データの全体出現頻度 $D_w(S, p)$ の合計を表しており，次の式で表される．

$$\sum D_w(S_i, p) = \sum_{l=N_{last}+1}^{N_{all}} D_w(S_{i_l}, p) \tag{5.9}$$

また，重み ω の計算に用いられる現在時刻 $time_{now}$ は，ここではアルゴリズム側から見た時刻を表しているので，S_i の発行時刻 $time_i$ と一致する．

例 5.6 図 5.1 の DPL 中の，節点 v_a を更新することを考える．今，$\epsilon = 0.1$, $N_{all} = 3$, $time_{now} = 3$, $wsize = 1$, $\lambda = 0.98$, $\tau = 3$, $\sum D_w(S, a) = 0.2$ の時，$\omega = 0.98^2 = 0.96$, 頻度 $\sum D_w(S, p) + \omega f + \Delta = 0.2 + 0.28 + 0 = 0.48$ となる．$\epsilon N_{all} = 0.3$ であるので，v_a の新しいラベルは $(f : 0.48, \Delta : 0, time_{last} : 3)$ と更新される．また，ある時点で v_c を DPL 中 から削除することを考える場合，その下位節点 $v_{[a,c]}, v_{[b,c]}, v_{[a,b]c}$ も削除される．それぞれの操作を反映した DPL を図 5.2 に示す．

5.5.3 選言パターン束からの頻出パターン集合抽出

構築された DPL からのパターン抽出は，**深さ優先探索** (*Depth-First Search*) で行う．以下にその手順を示す．

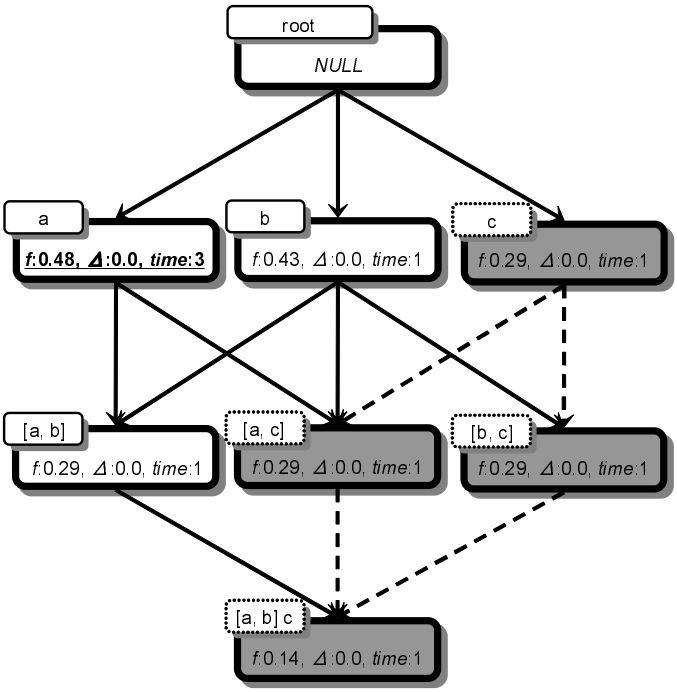

図 5.2: DPL に対しての更新・削除操作

入力: DPL $\mathcal{D}$, 最小支持度 σ, 許容誤差 ϵ
出力: 頻出パターン集合 $\mathcal{F}$
手順:

1. NULL 節点から開始して，$(\sigma - \epsilon)N_{all} \leq \omega f$ を満たす 1-頻出パターン節点を 1 つ選択し，その節点をスタック $\mathcal{F}$ へ出力する．
2. その節点の持つ 1 辺を選択し辿る．
3. 次の節点が，過去に訪れたことがなく，かつ $(\sigma - \epsilon)N_{all} \leq \omega f$ を満たせば，節点を $\mathcal{F}$ へ出力し (2) へ戻る．
4. 過去に訪れていれば，1 つ前の節点に戻り (2) から開始する．
5. すべての訪問可能な節点の探索が終わったら $\mathcal{F}$ を出力する．

ただし，重み ω の計算に用いられる現在時刻 $time_{now}$ は，ここではユーザ側から見た時刻となるので，ユーザの問い合わせた時刻そのものとなる．

例 5.7 図 5.1 において，$\sigma = 0.5$, $\epsilon = 0.1$, $N_{all} = 1$, $time_{now} = 2$, $\lambda = 0.98$, $\tau = 3$ とするとき，$(\sigma - \epsilon)N_{all} = 0.4$ 以上の度数を有するすべてのパターンを抽出する．この探索は図 5.3 のような順路に従い，頻出パターン集合 $\mathcal{F}$ を得る．図中の丸囲い数字は，探索の順序を示す．例えば，パターン a については，$\omega = 0.98$ であるので，$\omega f = 0.28$ となり頻出とはならない．しかし，パターン b については，$\omega f = 0.42$ となるので頻出パターンとして出力される．

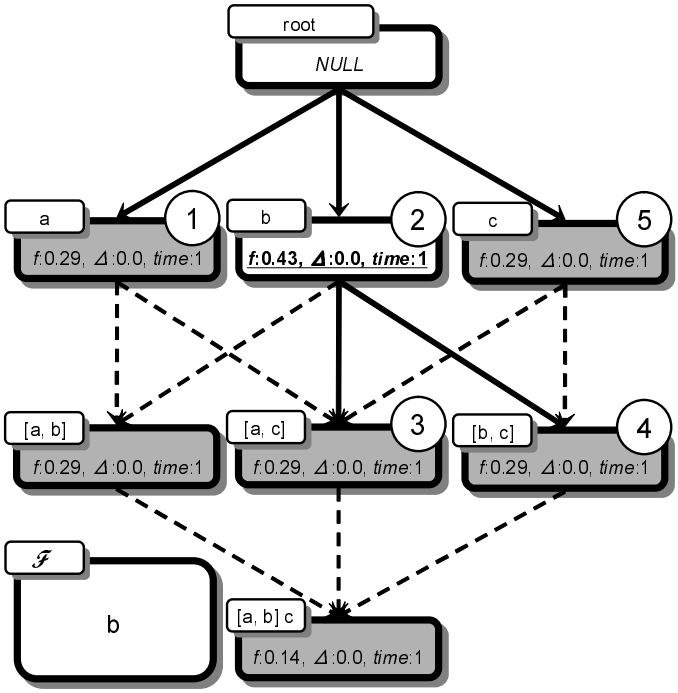

図 5.3: DPL からの頻出パターン抽出

5.6 実験

5.6.1 実験の方法

本稿では，4 種類の実験を行う．実験 1 では，大量のニュースストリームに対して，提案手法の規模耐性を評価する．実験 2 では，各しきい値と DPL 構築に要する負荷の関係を調べるため，提案手法の DPL 構築に関する性能を評価する．実験 3 では，DPL からの頻出パターン抽出性能を評価する．実験 4 では，DPL のスペース効率および時間変化への追従性を評価する．また最後に，実験結果の妥当性を検討し，提案手法で用いた忘却関数による重み付け法の意味を考察する．

なお，以下の実験で生成される候補パターンは，(a, [ab], [ab]c, [abc]d, [abcd]e 等のように) 選言の深さを 1 レベル，パターン長を 5 までに制限する．また，1 つのパターン中に選択パターンが 1 回だけ出現するものに限定する．

実験データとして，Reuters-21578 text categorization test collection を使用する．これは 1987 年のロイター社の発信記事を日時順に並べたコーパスであり，全記事数は 21,578 件である．今回は，この記事から 2200 件分の記事 (同発行時刻の記事 100 件を 1 組として，時系列順に 22 組) を用いる．時刻は日単位で扱い，それぞれの記事の発行時刻は，1 組目の発行時刻を基準とした相対時刻で考える．各記事については，不要語 (*stop word*) の削除および単語のステミング [36] を行ったものを実験データとして使用する．以下に記事内容の一例と，表 5.1 に実験データの詳細を示す.

```
shower continu week bahia cocoa zone allevi drought earli januari improv
prospect temporao humid level restor...
```

また実験では，4 種類の忘却定数 λ を用い，それぞれ 1.00(重み付け無し),1.00_7(有効期間 1 週間), 0.98(1 ヶ月で重み約 0.5),0.91(1 週間で重み約 0.5) である．

表 5.1: 実験データの詳細

データ [組]	1	2	3	4	5	6	7	8	9	10	11
時刻 [day]	0	5	7	11	14	15	19	21	25	26	28
差 [day]	-	5	2	4	3	1	4	2	4	1	2
組	12	13	14	15	16	17	18	19	20	21	22
時刻 [day]	32	34	36	40	41	46	95	96	113	235	236
差 [day]	4	2	2	4	1	5	49	1	17	122	1

5.6.2 実験結果

実験 1 では，許容誤差 ϵ を 0.005, 0.008, 0.010 と変化させ，忘却定数 λ を 0.98 に固定し，実験データ 2200 件に対して，50 件処理する毎に DPL 構築時間を計測し比較を行う．実験結果を図 5.4 に示す.

実験 2 では，(1) 許容誤差 ϵ, (2) 実験データの同時処理件数, (3) 忘却定数，の各しきい値と DPL 構築に要する負荷の関係を，DPL への節点の追加・更新・削除の各操作回数および実行時間で比較を行う．(1) では，実験データの同時処理件数を 50 件に固定し，ϵ を 0.005, 0.008, 0.010 と変化させ比較する．(2) では，ϵ を 0.008 に固定し，実験データの同時処理件数を 25, 50, 100 件と変化させ比較する．(3) では，実験データの同時処理件数を 50 件，ϵ を 0.008 に固定し比較する．実験結果を表 5.2, 表 5.3, 表 5.4 にそれぞれ示す.

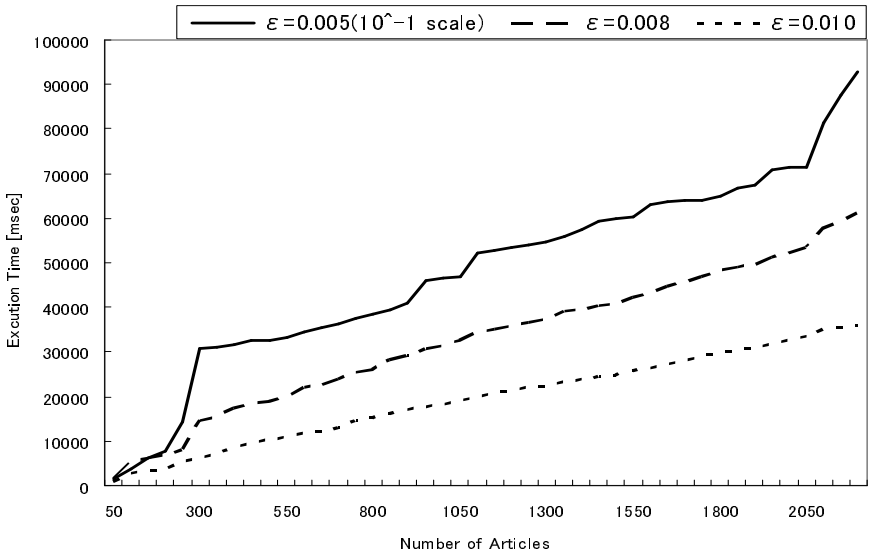

図 5.4: 規模耐性実験

表 5.2: ϵ を変化させた時の負荷

λ	1.00			1.00_7		
ϵ	.005	.008	.01	.005	.008	.01
追加 [回]	1301	461	258	1360	483	275
更新 [回]	1579	614	364	1368	530	316
削除 [回]	1125	400	223	1206	425	241
実行時間 [sec]	1041.1	67.8	36.2	1054.3	66.8	36.9
λ	0.98			0.91		
ϵ	.005	.008	.01	.005	.008	.01
追加 [回]	1388	487	281	1453	521	296
更新 [回]	1230	486	289	903	359	223
削除 [回]	1233	429	247	1293	462	261
実行時間 [sec]	929.1	61.2	36.1	738.6	56.5	35.7

表 5.3: 実験データの同時処理件数を変化させた時の負荷

λ	1.00			1.00_7		
同時処理数 [件]	25	50	100	25	50	100
追加 [回]	2597	461	172	2618	483	187
更新 [回]	1450	614	266	1328	530	221
削除 [回]	2494	400	134	2524	425	156
実行時間 [sec]	224.7	67.8	48.3	239.1	66.8	50.7
λ	0.98			0.91		
同時処理数 [件]	25	50	100	25	50	100
追加 [回]	2626	487	188	2643	521	203
更新 [回]	1210	486	196	1024	359	124
削除 [回]	2532	429	158	2549	462	172
実行時間 [sec]	208.9	61.2	44.3	196.4	56.5	39.4

表 5.4: λ を変化させた時の負荷

λ	1.00	1.00_7	0.98	0.91
追加 [回]	461	483	487	521
更新 [回]	614	530	486	359
削除 [回]	400	425	429	462
実行時間 [sec]	67.8	66.8	61.2	56.5

実験 3 では，許容誤差 ϵ を 0.005，実験データの同時処理件数を 100 件，忘却定数 λ を 0.98 に固定し，実験データを 300 件分用いて構築した DPL から，最小支持度 σ を 0.005,...,0.010(0.001 刻み), 0.015, 0.020, 0.030 と変化させ，頻出パターン抽出の実行時間を比較する．実験結果を表 5.5 に示す．

表 5.5: 頻出パターン抽出の実行時間

σ	.005	.006	.007	.008	.009	.01	.015	.02	.03
時間 [msec]	47	46	32	31	31	31	31	31	16
パターン数	101	101	72	51	39	37	7	5	2

実験 4 では，許容誤差 ϵ を 0.005，最小支持度 σ を 0.010，実験データの同時処理件数を 100 件に固定し，忘却定数 λ が 1.00 と 0.91 の時のそれぞれにおいて，DPL 中の節点 (候補パターン) 数と実際に頻出パターンとして抽出されるパターン数を比較する．また，同条件で実験データ 300 件分用いて構築した DPL から，7 日毎に DPL へ問い合わせを行い，各時点で抽出される頻出パターン数の変化を比較する．実験結果を図 5.5, 表 5.6 にそれぞれ示す．

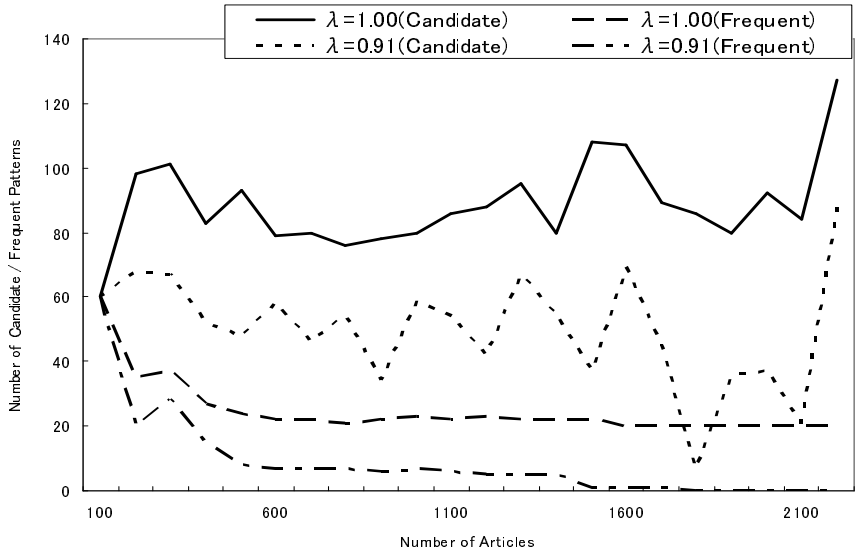

図 5.5: DPL 中の節点数と頻出パターン数

5.6.3 考察

実験結果の考察を行う．実験 1 において規模耐性実験を行っている．全体的には，許容誤差 ϵ が小さいほどパターンの探索範囲が増加するために効率は悪化し，ϵ が 2 倍

表 5.6: 経過日数による頻出パターン数

経過日数	0	7	14	21	28	35	42	49	56	63	70
$\lambda = 1.00$	37	37	37	37	37	37	37	37	37	37	37
$\lambda = 1.00_7$	35	0	0	0	0	0	0	0	0	0	0
$\lambda = 0.98$	37	31	20	18	8	7	7	6	5	4	2
$\lambda = 0.91$	28	7	2	0	0	0	0	0	0	0	0

で実行時間は 25.7 倍となる．また，入力される実験データに対して，実行時間はほぼ線形に推移するが，前半 (0 件から 300 件) では DPL への節点の追加が頻繁に行われるために効率が悪化し，後半 (2050 件から 2200 件) では大きな時間変化の影響により効率が悪化する．いずれの場合も，ϵ が小さいほどその影響も大きい．

実験 2 では DPL 構築の負荷を調べている．(1) 許容誤差 ϵ を変化させての比較では，ϵ が小さくなるほど負荷は増加しており，ϵ が 2 倍で忘却定数 λ が 1.00 の時に実行時間 28.8 倍，全操作回数 (追加, 更新, 削除の各操作の合計回数) は 4.74 倍の悪化，λ が 0.91 の時に実行時間 20.7 倍，全操作回数は 4.68 倍の悪化とそれぞれなる．これは，ϵ 減少に伴い生成される候補パターン数が増加することによると考えられる．

(2) 実験データの同時処理件数の比較を行っている．同時処理件数を増やすことで負荷は減少しており，同時処理件数が 4 倍で忘却定数 λ が 1.00 の時に実行時間 4.65 倍，全操作回数は 11.3 倍の改善，λ が 0.91 の時に実行時間 4.99 倍，全操作回数は 12.5 倍の改善とそれぞれなる．これは，各記事の処理ごとに発生する DPL への各操作が，同時処理件数を増やすことで一括的に実行されることによると考えられる．

(3) 忘却定数 λ の比較を行っている．λ が小さくなるほど負荷は減少しており，具体的には追加や削除の操作回数が増え，更新の操作回数が減少している．これは，λ を小さくすることで，時間変化の影響を大きく受けることとなり，DPL 中の候補パターンの頻繁な入れ替えによると考えられる．

実験 3 においては，DPL からの頻出パターン抽出実験を行っている．最小支持度 σ が小さいほどパターンの探索範囲が増加するため，実行時間も増加しており，σ が 6 倍で実行時間は 2.9 倍となる．しかし，最も遅い場合でも 47 [msec] であることから，オンライン分析により，非常に効率よく頻出パターンが抽出可能であるといえる．

実験 4 では，DPL のスペース効率と時間変化への追従性について調べている．スペース効率については，忘却定数 λ が小さくなると，削除前の候補パターンが DPL に残るため効率は悪くなり，λ が 1.00 の時に平均 28 %，λ が 0.91 の時に平均 17 %となる．これは，λ が小さいほど重みの影響を受け，非頻出となるまでの時間が早いことによると考えられる．

時間変化への追従性については，λ が 0.91 の時において 1600 件目以降，候補パターンおよび頻出パターン数が著しく減少していることがわかる．また，表 5.6 からも λ が小さいほど新しい話題をより重視する傾向が見られ，7 日後と 21 日後の頻出パターン

数は構築直後 (0 日) と比べそれぞれ，λ が 0.98 の時で 84 %と 54 %，λ が 0.91 の時で 25 %と 0 %となり，λ が 1.00_7 では 7 日後以降，頻出パターンは抽出されていない．これらの結果から，適切に重み付けを与えることで過去の候補パターンを削除し，時間変化に適切に追従できていることがわかる．

最後に，実験結果の妥当性を検討し，提案手法で用いた忘却関数による重み付けの意味を考える．上記の実験では，忘却定数 λ の小さいもの程，実行効率の改善が見られるが，これは言い換えれば，他の部分でその対価を払っていると考えることができる．そこで，許容誤差 ϵ を 0.005, 実験データの同時処理件数を 100 件と設定し構築した DPL から，最小支持度 σ を 0.01 として抽出できる全頻出パターンの内，その合計頻度が全頻出パターンの合計頻度の 81 %を占めるような支配的な 13 パターンを選択し，各パターンについて λ が 1.00 の時に対するカバレージ $Cover$ を比較する．$Cover$ は次の式で表される．

$$Cover[\%] = \frac{\text{任意の}\lambda\text{時のパターン抽出回数 (100 件毎問合せ)}}{\lambda = 1.00\text{ 時のパターン抽出回数 (100 件毎問合せ)}} \times 100 \qquad (5.10)$$

表 5.7 は，実験データ 2200 件での各パターンのカバレージを示している．ここでは，λ が小さいほどカバレージが低下していることがわかり，0.91 の時では 33.3 [%] にまで低下している．つまり，実行効率は頻出パターンのカバレージを犠牲にすることで得られている．忘却関数による重み付けは，前回の頻度更新からの経過期間が長いほど頻度を早く減衰させる．即ち，局所的に頻出となるようなパターンを制限し，その結果として実行効率が改善されている．

さらに，表 5.1 からもわかるとおり，実験データの 1800 件目以降は時間変化が急激になるため，これ以降は強制的に，前回更新からの経過期間が長くなる．そこで，この部分を除き，実験データ 1700 件でのカバレージを表 5.8 に示す．このとき，最大で λ が 1.00_7 の時に，実験データ 1700 件目から 2200 件目まででカバレージが 21.4 [%] 低下していることがわかる．このことからも，忘却関数による重み付けによって，全域的に頻出となるようなパターンを妥当な時間内に抽出できると言える．

5.7 結論

本稿では，ニュースストリームからの高速な DPL 構築法および DPL からの頻出パターン抽出法を提案した．提案した手法は，オンライン型の単一パスアルゴリズムであり，確率的にエラー保証がされた近似解を出力し，忘却関数による重み付け法を用いることで，時間変化に適切に追従することが可能である．また，いくつかの実験を通して，現実のデータに対して十分に有効であることを確認した．今後の研究として，データ分布の変化に対しての実行効率改善などが考えられる．

表 5.7: 各語のカバレージ (2200 件)

Cover[%]	said	mln	dir	pct	year	billion	campani
$\lambda : 1.00$	100	100	100	100	100	100	100
$\lambda : 1.00_7$	77.3	77.3	77.3	77.3	77.3	100	77.3
$\lambda : 0.98$	90.9	86.4	77.3	77.3	72.7	40.0	77.3
$\lambda : 0.91$	77.3	63.6	63.6	45.5	18.2	6.7	18.2
Cover[%]	bank	share	ct	net	loss	sale	**合計**
$\lambda : 1.00$	100	100	100	100	100	100	*100*
$\lambda : 1.00_7$	61.5	77.3	77.3	77.3	100	100	*78.6*
$\lambda : 0.98$	0	72.7	77.3	50.0	0	33.3	*67.1*
$\lambda : 0.91$	0	18.2	45.5	13.6	0	33.3	*33.3*

表 5.8: 各語のカバレージ (1700 件)

Cover[%]	said	mln	dir	pct	year	billion	campani
$\lambda : 1.00$	100	100	100	100	100	100	100
$\lambda : 1.00_7$	100	100	100	100	100	100	100
$\lambda : 0.98$	100	100	100	100	94.1	40.0	100
$\lambda : 0.91$	100	82.4	82.4	58.8	23.5	6.7	23.5
Cover[%]	bank	share	ct	net	loss	sale	**合計**
$\lambda : 1.00$	100	100	100	100	100	100	*100*
$\lambda : 1.00_7$	100	100	100	100	100	100	*100*
$\lambda : 0.98$	0	94.1	100	64.7	0	33.3	*82.6*
$\lambda : 0.91$	0	23.5	58.8	17.6	0	33.3	*44.6*

第6章　利用者の好みに基づくニュースストリームマイニング

頻出な選言パターンの抽出は，ニュースストリーム環境下において，損失数え挙げ (Lossy Counting) に基づくオンライン型単一パスアルゴリズムを論じることができ，精緻なデータストリームマイニング手法として知られている．本稿では上記のアルゴリズムを拡張し，頻度だけでなく利用者の好み情報を用いることで，対象となる選言パターンに対して重み付けを行い，それらのパターンを含むデータ構造構築のための動的なパラメタを推定する手法を提案する．また実験によりその有用性を検証する．

6.1　前書き

これまで，文書データから重要な語句や規則を抽出するために，統計的推定や，データマイニング手法などの定量的な分析を適用することは容易ではないと言われてきた [12, 17]．しかし，テキストを対象とした研究が注目されるにつれ，この手法を系列データ (sequence data)，特に文字列テキストに適用するテキストマイニングに関する研究が開始されている [37].

一方，近年のネットワークや Web 技術の発達により，ネットワーク上を流れる膨大な量の動的なデータであるデータストリーム (data stream) に対するデータマイニング研究が注目されている [14]．一般にデータストリームは，(1) 大量，(2) 高速，(3) データ分布の動的変化，(4) 連続的という特徴を持っている．これを対象とする手法には，限られた計算資源で高速かつ長期間の解析が要求され，任意の時点において正しい解が出力されるような手法であることが望ましい．しかし，過去のデータマイニング手法を適用するのは困難であると言われている [19, 14].

データストリームの中でも特に，時系列順のニュース記事や放送内容のアノテーション・トランスクリプションの配信を，ニュースストリーム (news stream) と呼ぶ．本稿ではこのニュースストリームを対象とする．一般に，ニュース記事の発行間隔は不定であるが，発行される記事は速報性が高いため，本稿では記事の発行時間と内容時間は一致すると仮定する．

ニュースストリーム環境下では，絶えず記事が発生しており，新しい記事と過去の記事が混在している．利用者は，新しい記事の内容に興味を持っていることが多く，興味のあるテーマに関してニュース記事が集中的かつ爆発的に生じる．ニュースストリー

ムの特性として両データを同等の重みで扱うことはできず，新しい記事に生じる語句ほど重要である．

APRIORI など静的なデータを対象としたデータマイニングの主要な研究では，探索空間の削減にパターン頻度の逆単調性が利用されるが，系列パターンでは成り立たない [2, 9]．著者らは**選言** (disjunctive) パターンを導入し，この上で逆単調性を満たす出現尺度を提案した [24]．

利用者が興味あるパターンを発見する場合，検出結果を見ながら設定条件を変化させるというプロセスを繰り返すことが多い．各ステップは効率よく処理できても，全体としてデータを繰り返し走査するため，膨大な読込みが生じ，最終結果を得るまでに多くの時間を要する．また，各処理が独立であるため，設定条件を変更するたびに再計算が必要となる．

これらの問題を回避するため，オンライン分析手法が提案されている [1]．特にデータストリーム環境下では，過去に遡った再計算が困難であるので，情報を集約した予備計算結果を利用するオンライン分析の考え方が重要となる．これまで筆者らは，オンライン分析に基づき，静的なデータ環境下で効率的な選言パターン抽出法を提案し [25]，またデータストリーム環境でこの考え方を一般化した手法を提案している [26]．

オンライン分析を用いた手法では，効率面での問題は解決できるが，パラメタを与えて計算した予備計算結果中に，利用者が興味を持つパターンが含まれているか否かは，実際の抽出を行うまでわからない．APRIORI に基づく手法の多くでは，頻度が重要度を表す．しかし，高頻度なパターンの多くは利用者にとって自明であることが多く，利用者が興味を持つパターンであると評価するのは困難であることから別の基準が検討されている [16]．

本稿では，利用者の好みをパターンの抽出結果に反映し，頻度だけに依らない選言パターン抽出方法を提案する．具体的には，利用者が興味を持つパターンの分布と抽出パターンの分布の類似度に基づき重み付けを行い，パラメタの動的推定や，利用者の好みを反映したパターンを含むデータ構造を構築する．

本論の構成は以下の通りである．2 章で関連研究を紹介し，3 章では選言パターンとその逆単調性を満たす出現尺度を示す．4 章で利用者の好みを反映したデータの構築法やパラメタの推定法，及びそのデータからの選言パターン抽出法の提案を行う．5 章で実験結果を示し，6 章で結びとする．

6.2 関連研究

自明な高頻度パターンを結果から取り除くため，安部ら [32] は最適パターン発見手法を提案している．ここでは，利用者が興味を持つ文書集合 (目的群) と，平均的な文書集合 (対照群) をデータマイニングシステムへの入力とし，目的群で高頻度かつ対照群で低頻度に出現するようなパターンを興味あるものとする．この枠組みでは，中頻度であっても対象内の頻度が低ければ興味あるパターンとして抽出できる．一方で，静

的なデータ環境下を対象としているため，利用者の興味や頻出となるパターンの変化等，時間経過を考慮する問題は想定されていない．

また，利用者の興味の評価方法として，サブパターンとスーパーパターンそれぞれの推定頻度を用いる sub+super 法が提案されている [42]．これは，対象となるパターンが包含するサブパターンと，包含されるスーパーパターンの両方から対象頻度を推定し，対象パターンの実際の頻度が推定頻度より大きくなる程，興味深いと判断する．この手法は，共起性の高いアイテム群を含むデータベースにおいて特に有効であるが，サブパターンとスーパーパターンの推定頻度も含めた計算に時間を要する．

6.3 選言パターン (Disjunctive Pattern)

与えられたアルファベット (あるいはアイテムとも言う) の集合 $I = \{i_1, .., i_K\}, K > 0$ に対し，**系列データ** (記事)S とはアルファベットの順序リスト $s_1...s_m, \infty > m > 0$ をいう．S には同じアルファベットが複数回生じてよい．

選言パターン (あるいは単にパターン) p とは $t_1t_2...t_n$ で表され，各 t_i はアルファベット a または，$t_i...t_{i+m}$ に対応する選択 $[a_1a_2...a_m], m > 0$ (各 a_j は相異なるアルファベットで，その順序は考慮しない) である．パターン $p = t_1t_2...t_n$，$q = v_1v_2...v_m, m \leq n$ があるとき，q が p の**部分パターン** である ($q \sqsubseteq p$ と記す) とは，$1 \leq j \leq k < ... < m \leq l \leq n$ があり，各 v_k は t_j に対応して次を満たす:v_k がアルファベット a のとき，$t_j = a$ もしくは選択中のアルファベット a ．$v_k...v_{k+m}$ が 選択 $[a_1a_2...a_m]$ のとき，$t_j...t_{j+l} = [b_1b_2...b_l]$ でかつ $\{a_1, .., a_m\} \subseteq \{b_1, .., b_l\}$.

例 6.1 “ac” は “abcd” の部分パターンである．同様に，“[ac]” は “[abcd]” の，“bd” は “[ab]b[cd]de” の，“b” は “[ab]” の，そして “ac”は “[ab][cd]” の部分パターンである．しかし，“ab” は “[ab]” の部分パターンではない．また “[ab]” は “ab” の部分パターンではない．

系列データ S に関してパターンから非負整数への関数 $\mathcal{M}$ が**逆単調性** (Anti Monotonicity, AM) を満たすとは，パターン p, q に対して $q \sqsubseteq p$ のとき $\mathcal{M}_S(q) \geq \mathcal{M}_S(p)$ が成り立つときを言う．この性質を用いれば，部分パターンの探索範囲を減少させることができる．本稿では逆単調性を満たすパターン出現尺度を提案する [24].

系列データ $S = s_1s_2...s_r$，パターン p を $t_1t_2...t_n$ とすると系列先頭頻度は以下のように表される．

$$H(S, p) = \sum_{i=1}^{r} Val(S, i, p) \tag{6.1}$$

ただし $Val(S, i, p)$ は次を満たすとき 1, さもなければ 0 であるとする: $S(i)$ を S の i 番目からの接尾辞 $s_i...s_r$ とする．t_1 がアルファベット a のとき，$s_i = a$ かつ $t_2t_3...t_n$ が $S(i+1)$ に適合する．$t_1...t_m$ が選択 $[a_1a_2...a_m]$ のとき，$s_i = a_j$ となる j があり (例えば $j = 1$)，$[a_2a_3...a_m]t_{m+1}...t_n$ が $S(i+1)$ に適合する．

$H(S,p)$ は S またはその接尾辞において p が先頭から適合する回数を表しているが，逆単調性を満たさない．そこで， p の全部分パターン q を調べ，その $H(S,q)$ のうちの最小の値を全体出現頻度とする．実際には，p の接尾辞 (n 個存在) のうち，長さの小さいものから順にその最小値を決定すればよい．

$$D(S,p) = \mathrm{MIN}\{H(S,p), D(S,p(2))\} \tag{6.2}$$

ただし，$p(2)$ は p より 1 つ長さの短い部分パターンである．このとき，$D(S,p)$ は逆単調性を示すことが知られている [24]．

ここで，系列先頭頻度について S の長さを考慮した重みを与える．ここでは，各頻出語の出現頻度として索引語頻度 (term frequency) を用いる [36]．すなわち，系列データ S における，アイテム数 n のパターン p に対する，重みを考慮した系列先頭頻度 $H_w(S,p)$ は，次で定義される．

$$H_w(S,p) = \frac{H(S,p)}{|S| - (n-1)} \tag{6.3}$$

ただし，$|S|$ は S 中の総単語数を表す．従って，全体出現頻度も以下のように書き直すことができる．

$$D_w(S,p) = \mathrm{MIN}\{H_w(S,p), D_w(S,p(2))\} \tag{6.4}$$

本稿では以降，この重み付けられた系列先頭頻度 $H_w(S,p)$ および全体出現頻度 $D_w(S,p)$ を用いる．

例 6.2 S を `cabcbba`, $p = ab$ とすれば $H(S,p) = D(S,p) = 1$，$H_w(S,p) = D_w(S,p) = 0.14$ である．また $p = [ac]$ とすれば $H(S,p) = 3$, $D(S,p) = 2$，$H_w(S,p) = D_w(S,p) = 0.29$ である．

6.4 提案手法

6.4.1 問題定義

本稿では，ニュースストリームからの選言パターン抽出問題を次のように定義する．

入力: ニュースストリーム NS, 最小支持度 σ, 許容誤差 ϵ

問題: 現在時刻 $time_{now}$ において，出現頻度が σ 以上となるような全ての選言パターンを出力せよ．

ニュースストリーム NS とは，系列データを要素とする無限長順序リスト $S_1 ... S_i ... S_j ...$ であり，S_i とは，S_1 を基準として NS 中の第 i 番目の系列データである．また，各 S_i はそれぞれ発行時刻 $time_i$ を持ち，$i \leq j$ ならば $time_i \leq time_j$ である．本研究では，発行時刻 $time_i$ が等しい複数の系列データ $S_{i_1}, ..., S_{i_{wsize}}$ は，直列に結合する単一

の系列データ S_i とみなす．ここで，$wsize$ をウインドウサイズと呼ぶ．$wsize$ は上限値を有すると仮定し，一度の計算で同時に処理できる系列データ数と考える．現在時刻 $time_{now}$ は問い合わせが行われた時刻を表す．本稿では最小支持度 σ 以上の頻度を持つパターンを**頻出パターン** (frequent pattern) と呼ぶ．

6.4.2 手法の概要

本稿では，5 つのフェーズからなるオンライン型単一パスアルゴリズムを提案する．

1. 系列データからの候補パターン集合生成
2. 候補パターンへの好みに基づく重み付け
3. 最適許容誤差の推定
4. 候補パターン集合からの選言パターン束構築
5. 選言パターン束からの頻出パターン集合抽出

このアルゴリズムは，ニュースストリームから系列データが入力されるごとに (1) から (4) を繰り返し実行する．利用者からの任意の問合せに対しては (5) を実行し，その時点での結果を出力する．

(1) では，系列データから，アイテム数 m で初期許容誤差 ϵ_0 以上の頻度を持つ m -**候補パターン** (candidate pattern) の生成を行う．

(2) では，利用者の好み情報に基づくパターンへの重み付けを処理する．利用者は正負それぞれの**興味パターン** (のリスト) を提示する．リスト中の各パターンと，候補パターンの分布を比較し，正もしくは負の重みを与えることで，利用者の好みを反映させる．正負どちらの興味パターンにも類似しないものは意外性があり，利用者は興味を持つ可能性があるとみなして大きな重みを与える．

また，(3) ではデータ構造を構築するときパラメタの許容誤差を推定し，この値を最適許容誤差として (4) で利用する．

(4) では，アイテム集合 I ，系列データ S に対して，図 6.1 のような，あるしきい値を持つアイテムのためのべき集合 2^I 上の**束** (lattice) を構築する．この束を**選言パターン束**（Disjunctive Pattern Lattice, 以下 DPL）と呼ぶ．DPL は，閉路の無いラベル付き根付き有向グラフ (V, E) である．V は節点の有限集合, E は有向辺の集合 $(E \subseteq V \times V)$ を表す．DPL にはただ一つの **根節** (root) が含まれ，この節はラベルを有さない．根からの距離が n の**節点** $v \in V$ は n -候補パターンを表し，3 項組 (推定頻度:f, 最大誤差:Δ, 節点最終更新時刻:$time_{last}$) を**ラベル**として持つ．ここで，f は $time_{last}$ の時点までに計算されたパターンの推定頻度値である．また Δ は，節点が DPL に最初に追加された時点までに処理された全系列データ数を表しており，節点の更新時には変更されない．$(n+1)$-候補パターン v' が v を部分パターンに持つとき，**有向辺** $e(v, v') \in E$ が存在する．

例 6.3 $S \in NS$ を cabcbba，$time = 1$, $\epsilon_{opt} = 0.1$ とした時，DPL は図 6.1 のように構築される．$v_{[a,\ b]}$ は，ラベルとして $(f : 0.29, \Delta : 0, time_{last} : 1)$ を持つ．また，$e(v_a, v_{[a,\ b]})$, $e(v_b, v_{[a,\ b]})$ から，v_a, v_b をそれぞれ部分パターンとして持つ．

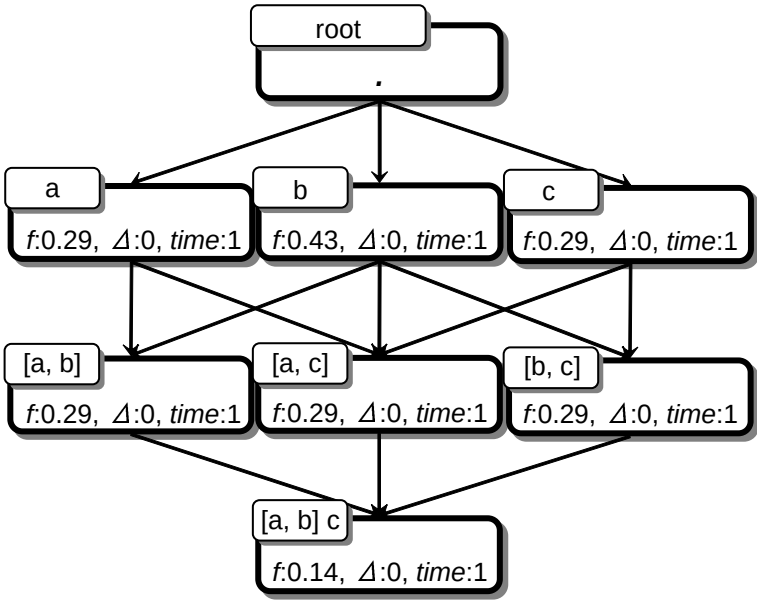

図 6.1: $\epsilon_{opt} = 0.1$ で構築された DPL

また (4) と (5) において，発行時刻に基づく重み ω を導入し，発行時刻が新しい系列データ上に生じる頻度ほど，より大きな重みを与える．すなわち，DPL 中の各節点へのアクセス時に**忘却関数** (decay function) による重みを用い，発行時刻の古い系列データ上で生じる頻度を指数的に減衰させる．

$$\omega = u\lambda^{(time_{now} - time_{last})} \tag{6.5}$$

ここで $time_{last}$ は，DPL 中の各節点の最終更新時刻である．λ は忘却定数 $0 \leq \lambda \leq 1$ を表す．この値が小さいほど ω も小さく，頻度の忘却の度合いが大きい．また u は，

$$u = \begin{cases} 1\ , (if\ time_{now} - time_{last} \leq \tau) \\ 0\ , (if\ time_{now} - time_{last} > \tau) \end{cases} \tag{6.6}$$

で表される単位ステップ関数 (unit step function) であり，τ は有効期間を表す．従って，節点の最終更新時刻からの経過期間が有効期間外であれば，λ がどのような値でも $\omega = 0$ となる．

6.4.3 系列データからの候補パターン集合生成

NS 中の系列データ S に対し，候補パターンの生成を行う．生成には，候補パターン p の S 中での出現開始位置 $head$ と終了位置 $tail$ を保持した**出現位置リスト**を用いる．すなわち，p の S 中での出現位置を 2 項組 $(head, tail)$ とするとき，p に関する出現位置リスト $list_p$ を次のように定義する．

$$list_p = \langle (head_{p_1}, tail_{p_1}), ..., (head_{p_h}, tail_{p_h}) \rangle \tag{6.7}$$

h は系列先頭頻度 $H(S, p)$ を表し，各 $head_{p_i} \leq tail_{p_i}, 1 \leq i \leq h$ は同じリスト中では重複しない．

ここでは出現位置リストを用いる．系列データ S からパターン中のアイテム数が n で，かつ頻度が初期許容誤差 ϵ_0 以上を満たすような n -候補パターン集合 C_n を生成する手順を以下に示す．

入力: 系列データ S, 初期許容誤差 ϵ_0
出力: n -候補パターン集合 C_n
手順:

$n = 1$ **の時:** S を 1 回走査して 1-候補パターンとその出現位置リストを生成し，頻度が ϵ_0 以上を満たすものを C_1 に追加する．

$n > 1$ **の時:** $(n-1)$-候補パターンの出現位置リストから，順次 n -候補パターンとその出現位置リストを取得し，頻度が ϵ_0 以上を満たすものを C_n に追加する．

上述の手順で，$(n-1)$-候補パターン p,q の出現位置リスト $list_p$, $list_q$ から n-候補パターン pq の出現位置リスト $list_{pq}$ を得る手順を示す．$list_p$ 中の $tail_{p_i}$ が $list_q$ 中の $head_{q_j}$ より小さいとき，pq は $head_{p_i}$ から $tail_{q_j}$ までの間に出現する．すなわち，pq の出現位置 $head_{pq_k}, tail_{pq_k}$ は，$head_{p_i}, tail_{q_j}$ と一致する．

入力: 出現位置リスト $list_p$, $list_q$
出力: 出現位置リスト $list_{pq}$
手順:

1. $list_p$, $list_q$ を突き合わせ，$tail_{p_i} < head_{q_j}$ を満たす $head_{p_i}$，$tail_{q_j}$ をすべて見つけ，それぞれを $head_{pq_k}, tail_{pq_k}$ として $list_{pq}$ に追加する．
2. すべての出現位置を追加し終わったら $list_{pq}$ を出力する．

例 6.4 S を cabcbba とすると，パターン a,b,c の出現位置リスト $list_a$, $list_b$, $list_c$ はそれぞれ,

$$\begin{aligned} list_a &= \langle (2,2),(7,7) \rangle \\ list_b &= \langle (3,3),(5,5),(6,6) \rangle \\ list_c &= \langle (1,1),(4,4) \rangle \end{aligned}$$

となり，パターン ab と $[ac]$ の出現位置リスト $list_{ab}$, $list_{[ac]}$ は,

$$\begin{aligned} list_{ab} &= \langle (2,3) \rangle \\ list_{[ac]} &= \langle (1,2),(2,4),(4,7) \rangle \end{aligned}$$

となる.

6.4.4 候補パターンへの好みに基づく重み付け

候補パターン集合 C_n 中の各パターン p に対し，好みに基づく重み付けを行う．与えられた正負の興味パターンの有限集合 $Pos = \{p_{pos_1}, ..., p_{pos_l}\}, Neg = \{p_{neg_1}, ..., p_{neg_m}\}(l, m \geq 0)$ に対し，**興味パターンリスト**とは $L = \langle Pos, Neg \rangle$ をいう.

以下に，興味パターンリストを用いた候補パターンへの好みに基づく重み付けを行う手順を示す.

入力: 候補パターン集合 C_n, 興味パターンリスト L, 更新判定しきい値 th_{new}
出力: 興味重み集合 W_{C_n}, 興味パターンリスト L
手順:

$n = 1$ **の時:** (1) C_1 中の各候補パターンと L 中の各興味パターンとの類似度を計算し，Pos, Neg, L との類似度をそれぞれ求める.
(2) 興味重み集合 W_{C_1} を計算し，同時に L との類似度が th_{new} 以下の候補パターンを L に追加する.

$n > 1$ **の時:** (1) C_n 中の各候補パターンの全部分パターンの興味重みを用い，その合計の平均値を候補パターンの興味重みとし，W_{C_n} を求める.

$n = 1$ の時の手順を詳細に示す．まず C_1 中の各 p に対して，Pos および Neg 中のパターン x との類似度 $sim(x, p)$ を計算する．本稿では類似度として，(6.8) 式の Kullback-Leibler(KL) 情報量を用いる．$Prob(p)$ および $Prob(x)$ はそれぞれ p ,x の確率分布を表す.

次に (6.9) 式で p と Pos, Neg との類似度 $sim(Pos, p), sim(Neg, p)$ および，p と L との類似度 $sim(L, p)$ をそれぞれを求め，(6.10) 式で p の興味重み w_p を求める．$th_{max_{pn}}$ とは，興味重みの最大値を決定する値であり，$th_{max_{ue}}$ は，意外性重みの最大値を決定する値である．また th_{ue} は，意外性を判断するしきい値である．w_p を求める一連の計算は，p が持つ興味の方向性とその大小を決定することに対応する．

$sim(L, p)$ が興味パターンリストの更新しきい値 $th_{new} \leq th_{ue}$ より小さければ，(6.11) 式に従い Pos もしくは Neg に p を新しい興味パターンとして追加する．

$$sim(x, p) = \sum Prob(x) log(\frac{Prob(x)}{Prob(p)}) \tag{6.8}$$

$$sim(L, p) = \mathrm{MIN}\{sim(\alpha, p) | \alpha \in L\} \tag{6.9}$$

$$w_p = \begin{cases} 1 + th_{max_{ue}}(\frac{sim(Pos,p)+sim(Neg,p)}{2}) \ , \\ \qquad\qquad (if \ sim(L, p) \geq th_{ue}) \\ 1 + th_{max_{pn}}(1 - sim(Pos, p)) \ , \\ \quad (if \ sim(Pos, p) < sim(Neg, p)) \\ 1 - th_{max_{pn}}(1 - sim(Neg, p)) \ , \\ \quad (if \ sim(Pos, p) > sim(Neg, p)) \end{cases} \tag{6.10}$$

$$\begin{cases} p \text{ を } Pos \text{ に追加} \ , (if \ 1 < w_p < 1 + th_{max_{pn}}) \\ p \text{ を } Neg \text{ に追加} \ , (if \ 1 > w_p > 1 - th_{max_{pn}}) \end{cases} \tag{6.11}$$

6.4.5 最適許容誤差の推定

候補パターン集合 C_n に対する興味重み集合 W_{C_n} を用いて，最適許容誤差 ϵ_{opt} の推定を行う．推定には F 値を用い，C_n 中の各候補パターン p の頻度が推定許容誤差 ϵ_{est} 以上で，W_{C_n} 中の興味重み w_p が 1 以上となるパターンを正解とする．本稿での F 値 F および再現率 $(Recall)$ β，適合率 $(Precision)$ γ の定義を以下に示す．

$$\begin{cases} \beta = \dfrac{\text{頻度 } \epsilon_{est} \text{ 以上かつ } w_p > 1 \text{ なパターン数}}{w_p > 1 \text{ なパターン数}} \\ \gamma = \dfrac{\text{頻度 } \epsilon_{est} \text{ 以上かつ } w_p > 1 \text{ なパターン数}}{\text{頻度 } \epsilon_{est} \text{ 以上なパターン数}} \\ F = \frac{2\beta\gamma}{\beta+\gamma} \end{cases} \tag{6.12}$$

上記の F を用い，初期許容誤差 ϵ_0 から繰り返し計算を行い，F を最大とするような ϵ_{est} を ϵ_{opt} とする．以下に最適許容誤差の推定を行う手順を示す．

入力: 興味重み集合 W_{C_n}, 初期許容誤差 ϵ_0, 許容誤差倍率 $rate$, 繰り返し回数 $loop$
出力: 最適許容誤差 ϵ_{opt}
手順:

$n = 1$ **の時:** (1) 推定許容誤差 ϵ_{est} を ϵ_0 から $\frac{\epsilon_0 rate}{loop}$ 刻みで大きくしていき，計 $loop$ 回の F 値計算を行う．
(2) 計算された全ての F 値から，F 値を最大とする ϵ_{est} を ϵ_{opt} として出力する．

$n > 1$ **の時:** (1) $n = 1$ で推定した ϵ_{opt} を出力する．

$rate$ は ϵ_{opt} が取りうる最大値が ϵ_0 の何倍となるかを定める許容誤差倍率を，また $loop$ は最適許容誤差計算の繰り返し回数を定める値を意味する．$rate$ が一定のとき，$loop$ を大きくすると ϵ_{opt} の推定精度は向上する．

6.4.6 候補パターン集合からの選言パターン束構築

系列データ S_i から生成される n -候補パターン集合中のパターン $p \in C_n$ について，DPL $\mathcal{D}$ への節点としての追加，更新，削除の各操作を次に示す．

入力: 候補パターン集合 C_n, 興味重み集合 W_{C_n}, 最適許容誤差 ϵ_{opt}
出力: DPL $\mathcal{D}$
操作:

追加: $\exists v_p \notin \mathcal{D}$ かつ $w_p \sum D_w(S_i, p) \geq \epsilon_{opt} wsize$ であれば，ラベル $(w_p \sum D_w(S_i, p), N_{last}, time_i)$ を持つ節点として $\mathcal{D}$ へ v_p を追加する．

更新: $\exists v_p \in \mathcal{D}$ かつ $(w_p \sum D_w(S_i, p) + \omega f + \Delta \geq \epsilon_{opt} N_{all})$ であれば，v_p のラベルを $(w_p \sum D_w(S_i, p) + \omega f, \Delta, time_i)$ として v_p を更新する．

削除: $\exists v_p \in \mathcal{D}$ かつ $(w_p \sum D_w(S_i, p) + \omega f + \Delta < \epsilon_{opt} N_{all})$ であれば，v_p およびその下位節点 ($n + 1$ 以降) を全て削除する．

N_{all} は S_i も含め，これまでに処理された全系列データ数を表す．また，N_{last} は (S_i を除く)，これまでに処理された系列データ数 $N_{all} - wsize$ を表す．

$\sum D_w(S_i, p)$ は $N_{last} + 1$ から N_{all} までの系列データの全体出現頻度 $D_w(S, p)$ の合計であり，次の式で表される．

$$\sum D_w(S_i, p) = \sum_{l=N_{last}+1}^{N_{all}} D_w(S_{i_l}, p) \tag{6.13}$$

発行時刻に基づく重み ω の計算に用いられる現在時刻 $time_{now}$ は，S_i の発行時刻 $time_i$ と一致することに注意したい．

例 6.5 図 6.1 の DPL 中の，節点 v_a を更新する．$\epsilon_{opt} = 0.1$, $N_{all} = 3$, $time_{now} = 3$, $wsize = 1$, $\lambda = 0.98$, $\tau = 3$, $\sum D_w(S, a) = 0.2$, $w_a = 1.5$ の時，$\omega = 0.96$, 頻度 $w_a \sum D_w(S, p) + \omega f + \epsilon_{opt}\Delta = 0.58$ となる．$\epsilon_{opt} N_{all} = 0.3$ であるので，v_a の新しいラベルは $(f : 0.58, \Delta : 0, time_{last} : 3)$ と更新される．また，v_c を DPL 中 から削除する場合，その下位節点 $v_{[a,c]}, v_{[b,c]}, v_{[a,b]c}$ も削除される．それぞれの操作を反映した DPL を図 6.2 に示す．

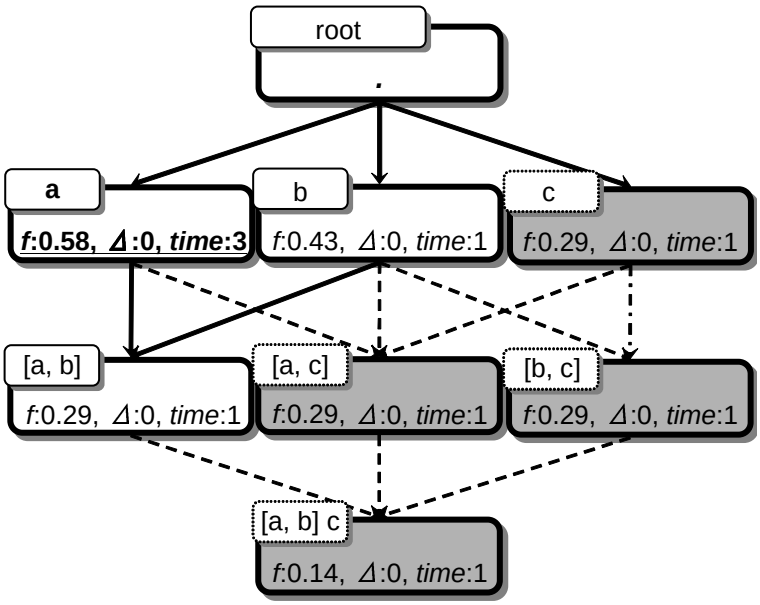

図 6.2: DPL に対しての更新・削除操作

6.4.7 選言パターン束からの頻出パターン集合抽出

DPL $\mathcal{D}$ からのパターン抽出を行うため，**深さ優先探索** $(Depth\text{-}First\ Search)$ を用いる．各節点について，$\sigma N_{all} \leq \omega f + \epsilon_{opt}\Delta$ を満たす節点は頻出であるとして頻出パターン集合 $\mathcal{F}$ に出力する．

本手法では，最適許容誤差 ϵ_{opt} を含めた頻度を用いて頻出パターン判定を行う．ϵ_{opt} は，各繰り返し時に推測され変動するが，ϵ_0 から $\epsilon_0 \cdot rate$ の間に収まる．

例 6.6 図 6.1 において，$\sigma = 0.4$, $\epsilon_{opt} = 0.1$, $N_{all} = 1$, $time_{now} = 2$, $\lambda = 0.98$, $\tau = 3$ とするとき，$\sigma N_{all} = 0.4$ 以上の度数を有するすべてのパターンを抽出する．この探索は図 6.3 のような順路に従い，頻出パターン集合 $\mathcal{F}$ を得る．図中の丸囲い数字は，探索の順序を示す．パターン a については，$\omega f = 0.28$ となり頻出とはならない．しかし，パターン b については，$\omega f = 0.42$ となるので頻出パターンとして出力する．

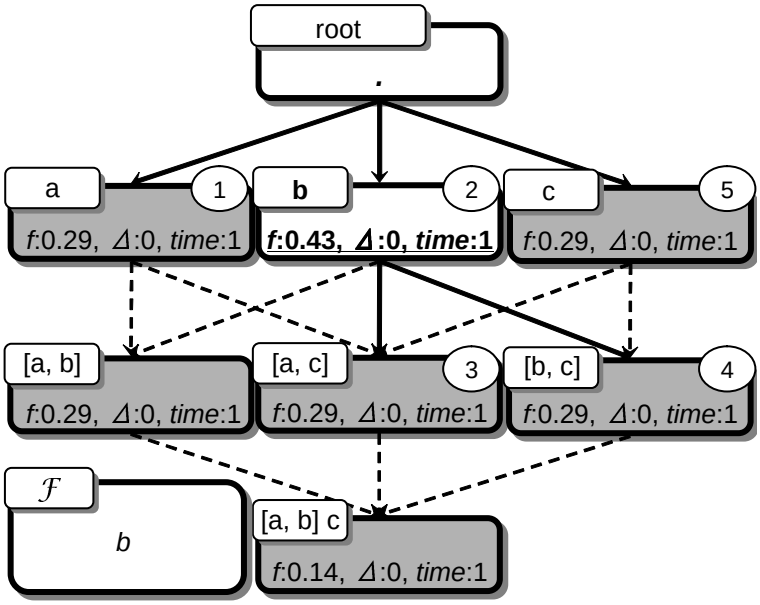

図 6.3: DPL からの頻出パターン抽出

6.5 実験

6.5.1 実験の方法

本稿では 3 つの実験を行う．実験 1 の目的は，頻出パターンの評価にある．実験 2 では，提案手法の実行効率および DPL 構築負荷の評価を，更に実験 3 は，提案手法が頻出パターンに与える影響と DPL の空間効率を評価することを目的とする．

本稿では，Reuters-21578 text categorization test collection をコーパスとして使用する．これは 1987 年のロイター社の発信記事を日時順に並べたものであり，21,578 件の記事からなるが，2200 件分の記事 (同発行時刻の記事 100 件を 1 組として，時系列順に 22 組) を用いる．日単位で扱い，それぞれの記事の発行時刻は，1 組目の発行時刻を基準とした相対時刻で考える．各記事については，予め 6 つのグループ {“銀行”, “会社経営”, “株式”, “商取引”, “決算報告”, “その他”} に分類しておく．初めの 5 グループに (重複を含め) 属する記事は，実験データの 70.2%にあたる 1544 件，いずれにも属さない”その他”に属する記事は，29.8%にあたる 656 件である．また，記事中の不要語 (*stop word*) の削除および単語のステミング [36] をしておく．表 6.1 に実験データ中の記事の例，表 6.2 に実験データの詳細を示す.

実験データ中に現れる全 16336 アイテム (語) の分布情報は，事前知識として利用する．また，実験で生成される候補パターンは，(a, [ab], [ab]c 等のように) 選言の深さを 1 レベルのみに制限する．

全ての実験で利用される興味パターンは，正の興味パターン $Pos = \{$year, compani, bank, share, net$\}$，負の興味パターン $Neg = \{$said, mln, dlr, sale, pct, ct$\}$ とする．

Pos には主に bank, share 等の金融関係のパターン，Neg には said のような新聞記事特有のパターンや，dlr, pct などの数値や通貨の単位を表すパターンをそれぞれ選択しておく．

表 6.1: “株式”グループに属する記事の例

shr **loss** 51 ct **loss** ct
net **loss** 7041000 **loss** 467000
rev 138 9 mln 131 4 mln
12 month
shr **loss** 64 ct profit 46 ct
net **loss** 8843,000 profit 63060000
rev 558 9 mln 556 7 mln
note: **net** **loss** 4th qtr 1986 includ charg
restructur 2 6 mln dlr tax 19 ct **share**
1986 **net** **loss** includ tax special
charg 2 7 mln dlr 20 ct **share**

6.5.2 実験結果

実験 1 では，実験データ 1700 件に対して，ϵ_0 を 0.0025, λ を 0.98，$wsize$ を 100 件，th_{new} を 0.1，th_{ue} を 0.3 とそれぞれ設定し，頻出パターン抽出を行う．比較手法には，提案手法である興味重み計算と最適許容誤差の推定を行う手法 (KL+F 手法)，どちらの計算も行わない (none 手法) の 2 手法を用いる．予備実験の結果から，KL+F 手法では $th_{max_{pn}}$ を 1，$th_{max_{ue}}$ を 5，$loop$ を 1000 回，$rate$ を 1.2 とそれぞれ設定を行うとともに，σ を 0.003 とした時の頻出パターン上位 10 件を表 6.3 に示す．パターン横の数字は，そのパターンの頻度を表している．

実験 2 では，実験 1 と同じパラメタを用い，実験データ 2200 件に対しての DPL 構築時間と負荷 (節点の追加, 更新, 削除の各回数) を計測する．比較手法には，KL+F 手法，none 手法に加え，興味重み計算のみを行う手法 (KL 手法) の 3 手法を用いる．実験結果を表 6.4 に示す．

実験 3 では，実験 1 と同じパラメタを用い，実験データ 1700 件に対し，KL+F 手法，KL 手法，none 手法を用いて DPL 構築を行い，総節点数および σ を 0.003 として抽出した際の頻出パターン数を計測する．また結果から，頻度パターン上位 30 件のカバレージ $Cover$ と，総接点数に対して頻出パターン数の占める割合を示す DPL 空間効率をそれぞれ計算する．$Cover$ は (6.14) 式で定義し，実験結果を表 6.5 に示す．

$$Cover[\%] = \frac{\text{比較 2 手法で共に抽出されたパターン数}}{30} \times 100 \qquad (6.14)$$

表 6.2: 実験データの詳細

データ [組]	1	2	3	4	5	6	7	8	9	10	11
時刻 [day]	0	5	7	11	14	15	19	21	25	26	28
銀行 [件]	13	15	11	29	20	14	16	27	33	10	16
会社経営 [件]	35	37	32	31	36	36	27	33	12	34	34
株式 [件]	18	25	22	24	20	17	24	24	17	25	22
商取引 [件]	11	13	36	14	17	15	17	13	5	18	16
決算報告 [件]	22	29	27	10	25	23	26	28	7	20	21
その他 [件]	32	19	14	28	26	30	29	28	42	32	30
データ [組]	12	13	14	15	16	17	18	19	20	21	22
時刻 [day]	32	34	36	40	41	46	95	96	113	235	236
銀行 [件]	10	23	14	10	10	36	17	20	24	19	12
会社経営 [件]	33	28	31	21	35	28	44	29	38	18	38
株式 [件]	24	16	23	11	19	24	20	15	25	31	25
商取引 [件]	13	10	23	10	17	14	13	10	15	14	11
決算報告 [件]	17	24	15	12	24	12	14	19	14	26	34
その他 [件]	38	22	23	55	27	30	30	36	34	30	21

表 6.3: 頻出パターン上位 10 件 (1700 件)

順位	KL+F	none
1	compani(17.1)	said(39.7)
2	share(16.5)	mln(22.0)
3	year(15.8)	dlr(18.0)
4	1(15.7)	[said, dlr](16.8)
5	billion(15.5)	ct(15.6)
6	net(13.9)	[said,mln](14.3)
7	shr(13.6)	pct(14.1)
8	bank(13.3)	[mln, dlr](13.2)
9	loss(11.3)	[said, pct](12.9)
10	april(11.0)	compani(8.59)

表 6.4: DPL 構築負荷と実行時間

手法	KL+F	KL	none
追加 [回]	3045	3092	2644
更新 [回]	1920	1943	1781
削除 [回]	2729	2772	2368
実行時間 [msec]	192888	190609	45999

表 6.5: カバレージと DPL 空間効率

手法	KL+F	KL	none
V.S.	KL	none	KL+F
Cover[%]	100	43.3	43.3
総節点数	690	722	417
頻出パターン数	43	43	30
DPL 空間効率 [%]	6.23	5.96	7.19

6.5.3 考察

実験結果の考察を行う．実験 1 では，頻出パターンを抽出した．頻出パターンと興味重みの有無については，表 6.3 より，KL+F 手法では `compani`, `share` などの正の興味パターンが高頻出となっており，興味重みの有効性が確認できる．

表 6.2 より，“会社経営”に属する記事はデータ 1 組あたり平均 31 件と最も多く，両手法で高頻出パターン `compani` が抽出できる．一方で，“銀行”, “株式”に属する記事はそれぞれ平均 18 件，21 件と少ないが，KL+F 手法では，高頻出パターン `bank`, `share` が抽出できる．すなわち，興味重みを考慮することで，関連する記事の多少にかかわらず，興味パターンを高頻出であるとして抽出できる．

実験データ中の記事と KL+F 手法で得られる頻出パターンを比較する．表 6.1 の “株式”に属する記事に関して，正の興味パターン `net` や `share`(太字) に加え，興味パターンではない `loss` や `shr`(下線) についても抽出できる．後者は，記事の具体的内容を表しており，また前者と類似しているために高頻出パターンとして抽出されると考えられる．このことから，興味重みを考慮して得られる結果は，利用者の好みに合うことがわかる．

実験 2 では提案手法の実行効率と DPL 構築負荷を解析した．実行効率に関しては none 手法が最も良く，実行時間は KL+F,KL 両手法と比べ約 4.2 倍高速である．しかし，KL+F 手法と KL 手法では，実行時間差は約 2 秒と差がなく，このことから，提案手法では興味重み計算が中心となっていることがわかる．また，KL+F 手法と KL 手法を用いたときの構築負荷を比べると，KL+F 手法の方が約 1.5%少なく，最適許容誤差の推測により，効率的に DPL 構築ができることがわかる．

実験 3 では提案手法が頻出パターン抽出結果に与える影響と，DPL の空間効率を調べている．none 手法と KL+F,KL 両手法との各カバレージは 43.3%となる．つまり，興味重みによる重み付けの影響によって，これまで非頻出とされたパターンが抽出され，抽出結果が大きく変わったことを示している．また KL+F 手法は，KL 手法とのカバレージが 100%となり抽出結果自体は変わらないが，最適許容誤差の推測により，DPL 中の空間効率が 0.27%改善されている．

6.6 結論

本稿では，ニュースストリームから利用者の好みを反映した頻出パターンを抽出するマイニング手法を提案した．提案手法は，オンライン型の単一パスアルゴリズムであり，利用者からの興味パターンリストを用いて候補パターンに重み付けを行うと共に，最適許容誤差の推測を行うことで DPL を効率よく構築可能である．また，実験を通して，現実のデータに対して十分に有効であることを確認した．

第7章　結論

本研究では，ニュースストリームのようなニュース文書集合から，頻出パターンを効率よく抽出するための手法の提案を行った．また，利用者が興味を持つ有用なパターンを獲得することができ，内容分析・把握をする際の手助けができることを示した．

まずはじめに，静的なデータ環境下における頻出パターン抽出を考えた．単一系列データ上で選言パターンを対象とし，逆単調性を満たすパターン出現尺度を与えることで，APRIORI に基づく手法を提案でき，抽象度の高いパターンを効率的に抽出できた．また，利用者が頻出基準となるパラメタを変更し，再計算を伴う環境に関する問題にも言及した．オンライン分析のアイデアを適用し，選言パターン束と呼ぶ特殊なデータ構造の構築を行い，これを再利用することで，実用的な実行時間で頻出パターンの抽出ができることを実験により示した．

次に，上記の手法を一般化し，データストリーム環境における頻出パターン抽出を考えた．提案手法はニュースストリームを対象としたオンライン型の単一パスアルゴリズムである．確率的に頻度を誤差保証することで，任意の時間において近似解を得ることができ，また，忘却関数による重み付けを行うことで，時間変化を考慮した頻出パターン抽出が可能となった．

最後に，利用者の好みを考慮することで，利用者が興味を持つ，有用な頻出パターンの抽出手法を提案した．利用者からの好み情報として興味パターンを利用し，対象となるパターンとの類似度に基づき重み付けを行った上で，頻出パターン抽出を行い，実験により，得られる結果が利用者の好みを反映した結果となることを確認した．さらに，選言パターン束構築時のパラメタを推測し最適化することで，構築負荷の軽減することも実現することができた．

本研究で提案した手法で得られる頻出パターンは，ニュース文書中で語られている重要な話題に関しての要約情報，またはラベル情報を表現することができる．さらに，自動要約手法や自動分類手法との連携を考えることで，例えば，携帯電話や PDA などの小型デバイスのディスプレイ上に，利用者が興味のある情報を効果的に配置し表示することが可能になるだろう．このように，提案手法は幅広い分野への応用が期待できる．

今後の課題として，まず，データ分布の変化に対しての対処が挙げられる．ニュース文書はその特徴として，新たな話題が発生すると，それに関連する爆発的な量のニュースが生成されることが知られている．本研究で提案した手法では，時間に関する重みを考慮し計算するため，それが実行効率の悪化に繋がることが分かっている．さらに，利用者の好みを考慮する上で，好みが移り変わることを考えなければならない．一般

に，利用者の好みは時間経過とともに変化するものであり，興味パターンに対して時間の概念を導入することで好みの変化を反映する，あるいは適当なタイミングで，対話的に入れ替えを行う方法などを検討する必要があろう．

謝辞

本研究を遂行するにあたり，日頃より適切な御指導，御鞭撻をいただいた，法政大学工学部情報電気電子工学科 三浦孝夫教授に深く御礼申し上げます.

並びに，産能大学経営情報学科 塩谷勇教授にも多くの有益なご助言をいただきました．深く感謝いたします.

データ工学研究室の先輩方，同輩，後輩たちにも，本研究の遂行にあたって数多くの助言と快適で能率的な研究室環境を整えていただきました．御礼申し上げます.

修士論文として私の研究をまとめることができたのも，多くの皆様方の御支援，御協力の賜物であります．この場をお借りしまして，厚く御礼申し上げます.

最後に，今までの学生生活を支えてくださった私の両親，兄弟に深く感謝したいと思います.

参考文献

[1] Aggarwal, C.C.. and Yu, P.S.: Online Generation of Association Rules, ICDE, 1998, pp.402-411

[2] Agrawal, R. and Srikant, R.: Fast Algorithm for Mining Association Rules, proc. VLDB, 1994, pp.487-499

[3] Agrawal, R. and Srikant, R.: Mining Sequential Patterns, proc. ICDE, 1995, pp.3-14

[4] Asai, T. Abe, K. Kawasoe, S. Arimura, H. Sakamoto, H. and Arikawa, S.: Efficient substructure discovery from large semistructured data, Proceedings of the 2nd SIAM International Conference on Data Mining (SDM '02), 2002, pp.158-174

[5] Garofalakis, M., Rastogi, R. and Shim, K.: SPIRIT : Sequential Pattern Mining with Regular Expression Constraints, proc. VLDB, 1999, pp.223-234

[6] Chen, Y., Dong, G., Han, J., Wah, B. W. and Wang, J.: Multi-dimensional regression analysis of time-series data streams, International Conference on Very Large Databases, 2002, pp.323-334

[7] DBLP, http://dblp.uni-trier.de/

[8] Feng, F. Hsu, W. Lee, M.L.: Efficient Pattern Discovery for Semistructured Data, International Conference on Tools with Artificial Intelligence (ICTAI), 2005, pp.294-301

[9] Goethals, B.: A Survey on Frequent Pattern Mining, Univ.of Helsinki, 2003

[10] Goethals, B., Hoekx, E. and Bussche, J.V.d.: Mining Tree Queries in a Graph, ACM SIGKDD, 2005, pp.61-69

[11] Han, J., Pei, J. Mortazavi, B., Chen, Q, Dayal, U. and Hsu, M-C.: FreeSpan: Frequent Pattern-Projected Sequential Patterns Mining, proc. KDD, 2000, pp.355-359

[12] Han,J. and Kamber, M.: Data Mining : Concepts and Techniques, Morgan Kaufmann, 2000

[13] Albert-Lorincz, H. and Boulicaut, J-F.: Mining Frequent Sequential Patterns under Regular Expressions: A Highly Adaptative Strategy for Pushing Constraints, proc. SIAM DM, 2003, pp.316-320

[14] Jian, N. and Gruenwald, L.: Research Issues in Data Stream Association Rule Mining, SIGMOD Record 35-1, 2006, pp.14-19

[15] Loo, K.K., Tong, I. and Kao, B.: Online Algorithms for Mining Inter-stream Associations from Large Sensor Networks, Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2005, pp.143-149

[16] Mcgarry, K.: A Survey of Interestingness Measures for Knowledge Discovery, The Knowledge Engineering Review Vol.20,Issue 1, 2005, pp.39-61

[17] Hand, D., Mannila, H. and Smyth, P.: Principles of Data Mining, MIT Press, 2001

[18] Mannila, H. and Toivonen, H. and Verkamo, I.: Discovery of Frequent Episodes in Event Sequences, *Data Mining and Knowledge Disvocery* 1(3), 1997, pp.259-289

[19] Manku G. S. and Motwani R.: Approximate frequency counts over data streams, VLDB, 2002, pp.346-357

[20] Metwally, A., Agrawal, D. and Abbadi, A. E.: Efficient computation of frequent and top-k elements indata streams, Proc. ICDT, 2005.

[21] Motoyoshi, M., Miura, T., Watanabe, K. and Shioya, I.: Temporal Class Mining for Time Series Data, proc. CIKM, 2002

[22] Ohuchi, H., Miura, T. and Shioya, I.: Document Retrieval using Projection by Frequency Distribution, IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2005, pp.356-361

[23] Pei, J., Han, J., Mortazavi, B., Pinto H., Chen, Q, Dayal, U. and Hsu, M-C.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Patter Growth, proc.ICDE, 2001, pp.215-224

[24] Shimizu, K. and Miura, T.: Disjunctive Sequential Patterns on Single Data Sequence and its Anti-Monotonicity, International Conference on Machine Learning and Data Mining (MLDM), 2005, pp.376-383

[25] Shimizu, K. and Miura, T.: Online Analysis for Disjunctive Sequential Patterns, ADBIS Workshop on Data Mining and Knowledge Discovery (ADMKD), 2006, pp.61-72

[26] Shimizu, K., Shioya, I. and Miura, T.: Mining Disjunctive Sequential Patterns from News Stream, 8th Intn'l Conf. on Intelligent Data Engineering and Automated Learning (IDEAL), 2007, pp.630-642

[27] Srikant, R and Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements, proc.EDBT, 1996, pp.412-421

[28] Toivonen, H.: Sampling Large Databases for Association Rules, VLDB, 1996, pp134-145

[29] Zaki, M.J.: Efficient Enumeration of Frequent Sequences, proc.CIKM, 1998, pp.68-75

[30] Zaki, M.J.: Efficiently mining frequent trees in a forest, ACM SIGKDD, 2002, pp.71-80

[31] 浅井, 有村: 半構造データマイニングにおけるパターン発見技法, 電子情報通信学会論文誌 VOL.J87-D1 No.2, 2004, pp.79-96

[32] 安部, 藤野, 下薗, 有村, 有川: テキストデータからの高速データマイニング : 探索的文書ブラウジングとウェブデータへの応用, 人工知能学会誌 Vol.15, No.4, 2000, pp.618-628

[33] 安部，川副, 浅井, 有村, 有川: 大規模半構造データストリームからの知識発見, データ工学ワークショップ (DEWS), 7C-02, 2003

[34] 石川, 北川: 忘却の概念に基づくクラスタリング手法の改良方式, DBSJ Letters Vol.2 No.3, 2003, pp.53-56

[35] 上嶋，三浦，塩谷: 不完全なニュース集合からのタイムスタンプ推定, データ工学ワークショップ (DEWS), 3C-o4, 2005

[36] 北, 津田, 獅子堀: 情報検索アルゴリズム, 共立出版, 2002

[37] 高野，岩沼，鍋島: 単一の長大なデータ系列上の系列パターンの出現尺度とその逆単調性, 第 3 回情報科学技術フォーラム (FIT), 2004, pp.115-118

[38] 清水, 三浦: 選言パターン抽出のオンライン分析, DBSJ Letters Vol.4 No.3, 2005, pp.9-12

[39] 長尾 真: 自然言語処理, 岩波書店, 1996

[40] 永田, 平田: "テキスト分類–学習理論の「見本市」–", 情報処理,vol.42(1),pp:32–37(2001)

[41] 本吉，三浦，塩谷: 回帰分析によるストリームデータのクラスタリング, 日本データベース学会 DBSJ Letters Vol.2,No.3, 2003, pp.45-48

[42] 吉田，太田，小林，湯上: サブパターンとスーパーパターンからの推定頻度に基づくパターンの興味深さの尺度の評価, DBSJ Letters Vol.3 No.1, 2004, pp.57-60

[43] 比土, 河野: 頻出順序木発見のための効率的な列挙手法の提案, DBSJ Letters Vol.4 No.1, 2005, pp.165-168

[44] 房延, 浅井, 有村, 宇野, 中野: 半構造データマイニングのための高速な無順序木パターン発見手法, データ工学ワークショップ (DEWS), 2004, 6A-o3