

多属性データに対する前処理の木構造を用いたモデル化

山田 有吉[†] 市瀬 龍太郎[‡] 沼尾 正行[†]

[†] 東京工業大学計算工学専攻 〒152-8552 東京都目黒区大岡山 2-12-1

[‡] 国立情報学研究所 知能システム研究系 〒101-8430 東京都千代田区一ツ橋 2-1-2

E-mail: [†] {yukichi, numao}@nm.cs.titech.ac.jp, [‡] ichise@nii.ac.jp

あらまし データベースに蓄積された膨大なデータから有用な知識を獲得するKDDプロセスにおいて、データを整理・加工する前処理の重要性は、多くの専門家によって指摘されている。しかし、前処理を専門に扱った研究やツールの開発は十分に進展していないのが現状である。そこで本研究では、試行錯誤的に行われてきた前処理に対する独自のモデル化を行い、複雑な処理群を半自動的にサポートするシステムを提案した。本研究では、視認性が高く、柔軟に構造変更を行うことのできる木構造を用いることにより、多くの属性を伴う複雑なデータに対する処理を、効率よく行うことを可能にした。また、実際の医療データに対して実験を行い、提案システムの有用性に対する検証を行った。

キーワード KDD, 前処理, データマイニング, 木構造

An Effective Pre-processing Model Using Layered Structure

Yukichi Yamada[†] Ryutaro Ichise[‡] and Masayuki Numao[†]

[†] Department of Computer Science, Tokyo Institute of Tech, 2-12-1 Oookayama, Meguro-ku, Tokyo, 152-8552 Japan

[‡] Intelligent Systems Research Division, National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430 Japan

E-mail: [†] {yukichi, numao}@nm.cs.titech.ac.jp, [‡] ichise@nii.ac.jp

Abstract Knowledge discovery in databases (KDD) requires huge data, which takes a long time to be preprocessed. Although each element of pre-processing is simple, it tends to be quite complicated and is hard to construct the whole plan. To reduce the load, we propose the original model for pre-processing and an interactive and dynamic planning tool for pre-processing, named TransX. This system is based on XML transformation, which enables to visualize the process by using a treelike notation and it allows a user to process data easily and understandably. We propose the original model for preprocessing scheme, which enables to define the preprocessing plan simply, and shows how the system, TransX, carries out the plan concretely

Keyword KDD, preprocessing, data mining, layered structure

1. はじめに

CPU の処理速度は最近になってますます加速されており、データを格納するストレージの容量もその増加は留まるところを知らない。これらマシン性能の向上と同期して、これまでは成し得なかったような大容量データを格納する大規模データベース技術が発展し、いたるところで大量のデータが蓄積されるにいたっている。このような、大規模データベースからの知識発見プロセスをKDD(Knowledge Discovery in Databases)と呼ぶ。KDDとは、目的に則した形で加工したデータを統計手法や機械学習手法を用いて解析を行うことにより、人にとって捉えることが困難な知識を発見するプロセスとして注目されている。

この KDD プロセスにおいて、データの処理・加工を行うステップを前処理と呼ぶ。前処理には構造の変形や

値の標準化などが含まれるが、これらの作業は事例によって処理が異なり、また経験の求められる複雑な作業である。そのため、KDD プロセスにおける前処理に必要とされる処理コストは、全体の 60%を要するといわれている[1]。

本研究はこの前処理における処理コストを削減することを目標とする。木構造によるデータ表現形式を使用することにより、属性間の関係性を可視化し、木構造の構造変更に対する柔軟性を利用することにより、動的に構造を変更しながらの処理を可能にする。また、木構造における処理要素を考慮することにより、前処理全体に対するモデル化を行い、処理の流れを規定する。

以下、第2章で関連研究を紹介し、第3章で本研究のアプローチを説明する。また、提案手法である木構造によるデータ処理と、前処理のモデル化に対する説明を第4

章、第5章でそれぞれ行う。また、第6章において実装に伴うシステムの説明を行い、第7章で従来手法である関係データベースツールとの比較実験と、処理されたデータの出力例を示す。そして最後に、第8章で本研究のまとめを述べる。

2. 関連研究

現在の前処理研究の動向としては、データクリーニングのようなデータ処理を専門に扱った研究よりもむしろ、データマイニングアルゴリズムとの親和性を高める方向で、アルゴリズムが許容する形にデータを変換する研究や、属性選択に関する研究などが盛んに行われている。

Wu は、2つの帰納学習アルゴリズムを目標データにかけ、その結果を比較することで前処理済みデータを求めようとしている[2]。これはユーザの負担を減らす、前処理の自動化の方向性を示唆するものである。また自動化の方向性とは別に、不完全な関係データベースから前処理を行うことなく直接関連ルールを発見する手法も提案されている[3] [4]。一方、属性選択の研究としては、データベース中の属性に与えられた、領域知識としての概念木による属性値の一般化と、重複度を考慮した属性除去を含めた冗長データの除去を行う属性指向アルゴリズムの提案が行われている[5] [6]。これは属性値を背景知識によって一般化し、他の属性値に置き換えることによって、単独ではルールとして出力されにくい個々の属性値のサポートを上げる手法である。

このように解析部との親和性を高めるような研究は行われているものの、実際に行われるデータ処理に関する研究は少なく、現時点で実際に前処理を行う場合は、単純だが有効性が明らかなものを人間であるオペレータが計画を立てて行っているのが実情である。

3. 本研究のアプローチ

本研究では前処理の支援範囲を、データの入力からデータマイニングアルゴリズムへの投入までのデータ処理に加え、ユーザがデータ構造を認識し、処理目標を決定するデータ解析もその範囲であると考えている。解析目標を定める意思決定は、KDDにおいて重要な意味を持ち、それはデータを理解することによって可能となる。そのため、データ構造の視認性は前処理を行う上で大きな意味を持つと考えられる。

現在のデータ表示については、これまでのデータベースに関する研究の流れを継承する形で、表形式によるものが多い。しかし、表形式はデータの格納効率や一義性に対しては利点を有するが、人間としてのユーザがデータを視認するのに適した形式ではないと考えられる。というのも、表形式においては各属性の立場は並列であり、属性数の増加に伴ってデータ構造は認識しづらくなるという欠点がある。また、表形式においてはデータの処理単位が基本的に列に対する処理となるため、単一の属性

値にアクセスすることのできる指標がフィールド名のみになってしまうという問題もある。

そこで本研究では、木構造を使用することによりデータの視認性を高める。属性をノードとして表現することにより、属性間の関係性を構造情報として可視化することが可能であるし、木構造は構造変形に対して柔軟であるため、動的に構造を変形しながらの様々なスコープに対する処理が可能となる。また、木構造に対する処理を前提として前処理要素を分類することにより、これまで経験的に行われてきた処理過程を体系化することが可能となる。

このように本研究では、木構造を用いた前処理工程をモデル化することによって、前処理に必要とされるコストを削減することを目的とする。

4. 木構造を用いたデータ処理

4.1. 木構造の定義

本研究では、木構造を使用することによってデータの可視化を行い、属性間の関係性を親子関係、兄弟関係として表現する。木構造は以下の定義に従って構成されるものとする。

1. 1つの属性を1つのノードとしてあらわし、属性値は各ノードが内部に保持する
2. 葉ノードは解析目標となる属性、内点¹はそれを識別するための属性とする
3. 同階層のノードは全て並列の関係を表す
4. 同階層のノードは全て同じ属性型を保持する
5. 同階層のノードは同じ階層名を保持する

4.2. 背景知識による階層化

前処理を行う際には、データに関する専門知識が必要とされる。というのも、データを操作する際には、属性や属性値の意味を把握することが必要とされるからである。特に、医療データのように100種もの検査項目が存在する場合、データそのものから得られる情報以外に、ユーザが保持しているデータに対する専門知識が重要となる。これらの専門知識を、本研究ではデータに対する背景知識として利用し、これら背景知識を直接データに附加しながら処理を進めることによって、属性を分類しながらデータ構造の変更を行う手法を提案する。

例えば図1の左木が入力として与えられた場合を考える。このとき、葉ノードは検査項目名をあらわしているが、それぞれのノードについての処理を行う際、各検査項目に対する情報はユーザに何も与えられていない。表形式についても同様で、各属性がフィールド名として作成された表を想定したとき、いくつもの画面に渡る属性名から、ユーザは必要とされる属性を検索し、それぞれの属性の意味を毎回考えながら多くの処理を行っていく。

¹ 葉ノード以外のノードを、内点と呼ぶこととする。

表 1：前処理における処理要素

操作	構造変換	データ型選択	表記の統一	離散化	データ結合	属性選択	欠損値補完
基本的な前処理	○	○	○				
特別な前処理	△			○	○	○	○

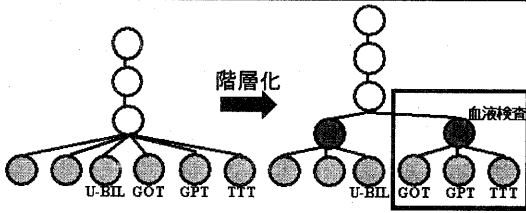


図 1：背景知識による階層化

ここで、“GOT”、“GPT”、“TTT”が血液検査に対する検査項目であるという背景知識を用いて、右木に変形されるような階層化を本システムではサポートする。このようにノードを追加して属性をグループ化することによって、以下のような情報がデータに加味されたことになる。

1. “GOT”と“GPT”は「検査区分」という視点から見たときに同等の立場にある
2. “GOT”と“U-BIL”は「検査区分」という視点から見たときに異なる立場にある
3. “GOT”は「血液検査」という属性に一般化することができる

このように、ユーザのデータに対する背景知識をデータに加えることにより、各属性に対する情報を可視化することができる。また、類似した処理が多くなる前処理において、このグループ化によって与えられた情報を用いることで、ユーザの負担を軽減することができると思われる。このような単純な階層化は関係データベース上でも可能であるが、いくつかの階層を生成し、多次元的に属性进行分类するような処理は、関係データベースでは扱うことができない。また階層化によって生成されたノードは、次節で紹介するダイシングと組み合わせることによって、より有効に利用することが可能である。

4.3. ダイシング

多次元データウェアハウスにおいて、多数ある次元の中から、ユーザの必要とする次元に則したデータを提示する手法をダイシングと呼ぶ。つまりダイシングとは、ユーザが必要とする次元に対して視点を変更する手法である。この手法により、分析に必要とされる次元によって参照可能なデータを、ユーザに提示することができる。

本研究ではこの概念を木構造に应用する。4.1節での定義から、全ての内点はその葉ノードを識別するためのラベルとして存在している。この内点を、その属性値をノ

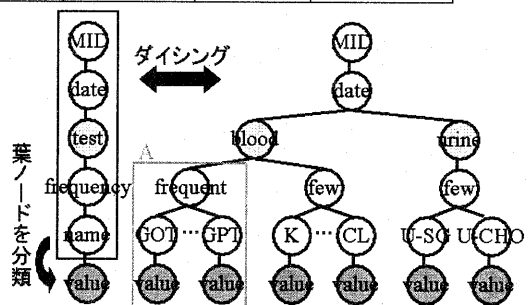


図 2：ダイシング

ード名として展開する処理を、本手法ではダイシングとして実現する。図 2はダイシングの一例である。図 2において、左木の“test”ノードには、属性値として“blood”、“urine”という2種の文字列が格納されている。ここで、このノードに対してダイシングを行うと、右木のように属性値をノード名とする2つのノードに展開される。また同様に不必要な階層に関しては、展開されたノードを縮小することも可能である。

このようにダイシングの概念を木構造に应用することにより、任意の属性に対して展開と縮小を行い、必要とされる指標によって葉ノード进行分类することが可能となる。またユーザの要求に応じて木構造を動的に変形することにより、属性間の関係性を容易に把握することが可能となる。また図 2の“A”で示したサブツリーは、テスト内容が“blood test”で、頻出度が“frequent”である属性値を指定している。このように、ダイシングによって得られた木構造上の、任意のサブツリーをスコープとしての処理を可能とすることにより、反復の多い前処理コストを削減できるものと考えられる。

5. 前処理モデル

前処理に必要とされる処理要素は、解析目標と使用するデータマイニングアルゴリズムによって多岐に渡る。本研究では、Collider らによって、KDD ツールの評価基準として指定されている処理要素について考えることとする[7]。

これら処理要素を、「基本的な前処理」と「特別な前処理」の二つのグループに分類する。これら分類の基準としては、フィードバックがかかった際に必要とされる処理であるかどうかとする。表 1に前処理要素とその分類分けを示す。このように前処理要素进行分类することで、次あげるようなメリットがあると考えられる。

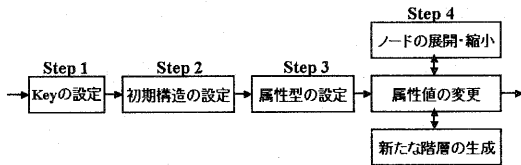


図 3：基本的な前処理プロセス

1. 各処理の位置付け：これまで雑多に配置されていた処理の位置付けを行うことにより、状況に対して必要な処理のみをユーザに提示することが可能となる
2. フィードバックの制御：フィードバックがかかった際に、前処理のどの過程に戻るかといった手がかりとなる
3. 一部処理群の自動化：フィードバック後に必要とされる処理とは、状況依存な処理である。その処理群をくりだすことによって、一部の処理群を自動化できると考えられる。

5.1. 基本的な前処理

「基本的な前処理」はフィードバックの起こらない処理群であり、おもにデータの初期設定とデータクリーニングを行うプロセスである。このプロセスは単一のデータに対して一度の処理で充足する処理であるが、データの整合性を高める上で一度は行わなくてはならない処理である。フィードバックの起こらないという性質から、本システムではこのプロセスの半自動化を行う。

具体的な処理プロセスを図 3に示す。基本的な前処理プロセスは 4つのステップからなっており、この図に則した形でシステムの提示に答えていくことによって、ユーザは対話的に基本的な前処理プロセスを行っていく。

またこのプロセスでは、動的に木構造を変形しながら処理を行っていく。ユーザの背景知識によって属性をグループ化しながら階層化を行い、ダイシングによってユーザの必要とする次元を参照することによって、様々なスコープに対する処理をサポートする。

5.2. 特別な前処理

「特別な前処理」はフィードバックのかかる、状況依存な処理群である。ここでフィードバックがかかるとは、解析目標の変化から処理の過程を変更するという事で、試行錯誤を必要とする KDD プロセスにおいてはこのような状況が多く存在する。このような場合に本手法では、基本的な前処理プロセス以前まで処理をさかのぼることはなく、図 4に示したように特別な前処理での処理内容を変化させることにとどめることができる。これは、前処理要素をフィードバックが必要とされるかどうかという方針で分類したためである。基本的な前処理

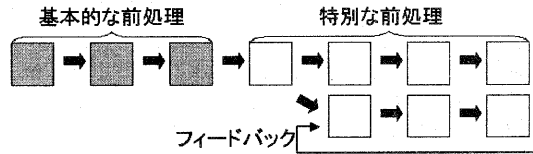


図 4：処理過程へのフィードバック

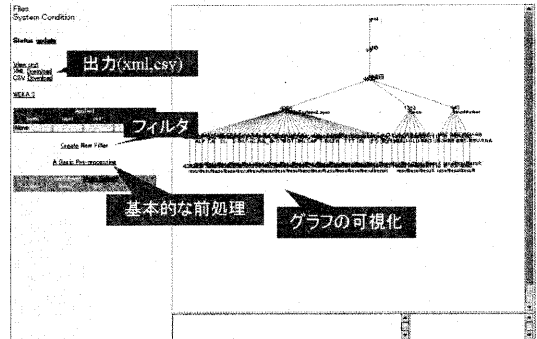


図 5：システムのインターフェース

ののちに特別な前処理を行うという方針によってモデル化がなされているために、解析目標が変更され、再度データの加工が必要となった際にも同様の処理を再度行うことなく、これまでの処理過程を利用することが可能となる。

6. 実装

6.1. TransX

本研究は、TransX システムを拡張する形で実装を行う。TransX とは、五十嵐らによって提案された、XML 形式でデータ操作を行うシステムである[8]。

システム内においては、XML として入力されたデータが、インターフェース上で木構造として表示される。ユーザはその木構造に対して、フィルタと呼ばれる処理単位の組み合わせである、フィルタパスを適用することによりデータ処理を行っていく。また、このフィルタパスはオブジェクトとしてプロジェクトに保存されており、処理の任意の過程に立ち返って処理を再開することが可能である。また、本研究で定義した木構造にそった形で構造変換を行うフィルタを用意し、それらのフィルタを提案モデルに沿う形でユーザに提示するインターフェースを実装することにより、システムの提示に従ってユーザは対話的に前処理を実現することが可能となる。システムのインターフェースを図 5に示す。

6.2. 処理の流れ

システムに実装された処理の流れを図 6に示す。入力

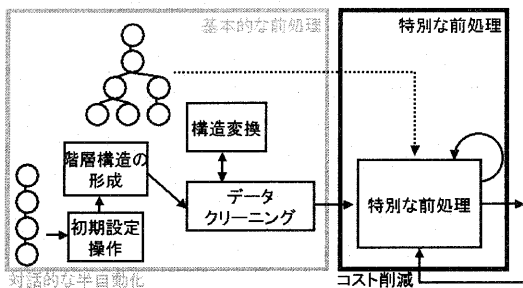


図 6: 処理の流れ

されたデータは、初期設定をした後に、ユーザのデータに対する背景知識をデータに附加する形で階層化を行う。そののちに、この階層構造を利用する形でデータクリーニングを行う。この際、ダイシングによって各ノードを展開、縮小しながら構造変換を行うことにより、ユーザは必要な属性情報を参照しながら、必要なスコープのサブツリーに対して処理を行うことができる。これらの処理は、対話的にステップが遷移していく中で行われていく、半自動化として実装されている。

データクリーニングを終えた時点で、処理は特別な前処理へと移行する。特別な前処理は状況依存的な処理であるため、必要な処理をユーザが選択することによって行っていく。この際、基本的な前処理で形成された背景知識による階層情報を利用することで、このプロセスでの処理コストを削減することができると考えられる。また、データマイニングアルゴリズムからのフィードバックは特別な前処理にのみかかり、基本的な前処理はデータの追加が行われない限り一度の施行で充足する。

7. 実験

一般的に使用されているデータベースツールである Microsoft Access との、処理ステップ数に対する比較実験を行った。

この場合のステップ数とは、それぞれのアプリケーションで用意されている最小単位の処理であり、本システムではフィルタ、Access においては GUI 上で用意されているクエリを指す。尚、実装の仕様によっては、複数の処理をひとつのフィルタとして用意することも可能であるが、Access における複数のクエリに相当するマクロ的なフィルタは存在していない。

7.1. 実験データ

本実験では、千葉大学附属医科大学第一内科から提供された、1982 年から 2001 年までに肝生検を受けた B 型・C 型の慢性肝炎患者についての検査データをメインデータとして使用した。今回はシステムの処理速度の関係から、患者 ID1~20 のデータのみを使用することとする。抽出データは 15,683 レコード、105 種の検査項目によ

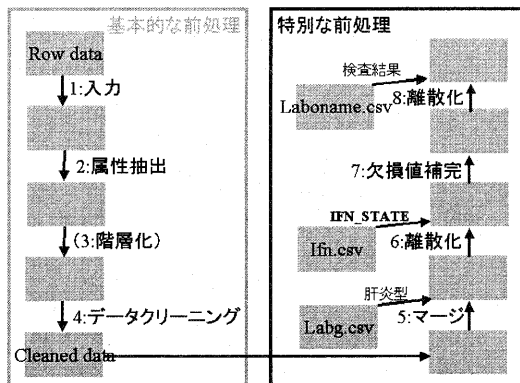


図 7: 処理手順

て構成される。データ量は csv で 352KB、XML で 3.22MB である。

7.2. 処理目標

今回の実験では、図 7 に示す工程を前処理目標として設定した。この前処理は、主に決定木などを用いて、インターフェロンの効果に対するルールを得るのに有効な処理工程であると考えられる。この処理工程において、データクリーニングまでの処理が基本的な前処理、その後の処理が特別な前処理にあたる。

以下に、具体的な処理の流れを説明する。

1. 入力: 属性型の決定などの初期設定を行った
2. 属性抽出: 105 種の検査項目を上位 42 項目に絞り込んだ。
3. 階層構造の形成: 検査区分、文字型、頻出度の 3 つの階層を附加することによって、検査項目を分類した。なお、この階層化は提案手法に固有の処理であり、既存手法には行われない。
4. データクリーニング: 名寄せ、及び表記ずれの修正をおこなった
5. 属性の追加: 肝生検データファイルから「肝炎型」という属性を抽出し、患者 ID についてマージした
6. 日付の離散化: 投薬データファイルの「投与開始日」「投与終了日」によって検査日を「投与前」「投与中」「投与後」「未投与」の 4 種に離散化した
7. 欠損値補完: 「検査結果」の欠損値を線形補完した
8. 値の離散化: 欠損値補完された「検査結果」を検査項目データファイルの「正常上限値」「正常下限値」によって、「上限異常」「正常」「下限異常」の 3 種に離散化した

7.3. 実験結果

実験結果を図 8 に示す。この結果において、処理の前半である基本的な前処理においては、本手法と Access では顕著な差は確認できていない。これは、この時点

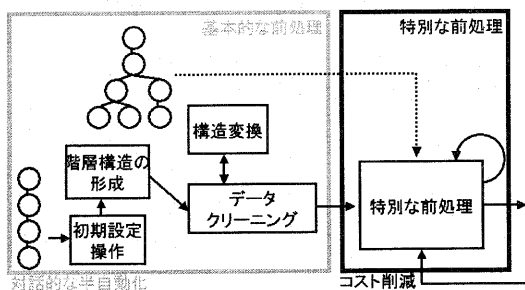


図 6: 処理の流れ

されたデータは、初期設定をした後に、ユーザのデータに対する背景知識をデータに附加する形で階層化を行う。そののちに、この階層構造を利用する形でデータクリーニングを行う。この際、ダイシングによって各ノードを展開、縮小しながら構造変換を行うことにより、ユーザは必要な属性情報を参照しながら、必要なスコープのサブツリーに対して処理を行うことができる。これらの処理は、対話的にステップが遷移していく中で行われていく、半自動化として実装されている。

データクリーニングを終えた時点で、処理は特別な前処理へと移行する。特別な前処理は状況依存的な処理であるため、必要な処理をユーザが選択することによって行っていく。この際、基本的な前処理で形成された背景知識による階層情報を利用することで、このプロセスでの処理コストを削減することができると考えられる。また、データマイニングアルゴリズムからのフィードバックは特別な前処理にのみかかり、基本的な前処理はデータの追加が行われない限り一度の施行で充足する。

7. 実験

一般的に使用されているデータベースツールである Microsoft Access との、処理ステップ数に対する比較実験を行った。

この場合のステップ数とは、それぞれのアプリケーションで用意されている最小単位の処理であり、本システムではフィルタ、Access においては GUI 上で用意されているクエリを指す。尚、実装の仕様によっては、複数の処理をひとつのフィルタとして用意することも可能であるが、Access における複数のクエリに相当するマクロ的なフィルタは存在していない。

7.1. 実験データ

本実験では、千葉大学附属医科大学第一内科から提供された、1982 年から 2001 年までに肝生検を受けた B 型・C 型の慢性肝炎患者についての検査データをメインデータとして使用した。今回はシステムの処理速度の関係から、患者 ID1~20 のデータのみを使用することとする。抽出データは 15,683 レコード、105 種の検査項目によ

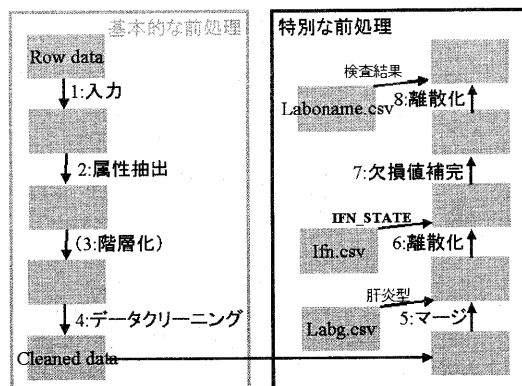


図 7: 処理手順

て構成される。データ量は csv で 352KB、XML で 3.22MB である。

7.2. 処理目標

今回の実験では、図 7 に示す工程を前処理目標として設定した。この前処理は、主に決定木などを用いて、インターフェロンの効果に対するルールを得るのに有効な処理工程であると考えられる。この処理工程において、データクリーニングまでの処理が基本的な前処理、その後の処理が特別な前処理にあたる。

以下に、具体的な処理の流れを説明する。

1. 入力：属性型の決定などの初期設定を行った
2. 属性抽出：105 種の検査項目を上位 42 項目に絞り込んだ。
3. 階層構造の形成：検査区分、文字型、頻出度の 3 つの階層を附加することによって、検査項目を分類した。なお、この階層化は提案手法に固有の処理であり、既存手法には行われない。
4. データクリーニング：名寄せ、及び表記ずれの修正をおこなった
5. 属性の追加：肝生検データファイルから「肝炎型」という属性を抽出し、患者 ID についてマージした
6. 日付の離散化：投薬データファイルの「投与開始日」「投与終了日」によって検査日を「投与前」「投与中」「投与後」「未投与」の 4 種に離散化した
7. 欠損値補完：「検査結果」の欠損値を線形補完した
8. 値の離散化：欠損値補完された「検査結果」を検査項目データファイルの「正常上限値」「正常下限値」によって、「上限異常」「正常」「下限異常」の 3 種に離散化した

7.3. 実験結果

実験結果を図 8 に示す。この結果において、処理の前半である基本的な前処理においては、本手法と Access では顕著な差は確認できていない。これは、この時点