STUDIA INFORMATICA 2002
Volume 23 Number 3 (50)

Stanisław CIEŚLA, Marek SIKORA, Arkadiusz TWARDOŃ
Politechnika Śląska, Instytut Informatyki

POZYSKIWANIE WIEDZY Z BAZ DANYCH Z WYKORZYSTANIEM TECHNOLOGII MICROSOFT .NET

Streszczenie. Artykuł przedstawia próbę stworzenia serwisu WWW w oparciu o technologię .NET firmy Microsoft wspomagającego pozyskiwanie wiedzy z baz danych.

WEB SERVICE FOR KNOWLEDGE DISCOVERY IN DATABASES USING MICROSOFT .NET TECHNOLOGY

Summary. The article presents proposal of creating web service for knowledge discovery in databases using Microsoft .NET.

1. Wprowadzenie

W ostatnich latach intensywnie rozwijającą się dziedziną informatyki jest pozyskiwanie wiedzy z baz danych (ang. *knowledge discovery in databases - KDD*), które jest jedną z gałęzi zgłębiania danych (ang. *data mining*). Pozyskiwanie wiedzy definiowane jest jako proces wydobywania prawdziwych, wcześniej nieznanych, zrozumiałych i użytecznych wzorców z danych.

Do wyznaczenia takich wzorców stosowane są algorytmy uczenia się pojęć na podstawie przykładów [3]. Systemy wykorzystujące takie algorytmy znajdują szerokie zastosowanie w rozwiązywaniu problemów pojawiających się w przemyśle [4], medycynie, informatyce oraz innych dziedzinach gospodarki.

Istotną cechą algorytmów uczących się opisów pojęć na podstawie przykładów jest to, iż wykorzystują one wnioskowanie indukcyjne, które nie jest wnioskowaniem niezawodnym, tzn. stosując to wnioskowanie, można przejść z prawdziwych przesłanek do fałszywych

wniosków, z tego względu efektywność systemów uczących się na podstawie przykładów weryfikowana jest poprzez obliczenie dokładności klasyfikacji na testowym zbiorze obiektów [9].

W dziedzinie *KDD* wyznaczone wzorce powinny nie tylko być opisami pojęć charakteryzującymi się dobrymi zdolnościami klasyfikacyjnymi, lecz także opisy te powinny być intuicyjnie zrozumiałe dla użytkownika. W szczególności *KDD* koncentruje się na uzyskaniu jak najprostszych opisów pojęć. Opisy te wykorzystywane są w systemach wspomagających podejmowanie decyzji.

Bardzo użyteczne w *KDD*, ze względu na prostotę języka opisu, są algorytmy uczące się regułowych opisów pojęć. W szczególności do wyznaczania regułowych opisów pojęć można wykorzystać teorię zbiorów przybliżonych [8], która pozwala generować reguły w postaci: $(a_1,V_{a_1})\wedge..\wedge(a_n,V_{a_k})\rightarrow(d,v_d)$ gdzie $a_1, ..,a_k$ należą do zbioru cech nazywanego potocznie zbiorem atrybutów warunkowych, d jest atrybutem decyzyjnym, V_{a_i} jest pewnym zbiorem wartości atrybutu a_i oraz v_d jest wartością atrybutu decyzyjnego.

Reguła wyraża zależność pomiędzy wartościami atrybutów warunkowych a wartością atrybutu decyzyjnego. Mówimy, że reguła $(a_1,V_{a_1})\wedge..\wedge(a_n,V_{a_k})\rightarrow(d,v_d)$ opisuje obiekt x (lub inaczej, reguła rozpoznaje obiekt x), jeśli $\forall_{i\in\{1,...,k\}}\ (a_i(x)\in V_{a_i})$. Zapis $a_i(x)$ oznacza wartość cechy a_i, jaką posiada obiekt x.

W procesie *KDD* wyznaczane mogą być reguły o różnej mocy opisowej, znaczy to, że część reguł może opisywać pewne szczególne zależności występujące w danym, konkretnym zbiorze treningowym. Zależności takie prawdziwe są jedynie dla niewielkiej liczby obiektów treningowych. Zjawisko generowania reguł o małej mocy opisowej jest nierozerwalnie związane z analizą danych dotyczących rzeczywistych zastosowań, gdyż dane takie zazwyczaj obarczone są niepewnością.

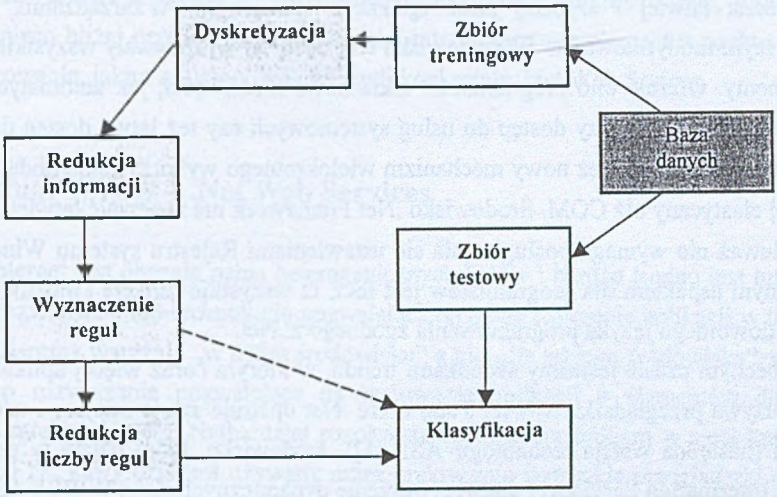
Zgodnie z definicją *KDD* takie szczególne zależności nie mogą być traktowane jako nowa odkryta wiedza, konieczne jest zatem stosowanie algorytmów pozwalających na generowanie tylko reguł o dużej mocy opisowej lub wskazujących w danym zbiorze reguł, reguły najsilniejsze (o największej mocy opisowej).

Ogólnie proces odkrywania wiedzy w bazach danych może być scharakteryzowany za pomocą schematu przedstawionego na rysunku 1.

Przed przeprowadzeniem procesu *KDD* konieczne jest skonstruowanie dwóch tablic. W jednej z nich znajdować się będą rekordy (obiekty) poddawane analizie (będzie to tzw. tablica treningowa), w drugiej znajdować się będą rekordy służące zweryfikowaniu jakości odkrytych reguł (będzie to tzw. tablica testowa). W tablicach treningowej i testowej nie powinny

występować obiekty o identycznych wartościach atrybutów (tzn. tablice powinny być rozłączne).

Jeśli wartościami pewnych pól (atrybutów) znajdujących się w tablicy treningowej są liczby rzeczywiste, to konieczne jest przeprowadzenie procesu dyskretyzacji (skalowania wartości) tych atrybutów [5].



Rys. 1. Generowanie reguł z baz danych
Fig. 1. Rules generation from databases

W dalszej części procesu *KDD* celowe może być zredukowanie liczby pól (atrybutów) występujących w rekordach znajdujących się w tablicy treningowej, pozwala to wyznaczać reguły prostsze, a zatem łatwiejsze w interpretacji, oraz skraca czas wyznaczania reguł.

W szczególnych przypadkach proces dyskretyzacji i redukcji informacji może zostać pominięty.

Efektywne zastosowanie metod *KDD* do dużych i różnorodnych baz danych napotyka jednak trudności, związane ze złożonością obliczeniową tych metod oraz z wyborem konkretnej metody *KDD*, która będzie adekwatna do konkretnego typu danych.

Możliwości zastosowania technologii Microsoft .Net do rozwiązania tej niedogodności opisane zostaną w dalszych rozdziałach.

2. Technologia Microsoft .Net – wprowadzenie

Microsoft .Net (czytaj „dot net") jest prefabrykowaną infrastrukturą do rozwiązywania typowych problemów występujących w aplikacjach internetowych. Jest to środowisko wykonywania, które działa w systemie operacyjnym Windows 2000. Jednym z podstawowych elementów infrastruktury .Net jest .Net Framework, czyli środowisko wykonywania, dzięki któremu można łatwiej i szybciej pisać aplikacje – łatwiejsze w zarządzaniu, wdrażaniu i późniejszym modyfikowaniu. W środowisku tym będą się wykonywały wszystkie programy i komponenty. Oferuje ono programistom takie nowe możliwości, jak automatyczne zarządzanie pamięcią, łatwiejszy dostęp do usług systemowych czy też łatwy dostęp do Internetu i baz danych. Zapewnia też nowy mechanizm wielokrotnego wykorzystania kodu, łatwiejszy i bardziej elastyczny niż COM. Środowisko .Net Framework ma zapewnić łatwiejsze wdrażanie, ponieważ nie wymaga posługiwania się ustawieniami Rejestru systemu Windows. Bardzo ważnym aspektem dla programistów jest fakt, iż wszystkie funkcje i możliwości są dostępne z dowolnego języka programowania zgodnego z .Net.

W obecnym czasie jesteśmy świadkami trendu, w którym coraz więcej aplikacji jako interfejsu używa przeglądarki WWW. Tutaj także .Net upatruje swoje miejsce i udostępniając ASP.Net (następna wersja technologii ASP [1]), środowisko, które działa w serwerze IIS (Internet Information Services) i ułatwia tworzenie dynamicznych stron WWW. ASP.Net oferuje nowy, niezależny od języka programowania, sposób pisania kodu i wiązania go z żądaniami kierowanymi z przeglądarek do serwerów i stron WWW. Środowisko to udostępnia także sprawne mechanizmy zarządzania stanem sesji i funkcje bezpieczeństwa, jest bardziej funkcjonalne i zawiera więcej rozszerzeń wydajnościowych w porównaniu z oryginalnym środowiskiem ASP.

Coraz większą popularność zyskują w oprogramowaniu tworzonym dla sieci Internet dedykowane oprogramowanie aplikacji klienta. Aplikacje takie w sposób lepiej zaprojektowany i przygotowany niż klasyczna przeglądarka będą umożliwiały przesyłanie przez sieć wywołań funkcji i odbieranie danych. Microsoft .Net oferuje nowy zbiór usług, za pomocą którego serwer może udostępniać swoje funkcje dowolnemu klientowi, działającemu na dowolnym komputerze z zainstalowanym dowolnym systemem operacyjnym. Oprogramowanie klienta będzie wywoływało funkcje udostępniane przez serwer, korzystając z najmniejszej wspólnej części wszystkich systemów przeznaczonych dla sieci Interent w chwili obecnej, to jest języka XML i protokołu HTTP. Zbiór udostępnianych w ten sposób funkcji nosi nazwę .Net Web Service.

Uzupełniając zbiór narzędzi przekazanych programistom .Net nie zapomina także o interfejsie użytkownika, udostępniając pakiet .Net Windows Forms, który ułatwia pisanie dedykowanych aplikacji klienta dla Windows przy użyciu środowiska .Net Framework.

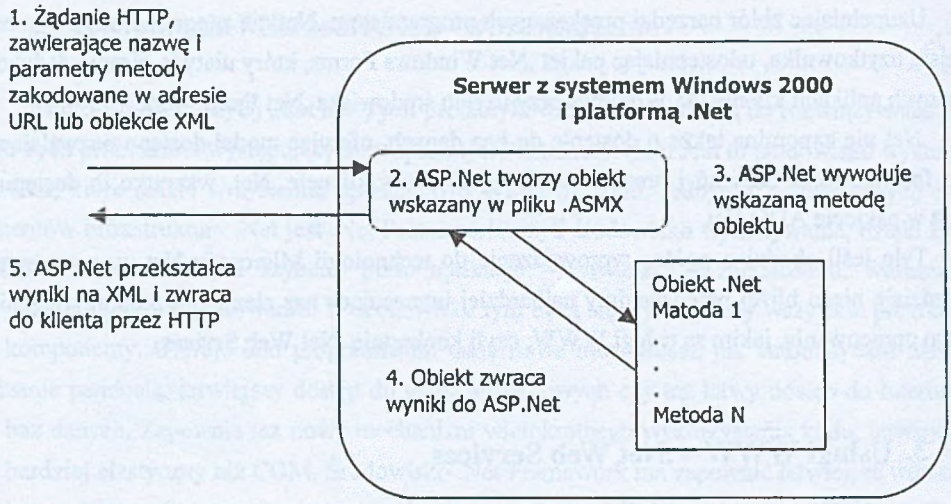
.Net nie zapomina także o dostępie do baz danych, oferując model dostępu pozwalający na formułowanie zapytań i prezentowanie wyników w kliencie .Net, wszystko to dostępne jest w pakiecie ADO.Net.

Tyle jeśli chodzi o ogólne wprowadzenie do technologii Microsoft .Net, w następnym rozdziale nieco bliżej przedstawimy najbardziej interesujący nas element z punktu widzenia tego opracowania, jakim są usługi WWW, czyli konkretnie .Net Web Srvices.

3. Usługi WWW – .Net Web Services

Sieć Internet jest obecnie pełna heterogenicznych bytów i bardzo trudno jest tutaj zaproponować rozwiązanie lub technologię pozwalającą na łatwe tworzenie aplikacji w takim środowisku, piszemy wyraźnie „w takim środowisku" a nie „dla takiego środowiska", gdyż chodzi nam o rozwiązanie pozwalające na budowanie aplikacji z elementów dostępnych w różnych miejscach sieci. Najbardziej rozpowszechnionym nośnikiem w sieci Internet jest protokół HTTP, który dziś jest używany przez praktycznie wszystkie przeglądarki do pobierania wyświetlanych w nich stron. Jako najbardziej uniwersalny język do kodowania przesyłanych informacji należy przyjąć HTML, a raczej jego nowszą i coraz bardziej popularną wersję, jaką jest XML.

Microsoft do realizacji takiej idei proponuje usługi WWW (.Net Web Services), usługi te to sposób przyjmowania przez obiekty na serwerze napływających żądań ze strony aplikacji klientów, posługując się najmniejszym wspólnym mianownikiem sieci Internet, to jest HTTP/XML. Utworzenie Web Service'u nie wymaga uczenia się nowego sposobu programowania, wystarczy utworzonemu obiektowi .Net ustawić atrybut mówiący, iż będzie on udostępniany klientom poprzez sieć Internet, a resztę przejmie na siebie ASP.Net. Środowisko to automatycznie podłączy prefabrykowaną infrastrukturę akceptującą żądania HTTP i odwzorowującą ją na wywołania zdefiniowanego obiektu. Tak utworzony i obudowany obiekt będzie mógł współdziałać z każdym innym elementem sieci, który posługuje się standardami HTTP i XML, czyli uniezależni to nasz obiekt od systemu operacyjnego i środowiska wykonania. Widok usług Web Srvices środowiska .Net po stronie serwera ilustruje rysunek 2.



Rys. 2. Widok usług Web Srvices środowiska .Net po stronie serwera
Fig. 2. Web Services .Net Environment from Server's point of view

Po stronie klienta .Net zapewnia klasy pośredniczące, które umożliwiają prosty, oparty na funkcjach dostęp do usług Web Services udostępnianych przez dowolny serwer akceptujący żądania HTTP. Narzędzie programistyczne odczytuje opis usługi Web Service i generuje klasę pośredniczącą zawierającą funkcje w dowolnym języku, jaki jest używany do pisania klienta. Kiedy oprogramowanie klienta wywołuje jedną z tych funkcji, klasa pośrednicząca tworzy żądanie HTTP i wysyła je do serwera. Gdy napłynie odpowiedź z serwera, klasa ta analizuje wyniki i zwraca je z funkcji. Pozwala to na realizację klientowi na interakcję z serwerem jedynie w oparciu o HTTP i XML.

Programistom piszącym aplikacje dla klientów, które to aplikacje będą korzystać z usług Web Services, należy udostępnić opis możliwości publikowanej usługi oraz metod korzystania z niej. Programista będzie potrzebował listy metod udostępnianej usługi, opisu wymaganych parametrów, protokołów i zwracanych rezultatów. Infrastruktura ASP.Net generuje taki opis na podstawie analizy metadanych zawartych w kodzie będącym implementacją usługi. Opis jest przechowywany w pliku o nazwie WSDL (Web Service Descriptor Language). Plik ten dla danej usługi jest czasem zwany „kontraktem", gdyż zawiera listę rzeczy, które usługa potrafi wykonać i informuje, jak z nich korzystać.

Dostęp do usług Web Services poprzez moduł odbiorczy ASP.Net jest możliwy na kilka sposobów przekazywania żądań: HTTP GET, HTTP POST i SOAP. Pierwsze dwa sposoby zapewniają głównie kompatybilność wsteczną, ponieważ cały kod dzisiejszych aplikacji WWW używa właśnie jednej z tych dwóch metod. W nowych projektach programistycznych

może się okazać, że łatwiej jest zastosować metodę SOAP, wykorzystując przygotowaną obsługę tego protokołu z poziomu technologii .Net.

Właśnie zagadnienie Web Services będzie dla nas szczególnie interesujące pod kątem usług obliczeniowych dyskutowanych w niniejszym opracowaniu.

4. Data mining a Microsoft .Net

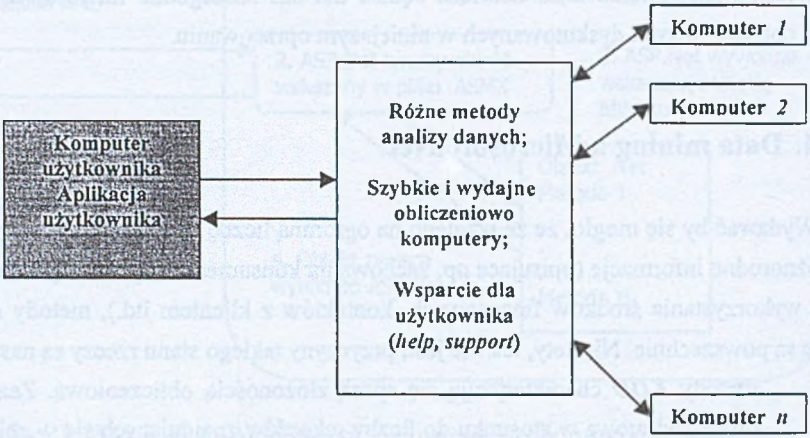
Wydawać by się mogło, że ze względu na ogromną liczbę baz danych, w których zawarte są różnorodne informacje (opisujące np. zachowania konsumentów, przebieg leczenia pacjentów, wykorzystania środków finansowych, kontaktów z klientem itd.), metody *KDD* stosowane są powszechnie. Niestety, tak nie jest, przyczyny takiego stanu rzeczy są następujące:

- metody *KDD* charakteryzują się sporą złożonością obliczeniową. Zazwyczaj jest ona kwadratowa w stosunku do liczby rekordów znajdujących się w zbiorze treningowym. Wynika stąd, że przeprowadzenie analizy danych złożonych z kilkudziesięciu tysięcy rekordów wymaga dostępu do komputerów o stosunkowo dużej mocy obliczeniowej, lub rozproszenia obliczeń pomiędzy kilka lub kilkanaście komputerów;
- zazwyczaj przed realizacją procesu *KDD* wymagane jest od użytkownika podanie pewnych parametrów pozwalających dostroić proces *KDD* do specyfiki analizowanych danych. Przykładowo, parametrami takimi mogą być: wybór metody dyskretyzacji, wybór metody wyznaczania reguł, wybór metody klasyfikacji itd.;
- nie do każdego typu danych treningowych można stosować każdą z metod *KDD*. Dokładniej, nie zawsze wybierając sposób reprezentacji szukanych wzorców (np. w postaci reguł) można proces *KDD* tak dostroić, aby wyniki analizy były satysfakcjonujące dla użytkownika. Nie znaczy to, że stosując inne metody analizy (np. analizę statystyczną) nie można by otrzymać interesujących użytkownika wyników. Powyższe rozumowanie potwierdza fakt, iż do tej pory nie podano metodologii, która wskazywałaby, jakie warunki musi spełniać zbiór danych treningowych, aby analizować go konkretną metodą.

Z powyższych względów idealna byłaby sytuacja, w której użytkownik mógłby przekazywać swoje dane treningowe i testowe do swego rodzaju centrum analizy, które zapewniałoby:

- możliwości różnorodnych analiz przesłanego zbioru danych,
- szybkie serwery zapewniające możliwość analizy dużych zbiorów danych,
- w miarę potrzeby rozpraszanie obliczeń, korzystając z innych dostępnych w danej chwili współpracujących komputerów,

- szeroko pojęte wsparcie dla użytkownika, obejmujące między innymi objaśnienia specyfiki każdej z oferowanej metody analizy danych.



Rys. 3. Współpraca użytkownika z „centrum analiz"
Fig. 3. User co-operation with "analyses centre"

Gdyby wyznaczone wzorce miały postać reguł, użytkownik otrzymywałby: zbiór wyznaczonych reguł, łącznie z oceną siły każdej reguły oraz dokładność klasyfikacji uzyskaną przez wyznaczone reguły na zbiorze testowym.

Uwzględniając powyższe postulaty, wydaje się być interesująca próba zastosowania technologii .Net, a konkretnie usług .Net Web Services do implementacji zagadnienia udostępniania metod *KDD*. Aspekty, które tutaj można wyróżnić jako możliwe do wykorzystania i dające bezpośrednie korzyści to:

- udostępnienie *KDD* jako usługi w sieci Internet pozwalające, okazjonalnym klientom na skorzystanie z metod bez konieczności implementowania algorytmów lub oprogramowania w swoich systemach,
- umiejscowienie i konfigurację procesów *KDD* na komputerze o większej mocy obliczeniowej lub też skonstruowanie aplikacji rozpraszającej obliczenia celem zwiększenia mocy obliczeniowej systemu, jednak realizowane jest to przez „usługę" i nie angażuje bezpośrednio zlecającego obliczenia,
- udostępnienie z wykorzystaniem komponentów .Net dostępu do usługi tak z poziomu przeglądarki WWW, jak i możliwość wykorzystania tej usługi w oprogramowaniu klienta w środowisku .Net Framework, gdzie klient w swoim oprogramowaniu wykorzystuje gotowy działający mechanizm obliczeniowy *KDD* samemu tworząc i rozwijając rozwiązanie swojego problemu budując pewnego rodzaju hybrydę.

Także z punktu widzenia autorów i twórców takiego rozwiązania technologia .Net pozwala na wykorzystanie w tworzonej usłudze już gotowych fragmentów kodu w postaci modułów wykonywalnych, ich połączenie z innymi elementami tworzonymi w innych językach programowania. Przyspiesza to zdecydowanie proces tworzenia aplikacji, gdyż unikamy ponownego pisania, kompilowania i testowania już sprawdzonego i poprawnie działającego kodu.

Ewentualne rozpraszanie obliczeń też może zostać oparte na sieci komputerów z zainstalowanymi na nich takimi samymi usługami obliczeniowymi *KDD*, wykorzystując właściwość procesu obliczeniowego, polegającą na tym, że nie trzeba rozpraszać danej procedury, tylko konieczne jest w *KDD* wykonanie tej samej procedury na prawie identycznym zbiorze danych wielokrotnie, a to mogą robić komputery niezależnie i nie muszą posiadać dużej mocy obliczeniowej. Takim procesem może sterować główna aplikacja usługi odbierająca od użytkownika żądania i rozdzielająca je na zarejestrowane w systemie maszyny, oraz odbierająca wyniki i przekazująca je do użytkownika.

5. Planowany eksperyment

Celem eksperymentu będzie zaadaptowanie na potrzeby Usługi WWW realizowanej jako .Net Web Service pewnej niewielkiej części biblioteki *TRS* (ang. *Tolerance Rough Sets*), która powstała w Zakładzie Teorii i Projektowania Systemów Komputerowych Instytutu Informatyki Pol. Śl. i jest efektem prac nad możliwością zastosowania zbiorów przybliżonych wyznaczanych przez relację tolerancji do celów *KDD*.

Przyjęto, że plik ze zbiorem treningowym jest plikiem tekstowym o następującym formacie

Cols: *Liczba_atrybutów*
Rows: *Liczba_Obiektów*
DecAttr: *Numer_atrybutu_decyzyjnego*

Typ_atr_1	*Typ_atr_2*	...	*Typ_atr_n*
Nazwa_atr_1	*Nazwa_atr_2*	...	*Nazwa_atr_n.*

W kolejnych wierszach umieszczone są dane poddawane analizie. W obecnej postaci biblioteki dopuszczalne są dwa typy atrybutów: *numeric* - oznacza atrybut o wartościach rzeczywistych, *string* - oznacza atrybut o wartościach symbolicznych. Przyjęto, że plik testowy również jest plikiem tekstowym, w pliku tym nie trzeba określać liczby obiektów i atrybutów,

nie trzeba także określać typów atrybutów. Wymagane jest, aby poszczególne kolumny danych testowych odpowiadały kolumnom znajdującym się w pliku treningowym.

Przyjęty format opisu plików treningowych i testowych jest podobny do formatów wykorzystywanych przez inne programy wspomagające proces *KDD* (np. Rosetta [7], AQ18 [2]).

Po analizie danych zwracany jest zbiór wyznaczonych reguł. Reguły mogą zostać zapisane w pliku tekstowym. Pojedyncza reguła decyzyjna zapisywana jest w postaci:

Nazwa_atr in ( ) and

:

Nazwa_atr in ( ) and

=> class is Nazwa_klasy.

W nawiasach po słowie *in* umieszczany jest zakres wartości deskryptora. Jeśli atrybut jest typu symbolicznego, zakres wartości wyliczany jest wprost, np. *Nazwa_atr in (zielony, czerwony)*. Jeśli atrybut jest typu rzeczywistego, zakres wartości reprezentowany jest jako przedział, np. *Nazwa_atr in (13.75..53.33)*. Reguły oddzielone są od siebie pojedynczą pustą linią.

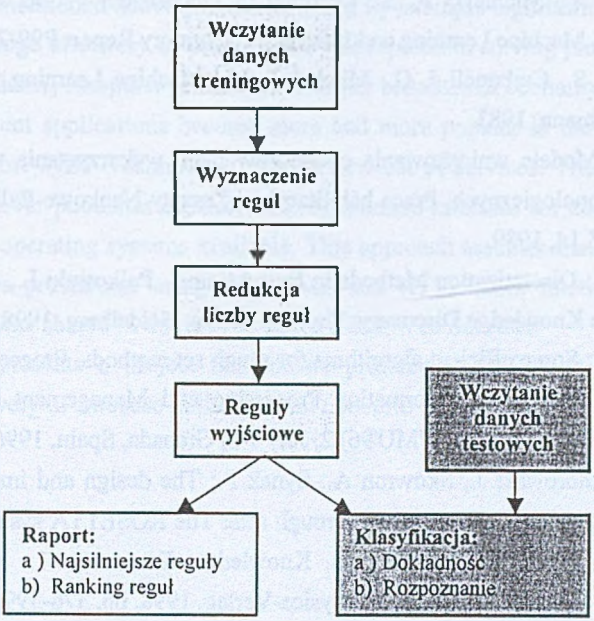
Do przekazywania danych do i z usług obliczeniowych oraz do transferu danych do i z programów wspomagających proces *KDD* zostanie wykorzystany język XML, jako aktualnie zyskujący coraz większą popularność standard wymiany danych. Przyjęcie takiego rozwiązania pozwoli na zunifikowanie przesyłanych danych oraz pozwoli na ich konwersję pomiędzy różnymi schematami, jak również umożliwi ich wykorzystanie w sposób bezpośredni, lub też łatwą modyfikację sposobu ich prezentowania za pomocą różnych narzędzi. Ten sam zbiór uzyskanych wyników może być prezentowany za pomocą przeglądarki WWW lub też za pomocą innej dedykowanej aplikacji prezentującej dane.

Przyjęto, że realizowany będzie następujący schemat analizy zbioru treningowego przedstawiony na rysunku 4.

Poza przygotowaniem zbiorów treningowego i testowego użytkownik musi dokonać wyboru jednej z dwóch metod generowania reguł. Reguły można wyznaczyć za pomocą algorytmu wyznaczającego jeden najkrótszy redukt [6] oraz algorytmu *RMatrix* [9].

Możliwe jest zażądanie wskazania najlepszych z wyznaczonych reguł oraz przeprowadzanie filtracji zbioru reguł. Przyjęto, że do wyboru są trzy kryteria oceniające poszczególne reguły [10]:

- accuracy (dokładność reguły),
- coverage (pokrycie reguły),
- Irep (funkcja łącząca ocenę dokładności i pokrycia reguły).



Rys. 4. Schemat analizy zbioru treningowego
Fig. 4. Schema of training file analyzes

Do filtracji reguł wykorzystano najprostszy algorytm pozwalający usunąć reguły o wartości oceny poniżej zadanego progu. Wartości progów przekazywane są przez użytkownika.

Klasyfikacja odbywa się przez reguły pasujące i odbywa się zgodnie ze sposobem opisanym m.in. w [9].

Efektem analizy jest:

- wyznaczony zbiór reguł posortowany malejąco względem wartości wybranej (jednej z trzech wspomnianych powyżej funkcji oceniających reguły),
- wartość dokładności klasyfikacji, jaką osiągnął wyznaczony zbiór reguł na zbiorze testowym,
- liczba obiektów testowych nierozpoznana przez żadną z wyznaczonych reguł.

LITERATURA

1. Cieśla S.: Wykorzystanie technologii Active Server Pages w dostępie do baz danych w sieciach Internet i Intranet, Zeszyty Naukowe Politechniki Śląskiej Informatyka Z.34, 1998.

2. Kaufman K. A., Michalski R. S.: Learning in Inconsistent World, Rule Selection in STAR/AQ18, Machine Learning and Inference Laboratory Report P99-2, 1999.
3. Michalski R. S., Carbonell J. G., Mitchel T. M.: Machine Learning vol. I. Los Altos: Morgan Kaufmann, 1983.
4. Mrózek A.: Modele wnioskowania operatorów i ich wykorzystanie w komputeryzacji obiektów technologicznych. Praca habilitacyjna. Zeszyty Naukowe Politechniki Śląskiej. Informatyka Z.14, 1989.
5. Nguyen S. H.: Discretization Methods in Data Mining. Polkowski L., A. Skowron (ed.): Rough Sets in Knowledge Discovery. Physica-Verlag, Heidelberg, 1998, pp. 451-482.
6. Nguyen S. H.: Some efficient algorithms for rough set methods. Proceedings of the Sixth International Conference, Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96) 2, July 1-5, Granada, Spain, 1996, pp. 1451-1456.
7. Ohrn A., Komorowski J., Skowron A., Synak P.: The design and implementation of a knowledge discovery toolkit bąsed on rough sets: The ROSETTA system. Polkowski L., Skowron A. (ed.): Rough Sets in Knowledge Discovery 1: Methodology and Applications. Germany, Heidelberg: Physica-Verlag, 1998, pp. 376-399.
8. Pawlak Z.: Rough sets: Theoretical aspects of reasoning about data. Dordrecht: Kluwer, 1991.
9. Sikora M., Proksa P.: Algorithms for generation and filtration of approximate decision rules, using rule-related quality measures. Bulletin of International Rough Set Society Vo. 5, No. 1/2 (Proceedings of the International Workshop on Rough Set Theory and Granular Computing), Matsue, Japan, 2001.
10. Sikora M.: Filtracja zbioru reguł decyzyjnych wykorzystująca funkcje oceny jakości reguł. Studia Informatica Vol. 22, No. 3, Gliwice 2001.

Recenzent: Dr inż. Krzysztof Nałęcki

Wpłynęło do Redakcji 10 kwietnia 2002 r.

Abstract

Knowledge Discovery in Databases (KDD) as a part of data mining is one of the most intensive developing branch of computer science. Finding true, unknown so far, understandable and useful data patterns define KDD process.

To calculate mentioned above patterns learning by example algorithms are used. Systems including knowledge discovery in database are wide spread in solving problems appearing in the industry, medicine, computer science and another branches of economy.

Dedicated client applications become more and more popular in the Internet developed software. Microsoft .NET Technology offers a new set of services. These services may be used when the server publishes application programmers interface for end users irrespective of hardware and operating systems available. This approach assumes that the client software is calling server's procedures using only XML and HTTP (core Internet standards). The package of functions shared in this way is called .NET Web Services.

This article presents a project that allows producing .NET Web Services to render knowledge discovery in database in the Internet network.