Nová GUHA-procedura ETree-Miner v systému LISp-Miner

Milan Šimůnek
Laboratoř pro inteligentní systémy Praha
Fakulta informatiky a statistiky, VŠE Praha
nám. W. Churchilla 4, 130 67 Praha 3
simunek@vse.cz

Abstrakt: *Tento článek představuje novou GUHA-proceduru ETree-Miner, která využívá nový koncept tzv. exploračních stromů a která byla právě naimplementována do akademického systému LISp-Miner vyvíjeného na Fakultě informatiky VŠE Praha. V článku je vysvětlen rozdíl v pojetí exploračních stromů oproti klasickým rozhodovacím stromům, stručně zmíněna metoda GUHA a systém LISp-Miner. Dále jsou popsány vybrané detaily implementace ETree-Mineru a konečně poukázáno na jeho využití v dalších projektech. Pro komunikaci se systémy třetích stran je používán standardizovaný výměnný formát PMML.*

Klíčová slova: GUHA, LISp-Miner, exploratční stromy, rozhodovací stromy, ETree-Miner, PMML

Abstract: *This paper presents a new GUHA-procedure called ETree-Miner which utilizes a new concept of so called exploration trees and which was recently implemented into LISp-Miner, an academic system for KDD developed at Faculty of Informatics, University of Economics Prague. Main differences between exploration trees and classical decision trees are explained. A brief summary of GUHA-method and LISp-Miner is included. Selected details of implementation ETree-Miner are described and finally, relations to other projects are mentioned. A de-facto standard PMML for description of data-mining models is used for communications with external systems.*

Keywords: GUHA, LISp-Miner, exploration trees, decision trees, ETree-Miner, PMML

1. Úvod

Rozhodovací stromy patří mezi nejčastěji používané techniky při dobývání znalostí z databází (DZD). Metoda GUHA pak mezi nejstarší známé metody DZD s bohatou teorií a schopností nabídnout uživateli všechny v datech platné vztahy odpovídající zadání. Tento příspěvek má za cíl informovat o konceptu tzv. *exploračních stromů* a o nově implementované GUHA-proceduře *LM ETree-Miner*, která tyto exploratční stromy generuje a nabízí i prostředky pro jejich interpretaci.

Článek je strukturován takto: v následující kapitole je vysvětlen koncept *exploračních stromů* a vztažen k obecně používaným a známým *rozhodovacím stromům*. Ve třetí kapitole je stručně představen systém LISp-Miner a metoda GUHA. Čtvrtá kapitola se pak týká přímo nové GUHA-procedury *ETree-Miner* – zmíněny jsou implementační detaily, přínosy implementace i řešené obtíže. Na závěr jsou naznačeny možnosti dalšího rozvoje procedury i s ohledem na souběžně řešené projekty.

2. Explorační versus Rozhodovací stromy

2.1 Rozhodovací stromy

Rozhodovací stromy se nejčastěji používají pro klasifikaci objektů/záznamů podle zvolené cílové třídy. Jsou obvykle vytvářeny od kořenového uzlu a jako větvící je z možných atributů zvolen ten, který na této úrovni a podle zvolené metriky nejlépe rozděluje podmnožinu záznamů s ohledem na cílovou třídu. Důležité je, že v daném uzlu může být jako větvící atribut zvolen vždy pouze jeden.

Rozhodovací stromy patří mezi zavedené techniky dobývání znalostí a propracované jsou i postupy jejich vytváření. Mezi nejčastěji používané algoritmy patří *ID3* resp. *C4.5* – viz [Quinlan, 1986] resp. [Quinlan, 1993] nebo *CART* – viz [Breiman a kol., 1984].

2.2 Explorační stromy

Explorační stromy byly navrženy v práci [Berka, 2011] a opět by měly sloužit především pro klasifikaci objektů a také pro jejich deskripci. Motivací pro jejich zavedení je odstranění nevýhod plynoucích u *rozhodovacích stromů* z výběru vždy pouze jednoho větvícího atributu. Omezení volby pouze na jeden větvící atribut může totiž vést k sub-optimálnímu řešení a tedy horší kvalitě klasifikace.

Explorační strom je vytvářen tak, že v místě větvení jsou všechny atributy potenciálně použitelné pro větvení seřazeny sestupně podle klesající metriky popisující jejich vhodnost pro to stát se větvícím atributem právě na tomto místě. Na základě parametrů zadání je pak pro větvení vybráno prvních *q* atributů ze seznamu (případně z nich pouze ty, které splňují zadaný minimální práh pro hodnotu metriky). Pro výpočet metriky vhodnosti větvení je prozatím použita charakteristika χ^2 daná vztahem:

$$\chi = \sum_i \sum_j \frac{a_{ij} - \frac{r_i s_j}{n}}{\frac{r_i s_j}{n}},$$

kde a_{ij} je četnost záznamů, které obsahují pro větvící atribut A *i*-tou hodnotu z možných hodnot atributu A a zároveň obsahují *j*-tou hodnotu cílového atribut (třídy); r_i je počet záznamů v analyzovaných datech, které obsahují *i*-tou hodnotu pro atribut A; s_j je počet záznamů v analyzovaných datech, které obsahují *j*-tou hodnotu pro cílového atributu; a konečně *n* je celkový počet záznamů v analyzovaných datech. Hodnota použité metriky jednak řadí atributy podle jejich vhodnosti pro větvení, ale zároveň vyjadřuje i absolutní hodnotu síly vztahu mezi větvícím a cílovým atributem (více viz [Berka, 2011]). Do budoucna se zároveň uvažuje o implementaci více možných způsobů výpočtu metriky, ze kterých si uživatel bude moci volit (viz dále).

Výběrem více než jednoho atributu větvení v daném uzlu stromu vzniká variantní větvení (pro každý z vybraných větvících atributů jedna varianta). Každou z variant je od této chvíle nutné při generování sledovat samostatně. Pro každou z nich volíme v dalším kroku na nižší úrovni opět možné atributy pro větvení, čímž vzniká další variantní větvení. Postup se stále opakuje a to až do okamžiku, kdy nejsou k dispozici žádné další atributy pro větvení nebo pro ně vypočtená metrika nedosahuje zadané prahové hodnoty.

Tímto způsobem postupně vznikne rozsáhlý *exploračí strom*, který v každém z uzlů obsahuje potenciálně několik možných variant větvení. Na obrázku 1 je ukázka části *exploračního stromu* pro klasifikaci klientů z hlediska poskytnutí úvěru. Na první úrovni (*L1*) je jako nejvhodnější atribut pro větvení (*L1.S1*) zvolen `Příjem`. Možné kategorie jsou `nízký`, `průměrný` a `vysoký`, a tak je první větev (*L1.S1.B1*) pro klienty s nízkým příjmem. Mimo obrazovku jsou pak i větve pro `průměrný` (*L1.S1.B2*) a `vysoký` (*L1.S1.B3*) příjem. Pro klienty s nízkým příjmem byla na druhé úrovni jako nejvhodnější větvící atribut zvolena výše konta (*L1.S1.B1.L2.S1*). Na příkladu klientů se střední výší konta (*L1.S1.B1.L2.S1.B2*) si pak můžeme všimnout variantního větvení, kdy jako úplně nejvhodnější byl pro větvení na třetí úrovni zvolen atribut `Nezaměstnaný` (*L1.S1.B1.L2.S1.B2.L3.S1*), ale alternativně je možné větvit i podle `Pohlaví` (*L1.S1.B1.L2.S1.B2.L3.S2*). Pro obě varianty jsou pak zpracovávány další větve, až do vyčerpání vhodných větvících atributů.

Obr. 1 – Explorační strom a variantní větvení v uzlu L1.S1.B1.L2.S1.B2 Konto = střední.

Kromě maximálního počtu vybraných variant větvení a prahové hodnoty pro metriku je v zadání možné omezit i maximální hloubku stromu a minimální četnost větve (po poklesu četnosti pod daný práh, není větev dále rozvíjena). V případě, že bychom maximální počet variant větvení omezili na jedna, tak bychom získali právě jeden *rozhodovací strom*, stejně jako v algoritmu *ID3*.

Z vygenerovaného *exploračního stromu* lze systematickým procházením všech možných variant odvodit potenciálně velké množství *rozhodovacích stromů* a vzniká tak „les“ *rozhodovacích stromů*. Pomocí každého z *rozhodovacích stromů* můžeme provést klasifikaci a výsledné doporučení pak složit z dílčích klasifikací – buď vážením (*weighted voting*), kdy váha každého stromu bude odpovídat jeho kvalitě (vypočtené na trénovacích nebo testovacích datech), nebo volbou nejčastěji doporučované klasifikace (*majority-voting*).

Výsledky v práci [Berka, 2011] ukazují, že klasifikace na základě celého *lesu stromů* jsou obvykle lepší než klasifikace na základě pouze jediného *rozhodovacího stromu* vzniklého výběrem vždy nejlepšího možného atributu. Tím, že v *exploračním stromu* jsou povoleny varianty větvení, mohou se na klasifikaci podílet i na dané úrovni méně úspěšné atributy, které však mohou umožnit expresi potenciálně důležité informaci na nižší úrovni.

3. Systém LISp-Miner a metoda GUHA

Systém LISp-Miner [LISp-Miner] je akademický systém pro DZD a je vyvíjen na Fakultě informatiky a statistiky VŠE Praha od přelomu let 1995/96 (více viz např. [Šimůnek, 2003]). Od počátku vývoje byl systém stavěn na metodě GUHA a na teoretických pracích, které se jí týkají.

3.1 Metoda GUHA

Metoda GUHA je původní českou metodou exploratční analýzy. Její počátky sahají do 60. let dvacátého století. Za základní publikaci o metodě GUHA lze považovat práci [Hájek & Havránek, 1978], ale v průběhu let byla publikována celá řada teoretických prací, z nichž zmíníme alespoň práce v oblasti observačního kalkulu a logiky asociačních pravidel (viz např. [Rauch, 2009]).

Cílem metody GUHA je poskytnout vše zajímavé k danému problému, tj. všechny zajímavé vztahy vyplývající z analyzovaných dat. Metoda GUHA je realizována tzv. GUHA-procedurami. GUHA-procedura je procedurou, na jejímž vstupu jsou analyzovaná data a (jednoduchá) definice velmi rozsáhlé množiny potenciálně zajímavých vztahů. Výstupem pak jsou všechny vztahy, které opravdu v datech platí. Zároveň jsou ve výstupu uváděny pouze takové vztahy, které nejsou logicky odvoditelné z některého jiného, na výstupu existujícího (jednoduššího) vztahu. Aktuální shrnutí historie vývoje metody GUHA je v práci [Hájek a kol., 2010].

Nejznámější a nejčastěji implementovanou GUHA-procedurou je ASSOC, která byla poprvé popsána ve zmíněné monografii [Hájek & Havránek, 1978]. Hledanými vztahy jsou rozšířená asociační pravidla, pracující nejen se *spolehlivostí* a *podporou* (později použité v konceptu *nákupních košíků* [Agrawal a kol., 1993]), ale i se vztahy založenými na statistických testech hypotéz. Oproti metodě *nákupních košíků* je pracováno s výrazně bohatší syntaxí asociačních pravidel, která tak v současné implementaci systému LISp-Miner (spolu s výsledky dalších teoretických prací) umožňují *více-četné koeficienty*, *podmíněná asociační pravidla*, *konjunkce/disjunkce/negace literálů*, *dílčí cedenty* nebo *třídy ekvivalence* (více viz např. [Rauch & Šimůnek, 2005]).

3.2 Systém LISp-Miner

Systém LISp-Miner se v současné době skládá z osmi GUHA-procedur (*4ft-Miner*, *CF-Miner*, *KL-Miner*, *SD4ft-Miner*, *SDCF-Miner*, *SDKL-Miner*, *Ac4ft-Miner* a zde popisovaný *ETree-Miner* jako nejnovější implementovaná procedura) a dalších modulů – zejména pro předzpracování dat (*LM DataSource*) a komunikaci se systémy třetích stran pomocí dokumentů ve formátu PMML, HTML či XML (*LM SwbImporter* a *LM SwbExporter*). Každá z implementovaných GUHA-procedur hledá v analyzovaných datech odlišný typ vztahů/vzorů (*patterns*), případně porovnává míru platnosti vztahů

mezi dvěma podmnožinami analyzovaných dat (procedury rodiny *SD*-). V případě *4ft-Mineru* se tak hledají *asociační pravidla* s rozšířenou syntaxí; v případě *KL-Mineru* pak *K×L polní tabulky četností* popisující vztah dvou kategoriálních atributů; a například v *Ac4ft-Mineru* dvojice *asociačních pravidel*, které vyjadřují *změnu stavu*, resp. *akci* od dosavadního stavu ke stavu žádoucímu (opět s bohatou syntaxí). Podrobný popis stavu systému k roku 2010 naleznou zájemci v [Šimůnek, 2010].

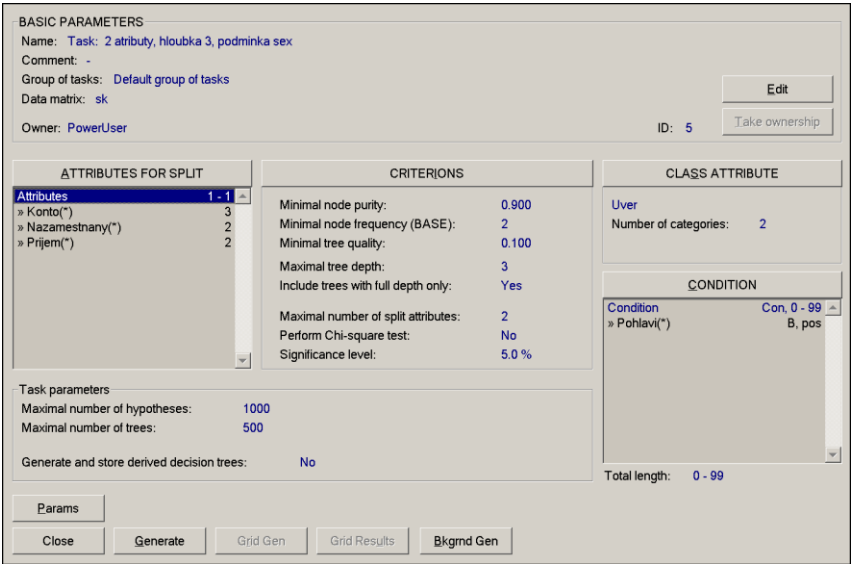
Systém je v prvé řadě používán pro výuku a pro výzkum v oblasti DZD. Je však využíván i jako nástroj DZD pro analýzu reálných dat a to jak ve výzkumných projektech (zejména v lékařských studiích), tak i v komerčních aplikacích (v oblasti bankovnictví, marketingu i CRM). V neposlední řadě je systém používán při práci na bakalářských, diplomových i disertačních pracích a to opět jak pro analýzu dat, tak i pro návrh nových postupů či technik v oblasti DZD.

Celý systém LISp-Miner, včetně zde popisované procedury *ETree-Miner* je volně ke stažení na adrese http://lispminer.vse.cz.

4. GUHA-procedura ETree-Miner

Nově implementovaná GUHA-procedura *ETree-Miner* rozšiřuje portfolio GUHA-procedur implementovaných v LISp-Mineru o *exploračních stromy*, resp. o „les“ *rozhodovacích stromů*, jak bylo vysvětleno výše.

Implementace celé procedury byla provedena v souladu s konvencemi použitými v ostatních procedurách a modulech systému LISp-Miner. Pro zadávání úloh a spouštění výpočtu slouží modul *ETree-Task*. Pro interpretaci výsledků a případné exporty pak modul *ETree-Result*.



Obr. 2 – Dialogové okno zadání ETree-Miner úlohy se seznamem atributů pro větvení (vlevo), cílovou třídou (vpravo), podmínky (vpravo dole) i parametry zadání (uprostřed)

K zadávání úlohy slouží dialogové okno na obrázku 2. V něm uživatel vybírá atributy, které se mohou použít pro variantní větvení na jednotlivých úrovních *exploračního stromu*; dále cílovou třídu, vůči které se počítá metrika vhodnosti atributu pro větvení; případně může prohledávaný stavový prostor omezit zadáním podmínky jako odvozeného *booleovského atributu* (více viz dále); a konečně zadává parametry ovlivňující hloubku stromu, maximální počet variant větvení atp.

Oproti návrhu v práci [Berka, 2011] byla provedena drobná změna ve způsobu zadání úlohy – původní parametr „*maximal node impurity*“ byl zaměněn za logicky opačný „*minimal node purity*“, tedy minimální požadovaná čistota uzlu, po jejímž dosažení se již dále strom nevětví a z uzlu se stává list. Důvodem bylo sjednocení s druhými dvěma prahovými hodnotami „*minimal node frequency*“ (minimální požadovaná frekvence uzlu) a „*minimal tree quality*“ (minimální požadovaná kvalita stromu).

#	Konto	Nazamestnany	Pohlavi	Prijem	Uver	Prediction
1	vysoké	ne	žena	vysoký	ano	ano
2	vysoké	ne	muž	vysoký	ano	ano
3	nízké	ne	muž	nízký	ne	ne
4	vysoké	ano	žena	nízký	ano	ano
5	vysoké	ano	muž	nízký	ano	ano
6	nízké	ano	žena	nízký	ne	ne
7	nízké	ne	muž	vysoký	ano	ano
8	nízké	ano	žena	vysoký	ano	ano
9	støední	ano	muž	nízký	ne	ne
10	støední	ne	žena	vysoký	ano	ano
11	støední	ano	žena	nízký	ne	ne
12	støední	ne	muž	nízký	ano	ano

Obr. 3 – Hromadná klasifikace pomocí váženého hlasování

Ve fázi interpretace výsledků implementované v modulu *ETree-Result* je pak i možnost hromadné klasifikace vstupních dat – a to jak váženým hlasováním (viz obrázek 3), tak i pro každý jednotlivý rozhodovací strom.

Oproti teoretickému návrhu v práci [Berka, 2011] je implementace v systému LISp-Miner dále rozšířena o následující:

- optimalizace generování *exploračních stromů* – bitové řetězce a operace nad nimi;
- oddělení fáze generování *exploračních stromů* od generování odvozených *rozhodovacích stromů*;
- možnost generování podmíněných *exploračních stromů*;
- provázání procedury *ETree-Miner* s ostatními moduly systému LISp-Miner.

4.1 Implementace generování exploračních stromů

Stromy jsou ze své podstaty rekurentní datové struktury a s tím přichází specifické obtíže při programování a zejména pak při ladění. Zároveň je třeba počítat se značným množstvím uzlů a větví, které při generování *exploračního stromu* (a následně *rozhodovacích stromů*) mohou vzniknout.

Při implementaci rekurentního algoritmu generování *exploračního stromu* byla použita technika pole, která je efektivnější oproti implementaci přes rekurentní volání funkce a zásobník. Na základě četnosti aktuálně zpracovávané větve jsou detekovány větve neperspektivní, které ani v nejlepším možném případě nemohou splnit zadáním požadovaný práh na čistotu uzlu, eventuálně kvalitu celého stromu. Zároveň byla redukována nutnost nové alokace dynamických proměnných a to důsledným *znovu-*

používáním struktur vytvořených původně pro větve, které se následně ukázaly jako neperspektivní.

Při implementaci byl kladen důraz na rychlost generování, a proto byl využit i koncept již dříve implementovaných a osvědčených *bitových řetězců* a optimalizovaných operací nad nimi. *Bitové řetězce* jsou jedním z hlavních důvodů vysoké rychlosti výpočtu úloh ve všech procedurách systému LISp-Miner. Umožňují velmi rychle spočítat četnosti záznamů splňujících jak jednoduché, tak i odvozené *booleovské atributy*. Tedy rychle ověřit, zda je daný vztah v analyzovaných datech platný (bez ohledu na to, jestli jde o *asociační pravidlo*, *K×L polní tabulku četností* nebo právě *exploračního/rozhodovací strom*). Založením implementace na *bitových řetězcích* je dosaženo lineární časové závislosti doby výpočtu úlohy na počtu řádků v analyzované matici dat. Podrobněji viz například [Rauch & Šimůnek, 2005]. V případě *ETree-Mineru* je pak doba nutná k vygenerování stromu zanedbatelná v porovnání s dobou nutnou pro uložení tohoto stromu do relační databáze.

Výpočet *ETree-Miner* úloh byl implementován jako dvoustupňový. Nejprve jsou generovány všechny *exploračních stromy*, které v analyzovaných datech platí v míře dané zadáním. Je-li *exploračního strom* úspěšně verifikován, je zařazen do výsledků. Pro každou *4ft-podmínku* je přitom generován jeden *exploračního strom*. Maximální počet *exploračních stromů* ve výsledcích úlohy je proto dán počtem možných variant *4ft-podmínky* podle zadání. Neobsahuje-li zadání podmínku, může být výsledkem buď právě jeden *exploračního strom* (byl-li nalezen takový odpovídající zadání), nebo žádný.

Teprve po vygenerování všech *exploračních stromů* je přistoupeno k tvorbě odvozeného „lesu“ *rozhodovacích stromů* pro každý *exploračního strom*. Pro jeden *exploračního strom* je obvykle vygenerováno velké množství z něj odvozených *rozhodovacích stromů* (viz výše). Protože se ukázalo, že ukládání takového množství záznamů do relační databáze trvá neúměrně dlouho, byla do zadání úlohy přidána uživatelská volba určující, zda mají být *rozhodovací stromu* skutečně ukládány, nebo zda mají být vytvářeny dynamicky až v okamžiku interpretace v modulu *ETree-Result*. Dynamické generování je výrazně rychlejší, protože zcela odpadá nutnost ukládat jakákoliv data do databáze a tedy i veškerá komunikace s databází přes rozhranní ODBC.

Uživatel však může ukládání všech *rozhodovacích stromů* podle potřeby zapnout – například v případě, že chce výsledky úlohy exportovat do dalších aplikací a kromě základního *exploračního stromu* potřebuje i všechny z něj odvozené *rozhodovací stromy*.

4.2 Podmíněné exploračního stromy

Součástí zadání úlohy pro *ETree-Miner* je i zadání podmínky v podobě *4ft-cedentu* – stejně, jako například v proceduře *4ft-Miner*. Podmínka je odvozený *booleovský atribut*, který je automaticky generován pomocí operací logické *konjunkce*, *disjunkce* a *negace* z jednoduchých *booleovských atributů*. Pro každý řádek analyzované matice dat nabývá podmínka logické hodnoty 1 nebo 0, podle toho, zda ji daný řádek splňuje, či nikoliv. Z řádků, které podmínku splňují, vznikne podmnožina datové matice, na které je teprve *exploračního strom* generován (více viz [Rauch & Šimůnek, 2005]).

Všechny atributy vystupující v podmínce jsou zafixovány na kategorie uvedené v podmínce a není je třeba dále uvažovat při výběrů atributů v místech větvení

generovaného stromu. To výrazným způsobem zmenšuje procházený stavový prostor, který je jinak kvůli více možným větvícím atributům na každé úrovni značně rozsáhlý.

Možnost definice podmínky výrazně pomohla tvůrcům aplikace *ARBuilder* v projektu SEWEBAR (viz dále). Nyní je tak možné postupně fixovat atributy a kategorie, které již uživatel při ručním návrhu asociačního pravidla použil. To umožňuje zadáním vhodné podmínky snížit komplexitu aktuálně řešené úlohy návrhu dalšího nejvhodnějšího atributu. Řešení je pak získáno v dostatečně krátkém čase, aby návrh pravidla mohl být interaktivní.

4.3 Vazba na ostatní moduly a na externí systémy

Výhodou začlenění procedury *ETree-Miner* do systému LISp-Miner je možnost využití výsledků práce s nástroji pro předzpracování dat. Do zadání úloh tak vstupují již vhodně kategorizovaná data, případně očištěná o chybějící a odlehlé hodnoty. Uvažovat lze i naopak o použití výsledků procedury *ETree-Miner* v některém z existujících modulů.

Zároveň jsou k dispozici i prostředky pro výměnu dat s jinými systémy a to ve formě PMML dokumentů (viz dále). Pomocí modulu *LM SwbExporter* lze výsledky úlohy exportovat zejména do projektu SEWEBAR (viz [SEWEBAR]), kde jsou zobrazeny ve formě *analytických zpráv* jako strukturovaný text, který je pro běžné uživatele srozumitelnější. Stejným způsobem lze exportovat zadání (i výsledky) úloh do jiným systémů DZD, které podporují standard PMML (více viz dále).

Pro druhý směr komunikace (směrem k systému LISp-Miner, resp. proceduře *ETree-Miner*) slouží modul *LM SwbImporter*. Ten dokáže načíst externě vytvořený PMML dokument a vytvořit zadání *ETree-Miner* úlohy, která může být následně spuštěna. Do systému LISp-Miner tak lze importovat úlohy vytvořené v jiných nástrojích DZD nebo i zadání strojově vygenerovaná.

Kombinaci modulů *LM SwbImporter* a *LM SwbExporter* používá i výše zmíněná externí aplikace *ARBuilder*. Ta připraví v každém kroku interaktivního návrhu asociačního pravidla zadání pro *ETree-Miner* ve formě PMML dokumentu. PMML dokument je předán modulu *LM SwbImporter*, který jej načte a vytvoří zadání v prostředí LISp-Mineru. Zavoláním modulu *LM TaskPooler* je úloha spuštěna. Po jejím skončení jsou nalezené výsledky vyexportovány modulem *LM SwbExporter* opět ve formě PMML. Z obdrženého PMML dokumentu aplikace *ARBuilder* zjistí, které atributy jsou nejvhodnější pro další rozšiřování pravidla a v sestupném pořadí je nabídne uživateli ve webovém uživatelském rozhraní.

4.4 Výměnný formát PMML

PMML (*Predictive Model Markup Language*, viz [PMML]) je široce používaný datový formát založený na XML pro definici a výměnu *data-miningových* a statistických modelů (viz také [Guazzelli a kol., 2010]). Pro potřeby projektu SEWEBAR a vzhledem ke značně bohaté syntaxi hledaných vztahů v GUHA-proceduře *4ft-Miner* bylo navrženo rozšíření formátu na tzv. GUHA AR PMML (*GUHA Association Rules PMML*) a to pomocí konstrukce *Extension* podporované přímo standardem PMML (více viz [Kliegr a kol., 2010]). Postupně pak byla připravena rozšíření PMML i pro další GUHA-procedury systému LISp-Miner.

Každá z implementovaných procedur systému LISp-Miner pracuje s jinými typy vztahů, které se snaží v analyzovaných datech hledat. Zároveň existuje více rozdílných požadavků na podobu výstupu z jedné procedury – např. podrobný (pro přípravu *analytické zprávy*) × zkrácený (optimalizovaný pro potřeby aplikace *ARBuilder*).

Aby bylo dosaženo maximální flexibility tvaru exportovaných PMML zpráv, byl navržen vlastní *šablonovací jazyk*. Pomocí něj je možné vytvořit libovolné množství šablon, které obsahují aktivní prvky na místech, kam se má vložit přímo hodnota z *metabáze* (uživatelské databáze) systému LISp-Miner nebo vypočtená hodnota. V modulu *LM SwbExporter* je implementována interpretace těchto šablon a výsledkem je PMML dokument v podobě odpovídající konkrétně použité šabloně.

Syntaxe *šablonovacího jazyka* nabízí větvení podle logické hodnoty, cykly pro procházení seznamů a i rekurzi pro procházení *exploračních stromů*. V aktivních prvcích je možné se odvolávat i na pomocné údaje – jako je číslo verze modulu, ze kterého byl export proveden; datum a čas exportu atp. *Šablonovací jazyk* dovoluje použití vložených souborů, takže ucelené části PMML dokumentů, které se používají ve více typech exportů (např. sekce *DataDictionary*) jsou definovány pouze v jedné šabloně, která je pak v ostatních vložena na správné místo.

```
    <AssociationRules>

      <?LM:Include:Task.FTCedentI.Include.Template.PMML?>

      <!-- Hypothesis -->

      <?LM:Loop:Task.Hypothesis?>
      <AssociationRule id="<?LM:MB:Task.Hypothesis.HypothesisID?>"
        <?LM:IFF:Task.Hypothesis.FTAntecedentBagI#Count:0:Not?>
        antecedent="DBA_Antecedent_<?LM:MB:Task.Hypothesis.HypothesisID?>"
        <?LM:EndIFF:Task.Hypothesis.FTAntecedentBagI#Count?>
        consequent="DBA_Succedent_<?LM:MB:Task.Hypothesis.HypothesisID?>"
        <?LM:IFF:Task.Hypothesis.FTConditionBagI#Count:0:Not?>
        condition="DBA_Condition_<?LM:MB:Task.Hypothesis.HypothesisID?>"
        <?LM:EndIFF:Task.Hypothesis.FTConditionBagI#Count?>
      >
        <Text><?LM:MB:Task.Hypothesis.Name?></Text>

//      <!-- InterestMeasures -->
        <?LM:Loop:Task.Hypothesis.InterestMeasure?>
        <IMValue imSettingRef="<?LM:MB:Task.Hypothesis.InterestMeasure.FTQuantifierID?>" name
        <?LM:EndLoop:Task.Hypothesis.InterestMeasure?>

//      <!-- Frequency table -->
        <FourFtTable a="<?LM:MB:Task.Hypothesis.FreqA?>" b="<?LM:MB:Task.Hypothesis.FreqB?>"
       </AssociationRule>

      <?LM:EndLoop:Task.Hypothesis?>
    </AssociationRules>
  </guha:AssociationModel>
</PMML>
```

Obr. 4 – Ukázka části šablony pro export výsledků úlohy do formátu PMML

Vlastními prostředky navržený *šablonovací jazyk* se osvědčil, protože bylo možné reagovat na všechny požadavky ohledně formátu exportovaného souboru. Za hlavní výhodu však považujeme jeho nezávislost ať již licenční, tak s ohledem na nároky na rozsah *instalačního image* počítačových učeben, na kterých je systém LISp-Miner používán při výuce.

5. Směry dalšího rozvoje

Při vytváření stromů je jako metrika pro řazení atributů podle jejich vhodnosti k větvení na daném místě použita charakteristika χ^2. Ta zaručuje výběr atributů s „nejlepším“ rozdělením kategorií podle cílové třídy. To je vhodné kritérium při vytváření celého stromu. V případě využití *ETree-Mineru* pro potřeby aplikace *ARBuilder* se však ukazuje, že by bylo vhodné vybírat, resp. řadit atributy i podle jiných metrik – nejlépe těch, které uživatel zvolí jako *míry zajímavosti* při interaktivním návrhu pravidla. V současné době jde o *Fundovanou implikaci* a *Nadprůměrné souvisení* (*Above-Average Dependency*). Aktuálně se proto pracuje na rozšíření zadání *ETree-Miner* úlohy, aby bylo možné volit z více nabízených metrik. O tuto volbu bude rozšířena i syntaxe importovaného PMML souboru, aby externí aplikace *ARBuilder* mohla v každém kroku vhodnou metriku zvolit.

Dalším možným směrem rozvoje je i vytvoření celého nového modulu v rámci projektu SEWEBAR, který by dovolil interaktivní analýzu dat pomocí exploračních stromů v prostředí webové aplikace.

Do budoucna se také počítá s rozšířením již připravené infrastruktury pro distribuovaný výpočet úloh na počítačovém gridu i na proceduru *ETree-Miner*. V současné době je distribuovaný výpočet implementován ve všech sedmi ostatních procedurách, a proto je implementace i pro *ETree-Miner* logickým krokem.

Použitá literatura

Agrawal, R.; Imielinski, T.; Swami, A., 1993. Mining associations between sets of items in massive databases. In *Proc. of the ACM-SGMOD 1993 Int Conference on Management of Data,* Wasghington D.C., s 207-216.

Breiman, L. – Friedman, J. H. – Olshen, R. A. – Stone, P. J.: *Classification and Regression Trees*. Wadsworth, 1984

Berka, Petr, 2011. ETree Miner: a new GUHA procedure for building exploration trees. In: *Foundations of Intelligent Systems.* New York: Springer, s. 96–101. ISBN 978-3-642-21915-3. ISSN 0302-9743.

Guazzelli, A., Lin, W. L., & Jena, T., 2010. *Unleashing the power of open standards for data mining and predictive analytics*. CreateSpace.

Hájek, P.; Havránek, T., 1978. *Mechanising Hypothesis Formation – Mathematical Foundations for a General Theory*. Berlin – Heidelberg – New York, Springer-Verlag, 396 pp.

Hájek, P.; Holeňa, M.; Rauch, J., 2010. The GUHA method and its meaning for data mining. *Journal of Computer and System Sciences,* 76, pp. 34-48

Kliegr a kol., 2010
Kliegr, Tomáš, Svátek, Vojtěch, Ralbovský, Martin, Šimůnek, Milan, 2010. *SEWEBAR-CMS: semantic analytical report authoring for data mining results*. Intelligent Information Systems [online], , s. 1–25. ISSN 0925-9902. URL: http://dx.doi.org/10.1007/s10844-010-0137-0.

LISp-Miner [online]. 2011. cit. 15. 2. 2012. Dostupný z WWW: http://lispminer.vse.cz

PMML [online]. 2011. cit. 15. 2. 2012. Dostupný z WWW: http://dmg.org/v4-0-1/GeneralStructure.html

Rauch, Jan, 2009. *Considerations on Logical Calculi for Dealing with Knowledge in Data Mining*. In: Ras, Zbigniew W., Dardzinska, Agnieszka. Advances in Data Management. Berlin : Springer-Verlag, s. 177–201. Studies in Computational Intelligence 223/2009. ISBN 978-3-642-02189-3. ISSN 1860-949X. URL: http://www.springerlink.com/content/d335j4128jh84481/?p=f7d847caa944444887621b6031e76ea1&pi=8

Rauch, Jan, Šimůnek, Milan, 2005. An Alternative Approach to Mining Association Rules. In: LIN, Tsau Young et.al.(eds.). *Foundations of Data Mining and Knowledge Discovery*. Berlin : Springer, s. 211–231. ISBN 3-540-26257-1. ISSN 1860-949X.

SEWEBAR – *SEmantic WEB and Analytical Reports* [online]. 2011. cit.15. 2. 2012. Dostupný z WWW: http://sewebar.vse.cz/

Šimůnek, M. 2003, Projekt LISp-Miner, *Systémová integrace*,10 (1), 99-110, ISSN 1210-9479 (print), ISSN 1804-2716 (online)

Šimůnek, Milan, 2010. *Systém LISp-Miner – akademický systém pro dobývání znalostí z databází, Historie vývoje a popis ovládání*. skripta VŠE, Praha, Oeconomica, 106 stran, ISBN: 978-80-245-1699-8

Quinlan, J. R., 1986. *Induction of decision trees*. Machine Learning, 1/1, 81-106

QUINLAN, J. R.: C4.5: *Programs for machine learni*

JEL: C40, C80