Moving Data Mining Tools toward a Business Intelligence System

Nittaya Kerdprasop, and Kittisak Kerdprasop

Abstract—Data mining (DM) is the process of finding and extracting frequent patterns that can describe the data, or predict unknown or future values. These goals are achieved by using various learning algorithms. Each algorithm may produce a mining result completely different from the others. Some algorithms may find millions of patterns. It is thus the difficult job for data analysts to select appropriate models and interpret the discovered knowledge. In this paper, we describe a framework of an intelligent and complete data mining system called SUT-Miner. Our system is comprised of a full complement of major DM algorithms, pre-DM and post-DM functionalities. It is the post-DM packages that ease the DM deployment for business intelligence applications.

Keywords—Business intelligence, data mining, functional programming, intelligent system.

I. INTRODUCTION

Data mining (DM) or Knowledge Discovery in Databases (KDD) has been defined [3] as the automatic discovery of previously unknown patterns or relationships in large and complex datasets. Most DM algorithms have been drawn from the areas of Statistics and Machine Learning adapted to induce knowledge from data contained within a database. The main objective of DM is to use the discovered knowledge for the purposes of explaining current behavior, predicting future outcomes, or providing support for business decision. The DM techniques used in business-oriented applications are also known as Business Intelligence (BI). BI is a general term to mean all processes, techniques, and tools that gather and analyze data for the purpose of supporting enterprise users to make better decisions [1], [7].

Despite its high claims and expectations, DM technology requires a highly trained professional to do an iterative, multi-step process of accessing and preparing data, choosing an appropriate algorithm to mine the data, analyzing the learned knowledge, and presenting nontrivial, valuable knowledge to executives or decision makers. Owing to advancement in the machine learning research, mining can be done efficiently on a dataset of large size. A hindrance of DM employment as an automatic knowledge acquisition tool in BI is the part of post-mining evaluation to obtain only a relevance and valuable knowledge.

The difficulty of discovering and deploying new knowledge in the BI context is due to the lack of intelligent and complete DM system. Most DM packages are comprised of learning algorithms integrated into a visual environment. Such graphical environment is a useful facility for experienced data analysts or data miners, but it provides limited functionalities for a novice to interpret and evaluate significance of the mining results.

As an example, consider Fig. 1 that shows the three different mining results obtained from three rule-induction algorithms: Ripple-Down Rule Learner, Ripper (Repeated Incremental Pruning to Produce Error Reduction), PART. The dataset is taken from the credit card promotion database [9]. The mining objective is to learn a profile for individuals likely to take advantage of a life insurance promotion advertised along with their credit card statement. A learned profile can help the credit card company to send the promotion materials only to a select group of individuals who are likely to take advantage of a life insurance promotion.

Fig. 1 Different mining results obtained from different algorithms
The mining results (i.e., classification models) shown in fig.
1 obtained from the three runs on WEKA system [10], [11].
The three classification models show the same accuracy when
tested with 10-fold cross validation technique, but all three
models produce different knowledge. The knowledge
conveyed by each model can be explained in Fig. 2.

Model 1: Ripple-down rule learner algorithm
All customers respond to the life insurance promotion.

Model 2: Ripper algorithm
If the customers are 45 years old or over,
then they do not respond to the life insurance promotion.
Otherwise, they respond.

Model 3: PART
If the customers are male and do not have credit card insurance,
then they do not respond to the life insurance promotion.
Otherwise, they respond.

II. DATA MINING CONCEPTUAL MODEL

DM is about learning patterns. Pattern is an expression
describing a subset of the data, e.g., \( f(x) = 3x^2 + 3 \) is a pattern
induced from a given dataset \( \{(0,3), (1,6), (2,15), (3,30)\} \),
whereas the term model refers to a representation of the source
generating the data, e.g., \( f(x) = ax^2 + b \). However, in his paper
we use the term pattern and model interchangeably.

According to [3], DM involves fitting models to, or
determining patterns from, observed data. Primary goals of
DM are prediction and description.

Prediction uses supervised learning technique to predict
values of data using known values found from different data.
DM tasks for prediction include classification, regression,
time-series analysis.

Description focuses on employing unsupervised learning
technique to find human-interpretable patterns describing the
data. DM tasks for description are clustering, summarization,
association, and sequence discovery.

To start building a DM methodology, it is necessary to set
up a conceptual framework into which data and its structures
might be classified. The terminology to denote structures of
the dataset is summarized in Fig. 3.

We adopt the ontology of DM methodology proposed by
[8] to abstract data and their relationships for guiding the DM
method selection. The simple ontology presented in Fig. 4
provides a means to explicitly guide the novice data miners
towards DM task selection. This guideline is data-driven in
that the data types and structures are used as a basis for
selecting appropriate DM method.

Our data mining system consists of three main parts: pre-
DM, DM, and post-DM. The ontology presented in Fig. 4 is
for a method selection in the DM part. We also need complete
ontology for the parts of pre-DM and post-DM as well as the
full specification of each DM task. These will be our future
work.
reduce the data to a reasonable and sufficient size with only relevant attributes.

The DM phase performs mining tasks including classification, prediction, clustering, and association. The post-DM phase involves evaluation, based on corresponding measurement metrics, of the mining results. DM is an iterative process in that some parameters can be adjusted and then restart the whole process to produce a better result.

The post-DM phase is composed of knowledge evaluator, knowledge reducer, and knowledge integrator. These three components perform major functionalities aiming at a feasible knowledge deployment which is important for the applications in BI. The overall architecture of our SUT-Miner system is presented in Fig. 5.

IV. RAPID PROTOTYPING WITH HASKELL

The implementation of SUT-Miner system is mainly based on the functional programming paradigm using Haskell language [2], [4]. Functional languages (FL) offer a number of advantages over imperative languages (IL). FL can be used to express specifications of problems in a more concise form than IL. This results in the creation of program source codes that are shorter and easier to understand. The following example shows C versus Haskell codes to compute a list of fibonacci numbers starting with zero.

C-code

```c
int * fib (int n)
{
    int a = 0, b = 1, i, temp;
    int * fibsequence;
    fibsequence = (int *) malloc ((sizeof int) *n);
    for (i = 0; i<n; i++)
    {
        fibsequence[i] = a;
        temp = a + b;
        a = b;
        b = temp;
    }
    return fibsequence;
}
```

Haskell-code

```haskell
fib :: [Int]
fib = 0: 1: [ a+b | (a, b) <- zip fib (tail fib) ]
```

Haskell is a pure FL having a polymorphic type system, i.e. a data type can take type variables as parameters. This feature provides a high level abstraction leading to generic programming. Haskell is also a lazy FL, i.e. a value is evaluated only when it is needed. This feature allows infinite structures, such as an infinite sequence of fibonacci numbers, to be defined. According to our experimentation, the speed of running Haskell program on a moderate-size dataset is quite impressive. The experimental results shown in Fig. 6 compare the running time of a Haskell program against a Java program for mining a linear regression model. The first experiment
computes a regression model of two variables. The number of variables is increased to six in the second experiment. The experimentations are performed on a computer notebook with CPU speed 1.8 GHz and main memory 512 MB.

Fig. 6 Mining regression model with Haskell and Java

V. CONCLUSION AND FUTURE WORK

We present work in progress on the development of the SUT-Miner, a complete data mining system. The system is complete in that the pre-DM and post-DM phases are also included in the DM process. Most DM packages contain only the DM modules, while some systems incorporate a pre-DM module as a data preparation phase.

According to our knowledge, a post-DM phase is omitted in most systems. Post-processing of DM is very essential to the success of DM utilization. This is due to the fact that discovered knowledge is sometimes voluminous and redundant. At present, knowledge evaluation and filtration have to be done by human experts. We thus design our system to include this knowledge processor as another major component of the mining system.

The implementation of the SUT-Miner system uses a Haskell functional language. The functional programming is a paradigm of our choice because of its advantages on modularity, conciseness, polymorphism, and formal specification which supports the proof of program correctness. We plan to extend our design to produce an approximate model by means of progressive mining. We currently investigate the feasibility of applying a Markov Chain Monte Carlo method in our approximate data mining scheme.

APPENDIX

The Haskell code to mine two-variable and six-variable regression models is provided here.

```
main = do
    hSetBuffering stdin LineBuffering
    ex <- readArff -- ex is example
    let exs = read ex :: [[Float]]
    let n = length (head (read ex :: [[Float]]))
    if (n==2) then solution_2 exs
    else if (n==3) then solution_3 exs
    else if (n==4) then solution_4 exs
    else if (n==5) then solution_5 exs
    else if (n==6) then solution_6 exs

    write x = do
        hdl <- openFile "matrix.txt" WriteMode
        hPutStr hdl x
        hClose hdl

    -- Function for Solutions
    -- Input/Output function
    multi :: [Float] -> [Float] -> [Float]
    my_length :: [a] -> Float
    sumsq :: [Float] -> [Float]

    my_length [] = 0
    my_length (x:xs) = 1 + my_length xs
    sumsq [] = []
    sumsq (x:xs) = [x*x] ++ sumsq xs

    multi (x:xs) (y:ys) = [x*y] ++ multi (xs) (ys)

    -- s2_Solution for Exponential Regression
    --
    solution_2 ex = do
        let aa = 2
        let ab = s2_sum_x ex
        let ba = s2_sum_x ex
        let bb = s2_sum_x2 ex
        let ya = s2_sum_y ex
        let yb = s2_sum_y ex
        let line_1 = "2
        let line_2 = show aa ++ "," ++ show ab ++ "," ++ show ba ++ "," ++ show bb ++ "," ++ show ya ++ "," ++ show yb
        let line_4 = "n"
        let line_5 = show ya ++ "+" ++ show yb
        let line_6 = write (line_1++line_2++line_4++line_5

    s2_sum_x :: [Float] -> Float
    s2_sum_y :: [Float] -> Float
    s2_sum_x2 :: [Float] -> Float
    s2_sum_multi :: [Float] -> [Float]
    s2_form_x :: [Float] -> [Float]
    s2_form_y :: [Float] -> [Float]
```

This code can be found in the appendix of the paper.
s2_sum_x [] = 0
s2_sum_x xs = sum(s2_form_x xs)
s2_sum_y [] = 0
s2_sum_y ys = sum(s2_form_y ys)
s2_sum_x2 [] = 0
s2_sum_x2 xs = sum(sumsq(s2_form_x xs))
s2_sum_xy [] = 0
s2_sum_xy s = sum(multi (s2_form_x s)(s2_form_y s))
s2_sum_multi [] = 0
s2_sum_multi s = sum(multi (s2_form_x s)(s2_form_y s))

-- s5_Solution for Exponential Regression
--
solution_6 ex = do
  let aa = 6
  let ab = s6_sum_x1i ex
  let ac = s6_sum_x2i ex
  let ad = s6_sum_x3i ex
  let ae = s6_sum_x4i ex
  let af = s6_sum_x5i ex
  let ba = s6_sum_x1i ex
  let bb = s6_sum_x1i2 ex
  let bc = s6_sum_x1x2 ex
  let bd = s6_sum_x1x3 ex
  let be = s6_sum_x1x4 ex
  let bf = s6_sum_x1x5 ex
  let ca = s6_sum_x2i ex
  let cb = s6_sum_x1x2 ex
  let cc = s6_sum_x2i2 ex
  let cd = s6_sum_x2x3 ex
  let ce = s6_sum_x2x4 ex
  let cf = s6_sum_x2x5 ex
  let da = s6_sum_x3i ex
  let db = s6_sum_x1x3 ex
  let dc = s6_sum_x2x3 ex
  let dd = s6_sum_x3i2 ex
  let de = s6_sum_x3x4 ex
  let df = s6_sum_x3x5 ex
  let ea = s6_sum_x4i ex
  let eb = s6_sum_x1x4 ex
  let ec = s6_sum_x2x4 ex
  let ed = s6_sum_x2x5 ex
  let ee = s6_sum_x4i2 ex
  let ef = s6_sum_x4x5 ex
  let fa = s6_sum_x5i ex
  let fb = s6_sum_x1x5 ex
  let fc = s6_sum_x2x5 ex
  let fd = s6_sum_x3x5 ex
  let fe = s6_sum_x4x5 ex
  let ff = s6_sum_x5i2 ex
  let ya = s6_sum_yi ex
  let yb = s6_sum_x1yi ex
  let yc = s6_sum_x2yi ex
  let yd = s6_sum_x3yi ex
  let ye = s6_sum_x4yi ex
  let yf = s6_sum_x5yi ex
  
let line_1 = "3\n" let line_2 = "aa++, ab++, ac++, ad++, ae++, af++, ba++, bb++, bc++, bd++, be++, bf++, ca++, cb++, cc++, cd++, ce++, cf++, da++, db++, dc++, dd++, de++, df++, ea++, eb++, ec++, ed++, ee++, ef++, fa++, fb++, fc++, fd++, fe++, ff++, ya++, yb++, yc++, yd++, ye++, yf"
write (line_1++line_2++line_3++line_4++line_5++line_6++line_7++line_8++line_9)
Nittaya Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. She received her B.S. from Mahidol University, Thailand, in 1985, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, USA, in 1999. She is a member of ACM and IEEE Computer Society. Her research of interest includes Knowledge Discovery in Databases, AI, Logic Programming, Deductive and Active Databases.

Kittisak Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. He received his bachelor degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, USA, in 1999. His current research includes Data mining, Artificial Intelligence, Functional Programming, Computational Statistics.

ACKNOWLEDGMENT

This work was fully supported by research fund of Suranaree University of Technology granted to the Data Engineering and Knowledge Discovery (DEKD) Research Unit, in which Kittisak Kerdprasop is a director and Nittaya Kerdprasop is a member and researcher. The authors are also partly supported, by National Research Council of Thailand (NRCT) and the Thailand Research Fund (TRF). The first author would like to thank all the research assistants who participated in the SUT-Miner project, especially Prapanpong Nopsuwan who coded linear regression in Haskell and Java.

REFERENCES