

USER INTERFACE DESIGN FOR THE
RELATIONAL INDUCTIVE LEARNING ALGORITHM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY

BY

UTKU KAPUCU

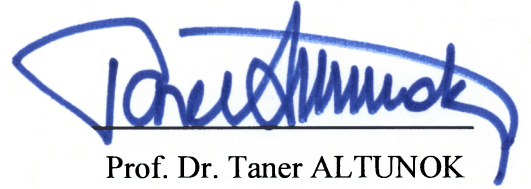
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2010

Title of the Thesis : **User Interface Design For The Relational Inductive Learning Algorithm**

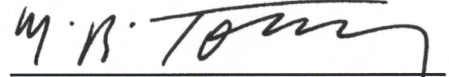
Submitted by **Utku Kapucu**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University



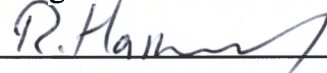
Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.



Prof. Dr. Mehmet Reşit TOLUN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.



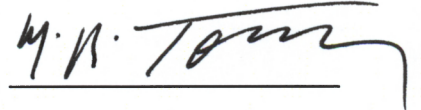
Assist. Prof. Dr. Reza Z. HASSANPOUR
Supervisor

Examination Date : 02.12.2010

Examining Committee Members

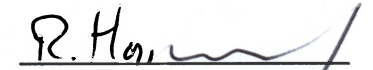
Prof. Dr. Mehmet Reşit Tolun

(Çankaya Univ.)



Assist. Prof. Dr. Reza Z. Hassanpour

(Çankaya Univ.)



Assoc. Prof. Ferda Nur Alpaslan


(METU)



STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name :Utku Kapucu

Signature : 

Date : 02.12.2010

ABSTRACT

USER INTERFACE DESIGN FOR THE RELATIONAL INDUCTIVE LEARNING ALGORITHM

Kapucu, Utku
M.S.c., Department of Computer Engineering
Supervisor : Assist. Prof. Dr. Reza Z. Hassanpour

December 2010, 52 pages

The study subject of this thesis is developing user interface for the learning algorithm named RILA which is designed for relational databases. RILA algorithm developed by adding new features on ILA and ILA2 which are inductive learning algorithms developed before RILA algorithm. While the user interface is developed, showing of the rules generated by RILA aimed to be understandable to the user. In addition, the logs of the process of rule generation are shown to the user. In developmental stage, the library which contains the graphical elements is used and platform-independent programming language was chosen. The user interface is also has ability to connect to multiple relational database. It is observed that; the software provides the possibility of adding new modules in the future as of the design.

Keywords: Relational Learning Algorithms, Graphical User Interface, Inductive Algorithm.

ÖZ

USER INTERFACE DESIGN FOR THE RELATIONAL INDUCTIVE LEARNING ALGORITHM

Kapucu, Utku
Yükseklisans, Bilgisayar Mühendisliği Anabilim Dalı
Tez Yöneticisi: Assist. Prof. Dr. Reza Z. Hassanpour

Aralık 2010, 52 sayfa

Bu tezin çalışma konusu ilişkisel veritabanları için tasarlanmış olan RILA adlı bir öğrenme algoritması için kullanıcı arayüzü geliştirmektir. Bu algoritma kendisinden önce geliştirilen tümevarımsal öğrenme algoritmaları olan ILA ve ILA2' nin üzerine yeni özellikler eklenerek geliştirilmiştir. Kullanıcı arayüzü geliştirilirken özellikle RILA' nın üretmiş olduğu kuralların kullanıcıya gösteriminin anlaşılabilir olması hedeflenmiştir. Bunun yanında kural üretimi sürecinin logları da kullanıcıya gösterilmektedir. Gelişim aşamasında, grafiksel elemanlar içeren kütüphane kullanılmış ve platformdan bağımsız bir yazılım dili seçilmiştir. Bu kullanıcı arayüzü, birden çok ilişkisel veritabanına bağlanabilme yetisine de sahiptir. Gözlenmektedir ki; yazılım, tasarımı itibari ile gelecekte yeni modüller ekleme imkanı sağlamaktadır.

Anahtar Kelimeler: İlişkisel Öğrenme Algoritması, Grafiksel Kullanıcı Arayüzü, Tümevarım Algoritması.

ACKNOWLEDGMENTS

The author wishes to express his thanks to Prof. Dr. Mehmet Reşit TOLUN and to his supervisor Assist. Prof. Dr. Reza Z. Hassanpour for their suggestions and comments.

The author would also like to deepest gratitude Dr. Mahmut ULUDAĞ for his guidance, advice, criticism, encouragements and insight throughout the research.

TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM.....	iii
ABSTRACT	iv
ÖZ	v
ACKNOWLEDGMENTS.....	vi
TABLE OF CONTENTS	vii
LIST OF TABLES.....	ix
LIST OF FIGURES	x
LIST OF ABBREVIATION	xii
CHAPTERS:	
1. INTRODUCTION.....	1
2. RILA- THE NEW RULE INDUCTION SYSTEM.....	3
2.1 Dimension Table	3
2.2 Referential Integrity.....	4
2.3 ILA and ILA2 Algorithms	5
2.3.1 The ILA algorithm	5
2.3.2. The ILA2 algorithm	6
2.4 New Features in RILA.....	7
2.5 Basic Architecture of RILA System.....	8
2.5.1 How it works?.....	8
2.5.2 Query generation.....	10
2.5.3 Pruning	12
2.5.4 Rule selection	13

2.5.4.1	Select early strategy.....	14
2.5.4.2	Select late strategy.....	17
2.5.5	Optimistic estimate pruning	20
2.5.6	Strategy selection.....	21
3.	GRAPHICAL USER INTERFACE FOR RELATIONAL INDUCTIVE LEARNING ALGORITHM.....	24
3.1	What is GUI?.....	24
3.2	Why does RILA need a GUI?	25
3.3	Why Java?	25
3.4	Software Architecture.....	27
3.4.1	GUI package.....	33
3.4.2	Example for the classes of GUI package	34
3.4.3	Implementation.....	48
4.	CONCLUSION.....	51
	REFERENCES.....	R1
	APPENDIX.....	A1
	CV	

LIST OF TABLES

TABLES

Table 1. Comparing of the Select Early and Select Late Strategies	22
Table 2. Components Used in GUI.....	27
Table 3. New Added Classes.....	28
Table 4. Class That Imports <i>MySingleton</i> Class.....	31
Table 5. Classes and Their Extendings	33

LIST OF FIGURES

FIGURES

Figure 1. The Basic Architecture of the RILA Induction System.....	8
Figure 2. The RILA Algorithm Using the Select Early Strategy	15
Figure 3. Simple Illustration of Rila Algorithm Using the Select Late Strategy	17
Figure 4. Rule Selection Algorithm When Using the Select Late Strategy	19
Figure 5. Java Byte Code and Platform Independence	26
Figure 6. InputPanel Calls <i>runAlgoritihm</i> Method of a Run Instance	28
Figure 7. Setting Part of <i>runAlgorithm</i> Method in Run Class.....	29
Figure 8. <i>runAlgoritihm</i> Runs Select Late Startegy or Select Early Strategy	30
Figure 9. Process Diagram of Selecting Strategy	30
Figure 10. Class Model for <i>MySingelton</i> Related Classes	32
Figure 11. <i>LoginPanel.java</i>	35
Figure 12. Code Segment of <i>LoginPanel.java</i> for Connection a Database	36
Figure 13. BorderLayouts of <i>InputPanel</i> and <i>JTreePanel</i>	36
Figure 14. <i>Entity</i> Class	37
Figure 15. <i>JTreePanel.java</i> Menu with respect to the Tree Node.....	38
Figure 16. <i>actionperformed</i> Method in the Inner Class <i>PopupActionListener</i> .	39
Figure 17. <i>Settingsparam</i> Class	40
Figure 18. Code Segment of <i>InputPanel.java</i> (Values are Set from <i>JTreePanel</i>)	42

Figure 19. <i>InputPanel</i> ScreenShot.....	42
Figure 20 Menu in <i>GuiMain</i>	43
Figure 21. Rules Output	44
Figure 22. Rules in Grid Panel	45
Figure 23. HTML Panel That Contains Rules Generated.....	46
Figure 24. <i>getValues</i> Method for Drawing Bar Chart	47
Figure 25. Bar Chart for <i>Gene Table</i> and <i>Localization</i> Column	48
Figure 26. SQL SERVER Connectivity	49
Figure 27. GUI on LINUX.....	50

LIST OF ABBREVIATIONS

ABBREVIATIONS

RILA	Relational Inductive Learning Algorithm
ILA	Inductive Learning Algorithm
GUI	Graphical User Interface
UI	User Interface
RDBMS	Relational Database Management System
DBMS	Database Management System
IDE	Integrated Development Environment
JDK	Java Development Kit
SQL	Structured Query Language
CPU	Central Processing Unit
GB	Gigabyte
Ghz	Gigahertz

CHAPTER 1

INTRODUCTION

Relational database management system (RDBMS) was initiated to cater to the ever increasing needs of storing complex data in an efficient way. Owing to its enhanced capabilities, relational databases stored by RDBMS can represent more complex and structured data as compared to the conventional single tables [1]; [2]. These benefits have made relational databases much more desirable for the storage and representation of modern scientific and commercial data.

Data mining systems offer advanced searching mechanism which has a large number of great benefits. One of these benefits is that only the required pattern of the data is loaded in the memory, which not only saves time and memory consumption but also keeps the data free for other queries. To enjoy the benefits offered by the data mining systems and to overcome the complexities of relational data, collaboration is formed between these two.

Traditional relational learning algorithms were called ILP-based algorithms [3], designed for relational data stored in Datalog/Prolog servers in the past. Efforts have been made to couple ILP-based algorithms with the modern relational database systems [4], however they have their limitations.

Data mining and machine learning both concern with retrieving interested data and unknown knowledge from databases [5]. According to [6] learning process that applied to a database which is used as a training set is called data mining. Learning rules from database can be made by an automated tool. During time, machine learning techniques has been developed and applied to large database to get knowledge in addition to learn rules for expert systems. Because importance of application of data mining has been rising [7].

So new relational learning algorithm explained first with its strategies and the need of UI for this learning algorithm is discussed. Then software architecture is described. It is finished by examining the GUI classes in an example. Finally whole work is concluded.

CHAPTER 2

RILA – The New Rule Induction System

This chapter is related to a new rule induction system known as RILA [8]. It can be used to extract recurring patterns from multiple relations which are interconnected. This rule induction system mainly comprises consists of four steps; Hypotheses Construction, Rule Selection, Pruning and Conversions to/from SQL. RILA can make use of two different strategies for rule selection according to the situation; Select Early Strategy and Select late Strategy. There is also a brief introduction to Dimension Tables.

2.1 Dimension Table

A Dimension Table is usually a set of interconnected tables which surrounds the Fact Table in a schema, whereas, a Fact Table has measurements, metrics or facts regarding a business. Fact Table has the Foreign Keys which are Primary Keys in the Dimension Table. Dimension Tables are used to summarize, constrain or group data according to specific criteria while performing data mining queries.

The attributes in the dimension tables portray the fact records in the Fact Table. Usually, they provide two different type of information to the analyst; descriptive information about the attributes in the Fact Table and information how the data in Fact Table should be grouped or summarized. This grouping or summarization is

possible due to hierarchies separating the products in to different categories in a Dimension Table e.g. a motor showroom containing cars, jeeps which can be subdivided into cars of different brands or models.

In dimensional modeling the attributes in each dimension are autonomous and do not depend on any attribute in the other dimension tables e.g. a motor showroom dimension table will contain data about the different showrooms only, a customer dimension table contains data about customers and a product dimension table contains information about products i.e. cars. But queries can join attributes in the different dimension tables to represent the required information. For example, a query might use the product, showroom, and time dimensions to ask the question "What was the cost of Mercedes sold in the northeast region in 2005?" Subsequent queries might drill down along one or more dimensions to examine more detailed data, such as "What was the cost of Mercedes-Benz SLR McLaren in New York City in the third quarter of 2005?"

The data in the warehouses is stored and can be used for many years to come. As the time passes changes in the attributes of a dimension table are becoming more and more evident. For example shipping address of a showroom may change after some time. This phenomenon can cause discrepancies in the data.

2.2 Referential Integrity

To avoid the discrepancies in the relational data, it is very necessary to maintain referential integrity between all the dimension tables and fact table as well. The primary keys of dimension tables reside as foreign keys in the fact table. Referential integrity means that each entry in the fact table must have a relevant record in the

dimensional table through primary key/foreign key relation. If there are some records missing, facts can be missed when the fact table is joined with the set of dimension tables and the queries will also fetch inconsistent results.

2.3 The ILA and ILA2 Algorithms

RILA has two predecessors; ILA and ILA2. Actually, RILA is based on ILA2 [9] which subsequently is the more advanced and noise-tolerant version of ILA [10]. It will be easier to understand the working of RILA to first go through a short review of its two forerunners and then the different new features which were added in RILA.

2.3.1 The ILA Algorithm

The ILA algorithm is an inductive algorithm for generating a set of classification rules for a collection of training examples i.e. extracting rules from a collection of examples in a given domain. The example is described with reference to a fixed set of attributes; with each one having its own set of possible values. ILA generates classifiers in form of ordered rules and due to its hypotheses evaluation criteria it always generates 100% correct rules for the training data [8].

The ILA algorithm works in a repetitive fashion. Each iteration of the algorithm searches for a rule which covers a large number of training examples of a single class. Once ILA has selected a rule it removes the examples covered by it from the training data by marking them, and appends the selected rule at the end of its set of rules selected so far. Instead of producing a decision tree ILA produces an ordered list of rules.

2.3.2 The ILA2 Algorithm

The ILA2 algorithm is a sophisticated and noise-tolerant version of ILA. The ILA2 algorithm has been designed to overcome the performance issues encountered in the ILA algorithm. These issues are eliminated by implementing a new hypothesis evaluation function by selecting multiple rules, instead of selecting a single rule as in ILA, respectively. Another difference from ILA is that the ILA2 takes the noise factor into account by using a penalty parameter defined by the user.

Generally a hypothesis evaluation function's score should increase both with the number t_p of the positive instances covered and with the number t_n of negatives not covered. The score should decrease in proportion to the number of negative instances incorrectly classified, f_n . However, the original ILA evaluation metric discards a hypothesis if the number of incorrect classifications, f_n , is greater than zero. For this reason, ILA does not make any distinction between a hypothesis which incorrectly classifies 100 instances and another hypothesis which incorrectly classifies only 1 instance. The ILA evaluation metric can be summarized using the following terms. If a hypothesis covers any of the negative examples of the current class then the score is zero. Otherwise the score is equal to the number of positive examples covered.

This metric assumes no noise to be present in the training data, searching for a concept description that classifies training data perfectly. However, application to real-world domains requires methods for handling noisy data.

2.4 New Features in RILA

RILA is based on ILA2 which in turn was based on ILA. Although RILA inherits many features from its predecessors yet many new features have been added in RILA in order to overcome the shortcomings of the other two algorithms. This new inductive learning algorithm adapts following main features; level-wise search [11];[12] and the example covering approaches from ILA and the hypothesis evaluation metric and the multiple rule selection idea from ILA2 algorithm. In addition to the new features for relational learning, RILA also has some new features that ILA and ILA2 do not have. Here is a brief summary of the new features in RILA.

- In addition to select early strategy, there is a more efficient rule selection strategy in RILA known as select late strategy.
- Implementation is carried out more efficiently as hypotheses can be refined by adding new conditions. They do not need to be generated from scratch in each learning loop in each level.
- New pruning strategies; the minimum support pruning, the minimum Fmeasure pruning, and the optimistic estimate pruning heuristics.
- The ILA2 hypothesis evaluation function is normalized by the total number of examples in the current class and in the other classes. This is needed to take into account also the varying number of examples in the active class, depending on the joins made when building a hypothesis.

2.5 Basic Architecture of RILA System

[13] stated RILA as a tightly-coupled data mining application. When RILA runs, complete training data does not have to be held in its working memory. Java is used as the coding language of this system and it uses JAVA JDBC API to communicate with the database management system. Figure 1 presents a simple illustration of the architecture of the system.

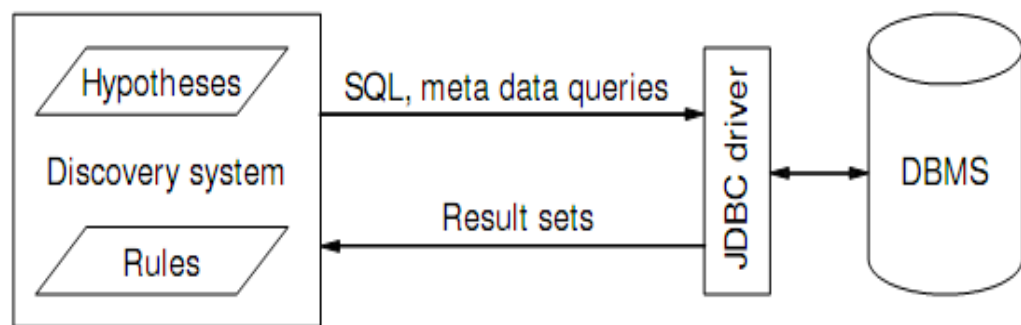


Figure 1. The Basic Architecture of the RILA Induction System [8]

To understand the architecture easily we can virtually divide it into two parts; one is the database server which not only stores and manages the data but also computes the results for the queries sent by the learning algorithm, second is the RILA learning algorithm which performs the actual search for rules by acting on certain steps which are explained in the next section.

2.5.1 How It Works?

By understanding the working of RILA one can also understand the benefits it offers while traversing the relational database, generating valid hypotheses and selecting rules. RILA has components which can construct hypotheses and select rules,

however, as it is a process for the relational databases, it also has components which can traverse the relational schema.

The first step in the process can be called initialization as the database is connected, tables are selected and parameters for rule selections are defined by the user. The user connects to the desired database and selects the set of tables which stores the training data. Following are the two general input options for the initialization phase:

1. The names of the tables that constitute the objects to be analyzed
2. The name of the target table, and the name of the class attribute.

Then the user starts the learning process. Meta-data queries are sent to the DBMS which fetch the descriptions of the columns, primary keys and foreign keys of the tables. The initialization phase ends here and the queries sent to the database after. These are generally for building valid hypotheses about the data. On the whole, the system sends SQL queries to the database system and then by analyzing the results of these queries it produces new hypotheses. At this point the system has the complete schema description of the training data.

The two main steps which are performed at this stage are:

1. The system sends SQL queries to the database system and analyzes the results.
2. Then the system analyzes the results of these queries and generates new hypotheses.

These two steps are repeated many times to further analyze the data. When a new row is selected the examples which are covered by the rule selected are removed

from the active search space. RILA does not delete the rows covered from the input table. Instead, it creates a temporary table to store the identifiers of the examples covered. It uses the primary keys in the target table as an identifier for every example that has already been covered by it. These examples are then excluded from the search space with the help of a join to the temporary table in the SQL.

The temporary table which stores information about the covered examples is also used to implement the 'effective cover'. Effective cover is merely used to avoid redundant rule selection. The system has a goal to keep the rule set size to minimum, therefore, redundant rules are not wanted. The effective cover of a rule is defined as the number of examples it covers that have not been covered so far by any other rule. If the effective cover of a candidate rule is zero, it means that examples covered by this candidate rule were already covered by the previously selected rules. So this rule is not considered and not appended to the final rule list.

The strategy applied by RILA makes sure that the input relational data stays in its original form and also stays available for the other process. The temporary table stores only the identifiers of the examples in the current class for which classification rules are being searched and it is cleared after each class is being processed.

2.5.2 Query Generation

Query generation is the basic functionality of both the hypotheses construction and rule selection however it acts differently for both of these different steps. During hypotheses generation the queries gather the recurring patterns and frequency information about the training data. This helps the system in making the initial and

subsequent valid hypotheses. When the rule selection process starts, query generation acts differently and it evaluates each candidate rule.

RILA traverses the schema by the foreign keys and then builds the initial hypothesis. This hypothesis is then refined by adding new conditions. The current attribute column is considered as the new condition. The initial hypotheses are based on only one condition. Here is the template used to generate the SQL queries for finding hypotheses and their frequency values.

```
Select attr, count (distinct targetTable.pk) from covered, path.getTableList() where  
path.getJoins() and targetTable.classAttr = currentClass and covered.id =  
targetTable.pk and covered.mark = 0 group by attr
```

In this query,

- attr is the name of the current attribute column,
- targetTable is the target table,
- pk is the name of the primary key column in the target table
- covered is the name of the temporary table where identifiers of the objects covered by the selected rules are stored,
- path refers to the path object that links the current table to the target table.
- classAttr is the column representing the class attribute for the learning task
- currentClass is the current class for which the hypotheses are being searched

2.5.3 Pruning

When RILA is applied on large relational data, we can expect a large number of hypotheses generated, therefore, some kind of heuristic is required. The procedure used to reduce the number of hypotheses to a reasonable size is called pruning, which trims or prunes the hypotheses selected. The technical term used for this process is called pruning heuristic.

There are diverse kinds of pruning heuristics available. Minimum Support Pruning Heuristic is considered to be the best one and used by the most of the data mining systems. No doubt, it is an effective approach and keeps the number of selected hypotheses small but it is not very effective for complex relational databases. The reason for this is that this pruning approach alone is not always good enough to avoid the weak hypotheses which are unlikely to produce strong hypotheses when they are refined.

Therefore, for the larger and more composite problems more advanced and complex pruning techniques are utilized. These techniques are more likely to produce comparatively stronger hypotheses which also produce strong hypotheses when refined. Optimistic Estimate Pruning is one of the most commonly used approaches by the traditional machine learning systems such as ICL [14] and m-FOIL [15]. The optimistic estimate pruning is also known as beam search because of its pruning method. This approach specifies a number of best 'n' solutions which are desired. Any hypothesis and its descendants which fail to fall in the top n solutions are pruned. However, the user must specify a reasonable size for the parameter 'n' because if the 'n' is not large enough the system may suffer from the myopia

problem i.e. only few hypotheses are selected and some hypotheses which may have been important are pruned.

RILA supports both minimum support pruning heuristic and optimistic estimate pruning heuristic.

2.5.4 Rule Selection

In inductive algorithm, there can be many different possible arrangements for hypotheses construction and rule selection, for example, one strategy may select rules every time a group of hypotheses is constructed, while another strategy may activate rule selection after all hypotheses have been constructed for the active class. RILA makes use of the two different rule selection strategies; The Select Early Strategy which is inherited from the ILA algorithm [16] and The Select Late Strategy which was developed with RILA. The difference between the two strategies is the activation of the rule selection process. The Select early strategy activates rule selection more frequently as compared to the select late strategy. The select early strategy activates the rule selection process as soon as the hypothesis is constructed for the current level whereas the select late strategy postpones the rule selection until all the hypothesis have been generated for all the levels of the active class. In turn, RILA works for each class, for example, if the class attribute has three different values the learning loop is repeated three times.

When RILA is working on the select early strategy, the examples covered by the new rules are removed from the active search space as soon as a new rule is selected which results in the reduction of search space. Although it helps in reducing the training time required for learning tasks but the rules selected towards the need of the

learning process are not based on as many examples as available during the early stages of the learning process. To overcome this problem [17] proposed the weighted covering algorithm. In the projected algorithm, already covered positive examples are not deleted from the search space, instead, the algorithm stores a count with each example which shows how many times the example has been covered. This information is later used by the weighted relative accuracy heuristic.

The select late strategy is free of this problem, however, because of the postponing the activation of rule selection until the enumeration of all the hypotheses, the select late strategy becomes more complex and needs a lot more computational resources. This more computational cost can cause efficiency problems. In order to avoid these efficiency problems the number of hypotheses generated is pruned by using optimistic estimate pruning heuristic.

Now let us understand the working of RILA when these two rule selection strategies are used separately:

2.5.4.1 Select Early Strategy

Select early strategy activates rule selection more frequently. Every time the hypotheses are generated for a level, the select early strategy activates the rule selection process for the hypotheses built so far.

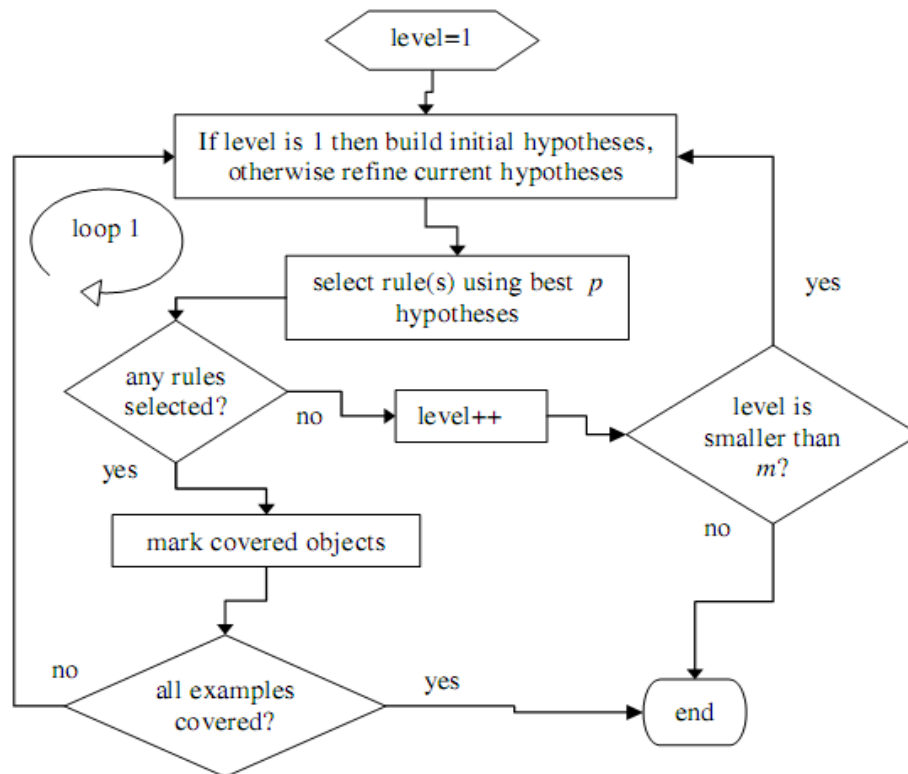


Figure 2. The Rila Algorithm Using the Select Early Strategy [8]

Figure 2 demonstrates the working of RILA algorithm while using the select early strategy for a single class. The process is repeated for every class attribute in this class. First hypotheses are generated for the current class with one condition by traversing the input schema graph. Furthermore, two relational queries are executed for the every attribute column traversed. RILA builds a set of hypotheses on the basis of the results fetched by these queries. When the schema graph is traversed completely and query results have been process, the rule selection step starts.

When the rule selection step starts, the hypothesis with maximum score is selected as the current new rule and this rule is removed from the active search space or hypothesis set. The ‘effective cover’ rule (described in the beginning) comes in to play and the examples which have been covered by the new rule are marked as

covered. The process of rule selection is repeated 'p' times (p is a predefined parameter).

Once RILA has selected the first rule using select early strategy, it checks whether the examples covered by the new candidate rule are already covered by the previous rules or not. If these examples are already covered then this rule is pruned, otherwise, the new candidate rule is asserted as a new rule in the output rule set.

The above mentioned process is repeated until the rule selection is completed. If the new rules have been selected and there is still data not covered by these rules then the initial hypothesis is rebuilt, however, this time that data is considered which is not covered by the already generated rules. This process is repeated until no new rule can be selected or all the examples in the currently active class have been covered by the generated rules. This indicates the completion of rule selection for level 1.

After the completion of level 1, RILA moves to level two. It refines the best n hypotheses generated in the previous level by traversing the input schema graph and executing the two relational queries. New hypotheses are built on the basis of the results fetched by these queries. Once schema is traversed completely and the results fetched by the queries have been processed, the rule selection process starts once again. In all the next levels rules are selected as described for the step 1. These steps are repeated until the system reaches the predefined parameter m, then the algorithm terminates.

2.5.4.2 Select Late Strategy

The select late strategy activates rule selection less frequently as compared to the select early strategy. The select late strategy activates the rule selection only after all the hypotheses have been generated for all the levels of the active class in the schema. Therefore, while using the select late strategy, the rules are selected after all the hypotheses have been constructed. These rule selection algorithms are more complicated. They have to ensure that the output rule set covers most of the instances in the training data after the hypotheses have been generated for all the levels of the current class instead of the rule selection at the end of every level. A simple illustration of this process is presented in Figure 3.

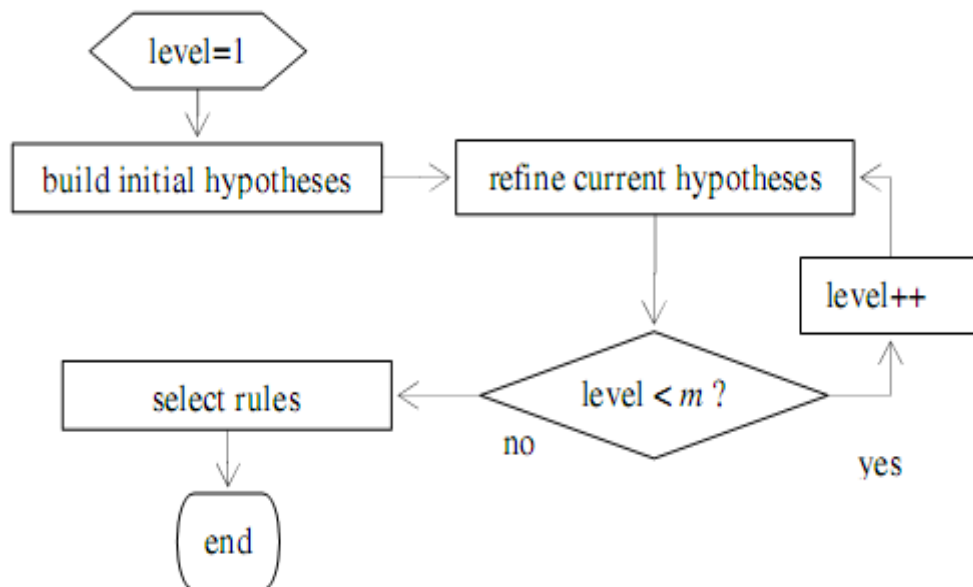


Figure 3. Simple Illustration of RILA Algorithm Using the Select Late Strategy [8]

In the select late strategy, the algorithm generates the hypothesis for the first level with one condition for the active class. As the rules selection is postponed by the

algorithm, it moves to level 2 and constructs new hypotheses by refining the n best hypotheses generated in the previous level, where n is the pruning factor which has been described previously in the pruning section. These steps are repeated until the level is equal to the predefined parameter m . When the level is equal to m , it means that hypotheses for all the levels of the active class have been constructed and rule selection process starts.

In both cases; the select early and the select late strategies, the construction of hypothesis for the first level and its refining for the subsequent levels is the same. However, the difference occurs that unlike select early strategy there is no rule selection between the levels when using select late strategy. Furthermore, the best n hypotheses which are in every next level are selected only from the hypotheses constructed in the preceding level, this is necessary to stop the algorithm from refining the same hypotheses again at different levels.

Figure 4 shows the rule selection algorithm when using the select late strategy. This process is also similar to the one used in select early strategy. During the rule selection, first the hypothesis which has the maximum score is considered the new rule. This rule is then removed from the search space of active hypothesis set. With the help of 'effective cover' all the examples covered by this rule in the temporary table are marked as covered, so they may not be reused during the rule selection process in future. Once the first rule is asserted, the next hypothesis with the highest score is selected as the new candidate rule. The effective cover of every candidate rule is determined by traversing the number of examples covered by the new candidate rule which are not already covered by any previous rule.

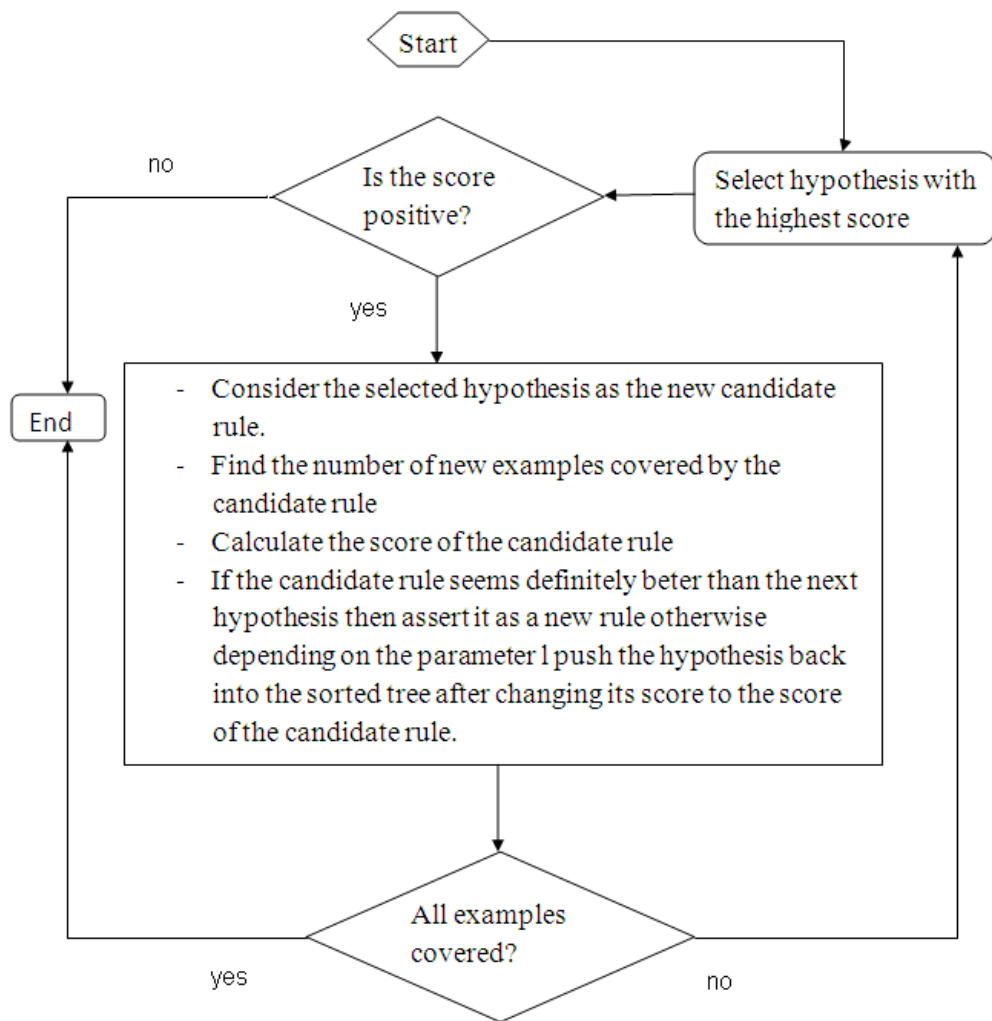


Figure 4. Rule Selection Algorithm When Using the Select Late Strategy [8]

Effective cover is used to recalculate the score of the candidate rule here again. Once the score is calculated it is compared to the score of the next hypothesis, already stored in the tree of hypotheses. If the score is higher than the next hypothesis, this candidate rule is asserted as a new rule and the hypotheses is removed from the active hypotheses set, to stop it from being compared again in the future. All the examples covered by this new rule, which are present in the temporary table, are marked as covered. But if the score of the current candidate rule is less than the score

of the next hypothesis then the parameter l is used for the decision making by the following two methods:

1- If the difference between the current score of the hypothesis and its original score before the rule selection started is more than the parameter l , then the candidate rule is selected as a new rule, and the examples covered by the new rule are marked in the temporary table as covered.

2- If their difference is less than the parameter l , the score of the hypothesis is set to score of the candidate rule and then the hypothesis is inserted back to the sorted tree of hypotheses. The rule selection process continues using the next hypothesis as it now becomes the hypothesis with the highest score. Rule selection is repeated until all the examples in the active class are covered by the generated rules or until there are no more hypotheses with a positive score.

2.5.5 Optimistic Estimate Pruning

Optimistic estimate pruning is usually used with select late strategy, and they both together make a good solution for rule selection process of RILA for complex relational databases. If this pruning strategy is not applied, the number of hypotheses generated by select late strategy can become impractically large even for a moderate size data. The optimistic estimate pruning heuristic exploits the fact that we are interested in the n best solutions, if a hypotheses or its descendents cannot make it in the top n list, this branch is pruned by the algorithm.

2.5.6 Strategy Selection

Both strategies; the select early strategy and the select late strategy, have a different process for the rule selection and as a result have different benefits to offer according to different situations. Choice of any rule selection strategy mainly depends on the performance criteria of a learning task.

As the select early strategy activates rule selection after the generation of hypotheses for each level, therefore, the individual rules are smaller. If the performance criteria of the learning tasks require the individual rules to be small, the select early option is the best choice. The computational cost of the select early strategy is also low, which can be a factor in selecting this strategy.

The select late strategy requires more computational resources but as it postpones the activation of rule selection until all the hypotheses have been generated, it selects the hypotheses with maximum score, as rules. If the learning task has the performance criteria to select the rules with maximum score, select late strategy is a better choice. The select late strategy can be optimized for computational expense by collaborating it with the optimistic estimate pruning heuristic.

Let us take a look at the Table 1 which shows an example of training data to demonstrate a case where the select late strategy performs better than the select early strategy.

Table 1. Comparing of the Select Early and Select Late Strategies [8]

Attribute A	Attribute B	Attribute C	Class
a1	b1	c1	A
a1	b1	c2	A
a2	b2	c3	A
a3	b2	c3	A
a4	b1	c3	B
a5	b1	c3	B
a1	b2	c4	B
a1	b2	c5	B

According to the training data in Table 1, the two hypotheses generated in the first level for the class A should be:

Hypothesis 1: IF attribute A = a2 THEN class = A (support = 1)

Hypothesis 2: IF attribute A = a3 THEN class = A (support = 1)

When RILA is using the select early strategy, these two hypotheses are generated and simultaneously asserted as the new rules at the end of the first level. But if RILA is using the select late strategy, and the rule selection process is delayed until the end of the next level, the following hypothesis is generated in the next level:

Hypothesis 3: IF attribute B = b2 AND attribute C = c3 THEN class = A (support = 2)

It is evident that the hypothesis 3 alone covers the two examples which were covered by hypothesis 1 and hypothesis 2 which means it will have better generalization capacity as compared to the first two hypotheses. The reason for this is that it is supported by more number of training examples as compared to the first two hypotheses. When the selection is completed for all the levels select early strategy selects a total of eight rules with each rule having one condition on the other hand the select late strategy selects only four rules with each rule having two conditions.

This example depicts that RILA or any rule induction algorithm would generate better rules when all hypotheses for one class are evaluated together at the end instead of evaluating rules for every level separately. However, this way, the number of hypotheses can become much larger especially in case of large size data. The solution to this is pruning which has been described earlier.

CHAPTER 3

GRAPHICAL USER INTERFACE FOR RELATIONAL INDUCTIVE LEARNING ALGORITHM

This chapter describes the Java-based graphical user interface which is developed for the relational inductive learning algorithm RILA. RILA GUI consists of “rila.mygui” package which has relations with other classes by the help of MySingleton class. Package ”rila.mygui” has direct relations “rila.run”, “rila.gui” and “rila.support” packages. By the help of “rila.mygui” package user can interact with RILA fast and the outputs of RILA can be observed easily. It is shown that Java programming language is very useful in developing graphical software applications. Also Java programming language can meet the user interaction requirements. The whole design of RILA GUI has been made in Java programming language.

3.1 What is GUI?

A GUI is a visual interface to a program with graphical icons, visual indicators, etc. via which the user can interact with the program easily [18]. An efficient GUI should provide to the user a consistent appearance and a control mechanism such as menus, buttons, check boxes etc. which also provides sufficient information about their functionality for an efficient use. After user performs an action it should be predictable how the program will behave. Therefore labels and texts on the GUI should indicate components in a right way.

3.2 Why does RILA need a GUI?

Increase of the interaction between human and the computer, more imposes the need to build up a GUI for computer programs. Every program has its own needs according to their focus area. RILA is a machine learning algorithm and used for rule generating in a database. It is actually difficult to connect a database and selecting related tables or columns inside of the RILA code. On top of all RILA's advance searching mechanism requires some parameters. Without a GUI it is difficult to input new parameters in every try of the user for generating rules. Furthermore it is stressful to read the generated rules from the output console of IDE.

3.3 Why Java?

Java is an object-oriented language so provides the advantages of object-oriented programming. These benefits can be summarized as below [19]:

- **Simplicity:** Java objects look like real world object, which reduces the complexity of the program structure.
- **Modularity:** Modularity allows to be developed individual modules. Separate modules can be implemented by different teams.
- **Modifiability:** Any minor changes in any class do not affect the other classes until their members are not related.
- **Extensibility:** New features can be added easily by simple modifications in existing objects and by adding some new ones.
- **Maintainability:** Objects to be maintained can be found with ease and can be fixed separately.
- **Re-Usability:** Different programs can use the objects.

Furthermore, Java is a platform independent programming language. Java source code is compiled into byte code which can be run on Java Virtual Machine(JVM). There are different JVMs for all operating systems so it is not important where source code is compiled. JVM interprets the byte code for the operating system or machine on which it runs. (Figure 5)

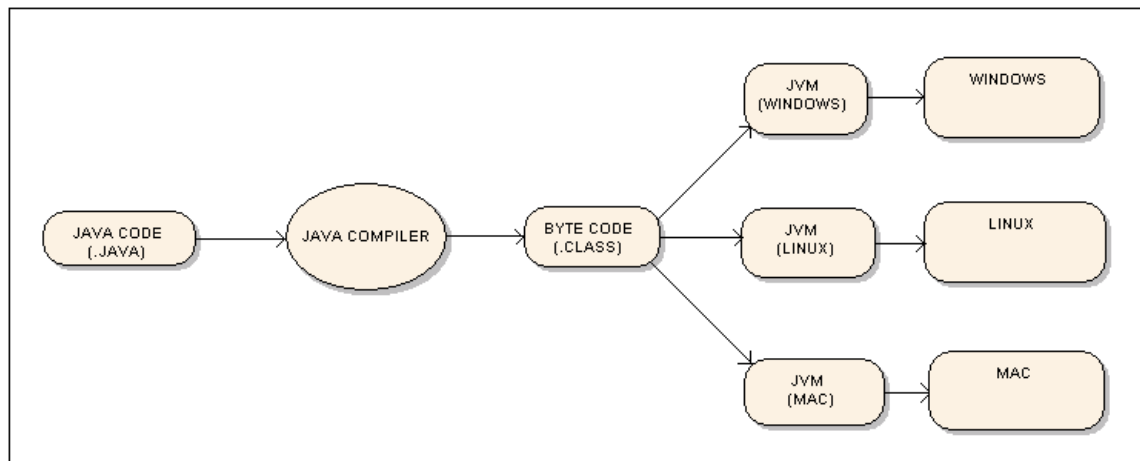


Figure 5. Java Byte Code and Platform Independence

Java is considered as an efficient tool, since it also provides additional features such as:

- *Swing* is a package in Java standard library which supplies a collection of GUI elements.
- Multi-threading makes available to execute different threads at the same time in the same Java program.

3.4 Software Architecture

RILA has several classes and a number of packages that surrounds these classes. One more package is added to RILA package hierarchy to develop its GUI and a fast library; [20] swing is added for this GUI package. Components of Java Swing library used in GUI package can be seen in the Table 2 [24].

Table 2. Components Used in GUI

COMPONENT NAME	DESCRIPTION
JLabel	Short text string display area for labeling components.
TextField	Single line text area.
PasswordField	It is a single line area. Original characters in this area is not shown.
TextArea	Multi-Line area that displays plain text.
ComboBox	An Editable field with a drop-down list.
CheckBox	A box which can be in a state of selected or deselected.
Button	It is a push button.
MenuBar	A bar that holds the menu.
Menu	A pop-up menu that contains menu items.
MenuItem	Any item in a menu.
Panel	It is a container.
EditorPane	It is a text component for editing text content in it.
TextPane	A text component that can be marked up with attributes
FileChooser	It helps user to be able to choose a file.
ScrollPane	Provides a scrollable view.
Table	It is used for editing and displaying 2D tables.
Tree	It helps to display a set of hierarchical data as an outline.

Table 3 shows new added classes and their packages. All the classes except Run and RilaDAO are in “rila.mygui” package. RilaDAO is used as a database access object for querying. HubCenterPanel uses RilaDAO to make a bar chart with the result of RilaDAO query. Run class is an API class. It provides the relation between GUI and the RILA. MySingleton class’ duty is catching outputs and some values and carrying them between GUI and RILA.

Table 3. New Added Classes

Class Name	Package Name
Entity	рила.mygui
GridPanel	рила.mygui
GuiMain	рила.mygui
HTMLPanel	рила.mygui
HubBarPanel	рила.mygui
HubCenterPanel	рила.mygui
HubPanel	рила.mygui
InputPanel	рила.mygui
JTreePanel	рила.mygui
LoginPanel	рила.mygui
MySingelton	рила.mygui
Settingsparam	рила.mygui
jEditorPane	рила.mygui
Run	рила.run
RilaDAO	рила.support

This relationship between RILA and GUI is established by the help of Run and MySingelton classes. Code segments from Run class and InputPanel class are presented as Figure 6 and Figure 7 to understand this relationship.

```
Run r = new Run();

r.runAlgorithm(minSupport, penaltyFactor, MaxSizeForRules,
ignoreUnknownValues, minF, maxNumHypothExtend,
conInfo.hubTable, conInfo.TargetAttribute,
conInfo.PrimaryKey, conInfo.ClassTable,
conInfo.dimTables, lateStrategy);
```

Figure 6. InputPanel Calls runAlgorithh Method of a Run Instance

API method runAlgorithm of Run class is called from the GUI class InputPanel with the parameters *minSupport*, *penaltyFactor*, *MaxSizeForRules*,

ignoreUnknownValues, minF, maxNumHypothExtend, hubTable, targetAttribute, PrimaryKey, ClassTable, dimensionTables and lateStrategy.

```
public void runAlgorithm(int minSupport, int penaltyFactor,
    int MaxSizeForRules, boolean ignoreUnknownValues, double minF,
    int maxNumHypothExtend, String hubTable, String targetAttribute,
    String PrimaryKey, String ClassTable,
    Collection<? extends String> dimensionTables,
    boolean lateStrategy)
{
    Globals.setMinSupport(minSupport);
    Globals.setPF(penaltyFactor);
    Globals.setMaxRuleSize(MaxSizeForRules);
    Globals.setIgnoreUnknownValues(ignoreUnknownValues);

    Globals.setMinFmeasure(minF);
    Globals.setNhypothesesToExtend(maxNumHypothExtend);

    Workbench.setOption("targetTable", hubTable);
    Workbench.setOption("targetAttr", targetAttribute);
    Workbench.setOption("primaryKey", PrimaryKey);
    Workbench.setOption("classTable", ClassTable);
}
```

Figure 7. Setting Part of *runAlgorithm* Method in Run Class

Method *runAlgorithm* sets these parameters for RILA with the code above and runs it with the code below in Figure 8. It can be said that all parameters come from GUI and are used in RILA.

```

try {
    if (lateStrategy == true) {
        new SelectLate(Globals.getDBAccess(), hubTable,
            targetAttribute,
            dimensionTables_,
            ClassTable).run();
    } else {
        new SelectEarly(Globals.getDBAccess(), hubTable,
            targetAttribute,
            dimensionTables_,
            ClassTable,
            maxNumHypothExtend).run();
    }
} catch (Exception e) {
    StackTraceElement[] err = e.getStackTrace();
    e.printStackTrace();
    JOptionPane.showMessageDialog(null, err, "ERROR",
        JOptionPane.ERROR_MESSAGE);
    return;
}

```

Figure 8. runAlgorithm Runs Select Late Strategy or Select Early Strategy

Figure 9 presents the running diagram of the code in Figure 8.

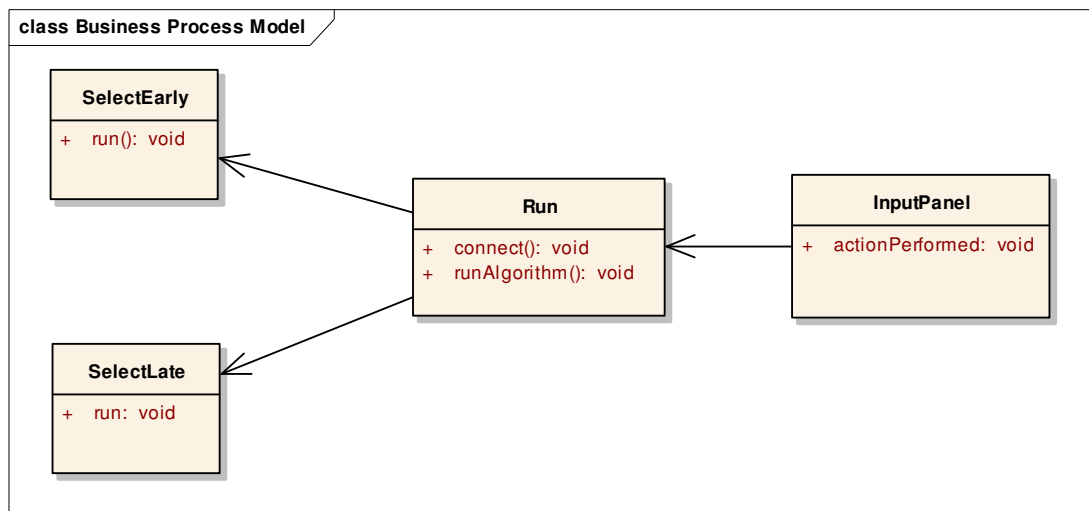


Figure 9. Process Diagram of Selecting Strategy

All the output of RILA is carried on MySingleton object. Thus MySingleton class has to be imported in some of the RILA packages. Table 4 shows these classes and their packages.

Table 4. Class That Imports *MySingleton* Class

Classes	Packages
RelationalILA	рила.алгоритмы
SelectLate	рила.алгоритмы
SelectEarly	рила.алгоритмы
MetaImporter	рила.relation
Hypothesis	рила.rule
RelationalRuleSet	рила.rule
Rule	рила.rule
Globals	рила.util
Run	рила.run

MySingleton class has some get / set functions to behave like an inter class between RILA and GUI. Here is the get / set functions:

- public String getMetaInfo()
- public void setMetaInfo(String meta)
- public String getRules()
- public void setRules(String rule)
- public String getRuleSetParameters()
- public void setRuleSetparameters(String str)
- public String getAppliedRules()
- public void setAppliedrules(String appl)
- public String getListofParameters()
- public void setListofParameters(String listOfParams)
- public String getLogofSearchingRule()

- public void setLogofSearchingRule(String str)
- public String getNumbers()
- public void setNumbers(String num)
- public String getListofgeneratedRules()
- public void setListofGeneratedRules(String str)

Above methods are used by the classes SelectEarly, SelectLate, RelationalILA, MetaImporter, Hypothesis, RelationalRuleSet, Rule, Globals, Run to set and by jEditorPane to get the final outputs. Figure 10 illustrated the diagram of above statement.

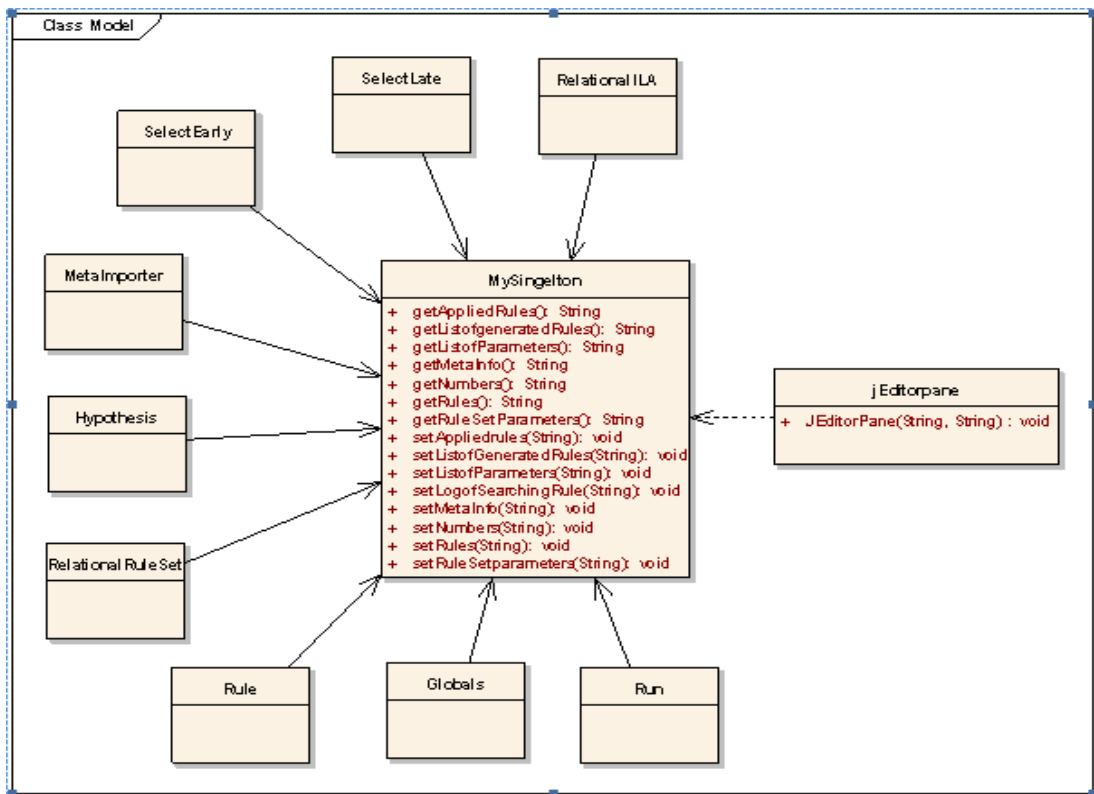


Figure 10. Class Model for *MySingleton* Related Classes

3.4.1 GUI Package

In Table 5 all the classes in this package and what they extends are presented.

Table 5. Classes and Their Extendings

Entity	-
GridPanel	JPanel
GuiMain	JFrame
HTMLPanel	JPanel
HubBarPanel	JPanel
HubCenterPanel	JPanel
HubPanel	JPanel
InputPanel	-
JTreePanel	JPanel
LoginPanel	-
MySingleton	-
Settingsparam	-
jEditorPane	JPanel

GuiMain class is the main class which is responsible for launching GUI. This class extends JFrame so every panel that is wanted to be displayed is added GuiMain's content pane. It is possible to write a panel for the needs and add the panel to RILA GUI anytime. An example is added with RILA GUI's screen shots to understand the classes of GUI package.

3.4.2 Example for the classes of GUI package

In this work RILA is tried on “gene” database which was first used in KDD Cup 2001 competition [21] and “ensembl” database which is used in ensembl project¹. Gene database is set on local mysql database server with a name “test”. MySQL Server 5.1 is installed on local machine and its url is **`jdbc:mysql://localhost:3306/test`**. In this database there are three tables: Composition, Gene, Interaction. If Select Early strategy is applied a table named “covered” is generated during process. Ensembl database does not let generating or dropping objects from outside. Thus only Select Late strategy is applied to ensemble database.

Its url is **`jdbc:mysql://ensemldb.ensembl.org:5306/aedes_aegypti_core_48_1b`**. After launching the code the first initialized form is GuiMain and LoginPanel is added to its contentpane. GUI is presented with its screen shots and some code segments in the package.

¹ EMBL - EBI and the Wellcome Trust Sanger Institute works on ensembl project to develop a software system which produces and maintains automatic annotation on selected eukaryotic genomes [25].

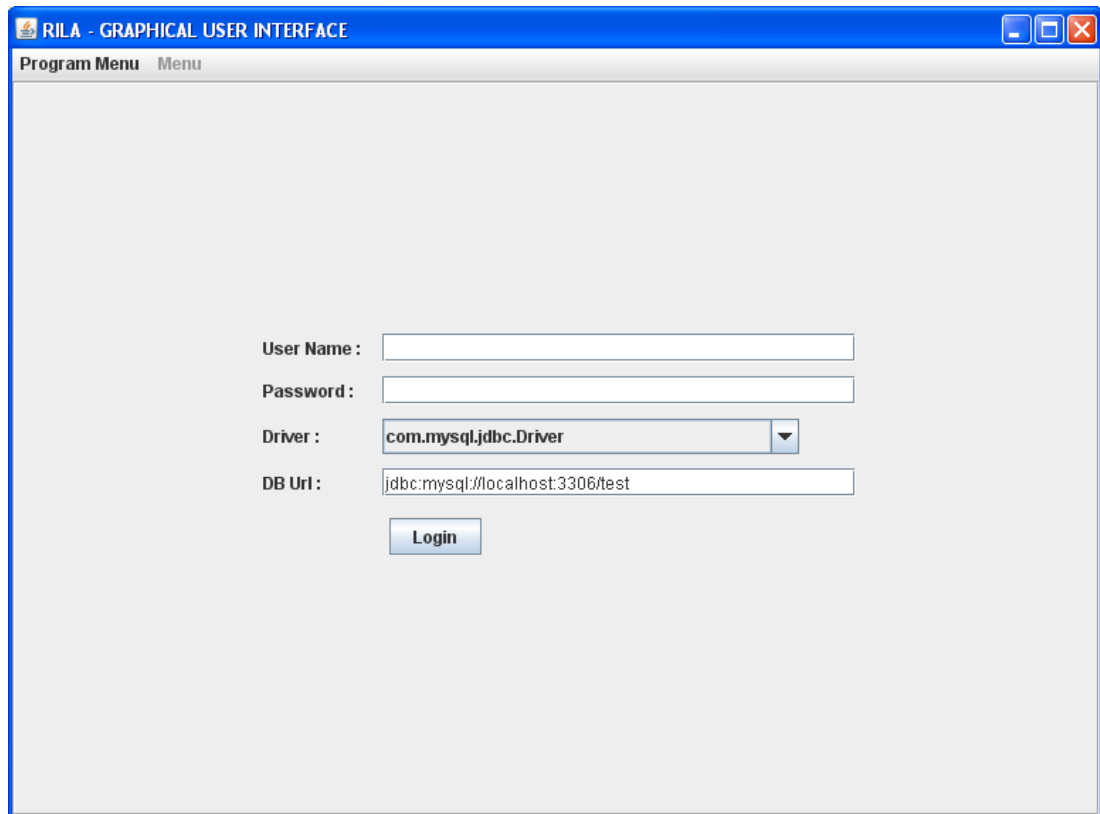


Figure 11. *LoginPanel.java*

To connect the database user name, password, database url must be correctly written in corresponding area. In the driver list there are class names of sql driver, mysql driver, derby driver. After pressing “Login” button if driver class is not found, it is handled by `ClassNotFoundException`. A connection is tried to establish by using “`DriverManager.getConnection(dbURL, userName, password)`” line. It is handled by `SQLException`. Look at Figure 12.

```

Connection connection = null;
try {
    Class.forName(driver);

    try {
        System.out.println("Connecting");
        connection = DriverManager.getConnection(dbURL,
            userName, password);
        System.out.println("Connected");

    } catch (SQLException e) {

        e.printStackTrace();

        return;
    }
} catch (ClassNotFoundException e) {

    e.printStackTrace();
    return;
}
}

```

Figure 12. Code Segment of *LoginPanel.java* for Connection a Database

After connection is set, *InputPanel* and *JTreePanel* are adding to the main frame. *JTreePanel* is located at the WEST and *InputPanel* is located at the CENTER(Figure 13).

```

JTreePanel treePanel = new JTreePanel(connection,
    connectionInfo, guiMain, percLabel);
guiMain.removeAll();guiMain.add(treePanel, BorderLayout.WEST);
guiMain.setBorder(null);
InputPanel inp = new InputPanel(treePanel,
    connectionInfo, guiMain);
treePanel.setInputPanel(inp);
MySingleton.getInstance().setTreePanel(treePanel);
MySingleton.getInstance().setInputPanel(inp);
MySingleton.getInstance().getMenu().setEnabled(true);
guiMain.add(inp, BorderLayout.CENTER);
guiMain.repaint();guiMain.revalidate();

```

Figure 13. BorderLayouts of *InputPanel* and *JTreePanel*

JTreePanel is a data presenter in hierarchical way. Database name is shown at the top. Below from database name, all of the table names are presented at the same level. When user clicks on any table name, column names relevant to the selected table name are opened. A mouseReleased event added to the elements of JTreePanel. In this event myPopupEvent method is triggered and in this method JPopupMenu object is generated and some menu items are added to the JPopupMenu object with respect to the EntityType of the tree node that the coordinate of the mouse is on it. In order to understand EntityType look at Figure 14.

```
public class Entity {  
  
    public enum EntityType(DATABASE, TABLE, COLUMN);  
    private String value;  
    private EntityType type;  
    public String getValue() {  
        return value;  
    }  
    public void setValue(String value) {  
        this.value = value;  
    }  
    public EntityType getType() {  
        return type;  
    }  
    public void setType(EntityType type) {  
        this.type = type;  
    }  
    @Override  
    public String toString(){  
        return this.value;  
    }  
}
```

Figure 14. Entity Class

Menu items of JPopupMenu with respect to the EntityType:

If the EntityType is TABLE:

- Select as hubtable

- Select as ClassTable
- Select as Dimension Table

If the EntityType is COLUMN:

- Select as PrimaryKey
- Select as Target Attribute

Below code provides the spread of menu items (Figure 15).

```
Entity entity = (Entity) obj.getUserObject();
if (entity.getType().equals(EntityType.TABLE)) {
    ActionListener actionListener = new PopupActionListener(
        entity);
    JPopupMenu popup = new JPopupMenu();
    JMenuItem item = new JMenuItem("Select as hubtable");

    item.addActionListener(actionListener);
    popup.add(item);
    popup.addSeparator();
    JMenuItem item1 = new JMenuItem("Select as ClassTable");
    item1.addActionListener(actionListener);
    popup.add(item1);
    popup.addSeparator();
    JMenuItem item2 = new JMenuItem("Select as Dimension Table");
    item2.addActionListener(actionListener);
    popup.add(item2);
    popup.addSeparator();
    popup.show(tree, x, y);
} else if (entity.getType().equals(EntityType.COLUMN)) {
    ActionListener actionListener = new PopupActionListener(
        entity);
    JPopupMenu popup = new JPopupMenu();
    JMenuItem item = new JMenuItem("Select as PrimaryKey");

    item.addActionListener(actionListener);
    popup.add(item);
    popup.addSeparator();
    JMenuItem item1 = new JMenuItem(
        "Select as Target Attribute");
    item1.addActionListener(actionListener);
    popup.add(item1);
    popup.addSeparator();
    popup.show(tree, x, y);
}
}
```

Figure 15. *Jtreepanel.java* Menu with respect to the Tree Node

For setting the parameters in the JPopupMenu, GUI uses Settingsparam get / set class. An inner class that implements ActionListener named PopupActionListener is used to set the parameters by using its actionPerformed method (Figure 16).

```
public void actionPerformed(ActionEvent actionEvent) {
    String action = actionEvent.getActionCommand();
    if (action.equals("Select as Target Attribute")) {
        params.setTargetAttribute(this.entity.getValue());
        inpPanel.setValues("ATTR", this.entity.getValue());
    } else if (action.equals("Select as hubtable")) {
        params.setHubTable(this.entity.getValue());
        inpPanel.setValues("HUB", this.entity.getValue());
    } else if (action.equals("Select as ClassTable")) {
        params.setClassTable(this.entity.getValue());
        inpPanel.setValues("CLASS", this.entity.getValue());
    } else if (action.equals("Select as PrimaryKey")) {
        params.setPrimaryKey(this.entity.getValue());
        inpPanel.setValues("KEY", this.entity.getValue());
    } else if (action.equals("Select as Dimension Table")) {
        if(!dimTableValue.contains(this.entity.getValue()))
        {
            dimTableValue.add(this.entity.getValue());
            inpPanel.setValues("DIM", this.entity.getValue());
        }
        else
        {
            JOptionPane.showMessageDialog(null,
                "You already add this table as dimension table.",
                "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

Figure 16 *actionPerformed* Method in the Inner Class *Popupactionlistener*

In the above code params value is an instance of Settingsparam class. Here is the class.

```
public class Settingsparam {  
  
    private String hubTable;  
    private String classTable;  
    private String TargetAttribute;  
    private String PrimaryKey;  
    private List<String> dimTable;  
    public String getHubTable() {...}  
    public void setHubTable(String hubTable) {...}  
  
    public String getClassTable() {...}  
    public void setClassTable(String classTable) {...}  
    public String getTargetAttribute() {...}  
    public void setTargetAttribute(String targetAttribute) {...}  
    public String getPrimaryKey() {...}  
    public void setPrimaryKey(String primaryKey) {...}  
    public List<String> getDimTable() {  
        return dimTable;  
    }  
    public void setDimTable(List<String> dimTable) {...}  
}
```

Figure 17. Settingsparam Class

Duty of *InputPanel* class is to give user an ability to set all parameters for running RILA. Here are the parameters:

- Penalty Factor
- Max # of Hypothesis to Extend
- Min F Value
- Max Size For Rules
- Minimum Support
- Select Late Strategy
- Ignore Unknown Values

Penalty Factor: For the evaluation of the hypothesis RILA needs a score. In the formula below tp represents the true positives and fn represents false negatives. For no sensitivity pf should set to 0 [22];[23].

$$score(h) = tp - pf * fn \quad (2.1)$$

Max # of Hypothesis to Extend: It determines the max number of hypothesis that can be extended in each level.

Min F Value: Minimum acceptable f measure value. By this value a hypothesis can be added to active hypothesis set. Thus it can be evaluated at the time of the rule selection process and new conditions also can be appended to this hypothesis.

Minimum Support: Generated hypothesis should cover this minimum number.

Select Late Strategy: If it is not checked, RILA uses Select Early Strategy

Ignore Unknown Values: If it is checked during rule generation RILA ignore unknown database values like “?”.

JTreePanel class sets the Hub table, Class Table, Dimension Tables, Primary Key, Target Attribute.

```

public void setValues(String componentName, String value) {
    if (componentName.equals("HUB")) {
        hubValue.setText(value);
    } else if (componentName.equals("CLASS")) {
        classValue.setText(value);
    } else if (componentName.equals("DIM")) {
        dimValue.append(value + "\n");
        dimScroll.setVerticalScrollBar().setValue(0);
    } else if (componentName.equals("KEY")) {
        keyValue.setText(value);
    } else if (componentName.equals("ATTR")) {
        attributeValue.setText(value);
    }
}
}

```

Figure 18. Code Segment of *InputPanel.java* (Values are Set from *JTreePanel*)

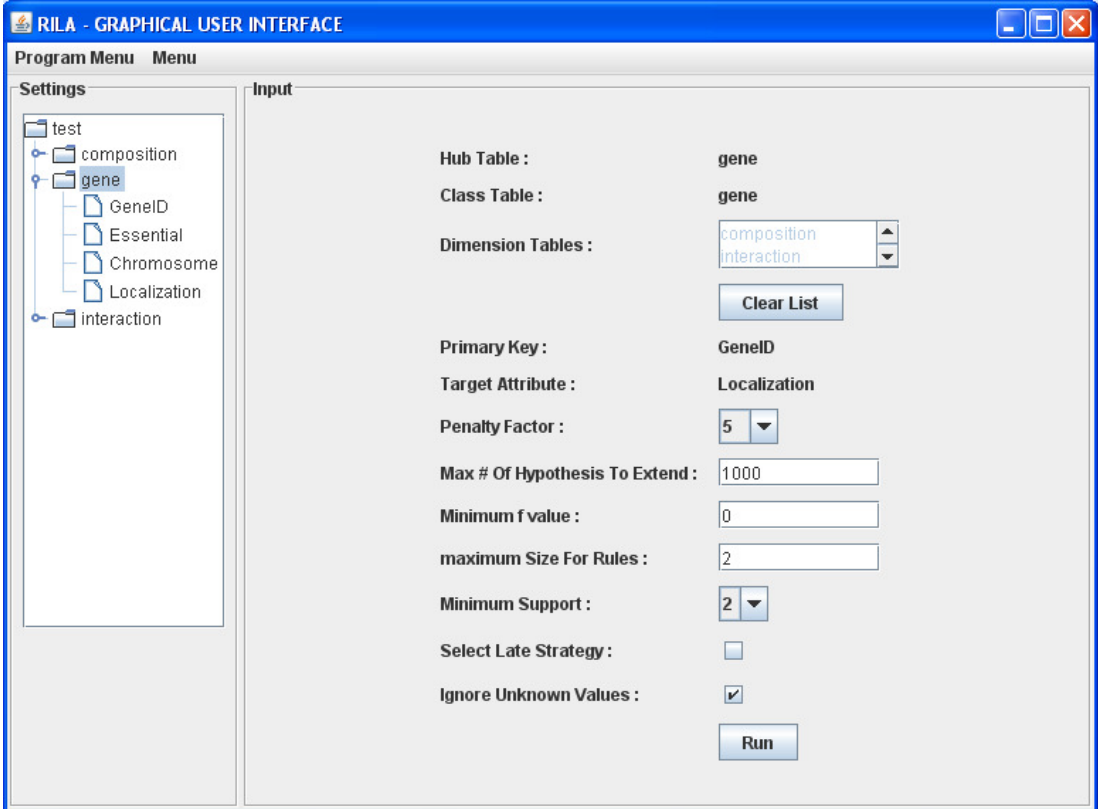


Figure 19. *InputPanel* ScreenShot

After pressing “Run” button RILA generates outputs. User can reach these outputs by using menu bar.

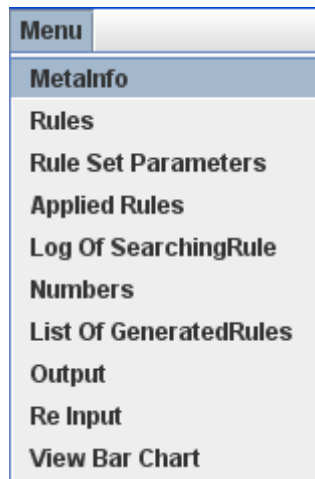


Figure 20 Menu in *GuiMain*

In “Menu” except “Re Input” and “View Bar Chart”, all of the menu items link to the relevant panel. By pressing “Re Input” user can set new values to RILA.”View Bar Chart” draws a bar chart for the tables in the connected database. All outputs except “Rules” which RILA generates are added to the appendix. “Rules” is given below for the parameters.

Hub Table: Gene

Class Table: Gene

Dimension Tables: Composition, Interaction

Primary Key: GeneID

Target Attribute: Localization

Penalty Factor: 5

Max # of Hypothesis to Extend: 1000

Minimum f Value: 0

Maximum Size for Rules: 2

Minimum Support: 2

Select Late Strategy: False

Ignore Unknown Value: True

Every output panel gives an opportunity to save output to a txt file (Figure 21).

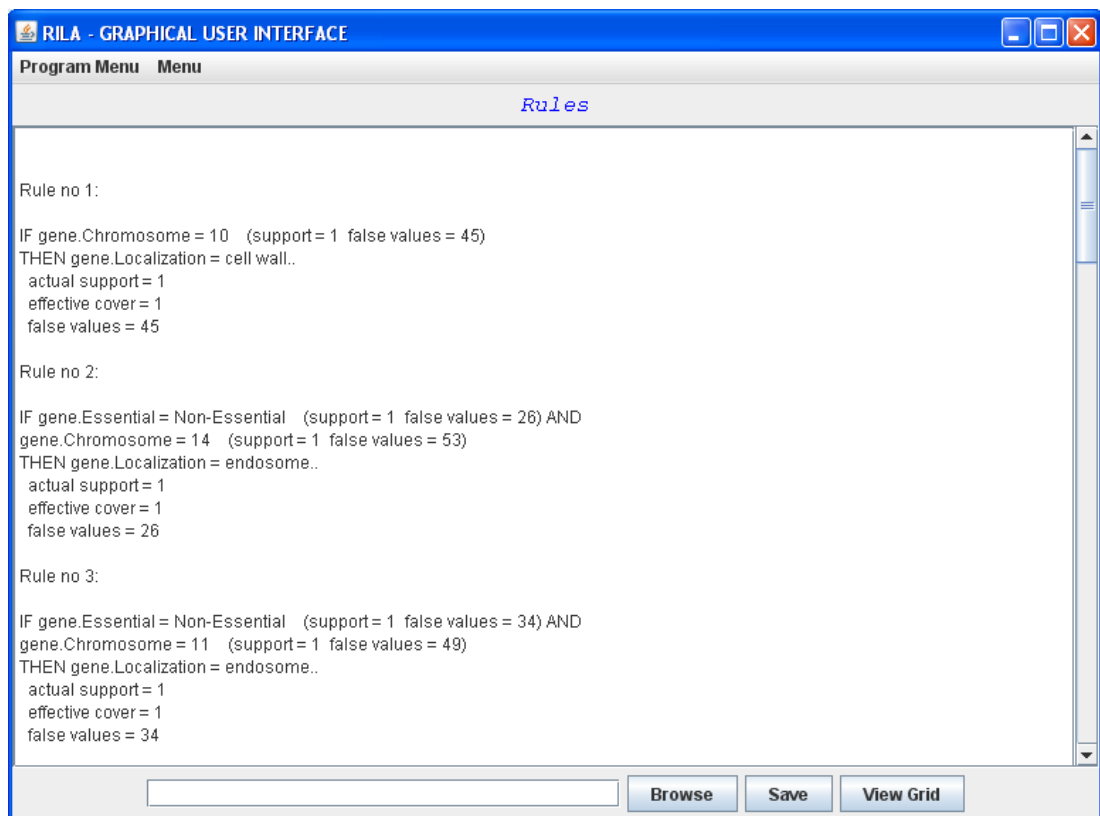


Figure 21. Rules Output

If the user presses “View Grid” button, rules are shown in a grid. GridPanel class is added to content pane of the frame. Look at Figure 22.

RILA - GRAPHICAL USER INTERFACE				
Program Menu Menu				
RULES	DECISION	ACCURACY%	Actual Support	FalseValues
gene.Essential = Essential AND gene.Chromosome = 9	gene.Localization = transport vesicles..	0,167	2	10
gene.Essential = Non-Essential AND gene.Chromosome = 1	gene.Localization = transport vesicles..	0,143	1	6
gene.Essential = Non-Essential AND gene.Chromosome = 14	gene.Localization = peroxisome..	0,074	2	25
gene.Essential = Non-Essential AND gene.Chromosome = 9	gene.Localization = peroxisome..	0,071	1	13
gene.Essential = Essential AND gene.Chromosome = 7	gene.Localization = nucleus..	0,792	19	5
gene.Chromosome = 1	gene.Localization = peroxisome..	0,083	1	11
gene.Essential = Non-Essential AND gene.Chromosome = 11	gene.Localization = endosome..	0,029	1	34
gene.Essential = Non-Essential AND gene.Chromosome = 14	gene.Localization = endosome..	0,037	1	26
gene.Chromosome = 10	gene.Localization = cell wall..	0,022	1	45
gene.Chromosome = 4	gene.Localization = lipid particles..	0,008	1	126
gene.Essential = Essential AND gene.Chromosome = 11	gene.Localization = golgi..	0,214	3	11
gene.Chromosome = 7	gene.Localization = extracellular..	0,012	1	79
gene.Chromosome = 10	gene.Localization = extracellular..	0,022	1	45
gene.Essential = Essential AND gene.Chromosome = 3	gene.Localization = ER..	0,286	2	5
gene.Essential = Non-Essential AND gene.Chromosome = 13	gene.Localization = integral membrane..	0,023	1	42
gene.Chromosome = 12	gene.Localization = integral membrane..	0,028	2	70

Print HTML

Figure 22. Rules in Grid Panel

If the user presses Print HTML, rules are shown in HTML table. Rules in HTML can be saved as html extension.

RILA - GRAPHICAL USER INTERFACE

Program Menu Menu

HTML

RULES	DECISION	ACCURACY%	Actual Support	False Values
gene.Essential = Essential AND gene.Chromosome = 9	gene.Localization = transport vesicles..	0.16666666666666666	2.0	10.0
gene.Essential = Non-Essential AND gene.Chromosome = 1	gene.Localization = transport vesicles..	0.14285714285714285	1.0	6.0
gene.Essential = Non-Essential AND gene.Chromosome = 14	gene.Localization = peroxisome..	0.07407407407407407	2.0	25.0
gene.Essential = Non-Essential AND gene.Chromosome = 9	gene.Localization = peroxisome..	0.07142857142857142	1.0	13.0
gene.Essential = Essential AND gene.Chromosome = 7	gene.Localization = nucleus..	0.7916666666666666	19.0	5.0
gene.Chromosome = 1	gene.Localization = peroxisome..	0.08333333333333333	1.0	11.0
gene.Essential = Non-Essential AND gene.Chromosome = 11	gene.Localization = endosome..	0.02857142857142857	1.0	34.0
gene.Essential = Non-Essential AND gene.Chromosome = 14	gene.Localization = endosome..	0.037037037037037035	1.0	26.0
gene.Chromosome = 10	gene.Localization = cell wall..	0.021739130434782608	1.0	45.0
gene.Chromosome = 4	gene.Localization = lipid particles..	0.007874015748031496	1.0	126.0
gene.Essential = Essential AND gene.Chromosome = 11	gene.Localization = golgi..	0.21428571428571427	3.0	11.0
gene.Chromosome = 7	gene.Localization = extracellular..	0.0125	1.0	79.0
gene.Chromosome = 10	gene.Localization = extracellular..	0.021739130434782608	1.0	45.0
gene.Essential = Essential AND gene.Chromosome = 3	gene.Localization = ER..	0.2857142857142857	2.0	5.0
gene.Essential = Non-Essential AND gene.Chromosome = 13	gene.Localization = integral membrane..	0.023255813953488372	1.0	42.0
gene.Chromosome = 12	gene.Localization = integral membrane..	0.027777777777777776	2.0	70.0

Browse Save

Figure 23. HTML Panel that Contains Rules Generated.

For the user generated rules become more meaningful with the bar chart. GUI draws the bar chart by using the getValues method of RilDAO class. This method contains a query which is illustrated in Figure 24.

```

public static List<Object> getValues(Connection connection, String table,
String columnName)
{
    List<Object> list = new ArrayList<Object>();
    String query = "select distinct(" + table + "."
        + columnName + ") rowName,count(" + table + "." + columnName + ")"
        + " RowNumber from " + table + " group by " + table + "." + columnName
        + " order by RowNumber desc";
    try {
        Statement stmt = connection.createStatement();
        ResultSet rst=stmt.executeQuery(query);

        while(rst.next()){
            String[] arr={rst.getString("rowName"),rst.getString("RowNumber")};
            System.out.println(">>>>>" +arr[0]+"----"+arr[1]);
            list.add(arr);
        }
    } catch (SQLException ex) {
        Logger.getLogger(RilaDAO.class.getName()).log(Level.SEVERE, null, ex);
    }
    return list;
}

```

Figure 24. *getValues* Method for Drawing Bar Chart

Figure 25 presents the bar chart which is drawn by the query executed in *getValues* method. In Figure 25 selected table is *gene* and selected column is *Localization*. Thus executed query is:

```

select distinct(gene.Localization) rowName,count(gene.Localization) RowNumber
from gene group by gene.Localization order by RowNumber desc

```

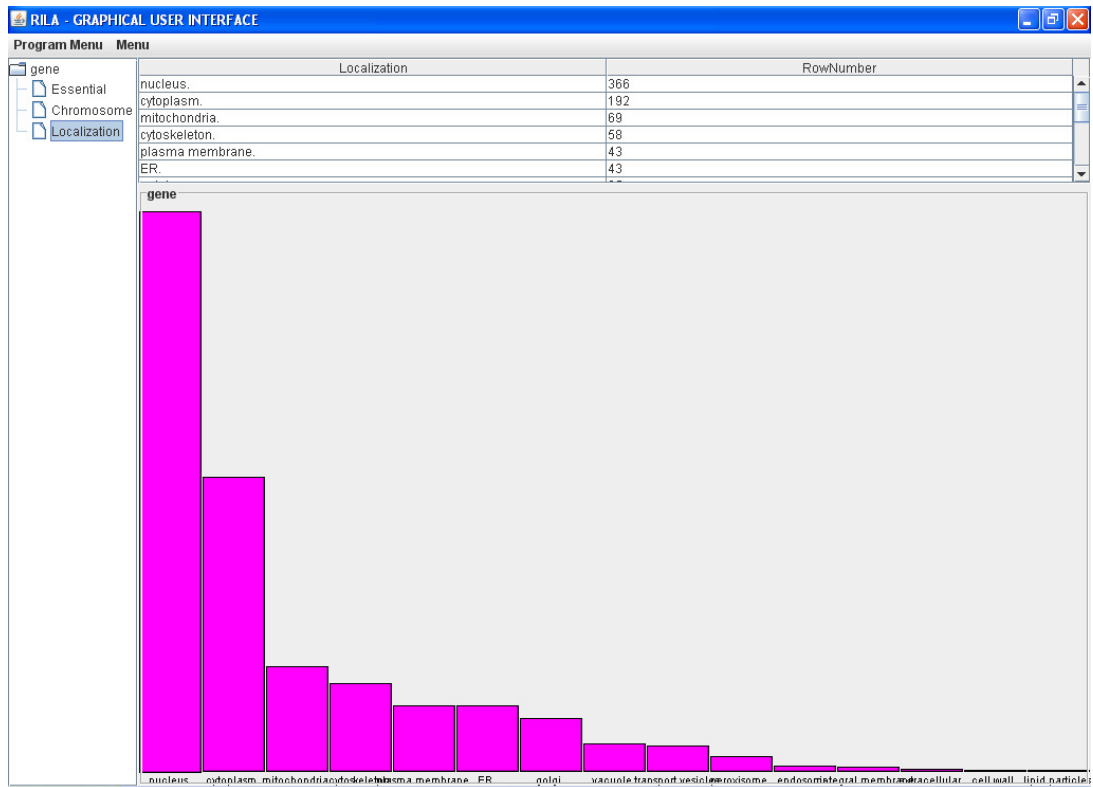


Figure 25. Bar Chart for *Gene Table and Localization Column*

3.4.3 Implementation

For development of the software multimedia PC was used with a 1.86 GHz Intel Pentium M CPU and 1.50 GB Ram. Operating system was Windows XP. In addition, MySQL Server 5.1 was installed on this computer. For testing the connectivity to other DBMS, MSSQL Server 2005 and Netbeans IDE 6.9.1 were installed on the computer which had 2.33 GHz Intel Core 2 Duo CPU and 3 GB Ram. Connectivity is tested locally.

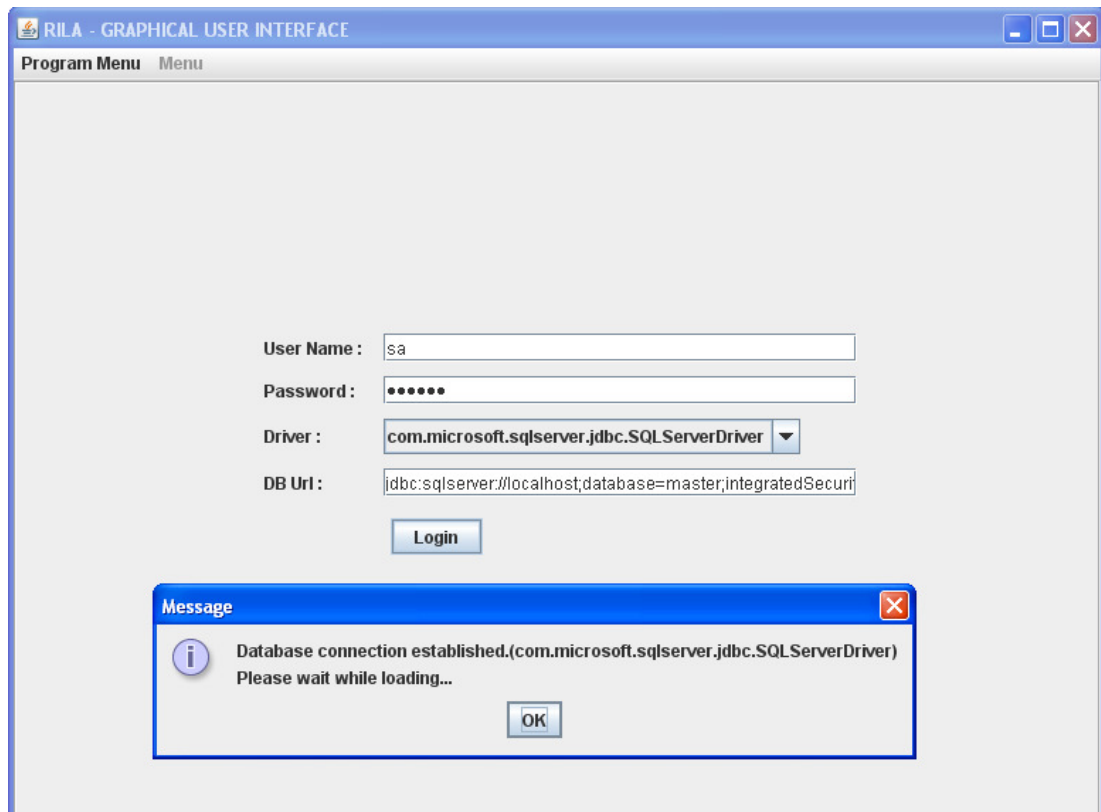


Figure 26. SQL SERVER Connectivity

For testing the platform independence of Java GUI, OpenSUSE 11.3 and Java Version of Netbeans IDE 6.9.1 were installed on the computer which had 2.33 GHz Intel Core 2 Duo CPU and 3 GB Ram. All of the platforms had JDK 1.6.0. No third party tool was used.

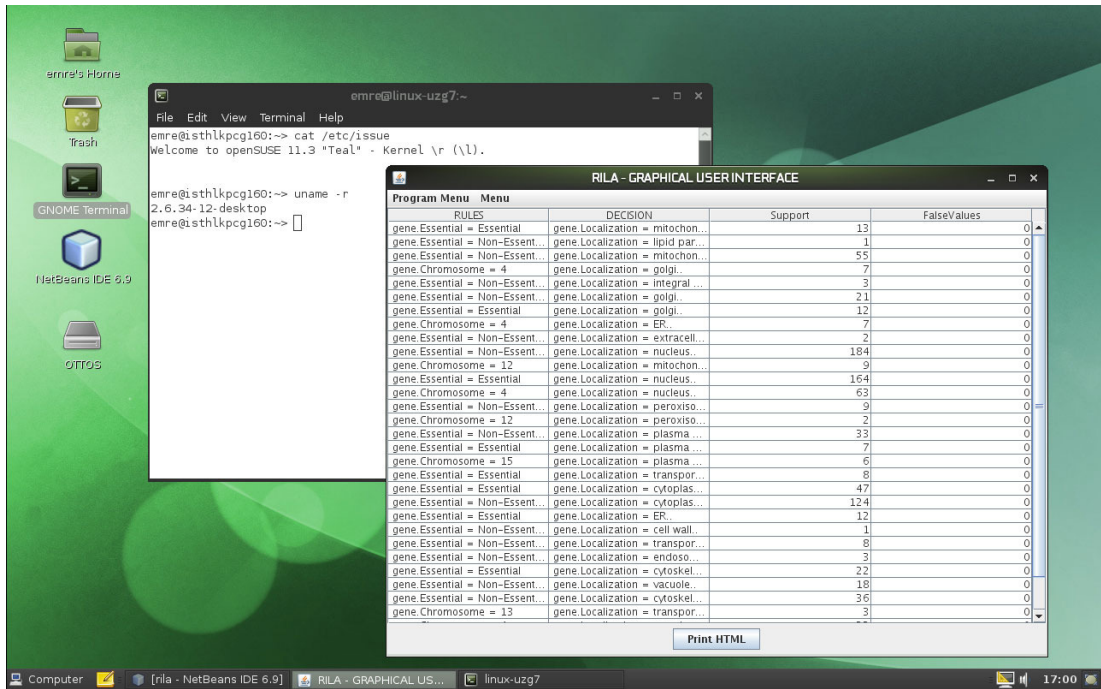


Figure 27. GUI on LINUX

CHAPTER 4

CONCLUSION

In chapter two the basic architecture, predecessors and the rule selection strategies of RILA have been explained. Two different strategies which are applied by RILA; select early and select late strategy, have been discussed which are very important to be considered for the efficient performance of the system according to the situation. In chapter three the main part of the work which was designing a user interface for RILA had been described.

In conclusion, RILA makes use of the SQL queries and directly uses the data in RDBS and collaborates with the DBMS. This way, it optimizes the query execution procedure. RILA is able to mine relational data stored in the relational database without requiring a local copy of the data. RILA is simple but powerful inductive algorithm used in the process of machine learning. However before this work it was difficult to work with RILA due to the lack of user-interface. Swing library has been used for making user interface graphical and more user friendly.

Designed GUI provided to users :

- A connection to jdbc database and hierarchy in the tables and columns get retrieved by this connection.

- A chance for selecting the “rule selection strategy” and a chance for setting parameters of the chosen strategy easily.
- Logs of the processes of the chosen algorithm.
- List of the rules with their support values.
- A bar chart which is designed according to the values in training set.

The design approach of GUI permits enlargement. There are some practical issues that can apply to the GUI. Showing the history of rules generated gives benefits to user. Also “select queries” executing ability without using any third party database tool provides user to work easy with RILA. Above ideas have considered as a future work.

REFERENCES

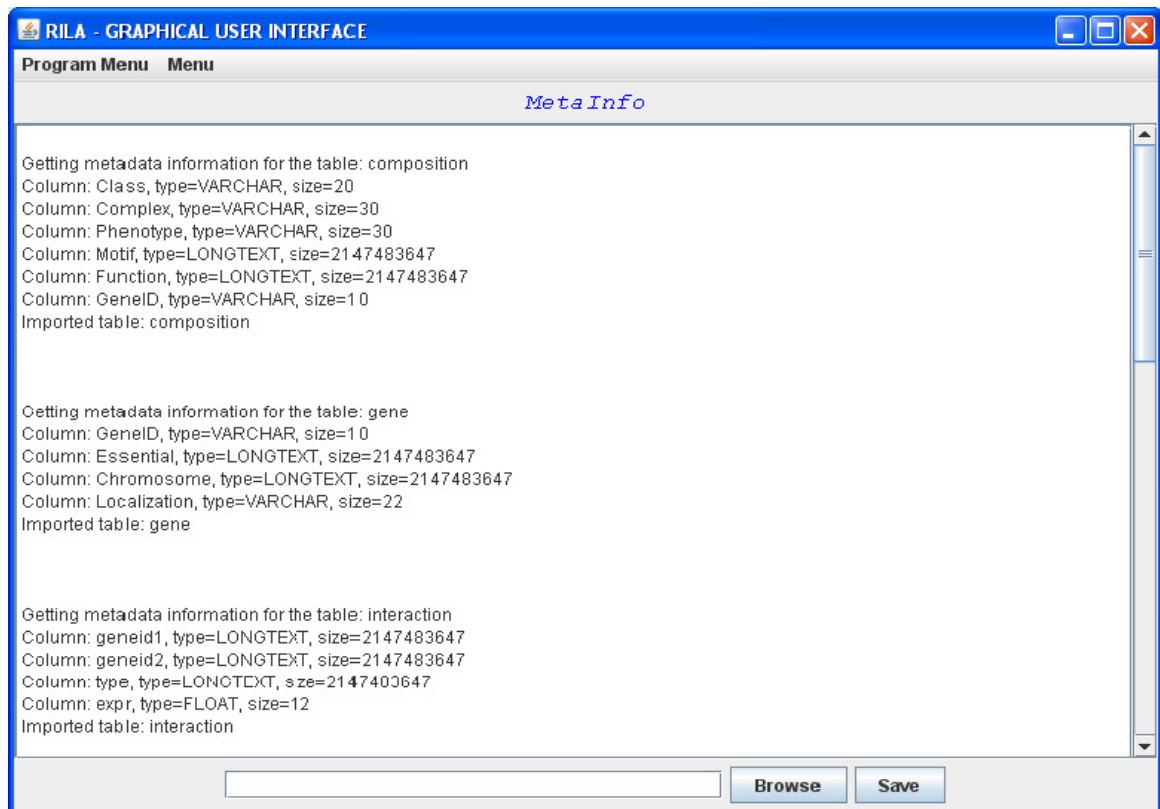
- [1] **ELMASRI, R., NAVATHE, S.B.** (1989), *Fundamentals of Database Systems*, Benjamin/Cummings, Redwood City, CA.
- [2] **SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S.** (2002), *Database System Concepts*, McGraw-Hill Companies Inc.
- [3] **MUGGLETON, S.** (1992), *Inductive Logic Programming*, Academic Press, London.
- [4] **BLOCKEEL, H., SEBAG, M.** (2003), Scalability and Efficiency in Multi-Relational Data Mining, *Special Issue on Multi-Relational Mining, SIGKDD Explorations*, 17-30. Vol.5
- [5] **DEOGUN, J. S.** et. al. (1997), Data Mining: Research Trends, Challenges, and Applications, *Proceedings of ACM CSC '95*, Kluwer Academic Publishers.
- [6] **HOLSHEIMER, M., SIEBES, A.** (1994), *Data Mining-The Search for Knowledge in Databases*, Report No. CS-R9406, CWI, Amsterdam, The Netherlands.
- [7] **FRAWLEY, W.J., PIATETSKY-SHAPIRO, G., MATHEUS, C.J.** (1991), *Knowledge Discovery in Databases: An Overview*, MIT Press, Cambridge.
- [8] **ULUDAĞ, M.** (2005), *Supervised Rule Induction for Relational Data*, Ph.D.Dissertation, Eastern Mediterranean University, Gazimağusa, Turkish Republic of Northern Cyprus.
- [9] **TOLUN, M.R.** et. al. (1999), ILA-2:An Inductive Learning Algorithm for Knowledge Discovery, *Cybernetics and Systems: An International Journal*, 609-628.
- [10] **TOLUN, M.R., ABU-SOUD, S.M.** (1998), ILA: An inductive learning algorithm for rule extraction, *Expert Systems with Applications*, 361-370.
- [11] **MANILLA, H., TOIVONEN, H.** (1997), Levelwise search and borders of theories in knowledge discovery, *Data Mining and Knowledge Discovery*, 241-258.

- [12] **KNOBBE, A.J., BLOCKEEL, H., SIEBES, A., VAN DER WALLEN, D.M.G.** (1999), Multi-Relational Data Mining, *Proceedings of the Benelearn'99 Conference*.
- [13] **AGRAWAL, R., SHIM, K.** (1996), Developing Tightly-Coupled Data Mining Applications on a Relational Database System, *Proceedings of the KDD 96 Conference*.
- [14] **DE RAEDT, L., VAN LAER, W.** (1995), Inductive Constraint Logic, *Proceedings of the Sixth Conference on Algorithmic Learning Theory*, Lecture Notes in AI, 80–94. Springer-Verlag.
- [15] **DZEROSKI S., LAVRAC, N.** (2001), *Relational Data Mining*, Springer-Verlag.
- [16] **Uludağ, M.** (1998), *Application of Rule Induction Algorithms to DNA Sequence Analysis*, M.Sc. Thesis, Middle East Technical University, Ankara, Turkey.
- [17] **LAVRAC N. et. al.** (2002), Adapting Classification Rule Induction to Subgroup Discovery, *Proceedings of the 2002 IEEE International Conference on Data Mining*, Maebashi City, Japan.
- [18] **DEITEL, H.M., DEITEL, P.J.** (2002), *Java How to Program*, Upper Saddle River, New Jersey.
- [19] **BLAHA, M.R., RUMBAUGH, J.R.** (2004), *Object-Oriented Modeling and Design with UML*, Prentice Hall.
- [20] **GUTZ, S., ROBINSON, M., VOROBIEV, P.** (1999), *Up to Speed With Swing*, Greenwich, Manning.
- [21] **CHENG, J. et. al.** (2002), KDD Cup 2001 Report, *SIGKDD Explorations*, 47-64.
- [22] **ULUDAĞ, M., TOLUN M.R., ETZOLD, T.** (2003), A Multi-Relational Rule Discovery System, *Proceedings of Eighteenth International Symposium on Computer and Information Sciences*, Antalya, Turkey.
- [23] **TOLUN, M.R., SEVER, H., ULUDAĞ, M.** (1998), Improved Rule Discovery Performance on Uncertainty, *Proceedings of the Second Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-98)*, Melbourne, Australia.
- [24] <http://download.oracle.com/javase/1.4.2/docs/api/javax/swing/package-summary.html> (2010)
- [25] <http://www.ensembl.org> (2010)

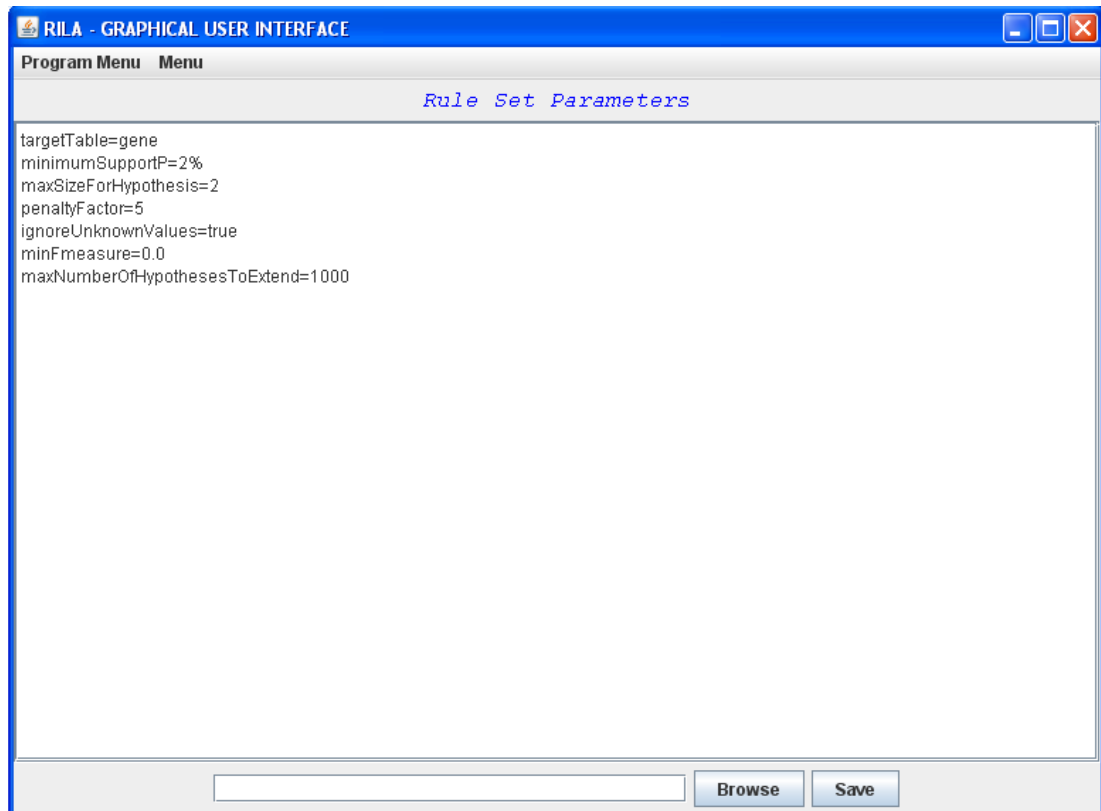
APPENDIX A

OUTPUTS OF RULES GENERATED BY RILA

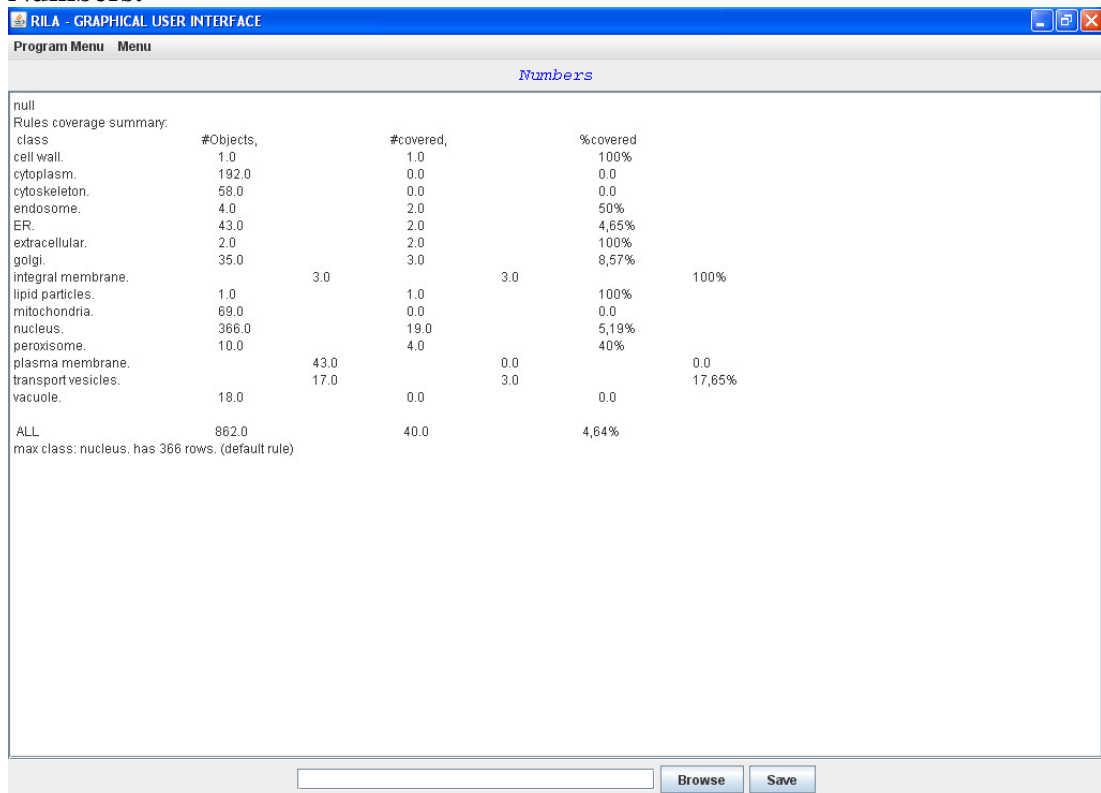
MetaInfo:



Rule Set Parameters:



Numbers:



Log of Searching Rules:

RILA - GRAPHICAL USER INTERFACE

Program Menu Menu

Log Of SearchingRule

Building hypotheses using table: gene
Building hypotheses using column: Essential
Building hypotheses using column: Chromosome
Built hypotheses using table: gene #of hypotheses=5

Generating rules for the class: ER.

Number of class examples: 43
Building hypotheses using table: gene
Building hypotheses using column: Essential
Building hypotheses using column: Chromosome
Built hypotheses using table: gene #of hypotheses=17

Generating rules for the class: extracellular.

Number of class examples: 2
Building hypotheses using table: gene
Building hypotheses using column: Essential
Building hypotheses using column: Chromosome
Built hypotheses using table: gene #of hypotheses=3
Rule selection out of 3 hypotheses

New rule generated:
IF
gene.Chromosome = 10 (support=1/2 false values=45/860 fmeasure=0,0417 score=0,2384)

THEN gene.Localization = extracellular.
actual support = 1
effective cover on the target table = 1
false values = 45

New rule generated:
IF
gene.Chromosome = 7 (support=1/2 false values=79/860 fmeasure=0,0244 score=0,0407)

THEN gene.Localization = extracellular.
actual support = 1
effective cover on the target table = 1
false values = 79

Browse Save

Applied Rules:

RILA - GRAPHICAL USER INTERFACE

Program Menu Menu

Applied Rules

gene.Chromosome = 10 (support=1/2 false values=45/860 fmeasure=0,0417 score=0,2384)

THEN gene.Localization = extracellular.
actual support = 1
effective cover on the target table = 0
false values = 0
True predictions: 0
False predictions: 0

IF
gene.Chromosome = 7 (support=1/2 false values=79/860 fmeasure=0,0244 score=0,0407)

THEN gene.Localization = extracellular.
actual support = 1
effective cover on the target table = 0
false values = 0
True predictions: 1
False predictions: 55

IF
gene.Chromosome = 4 (support=1/1 false values=126/861 fmeasure=0,0156 score=0,2683)

THEN gene.Localization = lipid particles.
actual support = 1
effective cover on the target table = 0
false values = 0
True predictions: 1
False predictions: 126

Default rule: gene.Localization = nucleus.
True predictions: 154
False predictions: 219

Total # of true predictions: 191
Total # of false predictions: 671

Accuracy and Predictions

Number of objects predicted: true=191, false=671, total=862

Accuracy=0,22157772621809746

Browse Save

List of Generated Rules:

RILA - GRAPHICAL USER INTERFACE

Program Menu Menu

List Of GeneratedRules

AND
gene.Chromosome = 9 (support=1/10 false values=29/852 fmeasure=0,05 score=-0,0702)

THEN gene.Localization = peroxisome.
actual support = 1
effective cover on the target table = 1
false values = 13
accuracy: 7,14%
class examples covered (support): 0%

Rule no 15:
IF
gene.Essential = Essential (support=2/17 false values=10/845 fmeasure=0,1379 score=0,117)
AND
gene.Chromosome = 9 (support=2/17 false values=28/845 fmeasure=0,0851 score=-0,0961)

THEN gene.Localization = transport vesicles.
actual support = 2
effective cover on the target table = 2
false values = 10
accuracy: 16,67%
class examples covered (support): 0%

Rule no 16:
IF
gene.Essential = Non-Essential (support=1/17 false values=6/845 fmeasure=0,0833 score=0,0233)
AND
gene.Chromosome = 1 (support=1/17 false values=11/845 fmeasure=0,069 score=-0,0063)

THEN gene.Localization = transport vesicles.
actual support = 1
effective cover on the target table = 1
false values = 6
accuracy: 14,29%
class examples covered (support): 0%
sum of support values: 40
sum of false covers: 475
Number of conditions in all rules: 26
Predicted average accuracy = 0,43

Browse Save

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Kapucu, Utku

Nationality: Turkish (TC)

Date and Place of Birth: 30 August 1980 , Karabük

Marital Status: Married

email: utkukapucu@gmail.com

EDUCATION

BSc. in Mathematics and Computer Science	Cankaya University	2002-2006
BSc. in Civil Engineering	Yildiz Technical University	1998-2001 (Not Completed)
High School	TED Karabük College	1994-1997

WORK EXPERIENCE

IT Expert	Prime Ministry Undersecretariat of Customs, Ankara	2006-Present
Software Developer	Bott Bilgisayar	September 2006 - November 2006
Software Developer	Basarsoft	June 2006 - August 2006

FOREIGN LANGUAGES

Fluent English, Basic German

HOBBIES

Playing Kemence and Baglama, Movies, Football.