Evaluating Knowledge Intensive Multi-agent Systems

Christos Dimou[1], Andreas L. Symeonidis[1,2], and Pericles A. Mitkas[1,2]

[1] Department of Electrical and Computer Engineering,
Aristotle University of Thessaloniki, 54 124, Thessaloniki, Greece
[2] Intelligent Systems and Software Engineering Laboratory,
Informatics and Telematics Institute/CERTH, 57 001, Thessaloniki, Greece

Abstract. As modern applications tend to stretch between large, ever-growing datasets and increasing demand for meaningful content at the user end, more elaborate and sophisticated knowledge extraction technologies are needed. Towards this direction, the inherently contradicting technologies of deductive software agents and inductive data mining have been integrated, in order to address knowledge intensive problems. However, there exists no generalized evaluation methodology for assessing the efficiency of such applications. On the one hand, existing data mining evaluation methods focus only on algorithmic precision, ignoring overall system performance issues. On the other hand, existing systems evaluation techniques are insufficient, as the emergent intelligent behavior of agents introduce unpredictable factors of performance. In this paper, we present a generalized methodology for performance evaluation of intelligent agents that employ knowledge models produced through data mining. The proposed methodology consists of concise steps for selecting appropriate metrics, defining measurement methodologies and aggregating the measured performance indicators into thorough system characterizations. The paper concludes with a demonstration of the proposed methodology to a real world application, in the Supply Chain Management domain.

1 Introduction

During the previous years, the exponential growth of the amount of available data has pushed user needs towards a more knowledge-demanding direction. Today's applications are, therefore, required to extract knowledge from large or distributed repositories of text, multimedia or hybrid content. The nature of this quest renders impossible to use traditional deterministic computing techniques. Instead, the challenge for open and dynamic solutions in knowledge discovery is addressed by various machine learning and other soft computing techniques. Most notably, Data Mining (DM) produces useful patterns and associations from large data repositories that can later be used as *knowledge nuggets*, within the context of any application.

Individual knowledge discovery activities, introduced by DM techniques are often required to be orchestrated, integrated and presented to end users in a unified way. Moreover, integrated knowledge should be exploited and embodied in autonomous software for learning purposes. Agent Technology (AT) is a promising paradigm that is suitable for modelling and implementing the unification of DM tasks, as well as for providing autonomous entity models that dynamically incorporate and use existing knowledge. Indeed, a plethora of agent-related solutions for knowledge-based systems can be found in the literature, and more specifically in the area of Agent-related data mining.

Despite the numerous related agent development methodologies, that deal with most of the steps of the development lifecycle, there is a remarkable lack of generalized evaluation methodologies for the systems in question. Evaluation of performance, a fundamental step of any development methodology, provides developers with countable, qualitative and verifiable attributes in an effort for better understanding the nature of a system at hand. Additionally, generalized and standardized evaluation procedures allow third parties to safely verify the acclaimed properties of systems or newly discovered scientific findings.

Existing evaluation approaches address either the DM algorithmic issues or the overall system performance. Both approaches come short in the case of AT and DM integration, due to the complex and dynamic nature of the produced systems. In the case of DM evaluation, focus is given on the statistical performance of individual techniques, in terms of precision and recall, ignoring the actual impact of the extracted knowledge to the application level. In the case of overall system evaluation, existing methods fail to deal satisfactorily with emergent agent behaviors that may not be known at design time.

In this paper, we present a generalized methodology for evaluating the performance of DM-enriched Multi-Agent Systems (MAS). A consice set of iterative methodological steps is presented, focusing on three fundamental evaluation aspects, namely the selection of a) metrics, b) measurement method, and c) aggregation methods. The proposed methodology is designed to assist developers as an off-the-shelf tool that can be integrated in the system development methodology. As an example of this incorporation, we briefly present Agent Academy, an open source platform for developing and training agents through data mining.

The remainder of this paper is organized as follows: in Section 2 we review the related bibliography; in Section 3, we present the proposed evaluation methodology in detail; in Section 4, we apply the evaluation guidelines to a real world demonstrator; finally, in Section 5 we summarize and provide future pointers.

2 Related Work

The proposed work draws from and extends related work in the direction of both AT-DM integration and evaluation of intelligent agents. In the field of AT-DM integration, there exist various related efforts that either use AT for enhanced DM or exploit DM advantages for incorporation in MAS. Most notably, in [1], inductive and deductive logic are combined for reasoning purposes in the field of customer care. In this work, deduction is used when complete information is available, whereas induction is employed to forecast behaviors of customers when the available information is incomplete. Inductive and deductive reasoning are also combined in [2], where logic terms of model data, information and knowledge are incorporated and processed by deductive agents. In [3], an integration of deductive database queries and inductive analysis on these queries and their produced knowledge is presented. Finally, [4] presents a unified methodology for transferring DM extracted knowledge into newly created agents. Knowledge models are generated through DM on the various levels of knowledge diffusion and are dynamically incorporated in agents. The iterative process of retraining through DM on newly acquired data is employed, in order to enhance the efficiency of intelligent agent behavior.

In our effort to study crucial performance issues for DM-enriched MAS, we present a literature review on intelligent agent evaluation. Although evaluation is an all encompassing term that may refer to either algorithmic performance or system performance, in this work we focus mainly on the latter. Indeed, within the context of DM and Machine Learning, a plethora of metrics and evaluation tools have been proposed, including precision, recall, F-measure, ROC Curves and fitness functions.

However, there is a remarkable lack of evaluation methodologies for intelligent systems that employ such algorithms for knowledge extraction purposes. Instead, researchers often have to devise their own ad-hoc metrics and experimental procedures. In fact, in some cases, the chosen parameters or input data are chosen so as to produce the best results for the -each time presented- method. Moreover, the findings are often supported by qualitatively arguments only, in favor of the proposed system and no debate with respect to its drawbacks is provided. Consequently, it is impossible for a third party to repeat the evaluation procedure and validate the quality of the proposed solution by concluding to similar results. The need for a generalized evaluation framework is, thus, evident.

In the literature, two general research approaches towards the direction of engineering aspects evaluation exist: a) bottom-up and b) top-down. The first approach represents the strong AI perspective on the problem, indicating that intelligent systems may exhibit any level of intelligence comparable to human abilities. Zadeh [5] argues that evaluating such systems is infeasible today, due to the lack of powerful formal languages for defining intelligence and appropriate intelligent metrics. The second approach represents the weak AI or engineering perspective, according to which intelligent systems are systems of increased complexity that are nevertheless well-defined in specific application domains, designed for solving specific problems. It is suggested that intelligent performance can be effectively evaluated after a concise decomposition of the problem scope and definitions of relative metrics and measurement procedures. Driven by the urging need to evaluate and compare existing or emergent applications, we adopt the top-down approach.

Ongoing domain-specific efforts for generalized metrics and evaluation methodologies exist in application fields, such as robotics and autonomic computing.

In robotics, evaluation efforts span from autonomous vehicle navigation (e.g. [6],[7]) to hybrid human-robot control systems (e.g. [8],[9]). In autonomic computing, emphasis is given to the quality assessment of the selected self-managing techniques [10]. Both fields provide usefull metrics and thorough methodological steps. However, neither of the above approaches are complete and mature nor do they provide us with relevant tools for the case of knowledge infusion in autonomous entities.

3 Evaluation Methodology

The proposed evaluation methodology serves as an off-the-shelf tool for researchers and developers in this field. Composed of theoretical analysis tools, it provides guidelines and techniques that can be used, adopted or extended for the application domain at hand. The methodology follows the top-down approach, mentioned earlier, and is therefore applicable to existing applications or applications that meet current agent oriented engineering concepts and follow the definitions for agent systems and DM terms provided in previous sections. Moreover, we only consider only observable agent behaviors that derive from the applied DM techniques. We therefore need to generalize existing DM metrics and introduce new intelligent performance metrics.

For establishing an evaluation framework that meets the above characteristics, we define:

- *Horizontal aspects*, the essential methodological steps, that if followed sequentially in an iterative manner, will comprise a complete evaluation methodology. The horizontal aspects of our methodology are:
  - *Definitions and theoretical background* on evaluation terms and relevant techniques.
  - *Theoretical representation tools* that can help designers chose what to measure, how to measure and how to integrate specific findings.
- *Vertical aspects* are specific techniques that may be part of any of the above horizontal aspects and deal with the following three terms[11]:
  - *Metrics* that correspond to system features to be measured.
  - *Measurement methods* that define the actual experimental procedure of assigning measurement values to the selected metrics.
  - *Aggregation* of the metric-measurement pairs in single characterizations for the system.

In the remainder of this section, we examine the above mentioned horizontal aspects in turn, analyzing each of their vertical aspects accordingly.

3.1 Definitions and Theoretical Background

The definitions of relevant terms and the corresponding theoretical background is of vital importance, in order to determine the scope and goals of evaluation. Any developer, before actually initiating his/her experiments, must have full grasp of what can and what cannot be evaluated. We, hereinafter, present relevant definitions and background theory with respect to: a) metrics, b) measurement methods, and c) aggregation.

Metrics. Metrics are standards that define measurable attributes of entities, their units and their scopes. Metrics are the essential building blocks of any evaluation process, since they allow the establishment of specific goals for improvement. A specific metric provides an indication of the degree to which a specific system attribute has met its defined goal. Deviation from the desired range of values indicates that improvement is needed in the related parts or modules of the system. With respect to a complete evaluation methodology, a metric is the answer to the question: "*What should I evaluate?*".

Measurement. Measurement is defined as "the process of ascertaining the attributes, dimensions, extend, quantity, degree of capacity of some object of observation and representing these in the qualitative or quantitative terms of a data language"[12]. Having selected the appropriate metrics, measurement is the next fundamental methodological step that systematically assigns specific values to these metrics. Typical measurement methods consists of experimental design and data collection. A measurement method is the answer to the question "*How should I perform the experimental evaluation?*".

Aggregation. Aggregation, or composition, is the process of summarizing multiple measurements into a single measurement is such a manner that the output measurement will be characteristic of the system performance. Aggregation groups and combines the collected measurements, possibly by the use of weights of importance, in order to conclude to atomic characterization for the evaluated system. For example, an evaluated system may perform exceptionally well in terms of response time metrics (timeliness), but these responses may be far from correct (accuracy). An aggregation process must weightedly balance contradicting measures and provide an overall view of parts or the whole of the system, within boundaries of acceptable performance. Aggregation is the answer to the question: "*What is the outcome of the evaluation procedure?*".

3.2 Theoretical Representation Tools

We next present a set of theoretical representation tools that aim to assist users throughout the designing of the evaluation procedure, by providing sets of options and guidelines for intelligent performance assessment.

Metrics. We introduce a metrics representation theoretical tool for metric categorization in the form of an acyclic directed graph. The graph is organized in layers or *views* of granularity from general to specific, as further explained below. A user may traverse the graph in a top-down manner and, depending on the choices made, he/she shall conclude to a set of suitable metrics. This graph is designed to be general, but also provides the option of extensibility for necessary domain specific metrics.

In the proposed approach, we organize a metrics graph into four views, as depicted in Figure 1:

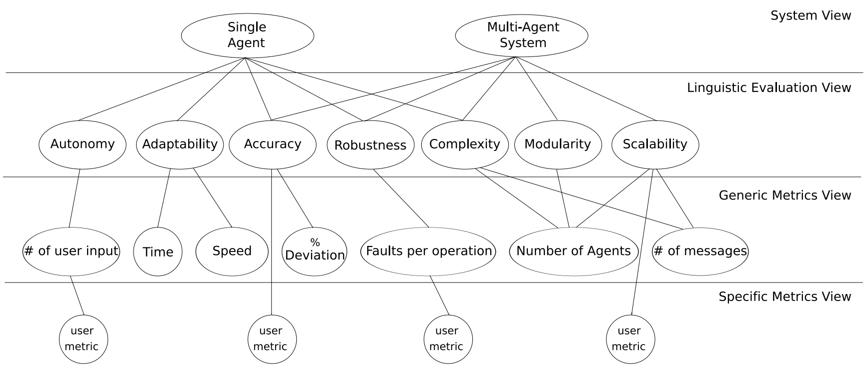

Fig. 1. Metrics graph

These views include:

1. *System view*: At the top-most level, the class of the application is selected. A user may chose between single-agent, multi-agent society and multi-agent competition, depending on the scope and focus of the evaluation effort.
2. *Linguistic evaluation view*: At this level, a user chooses the appropriate verbal characterizations of system aspects, such as accuracy, timeliness, robustness and scalability. These abstract high level characterizations exclude parts of the underlying metrics, while focusing on the aspects of interest to the evaluator.
3. *Generic metrics view*: This level consists of metrics that are general and independent of the application field, such as response time, number of agents and message exchange frequency. The user may either use directly these metrics or refine them by continuing to the next level.
4. *Specific metrics view*: The final level consists of metrics that are specific to the application field. These metrics are only defined by the user, since they are not known *a priori* to a generalized evaluation methodology. Newly defined metrics must conform to the metric definition and parametrization presented in the previous section. Finally, they must be appended to one of the graph nodes of the above levels with directed arcs.

After selecting the metrics from this graph, the user is requested to define a set of parameters for each metric, including the preferred scale of measurement and other attributes, such as frequency of measurement, time intervals etc.

Measurement Methods. Before implementing the actual measurement process, one must define the measurement method. Kitchenham [13] provides a categorization of measurement techniques, with respect to the types of properties employed and the nature of the experimental technique. Inspired by this work, we provide the following categorization:

1. Quantitative experiment: An investigation of the quantitative impact of methods/tools organized as a formal experiment
2. Quantitative case study: An investigation of the quantitative impact of methods/tools organized as a case study
3. Quantitative survey: An investigation of the quantitative impact of methods/tools organized as a survey
4. Qualitative screening: A feature-based evaluation done by a single individual who not only determines the features to be assessed and their rating scale but also does the assessment. For initial screening, the evaluations are usually based on literature describing the software method/tools rather than actual use of the methods/tools
5. Qualitative experiment: A feature-based evaluation done by a group of potential user who are expected to try out the methods/tools on typical tasks before making their evaluations
6. Qualitative case study: A feature-based evaluation performed by someone who has used the method/tool on a real project
7. Qualitative survey: A feature-based evaluation done by people who have had experience of using the method/tool, or have studied the method/tool. The difference between a survey and an experiment is that participation in a survey is at the discretion of the subject
8. Qualitative effects analysis: A subjective assessment of the quantitative effect of methods and tools, based on expert opinion
9. Benchmarking : A process of running a number of standard tests using alternative tools/methods (usually tools) and assessing the relative performance of the tools against those tests

Having selected the measurement method, one must thoroughly provide an experimental design prototype and a data collection procedure. As stated earlier, our methodology can only provide a set of guidelines that any designer may adjust to their specific application. A typical experimental design procedure must describe thoroughly the objectives of the experiments and ensure that these objectives can be reached using the specified techniques.

The last step of the measurement methodology is to carry out the data collection process. Here, the basic guidelines for the designer to follow are to ensure that the data collection process is well defined and monitor the data collection and watch for deviations from the experiment design.

Aggregation. Following the collection of measurement values and the construction of metric-measurement pairs, the problem of aggregation arises. In the evaluation process, aggregation occurs naturally in order to summarize the experimental findings into a single characterization of the performance, either of single modules, or the system as a whole. In the case of the metrics graph of the proposed methodology, after having the measurements collected, the user must traverse the graph in a bottom-up manner. From the *specific metrics view* and the *general metrics view*, he/she must proceed upwards and, at each view, apply aggregation techniques to provide single characterizations for every parent node.

It is apparent that a natural method for combining diverse and heterogeneous measurement information and linguistic characterizations is needed. We argue that *fuzzy aggregation* provides us with the appropriate natural functionality for this purpose. The term *natural* refers to the ability of the evaluator to express the evaluation findings in a manner that is coherent to their natural language. In other words, the fuzzy aggregation process translates the problem of combining numerical, ordinal or other measures into a collection of verbal characterizations for the system performance.

The proposed fuzzy aggregation method consists of four steps:

1. *Define weights in the metrics graph.* This process determines the importance of each node in the metrics graph with respect to the overall system performance. This decision relies heavily on the application domain as well as the requirements of each application. Hence, the determination of the weights may occur either a) semi-automatically, in case historical data on the importance of each node are available, possibly by an expert system, or b) directly by an expert user, the system designers in most cases.
2. *Define corresponding fuzzy scales for each metric.* The next step deals with the definition of fuzzy scales for the selected metrics. Fuzzy scales are defined by ordinal linguistic variables, such as *low*, *moderate*, *high* and membership functions that map numerical values to the above variables. Having the scales defined, one may already have scales for *natural* characterizations of performance, such as *high response time* or *moderate accuracy*, with respect to desired values.
3. *Convert actual measurements to fuzzy scales.* The conversion is a simple import of the selected measurements to the membership functions defined in the previous step.
4. *Apply a corresponding fuzzy aggregation operator at each view of the graph.* A wide variety of fuzzy aggregation operators exists [14], which can be categorized in:
   - Conjunctive operators, that perform aggregation with the logical "and" connection.
   - Disjunctive operators, that perform aggregation with the logical "or" connection.
   - Compensative operators, which are comprised between minimum and maximum, such as mean or median operators.
   - Non-compensative operators, that do not belong to any of the above categories, such as symmetric sums.

Theoretical Tools: Summary. In Table 1, we summarize the required methodological steps with respect to the theoretical tools, which take place at the

Table 1. Summarization of methodological steps

1. Traverse metrics graph and select metrics
2. Provide domain specific metrics (optionally)
3. Determine metrics parameters
4. Specify measurement method and parameters
5. Execute experiments
6. Define weights in the graph
7. Define fuzzy scales and convert measurements accordingly
8. Select and apply aggregation operators on the collected measurements

evaluation process of a development methodology. In section 4, we present a real world case study on which the presented methodology is thoroughly applied.

4 A Real World Demonstrator

For validating the proposed methodology, we have selected Supply Chain Management (SCM) as a representative domain for testing agents that utilize DM techniques. We have implemented an SCM agent under the name Mertacor that has successfully participated in past Trading Agent SCM Competitions. Mertacor combines agent features with DM techniques. In the remainder of this section, we provide an overview of the SCM domain, the competition scenario and Mertacor's architecture. We conclude by applying the proposed evaluation methodology to different implementations of Mertacor.

Supply Chain Management and TAC Competition. SCM tasks comprise the management of materials, information and finance in a network consisting of suppliers, manufacturers, distributors and customers. SCM strategies target at the efficient orchestration of the sequence of tasks, from raw materials to end-user service. Traditional SCM relied heavily on rigid and predefined contracts between participating parties. However, the need for dynamic configuration of the supply chain, as indicated nowaydays by global markets, became imperative. Modern SCM approaches focus on the integration, optimization and management of the entire process of material sourcing, production, inventory management and distribution to customers.

Mertacor Architecture. Mertacor, as introduced in [15], is an agent that has successfully participated in the Trading Agent Competitions (TAC) [16]. The architecture of Mertacor consists of four cooperating modules:

1. the *Inventory Module*(IM). Mertacor introduces an assemble-to-order (ATO) strategy, which is a combination of two popular inventory strategies, namely make-to-order and make-to-stock.

2. the *Procuring Module*(PM). This module predicts future demands and orders affordable components, balancing between cheap procurement and running needs in the assembly line.
3. the *Factory Module*(FM). This module constructs assembly schedules and provides the Bidding Module with information on the factory production capacity, based on simulation of customer demand for the next 15 game days.
4. the *Bidding Module*(BM). This module attempts to predict a winning bid for each order, by performing DM on logs of past games.

Mertacor's core integrates this modules into a transparently robust unit that handles negotiations with both customers and suppliers. This architecture provides flexibility and extensibility, permitting the application of Mertacor's strategy to other real-life SCM environments.

Evaluating Mertacor's Performance. In the remainder of this section, we apply the proposed evaluation methodology to various implementations of Mertacor. In our effort to assess the impact of DM in Mertacor's performance, we require that the experiments are planned is such way that deals with both DM algorithmic-specific efficacy and their impact on overall agent performance. We follow the methodological steps defined in Table 1.

Step 1: Traverse metrics graph and select metrics Starting at the *System view*, we select the *Single Agent* node and corresponding path. This choice is attributed to the nature of auctioning environments; we, being the developers of Mertacor, have complete control only on the agent's executing threat and observe the auctioning world only through Mertacor's perspective. We, therefore, need to focus on performance aspects that exclusively deal with this single agent.

At the *Linguistic Evaluation View*, we select the linguistic metrics of *Accuracy* and *Timeliness*. Indeed, from our experience in SCM auctions, these three characteristics are the most fundamental, as the outcome of each auction is heavily dependent on the deviation of the forecasted bid, the timely deliver of the bid and the ability of the agent to adapt in dynamic environments, respectively.

At the *Generic Metrics View* we only select *Time*, as standard metric for *Timeliness*. The rest of the metrics are domain specific and are, therefore, defined in the next methodological step.

Step 2: Provide domain specific metrics Metrics for *Accuracy* should directly refer to DM related performance, since the outcome of the application of DM is directly related to the selected bid. For this purpose, we have selected the *Correlation Coefficient* (cc), the *Relative Absolute Error* (RAE) and the *Root Mean Square Error* (RMSE) metrics.

An instance of the metrics graph for this evaluation effort is depicted in Figure 2

Step 3: Determine metrics parameters We now continue by defining the scale of each metric. For the two linguistic metrics, *Accuracy* and *Timeliness*, we define

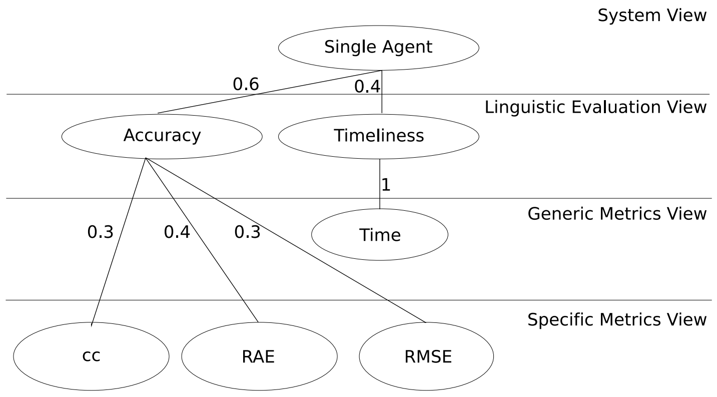

Fig. 2. Resulted metrics graph for Mertacor evaluation

the corresponding fuzzy scales in *Step 7* of the methodology. For the generic and specific metrics, we provide the following scales:

1. CC: The correlation coefficient is the degree at which the forecasted bid and the resulted price are correlated. The *cc* lies in the [-1,1] interval.
2. RAE: The Relative Absolute Error is a percentage indicator for the deviation of the above mentioned variables.
3. RMSE: The Root Mean Square Error is another well-known DM metric for the above mentioned variables.
4. Time: In TAC SCM auctions, bids are normally submitted just before the end of each predefined auction interval. One could argue that, since this time constraint exists, all agents have a time barrier to bid and therefore all bidding calculation procedures should be characterized either as successful or failed. In that context, timeliness is only a binary metric that provides no further performance indication. However, due to the modular architecture of Mertacor, the earliest possible decision on the bid, allows the agent to perform other game-related tasks in this interval. We therefore define *Time* as the time interval between the first call of the related bidding API function and the determination of the bidding value, in milliseconds.

Step 4: Specify measurement method and parameters Estimation of the winning price of the bids can be modeled as a regression problem, where the desired output is the agent's bidding price for clients' RFQs and the inputs are the parameters related to the bid that are known to the agent. The initial set of attributes considered are the demand (Total PCs requested each day), the demand in the product's market range, the due date of the order, the reserve price of components, and the maximum and minimum prices of same type PCs sold in the last days (2 previous days for maximum 4 for minimum), as shown in Table 2.

Table 2. Set of SCM auction attributes for DM

Attribute description	Attribute name
Demand (Total PCs requested the day the RFQ was issued)	demandAll
Demand in the product's market range	demandRange
Duedate	dueDate
Reserve price	reservePrice
Maximum price of PCs of same type sold in the last 1 day	max1
Maximum price of PCs of same type sold in the last 2 days	max2
Minimum price of PCs of same type sold in the last 1 day	min1
Minimum price of PCs of same type sold in the last 2 days	min2
Minimum price of PCs of same type sold in the last 3 days	min3
Minimum price of PCs of same type sold in the last 4 days	min4
Winning price of the bid	price

Finally, for training purposes, four different classification (regression) and two meta-classification schemes were applied, in order to decide on the one that optimally meets the problem of predicting the winning bid of an order:

1. Linear Regression
2. Neural Networks
3. SMOreg (Support Vector Machines)
4. the M5' algorithm
5. Additive Regression
6. Bagging

Step 5: Execute experiments In order to experiment on the data with a variety of training techniques and algorithms, the WEKA [17] was selected, providing with a wide range of filters for pre-processing, model evaluation, visualization and post-processing. The results of the experimental procedures are presented in Table 3.

Table 3. Results of experiments

Algorithm	CC	RAE (%)	RMSE	Time(ms)
Linear Regression	0.93	28.99	90.17	108
Neural Networks	0.93	32.91	94.69	111
Support Vector Machines	0.93	26.47	89.08	157
M5'	0.95	22.77	61.09	140
Additive Regr.	1.00	3.21	22.12	192
Bagging	0.98	14.89	52.02	201

Step 6: Define weights in the graph This step requires a subjective, expert-initiated attribution of weights to the corresponding edges of the metrics graph. Driven by our experience in the field, we assign a higher weight to *Accuracy* (0.7) and a lesser weight to *Timeliness* (0.3). The corresponding weights are illustrated in Figure 2

Step 7: Define fuzzy scales and convert measurements accordingly We provide the following fuzzy sets for the selected metrics:

- Fuzzy variables *very low*,*low*,*medium*,*high* and *very high* for the *RAE* and *RMSE* metrics
- Fuzzy variables *low* and *high* for the *CC* metric
- Fuzzy variables *low*, *medium* and *high* for the *Time* metric

We provide fuzzy variables *very low*,*low*,*medium*,*high* and *very high* for the *CC*, *RAE*, *RMSE* and *Time* metrics

Step 8: Select and apply aggregation operators on the collected measurements The final step of the methodology consists of the application of the selected aggregation method. As described in [14], the application of weighted operators result into a single characterization for every linguistic metric. After summarizing the results it can be seen that *Additive Regression* exhibit the best performance for all data subsets, as it balances between large accuracy and small time responses.

5 Conclusions and Future Work

As the number of application that integrate DM and AT increase, the need for assessing the overall system performance is imperative. In this paper, we have presented a generalized methodology for evaluating agents and MAS that employ DM techniques for knowledge extraction and knowledge model generation. The proposed methodology comprises a set of concise steps that guide an evaluator through the evaluation process. A novel theoretical representation tool introduces a metrics graph and appropriate selection guidelines for measurement and aggregation methods. A real world DM-enriched agent in the field of Supply Chain Management has used to demonstrate the applicability of the proposed methodology. Future work in this direction include the specification of a unique metrics ontology for the proposed metrics representation graph and the expansion of the graph with a complete set of real world metrics, borrowed either from the software engineering discipline or existing, ad-hoc efforts in intelligent systems evaluation. Finally, the proposed methodology must be thoroughly tested in a number of diverse and representative case studies.

References

1. Boris Galitsky and Rajesh Pampapathi: Deductive and inductive reasoning for processing the claims of unsatisfied customers. In Paul W. H. Chung, Chris J. Hinde, and Moonis Ali, editors, *IEA/AIE*, volume 2718 of *Lecture Notes in Computer Science* Springer (2003) 21–30

2. Alvaro A. A. Fernandes: Combining inductive and deductive inference in knowledge management tasks. In *DEXA '00: Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, page 1109, Washington, DC, USA (2000) IEEE Computer Society
3. Bob Kero, Lucian Russell, Shalom Tsur, and Wei-Min Shen: An overview of database mining techniques. In *KDOOD/TDOOD* (1995) 1–8
4. Andreas L. Symeonidis and Pericles A. Mitkas: *Agent Intelligence Through Data Mining*. Springer Science and Business Media (2005)
5. Lotfi A. Zadeh: In quest of performance metrics for intelligent systemsa challenge that cannot be met with existing methods. In *Proc. of the Third International Workshop on Performance Metrics for Intelligent Systems (PERMIS)*, 13-15 August 2002
6. Andrew L. Nelson, Edward Grant, and Thomas C. Henderson: Competitive relative performance evaluation of neural controllers for competitive game playing with teams of real mobile robots. In *Proc. of the Third International Workshop on Performance Metrics for Intelligent Systems (PERMIS)*, 13-15 August 2002
7. Noah Zimmerman, Craig Schlenoff, Stephen Balakirsky, and Robert Wray: Performance evaluation of tools and techniques for representing cost-based decision criteria for on-road autonomous navigation. In *Proc. of the Third International Workshop on Performance Metrics for Intelligent Systems (PERMIS)*, 3-15 August 2002
8. Jean Scholtz, Brian Antonishek, and Jeff Young: Evaluation of human-robot interaction in the nist reference search and rescue test arenas. In *Proc. of the Fourth International Workshop on Performance Metrics for Intelligent Systems (PERMIS)*, 13-15 August 2002
9. Jennifer L. Burke, Robin R. Murphy, Dawn R. Riddle, and Thomas Fincannon: Task performance metrics in human-robot interaction: Taking a systems approach. In *Proc. of the Fourth International Workshop on Performance Metrics for Intelligent Systems (PERMIS)*, 13-15 August 2002
10. Markus C. Huebscher and Julie A. McCann: Evaluation issues in autonomic computing. In *Proceedings of Grid and Cooperative Computing Workshops (GCC)* (2004) 597–608
11. Timothy K. Shih. Evolution of mobile agents. In *Proc. of the First International Workshop on Performance Metrics for Intelligent Systems (PERMIS)* 12-14 August 2000
12. Klaus Krippendorff: *A Dictionary of Cybernetics.* The American Society of Cybernetics, Norfolk, VA, USA (1986)
13. Barbara Ann Kitchenham: Evaluating software engineering methods and tool, part 2: selecting an appropriate evaluation method technical criteria. *SIGSOFT Softw. Eng. Notes*, 21(2) (1996) 11–15
14. Michel Grabisch, Sergei A. Orlovski, and Ronald R. Yager: Fuzzy aggregation of numerical preferences (1998) 31–68
15. Ioannis Kontogounis, Kyriakos Chatzidimitriou, Andreas L. Symeonidis, and Pericles A. Mitkas: A robust agent design for dynamic scm environments. In *LNAI, Vol. 3955* Springer-Verlag (2006) 127–136
16. Arunachalam R. Sadeh N. Ericsson J. Finne N. Collins, J. and S. Janson: The supply chain management game for the 2005 trading agent competition. Technical report, CMU (2004)
17. Ian H. Witten and Eibe Frank: *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco (2005)