Agent Based Frequent Set Meta Mining: Introducing EMADS

Kamal Ali Albashiri, Frans Coenen, and Paul Leng

Abstract In this paper we: introduce EMADS, the Extendible Multi-Agent Data mining System, to support the dynamic creation of communities of data mining agents; explore the capabilities of such agents and demonstrate (by experiment) their application to data mining on distributed data. Although, EMADS is not restricted to one data mining task, the study described here, for the sake of brevity, concentrates on agent based Association Rule Mining (ARM), in particular what we refer to as frequent set meta mining (or Meta ARM). A full description of our proposed Meta ARM model is presented where we describe the concept of Meta ARM and go on to describe and analyse a number of potential solutions in the context of EMADS. Experimental results are considered in terms of: the number of data sources, the number of records in the data sets and the number of attributes represented.

Keywords: Multi-Agent Data Mining (MADM), Frequent Itemsets, Meta ARM, Association Rule Mining.

1 Introduction

In this paper an extendible multi-agent data mining framework that can enable and accelerate the deployment of practical solutions to data mining problems is introduced. The vision is a collection of data, data mining and user agents operating under decentralised control. Practitioners wishing to participate in the framework may add additional agents using a registration strategy. We envision a collection of scattered data over the network, accessed by a group of agents that allow a user to pose data mining queries to those data sources with no requirement to know the location of the supporting data, nor how the agents are materialised through an inte-

Kamal Ali Albashiri, Frans Coenen, and Paul Leng
Department of Computer Science, The University of Liverpool,
Ashton Building, Ashton Street, Liverpool L69 3BX, United Kingdom
e-mail: {ali,frans,phl}@csc.liv.ac.uk

Please use the following format when citing this chapter:

Albashiri, K.A., Coenen, F. and Leng, P., 2008, in IFIP International Federation for Information Processing, Volume 276; *Artificial Intelligence and Practice II*; Max Bramer; (Boston: Springer), pp. 2332.

gration and ranking process. We also envision that the inclusion of a new data source or data mining techniques should be a simple process of adding new agents to the system. To investigate the potential of this approach we have built EMADS (Extendible Multi-Agent Data mining System). The use of EMADS offers a number of advantages, includes: decentralised control, distribution of computational resources, interoperability, distribution of expertise, task and data matching, Result evaluation, simple extendibility and security.

To illustrate some of the features of EMADS a Meta ARM (Association Rule Mining) scenario is considered in this paper. We define the term Meta Mining as the process of combining the individually obtained results of N applications of a data mining activity. The motivation behind the scenario is that data relevant to a particular ARM application is often owned and maintained by different, geographically dispersed, organizations. Information gathering and knowledge discovery from such distributed data sources typically entails a significant computational overheads; computational efficiency and scalability are both well established critical issue in data mining [1]. One approach to addressing problems such as the meta ARM problem is to adopt a distributed approach. However this requires expensive computation and communication costs. In distributed data mining, there is a fundamental tradeoff between accuracy and cost of computation. If we wish to improve the computation and communication costs, we can process all the data locally obtaining local results, and combine these results centrally to obtain the final result. If our interest is in the accuracy of the result, we can ship all the data to a single node (and apply an appropriate algorithm to produce this desired result). In general the latter is more expensive while the former is less accurate. The distributed approach also entails a critical security problem in that it reveals private information; privacy preserving issues [2] are of major concerns in inter enterprise data mining when dealing with private databases located at different sites.

An alternative approach to distributed data mining is high level learning which adopts strategies to allow all data to be locally analyzed, local results (models) are then combined at a central site to obtain the final result (global model). This approach is less expensive but may produce ambiguous and in- correct global results. To make up for such a weakness, many researchers have attempted to identify further alternatives to combining local models built at different sites. Most of these approaches are agent-based high level learning strategies such as: meta-learning [4], mixture of experts [5] and knowledge probing [6]. Bagging [7] increases the accuracy of the model by generating multiple models from different data sets chosen uniformly with replacement and then averaging the outputs of the models. However, these approaches still only have the ability to estimate a global data model through the aggregation of the local results, rather than generating an exact correct global model.

In EMADS a distributed computation framework is defined in terms of a Multi-Agent System (MAS), i.e. a system composed of a community of agents, capable of reaching goals that are difficult to achieve by an individual system [3]. In addition, a MAS can display self-organizational and complex behaviours, even when the capabilities of individual agents are relatively simple. The fundamental distinc-

tion between a distributed architecture and a MAS architecture is one of control. In a distributed system control is centralized; in a MAS control is decentralized in that agents are self motivating, and problem solving is achieved through inter-communication between agents.

The rest of this paper is organised as follows. Section 2 provides the motivation behind the material presented and discusses some related work. For completeness a brief note on Meta ARM Algorithms is then presented in Section 3. In section 4 our Meta ARM model architecture and functionality are described. Section 5 discusses the experimental results. Finally, Section 6 concludes the paper.

2 Related Work

There are a number of reports in the literature of the application of Agent techniques to data mining. Kargupta, Stafford, and Hamzaoglu [11] describe a parallel data mining system (PADMA) that uses software agents for local data accessing and analysis, and a Web based interface for interactive data visualization. PADMA has been used in medical applications. The meta-learning strategy offers a way to mine classifiers from homogeneously distributed data. Perhaps the most mature systems of agent-based meta-learning systems are: JAM [4], BODHI [12], and Papyrus [13]. In contrast to JAM and BODHI, Papyrus can not only move models from site to site, but can also move data when that strategy is desired. Papyrus is a specialized system which is designed for clustering while JAM and BODHI are designed for data classification. Basically, these systems try to combine local knowledge to optimize a global objective.

The major criticism of such systems is that it is not always possible to obtain an exact final result, i.e. the global knowledge model obtained may be different from the one that might have been obtained by applying the one model approach to the same data.

3 Note on Meta ARM Algorithms

Association Rule Mining (ARM) is concerned with the identification of patterns (expressed as "if ... then ..." rules) in data sets [8]. ARM typically begins with the identification of *frequent sets* of data attributes that satisfy threshold requirements of *relative* support in the data being examined. The most significant issue when combining groups of previously identified frequent sets is that wherever an itemset is frequent in a data source A but not in a data source B a check for any contribution from data source B is required (so as to obtain a global support count). The challenge is thus to combine the results from N different data sources in the most computationally efficient manner. This in turn is influenced predominantly by the magnitude (in terms of data size) of returns to the source data that are required.

To investigate and evaluate our ideas on EMADS a study of Meta ARM is presented here. Five Meta ARM algorithms are considered, all founded on the well known TFP ARM algorithm [9, 10] where results are stored in a T-tree. For the Meta ARM these trees must then be merged in some way. The structure of the T-tree, and the algorithms used in its construction, are described in [10]; the details of this are not relevant to the present paper, and in principle any algorithm for generating frequent sets could have been employed. As with all such algorithms, the merging of locally frequent sets to produce global totals may require additional computation to complete the counts of some sets. Each of the Meta ARM algorithms/agents makes use of *return to data* (RTD) lists, at least one per data set/agent, to hold lists of itemsets whose support was not included in the current T-tree and for which the count is to be obtained by a return to the originating raw data agent. The processing of RTD lists may occur during, and/or at the end of, the Meta ARM process depending on the nature of the algorithm. The algorithms can be summarised as follows:

1. Brute Force: Merges the T-trees one by one starting with the largest tree generating (N) RTD lists, processes RTD lists and prunes the T-tree at end of the merge process.
2. Apriori: Merges all T-trees level by level starting from the first level ($K = 1$) generating ($K * N$) RTD lists, processes RTD lists and prunes the T-tree at each level. The objective of the Apriori Meta ARM algorithm is to identify unsupported itemsets earlier in the process.
3. Hybrid 1: Commences by generating the top level of the merged T-tree in the Apriori manner described above (including processing of the RTD list); and then adds the appropriate branches, according to which top level nodes are supported, using a Brute Force approach.
4. Hybrid 2: Commences by generating the top two levels of the merged T-tree, instead of only the first level, as in the Hybrid 1 approach. Additional support counts are obtained by processing the RTD lists. The remaining branches are added to the supported level 2-nodes in the merged T-tree sofar (again) using the Brute Force mechanism.
5. Bench Mark: It is a bench mark algorithm against which the identified Meta ARM algorithms were to be compared.

Full details of the Meta ARM algorithms can be found in Albashiri et al. ([14]). Note that the overview given here is in the context of MADM (Multi-Agent Data Mining) whereas the original algorithms proposed by Albashiri et al. did not operate in an agent context.

4 Meta ARM Model

In order to demonstrate the feasibility of our EMADS vision a peer to peer agent-based framework has been designed and implemented, which uses a broker mediated-based architectural model [15].

Fig. 1 shows the Meta ARM model architecture of EMADS framework which is built with the JADE Toolkit [16]. The system consists of one organization site (mediator host) and several local sites (sites of individual hosts). Detailed data are stored in the DBMS (Data Base Management System) of local sites. Each local site has at least one agent that is a member of the organization. The connection between a local agent and its local DBMS is not included. There are two special JADE agents at the organization site (automatically started when the organization site is launched). The AMS (Agent Management System) provides the Naming Service (i.e. ensures that each agent in the platform has a unique name) and represents the authority in the platform. The DF (Directory Facilitator) provides a Yellow Pages service by means of which an agent can find other agents providing the services required in order to achieve its goals. All Routine communication and registration are managed by these JADE agents. Data mining tasks are managed through the P2P model by the other agents.

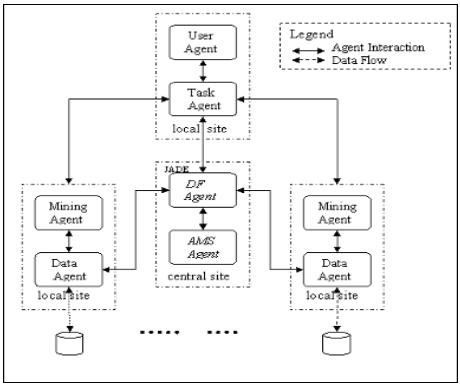


Fig. 1. Meta ARM Model Architecture

In this framework, agents are responsible for accessing local data sources and for collaborative data analysis. The architecture includes: (i) data mining agents, (ii) data agents, (iii) task agents, (iv) user agents, and (v) mediators (JADE agents) for agents coordination. The data and mining agents are responsible for data accessing and carrying through the data mining process; these agents work in parallel and share information through the task agent. The task agent co-ordinates the data mining operations, and presents results to the user agent. Data mining is carried out by means of local data mining agents (for reasons of privacy preservation). In the context of Meta ARM activity each local mining agent's basic function is to generate local item sets (local model) from local data and provide this to the task agent in order to generate the complete global set of frequent itemsets (global model).

4.1 Dynamic Behaviour of System for Meta ARM operations

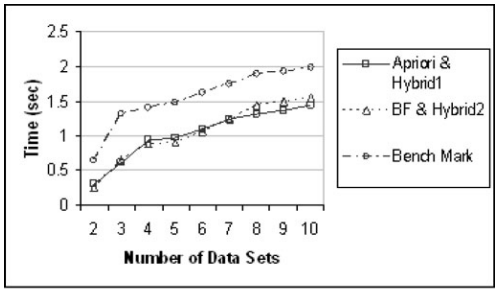
The system Initially starts up with the two central JADE agents. When a data agent wishes to make its data available for possible data mining tasks, it must publish its name and description with the DF agent. In the context of Meta ARM, each mining agent could apply a different data mining algorithm to produce its local frequent item sets T-tree. The T-trees from each local data mining agent are collected by the task agent, and used as input to Meta ARM algorithms for generating global frequent item sets (merged T-tree) making use of return to data (RTD) lists, at least one per data set, to contain lists of itemsets whose support was not included in the current T-tree and for which the count is to be obtained by a return to the raw data.

5 Experimentation and Analysis

To evaluate the five Meta ARM algorithms, in the context of EMADS vision, a number of experiments were conducted. These are described and analysed in this section. The experiments were designed to analyse the effect of the following:

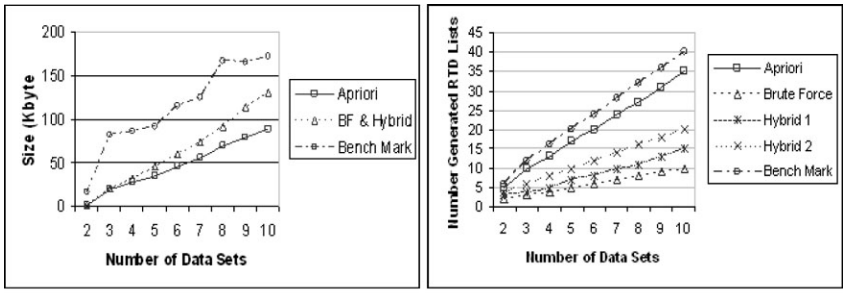
1. The number of data sources *(data agents)* .
2. The size of the datasets (held at data agents) in terms of number of records.
3. The size of the datasets (held at data agents) in terms of number of attributes.

Experiments were run using two Intel Core 2 Duo E6400 CPU (2.13GHz) computers with 3GB of main memory (DDR2 800MHz), Fedora Core 6, Kernel version 2.6.18 running under Linux except for the first experiment where two further computers running under Windows XP were added. For each of the experiments we measured: (i) processing time (seconds/mseconds), (ii) the size of the RTD lists (Kbytes) and (iii) the number of RTD lists generated. The authors did not use the IBM QUEST generator [17] because many different data sets (with the same input parameters) were required and it was found that the quest generator always generated the same data given the same input parameters. Instead the authors used the LUCS KDD data generator [1]. Note that the slight oscillations in the graphs result simply from a vagary of the random nature of the test data generation.



(a) Processing Time

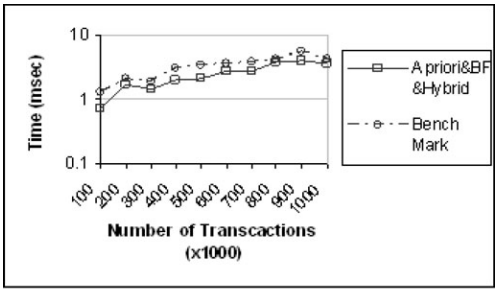
[1] *http://www.csc.liv.ac.uk/frans/KDD/Software//LUCS-KDD-DataGen/*



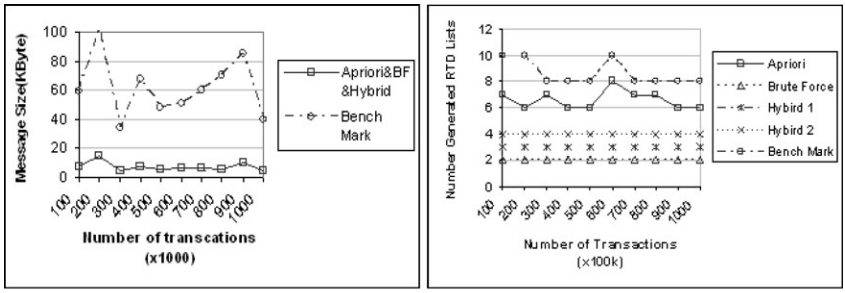
(b) Total size of RTD lists (c) Number of RTD lists

Fig. 2. Effect of number of data sources

Figure 2 shows the effect of adding additional data sources. For this experiment ten different artificial data sets were generated and distributed among four machines using $T = 4$ (average number of items per transactions), $N = 20$ (Number of attributes), $D = 100k$ (Number of transactions). The selection of a relatively low value for N ensured that there were some common frequent itemsets shared across the T-trees. Experiments using $N = 100$ and above tended to produce many frequent 1-itemsets, only a few isolated frequent 2-itemsets and no frequent sets with cardinality greater than 2. For the experiments a support threshold of 1% was selected. Graph 2(a) demonstrates that all of the proposed Meta ARM algorithms worked better then the bench mark (start from "scratch") approach. The graph also shows that the Apriori Meta ARM algorithm, which invokes the "return to data procedure" many more times than the other algorithms, at first takes longer; however as the number of data sources increases the approach starts to produce some advantages as T-tree branches that do include frequent sets are identified and eliminated early in the process. The amount of data passed to and from sources, shown in graph 2(b), correlates directly with the execution times in graph 2(a). Graph 2(c) shows the number of RTD lists generated in each case. The Brute Force algorithm produces one (very large) RTD list per data source. The Bench Mark algorithm produces the most RTD lists as it is constantly returning to the data sets, while the Apriori approach produces the second most (although the content is significantly less).



(a) Processing Time

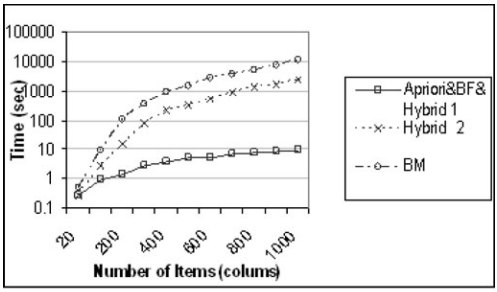


(b) Total size of RTD lists (c) Number of RTD lists

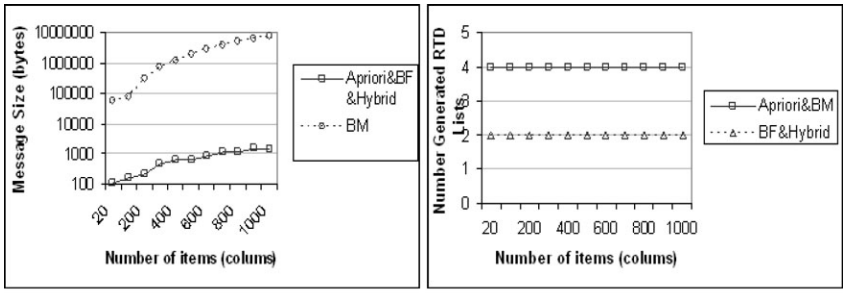
Fig. 3. Effect of increasing number of records

Figure 3 demonstrates the effect of increasing the number of records. The input data for this experiment was generated by producing a sequence of ten pairs of data sets (with $T = 4, N = 20$) representing two sources on two different machines. From graph 3(a) it can be seen that the Brute Force and Hybrid 1 algorithms work best because the size of the return to data lists are limited as no unnecessary candidate sets are generated. This is illustrated in graph 3(b). Graph 3(b) also shows that the increase in processing time in all cases is due to the increase in the number of records only; the size of the RTD lists remains constant throughout as does the number of RTD lists generated (graph 3(c)).

Figure 4 shows the effect of increasing the global pool of potential attributes (remember that each data set will include some subset of this global set of attributes). For this experiment another sequence of pairs of data sets (representing two sources) was generated with $T = 4$, $D = 100K$ and N ranging from 100 to 1000. As in the case of experiment 2 the Brute Force and Hybrid 1 algorithms work best (for similar reasons) as can be seen from graph 4(a). However in this case (compared to the previous experiment), the RTD list size did increase as the number of items increased (graph 4(b)). For completeness graph 4(c) indicates the number of RTD lists sent with respect to the different algorithms. The reasoning behind the Hybrid 2 algorithm proved to be unfounded; all the 1-itemsets tended not to be all supported.



(a) Processing Time



(b) Total size of RTD lists (c) Number of RTD lists

Fig. 4. Effect of increasing number of items (attributes)

All the Meta ARM algorithms outperformed the bench mark (start from scratch) algorithm. The Hybrid 2 algorithm performed in an unsatisfactory manner largely because of the size of the RTD lists sent. Of the remainder the Apriori approach coped best with a large number of data sources, while the Brute Force and Hybrid 1 approaches coped best with increases data sizes (in terms of column/rows) again largely because of the relatively smaller RTD list sizes. It should also be noted that the algorithms are all complete and correct, i.e. the end result produced by all the algorithms is identical to that obtained from mining the union of all the raw data sets using some established ARM algorithm. Of course our MADM scenario, which assumes that data cannot be combined in this centralised manner, would not permit this.

6 Conclusions and Future Work

Traditional centralized data mining techniques may not work well in many distributed environments where data centralization may be difficult because of limited bandwidth, privacy issues and/or the demand on response time. Meta-learning data mining strategies may offer a better solution than the central approaches but are not as accurate in their results. This paper proposes EMADS, multi-agent data mining framework with peer-to-peer architecture as an application domain to address the above issues. The use of EMADS was illustrated using a meta ARM scenario. Four meta ARM algorithms and a bench mark algorithm were considered. The described experiments indicated, at least with respect to Meta ARM, that EMADS offers positive advantages in that all the Meta ARM algorithms were more computationally efficient than the bench mark algorithm. The results of the analysis also indicated that the Apriori Meta ARM approach coped best with a large number of data sources, while the Brute Force and Hybrid 1 approaches coped best with increased data sizes (in terms of column/rows). The authors are greatly encouraged by the results obtained so far and are currently undertaking further analysis of EMADS with respect to alternative data mining tasks.

References

1. Kamber, M., Winstone, L., Wan, G., Shan, S. and Jiawei, H., "Generalization and Decision Tree Induction: Efficient Classification in Data Mining". Proc. of the Seventh International Workshop on Research Issues in Data Engineering, pp. 111-120, 1997.
2. Aggarwal, C. C. and Yu, P. S., "A Condensation Approach to Privacy Preserving Data Mining". Lecture Notes in Computer Science, Vol. 2992, pp. 183-199, 2004
3. Wooldridge, M., "An Introduction to Multi-Agent Systems". John Wiley and Sons Ltd, paperback, 366 pages, ISBN 0-471-49691-X, 2002.
4. Stolfo, S., Prodromidis, A. L., Tselepis, S. and Lee, W., "JAM: Java Agents for Meta-Learning over Distributed Databases". Proc. of the International Conference on Knowledge Discovery and Data Mining, pp. 74-81, 1997.
5. Xu, L. and Jordan, M. I., "EM learning on a generalised finite mixture model for combining multiple classifiers", In Proc. of World Congress on Neural Networks, 1993.
6. Guo, Y. and Sutiwaraphun, J., "Knowledge probing in distributed data mining". In Advances in Distributed and Parallel Knowledge Discovery, 1999.
7. Breiman, L., Bagging predictors, Machine Learning, 24, 123-140, 1996.
8. Agrawal, R., Imielinski, T., and Swami A., "Mining Association Rules between Sets of Items in Large Databases". In Proc. of ACM SIGMOD Conference on Management of Data, Washington DC, May 1993.
9. Goulbourne, G., Coenen, F.P. and Leng, P., "Algorithms for Computing Association Rules Using A Partial-Support Tree". Proc. ES99, Springer, London, pp. 132-147, 1999.
10. Coenen, F.P. Leng, P., and Goulbourne, G., "Tree Structures for Mining Association Rules". Journal of Data Mining and Knowledge Discovery, Vol 8, No 1, pp. 25-51, 2004.
11. Kargupta, H., Hamzaoglu, I., and Stafford, B., "Scalable, Distributed Data Mining Using an Agent Based Architecture". Proc. of Knowledge Discovery and Data Mining, AAAI Press, 211-214, 1997.
12. Kargupta, H., Hersh berger, D., and Johnson, E., Collective Data Mining: A New Perspective Toward Distributed Data Mining. Advances in Distributed and Parallel Knowledge Discovery, MIT/AAAI Press, 1999.
13. Bailey S., Grossman, R., Sivakumar, H., and Turinsky, A., "Papyrus: a system for data mining over local and wide area clusters and super-clusters". In Proc. Conference on Supercomputing, page 63. ACM Press, 1999.
14. Albashiri, K.A., Coenen, F.P., Sanderson, R. and Leng. P., Frequent Set Meta Mining: Towards Multi-Agent Data Mining. To appear in Research and Development in Intelligent Systems XXIV, Springer, London, (proc. AI'2007).
15. Schollmeier, R., "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications". First International Conference on Peer-to-Peer Computing (P2P01) IEEE. August 2001.
16. Bellifemine, F., Poggi, A., and Rimassi, G., "JADE: A FIPA-Compliant agent framework". Proc. Practical Applications of Intelligent Agents and Multi-Agents, April 1999, pg 97-108 (See http://sharon.cselt.it/projects/jade for latest information)
17. Agrawal, R., Mehta, M., Shafer, J., Srikant, R., Arning, A. and Bollinger, T.,"The Quest Data Mining System". Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining, (KDD1996).