Multiagent Collaborative Learning for Distributed Business Systems

Yutao Guo and Jörg P. Müller
Intelligent Autonomous Systems
Siemens AG, Corporate Technology
Otto-Hahn-Ring 6
81730 Munich, Germany
joerg.p.mueller@siemens.com

Abstract

This paper presents a multiagent architecture and algorithms for collaborative learning in distributed and heterogeneous business systems, where the participating agents have local, incomplete knowledge used to make predictions about parameters of a business transaction. We propose two collaborative learning strategies which differ in the nature and amount of information that is exchanged during collaboration, and which are hence suitable for different organisational settings. The first algorithm relies on the exchange of information about a transaction instance, whereas the second algorithm uses qualitative information provided by individual agents, such as the results of predictions from the agent's local perspective. We apply the architecture and strategies to a distributed supply chain prediction problem. Experiments run on a large real-world order data set indicate that our approach effectively improves the learning performance based on limited additional communication between the participating agents.

1. Introduction

Globalisation and mass customisation are business trends that have created a development where business processes that used to be managed by a single enterprise are now distributed and need to be planned and enacted across multiple partners. This leads to highly distributed architectures of business systems with amplified complexity and increased need for collaboration among the participating organizations. Multiagent systems [7] are a powerful technology to tackle this complexity and to enable flexible business systems. Software agents autonomously monitor business environments, observe and react to unforeseen events, provide human users with decision support, engage in automated action on behalf of humans and organizations, and learn from their experience. At the same time, the use of peer-to-peer, multiagent collaboration architectures in business applications [11] supports flexible and self-organising structures and decentral problem solving.

In distributed business systems, the participating organizations (and the software agents that represent these) usually have incomplete knowledge based on their partial local views on process definitions, (sub-)process status and outcomes, and information structures. In our paper, we represent this view by limiting the view of an agent to a projection of attributes from a relational table denoting e.g., the execution history of a transport order. In addition, the organizations and agents are embedded in an organizational structure. In our paper, we use a simplified model based on a graph topology which imposes constraints on the ability of individual agents to communicate with each other, and which (in future work) will allow us to associate cost and utility of communicating along an edge in the graph. A main reason for choosing a decentral, loosely coupled architecture is that centralization of data is often neither feasible nor desirable in many cases. Firstly, high transaction volumes and quickly changing local situations mean that enforcing complete knowledge involves a prohibitive communication overhead. Secondly, the different organizations participating in a distributed business system may not be willing to reveal private data to their competitors or customers.

In this paper, we present a multiagent architecture and two strategies and algorithms for collaborative learning in distributed, decentralized business systems. Our application is the prediction of delays in a supply chain logistics scenario. The first learning strategy aims at improving the quality of local prediction by incorporating process information available at other agents'. The second strategy aims at improving the prediction quality by exchanging qualitative information, i.e., opinions of other prediction agents, thus minimizing the amount of information which needs to be communicated. Experiments run on a large real-world or-

der data set indicate that the collaborative multiagent learning approach effectively improves the learning performance based on limited additional communication between the participating agents.

This paper is structured as follows: Section 2 discusses application scenarios and describes the architecture of our system. Section 3 gives an overview of related research. In Section 4, two collaborative learning algorithms are described. Section 5 describes the results of an empirical evaluation of the two approaches based on a large real-world data set. Finally, we present conclusions and perspectives of future research in Section 6.

2. Application scenario and system architecture

Many business systems are in distributed architecture and collaborative learning can help improving their performance. For example:

- **Supply chain management**: Supply networks reveal highly distributed architectures with heterogeneous information distribution, i.e. the partners have incomplete knowledge based on their local views of process definitions, status, and outcomes. Here, our claim, which we justify in this paper, is that collaborative learning can improve adaptive process planning, monitoring, and coordination of the overall process.
- **Customer relation management**: Customer services are usually implemented by different business units, such as product sales, delivery, and technical support. Each unit keeps specific service information, which can be seen as partial views of the overall customer information. Collaborative learning of these business units can help towards better understanding of the customers' context and preferences, and provide assistance to product recommendation or fraud detection applications.

In this paper, we illustrate our collaborative learning strategies using the supply chain scenario, and, in particular, proactive monitoring of logistics processes. We draw our results from a real-world use case and real application data obtained from a major logistics service provider. This service provider is responsible for the administration and supervision of order delivery. Customers are business units who wish to deliver products to their customers, as shown in Figure 1. Orders are characterised by origins and destinations around the world, various product types from automation systems to medical equipment, and different packaging. To carry out an order, different local and international transportation service providers may be selected. Often, fulfilling an order requires a combination of transport means and service providers. Due to the high volume of transactions (several millions of orders over the period of one year), short planning/execution cycles, and a variety of unexpected events that happen during execution, monitoring each transaction manually is infeasible for humans users.

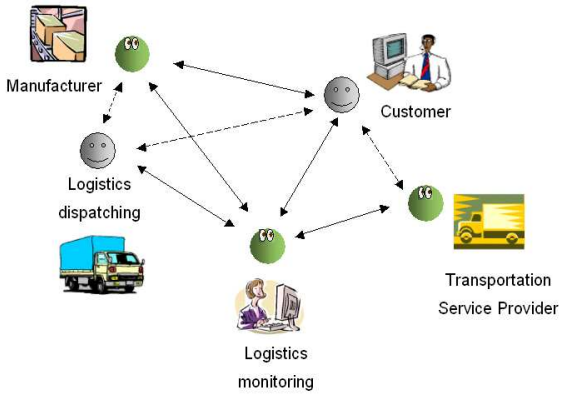

Figure 1. The system architecture of agent assistance in the logistics process.

Figure 1 shows the multi-agent architecture that we designed to support a decision support solution for this application. The human dispatcher – located at the logistics monitoring node in Figure 1 interacts with a decision support system that uses dedicated prediction agents to obtain predictions for order delays. These predictions can be used for dispatching incoming orders (e.g., by proposing suitable transport service providers) as well as for execution monitoring (e.g., by integrating different types of information and predict which orders have high delay probabilities). Apart from a team of *local learning agents* (omitted in Figure 1) that provide different means for making predictions based on analysing available local data (see [5] for a more detailed description of this part of the system), there is a distributed group of *coordination agents* whose task it is to exchange information in order to improve the quality of prediction across the boundaries of individual organizations. These agents can be located everywhere in the supply chain, at manufacturers, logistics services, transport service providers, and customers. Our focus in this paper is on the latter group of agents, the coordination agents.

3. Related work

Agent technology is becoming increasingly often used to model and implement distributed business systems. TRACONET [14] is a vehicle routing system with an extended version of the contract net protocol. A multiagent approach for cooperative transportation scheduling is investigated in

[3]. [4] describes an agent-based logistics monitoring architecture. A variety of approaches from multiagent systems, maybe most notably represented by the ADEPT project [6], have explored the use of multiagent collaboration and coordination techniques in the context of business applications. However, these approaches were generally not focused on aspects of adaptation and learning. There have been some researches on agent learning in Markov games with reinforcement learning [17]. Whereas the research in this paper focuses on multiagent collaborative learning in distributed and heterogeneous business systems for decision making.

Learning from heterogeneous distributed data is an active and challenging area [12]. The collective Bayesian network learning algorithm proposed in [1] works in two stages: first it learns a local Bayesian network at each site using the local data. Then each site identifies the observations that are most likely provide evidence for coupling between local and non-local variables and transmits a subset of these observations to a central site. A second Bayesian network is learnt at the central site using the data transmitted from the local site. The local and central Bayesian networks are combined to obtain a collective Bayesian network, that models the entire data. This approach suffers from the fact that it requires centralization of part of the data set at the training stage and that it relies on the exchange of row data in the prediction stage, thus leading to a considerable communication overhead.

To prevent centralization and reduce the amount of information exchanged, [10] propose a distributed decision tree (DDT) based method enabling a multiagent learning of classification tasks. Their algorithm performs a breath-first search for the path through some decision tree. The algorithm does not provide a training stage, nor is an explicit model learnt. At the prediction stage, the results of the iterative learning at the individual agents is correlated, also leading to a considerable computation and communication effort, since the whole process is performed each time an instance is classified.

The algorithms and results presented in this paper extends these lines of research by reducing the amount of communication and by allowing for loosely coupled collaboration among the learning agents, making it suitable for distributed business systems.

4. Agent collaborative learning algorithm

4.1. Overview

We formalize our approach by introducing three models, labelled *transaction data model*, *information distribution model*, and *organization model*.

The transaction data model describes the structure of business transaction information. A business transaction $\mathbf{b}$ is defined as an n-tuple $\mathbf{b} = (x_1, x_2, \ldots, x_n)$, where $x_i \in X_i$ is the value of the i-th attribute of $\mathbf{b}$.

The information distribution model describes how the business transaction information is distributed over a group of agents. In this paper, we assume a vertical information distribution model, i.e., agents only know (possibly overlapping) sub-sets of the transaction attributes. So each agent has a local, partial view of the overall transaction-related information and uses this view to construct models, e.g., used for predictions on transaction parameters.

The agents in our business systems are arranged according to an organization model $OM = (A, S, OR)$, where $A = \{a_1, a_2, \ldots, a_n\}$ is the set of agents, $S = s_1, \ldots, s_m$ is the set of services, and OR is the definition of the collaboration schemes. Each agent can provide a number of services to its environment. In this paper, and without loss of generality, we assume that S is the union of all services offered by the agents in A. The collaboration scheme OR allows us to define constraints on service provisioning among single agents. We define OR as $OR = \{(a_i, a_j, S_{ij}) | 1 \leq i, j \leq n, i \neq j\}$ where S_{ij} is a bit vector of length m with the l-th bit in S_{ij} being 1 iff a_i provides service s_l to a_j. See below for an example.

Collaborative learning is influenced by both the information distribution model and the organization model. Besides improving local learning performance, there are two issues to be considered:

- Minimize information exchange between agents to reduce communication costs and protect privacy.
- Provide loose coupling of the agents to achieve flexibility. For example, due to the chosen communication mechanism and different system configurations at each agent, it may be difficult to ensure the agents perform the learning tasks in a synchronous (tightly coupled) manner.

Two collaborative learning strategies are proposed in this paper to meet these requirements. They can also form hybrid method for complex business scenarios.

- **Collaboration based on instance information (CBI)**: An agent acquires partial and explicit business transaction information from other agents for better learning performance. With this strategy, an agent requires information about certain attributes of other agents, but does not need other agents make learning and predictions at the same time. We propose a method for an agent to determine the most relevant partner and acquire informative data without having to enumerate the transaction-related attribute space of other agents. See Section 4.3.
- **Collaboration based on agents' predictions (CAP)**: An agent ensembles other agents' opinions to improve

its own predictions. So explicit transaction-related information of other agents is not required; rather other agents need to provide their predictions simultaneously. We achieve this by a stacking based method with Support Vector Machine for an agent to ensemble other agents' opinions. See Section 4.4.

We incorporate this into the organization model described above by introducing two services that the agents in our system can provide to each other: An agent can offer another agent to provide transaction-related information in terms of additional attribute values for a given transaction instance (labelled service s_1), and an agent can offer to provide prediction results for a given transaction instance (labelled service s_2). Each agent may choose whether to offer each service. Clearly, service s_1 will lead to a higher volume of information to be exchanged while service s_2 will be more computation-extensive at the provider side, while allowing the service provider to hide transaction-related information and only reveal qualitative information. We can now give an example of a collaboration schema between two agents a_i and a_j: $(a_i, a_j, (1, 0))$ denotes that a_i will provide transaction-related information (service s_1) to a_j, whereas it will not provide qualitative prediction results (service s_2). Collaboration schema information can be stored centrally or locally for each agent. It will guide the behavior of an agent in collaborations and constrain the possible forms of collaborative learning applicable in a certain situation.

In the following, we describe the local prediction algorithms and the two collaborative learning strategies.

4.2. Local prediction algorithms

We view the problem of predicting transaction parameters (e.g., whether a transaction will be delayed or not) as a classification problem. Given incomplete knowledge about other attributes of the transaction, we shall classify the attribute *order delivery status* either as *delay* or *on-time*[1].

We adopt the Bayesian classifier method to implement the agents' local basis prediction algorithms used to build their individual models (based on a training set e.g., extracted from past transaction execution records), as most attributes of the instances of logistics orders have nominal data types. Bayesian classifiers assign the most likely class (e.g., *delay* or *on-time* in the above example) to a given instance described by its feature vector. The simplified Bayesian classifiers, Naive Bayes, assumes that attribute values are conditionally independent given the class values. Despite this assumption is unrealistic, Naive Bayes is remarkably successful in practice, often competing with much more sophisticated techniques [8].

1 Note that more fine-grained definitions of delay are possible using this approach

$$y = \underset{y_j \in Y}{\operatorname{argmax}} P(y_j) \prod_{i=1}^{n} P(x_i|y_j) \quad (1)$$

where y is the predicted class, $\mathbf{x} = (x_1, \ldots, x_n)$ is the attribute vector describing an instance to be classified, Y is the class value set. The various $P(y_j)$ and $P(x_i|y_j)$ terms can be estimated based on the training set. Given an instance with unknown label, the learned model will output the probability of success/delay. Hence, a criterion is needed to provide explicit labelling. In this paper, we label the instances using a relative measure of delay instead of a fixed probability threshold. We achieve this by sorting the instances according to their probability of delay, and then labelling the top d percent of instances as *delay*. A realistic value for d is determined by computing the average percentage of delay instances in the training set.

4.3. Collaboration based on instance information

In this learning strategy, an agent integrates partial information from other agents for better learning performance. The motivation for this is that research on using selective Bayesian classifiers [9] for feature selection has proved that a classifier built on an properly selected subset of attributes may have the same or even better performance than one created using all attributes. Another approach [20] uses naive Bayes classifier committees for improving classification performance. Each committee member is a naive Bayesian classifier built using different attribute subsets in sequential trials, and the final class predictions are made through committee voting. Although in a distributed business system, the attribute space is partitioned physically based on the business collaboration structure, we still can add relevant transaction attribute data to an agent's existing data to effectively improve its learning performance.

The optimized selection of the attribute subsets in [20] is based on greedy search of the whole attribute space; in our approach, we focus on minimizing communication costs and protection of privacy relating to an underlying organization model. We propose a method for an agent determine most relevant partner and get informative information without having to enumerate the attribute space of other agents. The method is implemented in two steps:

1. Each agent i finds the most relevant agent j by correlation analysis on their prediction behavior on the *context set*.
2. Agent j provides agent i its high informative attributes derived by mutual information analysis between all attributes and the instance class.

Here, the *context set* is a data set containing a small number of transactions, and with similar properties as the test

set. It is used for analyzing the agents models obtained from their *training sets*. In this paper, we use recent data as the *context set*. The *context set* is also an important component in the second collaborative learning strategy described in Section 4.4, below.

In the first step, an agent performs correlation analysis on the other agents' predictions on the *context set*. Agents with smaller resulting correlation will be more helpful to improve current agent's learning performance. The correlation of the agents prediction behavior on the *context set*, $\mathbf{p}^{agent_i}$ and $\mathbf{p}^{agent_j}$, is measured by the Bravais Pearson correlation coefficient as Equation (2).

$$c_{i,j} = \frac{\sum_{k=1}^{n}(p_k^{agent_i} - \bar{p}^{agent_i})(p_k^{agent_j} - \bar{p}^{agent_j})}{\sqrt{\sum_{k=1}^{n}(p_k^{agent_i} - \bar{w}^{agent_i})^2 \sum_{k=1}^{n}(p_k^{agent_j} - \bar{p}^{agent_j})^2}} \quad (2)$$

where $p_k^{agent_i}$ and $p_k^{agent_j}$ are the predicted probability of delay of instance k in the context set by agent i and agent j, $\bar{p}^{agent_i}$ and $\bar{p}^{agent_j}$ are the mean values, and n is the number of instances in the context set.

In the second step, an agent determines the most informative attribute based on *mutual information* analysis [15]. Mutual information of two variables is defined as the reduction in uncertainty ($Entropy$) of the value of one variable given knowledge of the value taken by other variable. The relevance of an attribute is determined by the mutual information $H(y; x_i)$ between the instance class y (in our example denoting the order delivery status, which could take the values *delay* or *on-time*) and the value of attribute x_i of the instance. Here, $H(y; x_i)$ measures the amount of information the attribute value conveys about the order delivery status. In case attribute value are continuous values, computing the mutual information requires to divide their ranges into predefined intervals [18]. The mutual information $H(y; x_i)$ between an instance class y and an attribute x_i is computed as:

$$H(y; x_i) = \sum_v \sum_c P(y = c, x_i = v) log_2 \frac{P(y = c, x_i = v)}{P(y = c)P(x_i = v)} \quad (3)$$

where $p(y = c)$ is the probability of delivery status being c, and $p(x_i = v)$ is the probability of value v for attribute x_i occurring in the respective training set.

The algorithm describing the collaborative learning strategy can now be defined as follows:

1. Each agent i learns its model C_i using the Bayesian classifier on its local training data set as described in Section 4.2.
2. Based on C_i, each agent i makes predictions on the *context set V*.
3. Agents exchange the prediction results for the context set.
4. Agent i finds the most relevant agent j by correlation analysis on the other agents' predictions on the context set.
5. Agent j provides its most informative attribute to agent i; this is obtained by mutual information analysis between the attributes and the instance class.
6. Each agent i re-learns its models C_i again with its local training data set, adding the attributes acquired from other agents.
7. The agents use C_i to make predictions on the test set.

4.4. Collaboration based on agents' predictions

This collaborative learning strategy enables an agent to integrate other agents' predictions with its local model to improve learning performance. We propose a *meta learning* approach to implement this strategy, as it provides flexible ensemble of multiple learners while limiting communication. Various meta-learning strategies were described in the literature [2, 13]. *Voting* is the simplest method of combing predictions from multiple classifiers. In its simplest form, majority voting, each model contributes a single vote. The collective predictions is decided by the majority of the votes. In weighted voting, the classifiers have varying degrees of influence on the collective prediction that is relative to their prediction accuracy. Each classifier is associated with a specific weight determined by its performance on a validation set. The final prediction is decided by summing over all weighted votes and by choosing the class with the highest aggregate. The voting method combine the set of models in linear fashion and is not effective in some scenarios.

In this paper, we adopt and extend the *stacking* method [19], as one of the most effective meta learning strategies. Stacking integrates independently computed base classifiers into a higher-level classifier by learning over a so-called *meta set*. A meta set is generated by using the predictions of the base classifiers as attributes and keeping the original class labels. Then a *meta classifier* can be trained with the meta set; this meta classifier is then used to classify unlabelled instances. The aim of this strategy is to correlate the predictions of the base classifiers by learning the relationship between these predictions and the true labels. So stacking allows effective non-linear combination of base classifiers. In this work, we added several improvements to the standard stacking method:

- Use Support Vector Machine (SVM) as the meta learner. SVM is an efficient learning method for learning problems based on small training sets. See below for a short description of SVM.
- The meta set is generated with model predictions, as well as the original attributes of the orders. Only highly

relevant attributes are selected to be the order attributes in the meta set, based on the results of a mutual information analysis (see Section 4.3). Not considering the less relevant attributes actually improves learning performance in our experiments, thus supporting Langley's results reported in [9].

- The meta set uses the predicted probability of delay instead of predicted class, which provides more information for meta learning.

The learning algorithm consists of the following steps:

1. Each agent i learns its model C_i using the Bayesian classifier method on its own training data set.
2. Each agent i makes predictions on the *context set V* based on C_i.
3. Agents exchange the prediction results for the context set.
4. Generate a $meta\ set\ T$ with the agent predictions on the context set.
5. Learn a $meta\ model\ M$ on the $meta\ set\ T$ using the SVM algorithm.
6. Use the $meta\ model\ M$ to make predictions on the test set.

The context set and the learned individual models are used to compute the meta set. In this paper, the attributes of a meta set data instance are composed of the model predictions and the original attributes of the orders. Thus, we define the meta set T as $T = \{y, C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_k(\mathbf{x}), attribute\_vector(\mathbf{x}))|$ $\mathbf{x} \in V\}$, where $\mathbf{x}$ is an instance in the $context\ set\ V$, y is the true class of $\mathbf{x}$, $C_i(\mathbf{x})$ is the prediction of model i on $\mathbf{x}$, and $attribute\_vector(\mathbf{x})$ contains the attributes of $\mathbf{x}$. The predicted probability of delay is used instead of predicted labels as the model prediction attributes, and only the high relevant attributes are selected to be the order attributes in the meta set.

The meta learner is trained with the meta set to make delay predictions on an instance of an order. We use Support Vector Machine (SVM) classifier, which has proven empirical performance in pattern recognition [16]. Based on statistical learning theory, the SVM learner has good generalization ability and reveals good performance even when the size of the available training data is limited. This enables us to use small context set. The SVM classifier projects data in the input space $\mathcal{X}$ into a higher dimensional feature space $\mathcal{H}$, with a kernel Function $k(\mathbf{t_i}, \mathbf{t_j}) = \phi(\mathbf{t_i})^T \phi(\mathbf{t_j})$, where $\mathbf{t_i}$ and $\mathbf{t_j}$ are instances in the meta set. Vapnik shows that the Structural Risk Minimization principle in statistical learning theory can be translated into finding the hyperplane with maximum margin for separating the positive and negative training instances. Computing this hyperplane is by solving a quadratic optimization problem. The solution will be a set of *support vectors*, which determine the position of the hyperplane. With the learned SVM model, the unlabelled orders can be classified as $on - time$ (positive) and $delay$ (negative). By calculating distances of order vectors to the trained hyperplane, the SVM model can give a straightforward ranking of orders. In the experiments, as the meta set is unbalanced, i.e., the percentage of success orders is much higher than that of delay orders, so higher weight is given to the delay orders when training the SVM model.

5. Empirical analysis

In this section, we describe experimental results which we conducted to evaluate the performance of the two proposed collaborative learning strategies.

5.1. Experimental settings

The empirical analysis is based on a large data set from the data warehouse of a major logistics service provider collected over 2003. Each data record describes a single order execution transaction, including the attributes: origin, destination, product type, packaging, local transport provider, international transport provider (where applicable), planned customer delivery date, planned pickup date, actual pickup date, and actual delivery date. The data sets used for collaborative learning are as follows:

- Training set: 390505 transaction instances, vertically partitioned into five separate groups and maintained by five agents: agent 1 to agent 5. As described in Section 2, these agents would represent different parties in the supply chain, including logistics services, transport service providers, or customers.
- Test set: 87994 transaction instances used for evaluation. These orders were August 2003 data without the first week in August, which was used as the context set.
- Context set: data from the first week of August 2003, 6088 instances.

All experiments were repeated ten times with differing data sets, all reported figures are average figures.

The following experiments are performed on the test set:

- EXP_1 to EXP_5: Results of local predictions of the five agents, as described in Section 4.2.
- EXP_6: Results of local predictions, however, based on the union of the information views of the five agents, i.e., on complete information.
- EXP_7: One agent, e.g., agent 3 in this case, performs collaborative learning based on instance information (CBI), as described in Section 4.3.

Experiment	Learning model	DR	DP
EXP_1	Agent 1	0.43	0.39
EXP_2	Agent 2	0.44	0.40
EXP_3	Agent 3	0.37	0.31
EXP_4	Agent 4	0.40	0.36
EXP_5	Agent 5	0.42	0.38
EXP_6	Centralized	0.49	0.42
EXP_7	CBI at Agent 3	0.45	0.41
EXP_8	CAP at Agent 3	0.59	0.40

Table 1. Learning performance of individual and collaborative learning

- EXP_8: One agent, i.e., agent 3 in this case, performs collaborative learning based on agents' predictions (CAP), as described in Section 4.4.

To evaluate the learning performance, we adopted two standard measures from information retrieval: *delay recall* (DR) and *delay precision* (DP), as defined in Equations 4 and 5.

$$DR = \frac{n_{D\to D}}{n_{D\to D} + n_{D\to S}} \tag{4}$$

$$DP = \frac{n_{D\to D}}{n_{D\to D} + n_{S\to D}} \tag{5}$$

where $n_{D\to D}$ is the number of correctly predicted *Delay* orders, $n_{D\to S}$ and $n_{S\to D}$ are the number of $Delay \to Success$ and $Success \to Delay$ errors. Intuitively, delay recall measures the share of actually delayed orders that the mechanism could successfully predict as *Delay*, whereas delay precision is a measure of the inverse noise ratio, i.e., the share of orders predicted by the system as *Delay* that were actually delayed. It is known from information retrieval that there is a trade-off between recall and precision, and that it is rather easy to optimise either of them, but very hard to provide good results for both.

5.2. Experimental results and analysis

The experimental results of individual and collaborative learning are shown in Table 1. The correlation analysis in CBI (for EXP_7) is shown in Table 2; Table 3 illustrates the mutual information analysis results used by the agents to determine the informative attributes.

The experimental results show that both collaborative learning methods, CBI and CAP, can effectively improve the learning performance of agent local learning. CAP has better performance than CBI by assembling higher level knowledge of the agents. It's also revealed in our experiments that the proposed method in CBI is effective in helping an agent find the relevant partner to get most informative information. Agents with low correlation of prediction behaviors are most informative and lead to better values for delay precision and recall (see Table 2). Interestingly, the comparison of EXP_6 with EXP_8 indicates that in our experiments, the CAP strategy even manages to outperform a complete knowledge Naive Bayes classifier algorithm. While this partly means that naive Bayes is not the best possible learning algorithm, it does mean that CAP provides good performance in a distributed multi-agent environment that compares favorably with established central algorithms. Also, as described before this result supports Langley's observation in [9] that having more information available to solve a classification problem does not automatically mean having a better performance.

Agent	Correlation	DR	DP
Agent 1	0.32	0.45	0.41
Agent 2	0.40	0.43	0.40
Agent 4	0.77	0.39	0.36
Agent 5	0.41	0.44	0.40

Table 2. Correlation between agent 3 and other agents, and learning performance with CBI

Agent id	Attribute name	Mutual Information
Agent 1	Destination Country	0.104
Agent 2	Origination Point	0.054
Agent 3	National Carrier	0.041
Agent 5	International Carrier	0.088

Table 3. Top informative attribute of different agents

6. Conclusions

A key contribution of this paper is that it integrates techniques and results from distributed learning with underlying organization models constraining the ability of agents to communicate and to make use of the services and computation resources provided by each other. We proposed two multiagent collaborative learning strategies, one of them relying on the exchange of explicit transaction data, the other being based on the communication of qualitative data, i.e., prediction results. Using empirical data from a real-world supply chain application scenario, we could show that both collaborative methods can improve the learning performance of agents endowed with a limited local view, and meets the requirement of different organization scenarios.

The collaborative learning algorithms were developed in the context of extending a pilot logistics decision-support solution for a major logistics service provider, which provides support for order dispatching, monitoring, and re-planning (see [5]). Our hope is to be able to improve the current local prediction algorithms which work fine in a centralized settings, but which do not scale to a decentralized business environment.

Further research will investigate hybrid learning strategies geared towards complex distributed business systems with both heterogeneous information distribution and organization models. Also, the research shown in this paper focused on the influence of the information distribution models on learning performance; we did not consider the effect of varying parameters of the organization model. This will be another topic of future research. Finally, we have not yet theoretically explored the communication overhead involved with the different collaborative learning strategy. In our experiments, however, communication was not a critical factor, since both collaborative approaches attempt to reduce communication. However, a formal analysis of this aspect is subject of future work.

7. Acknowledgments

This paper is partly funded by the E.C. through the Athena IP. It does not represent the view of E.C., and authors are responsible for the paper's content.

References

[1] R. Chen, K. Sivakumar, and H. Kargupta. Collective mining of bayesian networks from distributed heterogeneous data. *Knowledge and Information Systems*, 6(2), 2004.

[2] T. G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.

[3] K. Fischer, J. P. Müller, and M. Pischel. Cooperative transportation scheduling: an application domain for DAI. *Journal of Applied Artificial Intelligence*, 10(1), 1996.

[4] D. Frey, T. Stockheim, P.-O. Woelk, and R. Zimmermann. Integrated multi-agent-based supply chain management. In *Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-2003)*, pages 24–29. IEEE, 2003.

[5] Y. Guo, J. Müller, and B. Bauer. A multiagent approach for logistics performance prediction using historical and context information. In *Proceedings of The Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, New York, 2004. IEEE Press.

[6] N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, M. E. Wiegand, C. Voudouris, J. L. Alty, T. Miah, and E. H. Mamdani. ADEPT: Managing business processes using intelligent agents. In *Proceedings of the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems (ISIP Track)*, pages 5–23, Cambridge, UK, 1996.

[7] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[8] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 399–406. AAAI Press, 1992.

[9] P. Langley and S. Sage. Induction of selective bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406. Morgan Kaufmann, 1994.

[10] P. J. Modi and W.-M. Shen. Collaborative multiagent learning for classification tasks. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 37–38, Montreal, 2001. ACM Press.

[11] J. P. Müller, B. Bauer, and M. Berger. *Software Agents for Electronic Business: Opportunities and Challenges*, volume 2322 of *Lecture Notes in Computer Science*, pages 61–106. Springer, 2002.

[12] B.-H. Park and H. Kargupta. Distributed data mining: Algorithms, systems, and applications. In N. Ye, editor, *The Handbook of Data Mining*. Lawrence Erlbaum, 2003.

[13] A. Prodromidis and P. Chan. Meta-learning in distributed data mining systems: Issues and approaches, 2000.

[14] T. Sandholm. An implementation of the contract net protocol based on marginal-cost calculations. In *Proc. 11th National Conference on Artificial Intelligence (AAAI-93)*, pages 256–262, 1993.

[15] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technology Journal*, 27(1,3):379–423 and 623–656, July and October 1948.

[16] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[17] X. Wang and T. Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. In *Proceedings of the 16th Neural Information Processing Systems: Natural and Synthetic (NIPS) conference*, Vancouver, 2002.

[18] D. Wettschereck, D. W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, July and Oct. 1997.

[19] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[20] Z. Zheng. Naive bayesian classifier committees. In *Proceedings of the Tenth European Conference on Machine Learning*, pages 196–207. Springet-Verlag, 1998.