Collaborative learning agents supporting service network management

Mulder, W.; Meijer, G.R.; Adriaans, P.W.

Published in:
Service-Oriented Computing: Agents, Semantics, and Engineering

DOI:
10.1007/978-3-540-79968-9_7

Collaborative Learning Agents Supporting Service Network Management

W. Mulder[1,2], G.R. Meijer [1], P. W. Adriaans[2]

[1] LogicaCMG, Prof. Keesomlaan 14, 1180 AD Amstelveen, Netherlands
{Wico.Mulder, Geleyn,Meijer}@logicacmg.com
[2] University of Amsterdam, Kruislaan 419 Matrix 1, 1098 VA Amsterdam
adriaans@science.uva.nl

Abstract. Service oriented systems need to be maintained to keep the requested level of service. This is challenge in large grid- and saas based networks that are managed by numerous entities. This paper is about supporting multi agent systems that operate in the network and support its management by learning actual structures from life observed logging data. We focus on a collaborative grammar induction mechanism in which agents share local models in order to retrieve a model of the structure of the total service network. We studied the performance of groups of agents while varying the size and degree of communication. We motivate the application of the mechanism in the domain of service oriented system and show the results of experiments using a distributed agent-based monitoring system. We promote further research in the overlapping scientific disciplines of multi agent systems and machine learning in the application domain of service oriented systems.

1. Introduction

Service networks are networks of computer systems that are used to deliver end-user applications in a dynamic and personalized way. They are highly scalable, heterogeneous, and operate in agile, demand-driven environments. Service networks are interesting from business as well as technical perspectives; we have the business of cooperative industrial organizations providing federated services towards consumers, and we have the technical ICT infrastructures that provide, support and enable these business services in a dynamic and personalized way. These infrastructures consist of hardware and software, are based on service oriented architectures and often consist of web-services that work together in various end-user applications.

Services networks need to be maintained to keep the requested level of service. This is a particular challenge in the case of distributed networks in which the nodes are maintained by separate entities, such as in a grid based network1. Large service networks can be complex in terms of dimensions, interactions or level of

[1] http://www.nessi-europe.com

heterogeneity. We study the operational management of such networks, and focus on maintenance support by means of autonomous software agents [12]. While operating in the service network itself, the task of these agents is to provide information about the status of that network and the services that are provided by it. In order to deal with the complexity of the network, the agents are designed to be adaptive, and share information with each other. A constraint often encountered in service networks is that its components only have access to local parts of the network, which is often combined with communication constraints due to security reasons or limited physical bandwidth. We believe that the dynamic and heterogeneous aspects of the environment in which the agents operate force them to collaborate in their learning and information provisioning tasks. In this way, the agents form a network themselves, providing a robust and redundant way of information provisioning and management support. Figure 1 shows the situation in which an agent network analyses the status of a service network and support the responsible network managers in their task.

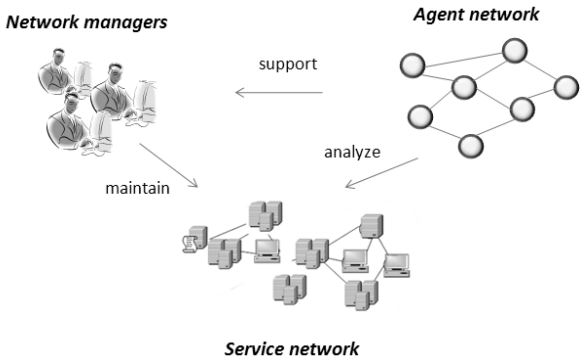

Fig. 1. service network management support

In previous work [9] we talked about Collaborative Information Services in multi domain networks. In this paper we focus on the collaborative learning mechanism of these agents. We look at the application scenario in which the agents support network managers and system administrators to obtain the actual status and structure of a complex service network. The agents look at different sets of provenance[2] data and try to induce a grammars and structures in it. This is done by means of DFA learning, as explained in the next section where we propose a distributed DFA modeling approach.

The field of grammar induction is a well known area that has been studied from many perspectives during the last decades [6]. In a grammar induction process, a learning algorithm is used to obtain a grammar that should explain the structure of a given set

[2] Wikipedia: *Provenance* is the origin or source from which something comes, and the history of subsequent owners. Provenance information of some data the documentation of the process that led to the data [3]. It can be generated from the static information available in original workflow specification together with the runtime details obtained by tracing the execution of the workflow.

of data. This grammar represents a model of the dataset. The aim is to learn from sample data (usually a list of words) an unknown grammar which explains this data. The model is also used to verify whether unknown samples follow the rules of the grammar.

A Deterministic Finite Automaton (DFA) is a common algorithm used to classify structures (languages) and represent grammars in the form of graphs 0. A DFA can be seen as a model which captures the underlying rules of a system, from observations of its behavior or appearance. These observations are often represented by strings that are labeled "accepted" or "rejected". Every string in the dataset is represented as a path in the graph. Figure 2 shows an example of a DFA-tree for the strings *abcd* and *abcbcd*. By merging or clustering data using heuristics the algorithm learns to represent the data in a more structured way.

Since creating some DFA that is consistent with training data is trivial, it is usual to add two further constraints, that the DFA should generalize to unseen test data and that the challenge is to find the smallest DFA that is consistent with the training set. For the latter, we use MDL (minimum description length) as a criterion.

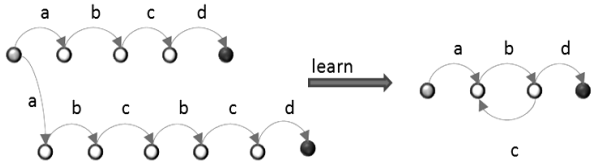

Fig. 2. example of a unfolded DFA (left) and a folded (learned) DFA (right)

In our experiments, we use DFA as a common grammar induction method for learning individual as well as collaborative (global) models. We take samples of provenance datasets and define a set agents that learn local topology structures. The agents observe data and communicate the induced structures with each other. The goal is that each agent learns a model of the dataset as a whole.

The rest of the paper is organized as follows: Section 2 discusses our agent model and the strategies used for communication and learning. In section 3 we show the results of our first experiments. Section **Fout! Verwijzingsbron niet gevonden.** contains discussion and ideas for future work, and section 5 ends with the conclusions.

2. Introduction

Our agents are designed to learn grammar in a collaborative way. We used groups of agents of different sizes, in which each agent analyzes a part of a given dataset. They observe a part of the dataset, learning a grammar from it and communicate the results with the other agents. This is done by means of two separated models per agent and a series of chosen strategies, which will be explained below.

Each agent keeps two models of the dataset; the first model, called the *individual model* reflects the structure of the observed dataset; the second model, called the *collaborative model* or *global model,* reflects the structure of the dataset as a whole. Both models are in the form of a DFA. By means of these two models per agent, we intend to have a clear separation between its local information and its shared information. Taking into account the constraints of the application domain, we cannot always communicate the original samples of the individual datasets. Therefore we have chosen to communicate the models and generate new samples from them at the moment of their arrival at the receiving agent. Figure 3 shows a schematic picture of a set of agents and their models. The arrows indicate the flow of information.

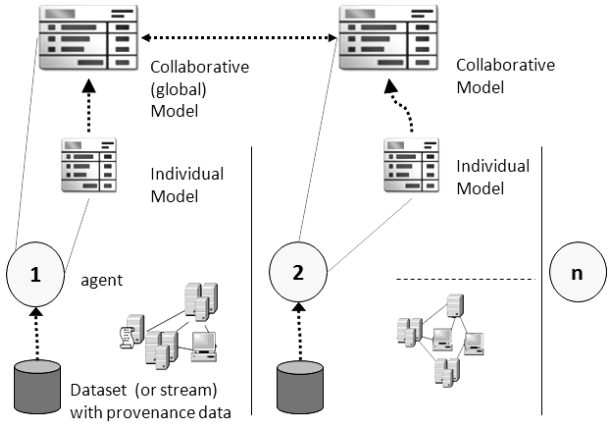

Fig. 3. n agents analyzing (workflow) data. Each agent observes data from its own local environment. The vertical lines denote the constraint that the agents do not (always) have access to the datasets of each other.

We designed our collaborative learning mechanism to work with four types of strategies: an *individual learning* strategy, a *collaborative learning* strategy, a *dispatch* strategy and an *acceptance* strategy.

The individual learning strategy reflects the way that an agents learns its individual model. In our case this is grammar induction using DFA learning. The collaborative learning strategy describes the way the collaborative model is maintained. In our experiments we combine generated samples from the individual model and incoming models from other agents and use them to build a DFA tree.

The communication process of the agents is characterized by a *dispatch strategy* and an *acceptance strategy*. Agents share their collaborative model with each other in the form of messages, called DFA-hypotheses. The dispatch strategy defines *to whom*, and *when* this communication takes place as well as the information content, the *what*, or simply the *hypothesis content*. When receiving these hypotheses, an agent uses its

acceptance strategy to judge whether it should accept the incoming hypotheses and how they are merged with their own collaborative model.

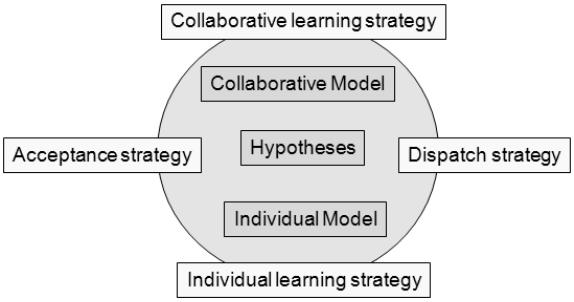

Fig. 4. an agent's internal models and strategies

While observing the local data sets, the agents update both, their individual model as well as their collaborative model. First, after taking a number of samples, the individual model is trained. Then data for the collaborative model is assembled from both, the individual model as well as models from other agents. The updated collaborative model is then shared again with the other agents.

Sharing takes place by communicating the model, and merging it at its arrival with the existing model. In our experiments we simply merge by building a new model from a number of generated samples from both, the previous model and the incoming one. The goal is that each agent has a good model of the total dataset.

Note that each agent has its own collaborative model, but mechanism allows these models to be slightly different (like in Plato's theory of Forms). It is common to share information between agents using blackboards [1][7]. In our mechanism the agents create and share their own collaborative model which can be regarded as an implicit, redundant, distributed blackboard.

3. Experiments

We used small datasets containing strings that represent. workflow execution orders of web-services situated in two isolated environments of a service network. An example of such a situation is shown in figure 5.

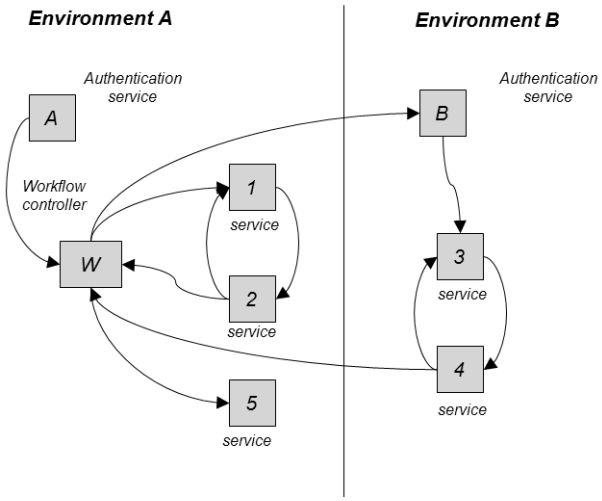

Fig. 5. webservices and workflow orders in two isolated environments

We looked at the models of the two agents, one observing data from environment A, the other observing data from environment B. An example of a learned DFA model is shown in figure 6.

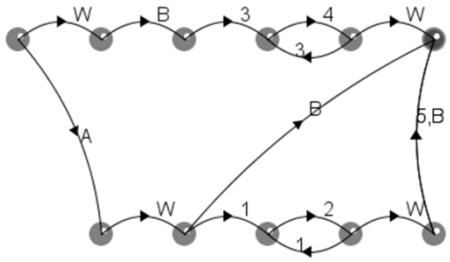

Fig. 6. a DFA model obtained from both agents

We compared the collaborative models with models obtained by a single agent observing the data from both environments. Except from extra end-states, they were found to be similar.

In order to study our collaborative learning approach in detail, we carried out a series of experiments in which agents are allowed to take randomly a number of samples from a shared dataset. The agents learn individual DFA structures and share their collaborative models in order to model the total dataset.

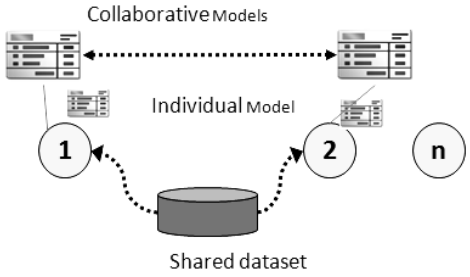

Fig. 7. multiple agents observing a shared dataset

We studied the performance of the group while varying the number of agents (**n**), number of learning steps (**t**), the number of samples taken per learning step (**m**) and the number of samples generated from a particular model during a merge process (**s**). In each experiment we took the same web-service scenario and used a dataset of 20 different samples.

The structure of an experiment is given below.

```
Experiment (n,t,m,s):
    n agents, for each agent:
        Start with new individual model and collaborative model
        Repeat for t learning steps
            take m samples from the training set
            add unique samples to the individual model
            learn DFA from individual model samples
            generate m samples from the individual DFA
            add generated samples, if unique, to the collaborative model
            generate s samples from each incoming hypothesis
            add generated samples, if unique, to the collaborative model
            learn DFA from collaborative model
            send DFA as hypothesis to other agents
        until t learning steps
    take whole trainingset
    verify DFA of collaborative model of each agent, determine mean score
    sum mean scores and take mean and std.
End experiment
```

We developed an agent framework[3] allowing us to control the experiments and study the behavior of the agent network for different values of n,m,t, and s.

The collaborative model of each agent is stored in a result-database allowing us to validate and verify the performance of the agents. The collaborative model of each agent is used to classify the samples of the whole dataset. The mean number of

[3] Existing frameworks, such as e.g. Jade, are under investigation of using instead.

samples classified as valid to the grammar, is defined as the score per agent. The mean score of a set of agents indicates the score of the particular experiment.

For a single agent, varying the value of m, the results are shown in fig 8. The figure indicates that for this particular dataset, the number of samples to learn a complete model for a single agent is roughly the number of samples in the dataset, which is 20.

The variance in the score strongly depends on the structure of the individual samples: during a particular experiment, samples are taken randomly and generic samples might give a valid classification result for other samples as well.

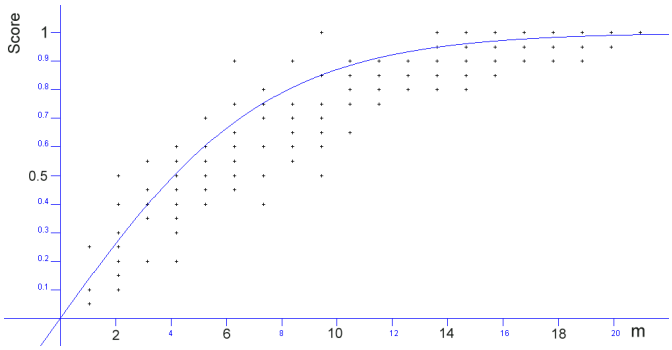

Fig. 8. score-graph for n=1, t=1, s=0, m=1..20, the fitted logistic curve has a=1, b=0.28.

The growth of the score behaves exponentially with the increase of m and saturizes to the value of 1. The steepness of this growth reflects the learning performance of the agent network for this particular dataset. We fitted the score as a function of m to a logistic curve[4], where the values 1 and 2 are used to scale the offset, parameter a is taken to be 1, and parameter b fitted as an indicator of the steepness of the learning behavior.

$$f(x) = -1 + \frac{2}{1 + ae^{-bx}}$$

For a network of two and four of agents, we varied m, s and t. Figure 9 shows the results of two agents (n=2). The fitted value of b increases with the number of samples during a merge (s) and the number of learning steps (t). This means that the score climbs faster to 1 or, in other words, the agents learn from each other's examples.

[4] The logistic function has applications in areas of population statistics and biology. An example can be found in Rasch-modeling theory [11] where the probability of responses is modeled using as person and item parameters. In our model, in similar ways, the score of the agent is a mean of individual scores which in their turn are based on accepting individual samples.

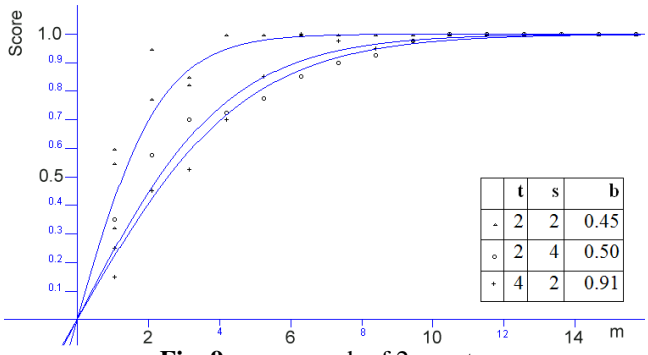

Fig. 9. score graph of 2 agents

Figure 10 shows the same, but then for a network of 4 agents.

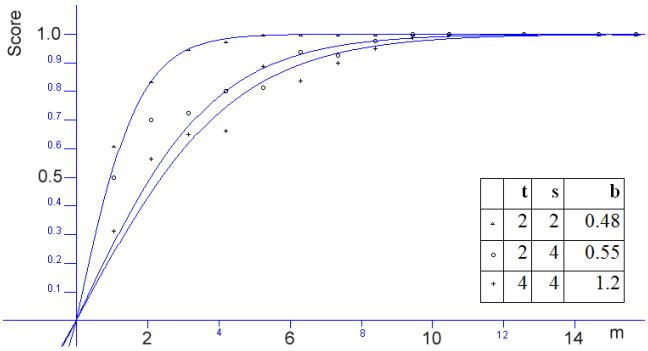

Fig. 10. score graph of 4 agents

4. Discussion and Future Work

The area of distributed learning recognizes that in many cases agents cannot simply solve problems individually and need to combine their models. Shen and Lesser [10] have studied Distributed Bayesian Networks, where agents generate local solutions based on their own data and then transmit these high level solutions to other agents.

Network management using agents is being studied in the fields of grid computing and provenance management. Forestiero [4] studies ant-based resource management and discovery where agents copy and move resource metadata among grid hosts. Feng [3] describes a decentralized provenance recording and collection mechanism in which mobile agents collect information about jobs in workflow executions.

In our research we focus less on the actual distribution of provenance data. Instead of proper metadata management, we study the learning behavior of a set of communicating agents having collaborative models.

Since DFAs are common and fundamental in the area of unsupervised learning, we used this algorithm in our learning tasks. We do not try to improve DFA algorithms themselves, but focus on the collaborative learning behavior of the agents. Since the agents take into account hypotheses from other agents whilst they are learning themselves, our mechanism can be regarded as a 'distributed on-line learning mechanism'.

In our design and implementation of the prototype we took in account that the strategies, which are currently rather simple and straightforward, can be replaced by other, more sophisticated ones. We intend to improve the communication strategy including decision and dissemination algorithms in which receiving agents can actively ask for information as well.

We want to improve the mechanism of merging incoming hypothesis-DFAs with the model of a receiving agent using genetic algorithms; rather than searching for the best hypotheses to be taken into account for merging models, mutation and recombination of the best currently known may lead to evolutionary learning behaviors. An agent that receives hypotheses can learn to choose optimal (listening) actions to achieve its goal. As a feedback, an agent might provide a reward or penalty in reaction to an incoming and accepted hypothesis. This could be done by comparing the fitness of the collaborative model before and after the included hypothesis. On the level of meta-learning, we think of using a kind of feedback to the individual learning process; the learning process of the agent itself might be affected by incoming hypotheses from other agents.

Last but not least, for the work described in this paper, we used simple example data. Since our motivations for this research are based on expected needs and constraints in the application domain of service networks, we plan to apply our methods in this area dealing with real workflow execution data.

5. Conclusions

In this paper we talked the application domain of service oriented systems and dynamic infrastructures, in which these agents can support the operational and technical maintenance. We focused on the provision of workflow topology information obtained from provenance datasets , explained the architectural model of our agents as well as our approach of distributed DFA learning.

We have presented an approach for distributed grammar induction using collaborative agents. We showed the results of experiments in which a agents learned individual DFA models from local datasets and shared these models in order to obtain a DFA model that represents the total dataset.

We showed the results of our experiments where different groups of agents learned structures from a shared dataset, while sharing their intermediate results with each other. We analyzed the learning behavior of the agent network, and showed the relationship of its steepness with the number of agents and level of communication.

We suggested a number of improvements and promoted further research in combined fields of machine learning, multi agent systems and service oriented systems.

References

[1] O. Cicchello and S. C. Kremer. Inducing grammars from sparse data sets: A survey of algorithms and results. Journal of Machine Learning Research, 4:603–632, 2003.

[2] Daniel D. Corkill. Collaborating software: Blackboard and multi-agent systems & the future. In Proceedings of the International Lisp Conference, New York, New York, October 2003. (Invited presentation.)

[3] Yuhong Feng, and Wentong Cai, “Provenance Provisioning in Mobile Agent-based Distributed Job Workflow Execution”, accepted by 7th International Conference on Computational Science (ICCS 2007), Beijing, China, 27-30 May 2007.

[4] Forestiero, A., Mastroianni, C, Spezzano, G., A decentralized ant-inspired approach for resource management and discovery in grids, Journal of Multiagent and Grid systems, 2007, 43-63, IOS Press

[5] Gannon, D., Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications, Journal of Cluster computing, Special issue on Grid Computing, July 2002

[6] Higuera, C de la, "A bibliographical study of grammatical inference", Pattern Recognition 38 (2005): 1332—1348, Elsevier Science

[7] Jiang Y.C., Yi, P., Zhang, Z.Y.,Zhong, Y.P., Constructing agents blackboard communication architecture based on graph theory, Elsevier Science,Computer Standards & Interfaces 27 (2005) 285–301

[8] Mitchell, T. Machine Learning, McGraw Hill, 1997, ISBN...

[9] Mulder, W., Meijer, G.R., Distributed information services supporting collaborative network management, in IFIP International Federation for Information Processing, Volume 224, "Network-Centric Collaboration and supporting frameworks", Proceedings PROVE06, p. 491-498, Springer, ISBN 0-387-38266-6, 2006

[10] Shen, J., Lesser, V., Communication Management Using Abstraction in Distributed Bayesian Networks, in Autonomous Agents and Multiagent Systems, 2006, AAMAS 2006, Proceedings of the fifth international joint conference on, ISBN:1-59593-303-4 p. 622 - 629

[11] http://en.wikipedia.org/wiki/Rasch_model, http://en.wikipedia.org/wiki/Logistic_function

[12] Wooldridge, M. MultiAgent Systems, Introduction to MultiAgent Systems, John Wiley and Sons, 2002, ISBN 047149691X