Self-Calibration: Enabling Self-management in Autonomous Systems by Preserving Model Fidelity

Fahad Javed, Malik Tahir Hassan, Khurum Nazir Junejo, Naveed Arshad, and Asim Karim
Dept. of Computer Science,
LUMS School of Science and Engineering
Lahore, Pakistan
{fahadjaved, mhassan ,junejo, naveedarshad, akarim } @lums.edu.pk

***Abstract*—Autonomic and autonomous systems exist within a world view of their own. This world view is created from the training data and assumptions that were available at their inception. In most of these systems this world view becomes obsolete over time due to changes in the environment. This brings a level of inaccuracy in the autonomic behavior of the system. When this degradation reaches a certain threshold self-healing or self-optimizing systems generally recreate the world view using current data and assumptions. However, the self-optimization process is akin to kill a fly with a hammer for minor tuning of the world view. Instead we propose the idea of self-calibration for self-managing these systems. We define self-calibration as the ability of the system to perceive the need for and the ability to execute minimal tuning to bridge the gap between system's world view and incoming information about the outside world. In this paper we present a case for considering self-calibration as a self-* enabling property of systems specifically for time-critical systems using data-centric AI technologies. We present our case by discussing three case studies from different domains where self-calibration enables a system to become self-healing or self-optimizing. We then place self-calibration in a generic system and explicitly describe the types of systems in which self-calibration can be implemented and the benefits that one can accrue from its inclusion.**

I. Introduction

Autonomous intelligent systems in general create a world view to reason, analyze and plan their actions. This world view, or system's internal model, is created from various sources including the training data, assumptions, and modeling artifacts available at time of inception. It is accepted that this world view is an approximation of the actual world in which autonomous intelligent system exists. Furthermore, as the world evolves over time accuracy of representation of this model is expected to drop.

Our interest is to maintain an appropriate model of the world despite the changes in the world so that autonomous system operates accurately. Online models are a solution for such problem [6], [17]. However, for a range of time critical applications such as spam filtering, demand response systems in smart grids, web 2.0 data handling etc., online algorithms are not suitable due to various factors. These factors primarily are concerned with response time and scalability issues with respect to size of data and its dimensions. For these time-critical systems, the state of the art is to model offline and later use this model to make its real-time decisions. But this brings us back to the issue of timely adaptation to changes in the world.

A possible way to solve this problem is to make the autonomous intelligent system self-managing so that it maintains fidelity of its model with the world. Based on the system's operations this can produce self-healing or self-optimizing or even self-protecting properties in the system. But if we look at this task through the generally accepted self-* definition or analyze operational requirements to maintain model fidelity we see that this task is subtly different from self-healing, self-optimization and self-protection.

As we will argue in section II, the task of maintaining model fidelity is more akin to intelligent, automatic calibration of system than the task of maintaining health or optimizing system's performance or protecting system from threats. Though this task of maintaining model fidelity results in a self-healing or self-optimizing system, the task is inherently different from the accepted norms of traditional self-* properties.

We can also compare this task to adaptive control. Adaptive controls or self-tuning controls calibrate their controllers according to the changes in environment. For instance, a common example is the adaptive control of an aircraft. The parameters to control an airplane are dependent on weight of the plane. But as the plane flies, it's weight changes due to consumption of fuel. Adaptive controllers adapt their working by observing the system and correcting themselves due to change in environment. Adaptive controllers such as model reference adaptive controller (MRAC) calibrate the base controller against a reference model to correct the system errors due to environmental evolution.

In various applications, however, standard controllers are not applicable and other intelligent computing solutions are applied. For these systems, calibrating of the controller or the autonomous manager is not well defined according to our study.

In this paper we present this concept of self-managing calibration or self-calibration for data-driven time-critical systems. We provide an abstract system definition and de-

scribe ways to achieve self-management of model through self-calibration in this abstract system. In addition we describe two architectures to implement self-calibration in this abstract system. To ground the problem we provide case studies from three different domains- web 2.0 application, spam filtering, and sensor networks- which benefit from use of self-calibration. Our contributions in this paper are (i) We describe how self-management of a model can be achieved through self-calibration in an abstract system. (ii) We define two architectures which can be used to engineer self-calibration in the class of systems we consider in our scope. (iii) We provide case studies to ground the generic system in concrete examples.

The paper is organized in the following way: Section II discusses the reason for proposing self-calibration. We follow this we definition of self-calibration in section III. In section IV we describe an abstract model to position and define self-calibration. Section V describes the process of incorporating self-calibration in the abstract system. This is followed by case studies, future work and conclusion.

II. Motivation

A range of time and content driven autonomous intelligent systems such as spam filtering software, forecasting engines, online tag recommendation, etc. are based on off-line learning algorithms. Here by content driven engine we mean systems where operations, goals or primary services are dependent on and driven by content. In these systems the costly step of model creation is done off-line and decisions are made during execution using this model [12], [13], [14], [15]. Since the content changes over time, the static off-line model becomes obsolete after some time resulting in drop in correctness of the system. On the other hand, current online algorithms are not viable due to time constraints, or scalability issues with respect to large variable space or data size. There are two measures of concern here: accuracy and efficiency. Accuracy is the measure of correctness of system and efficiency is the measure of how the system operates. This includes response time, resources utilized, etc., Whereas an off-line model affects the accuracy, online models affect the efficiency of the system.

We observed that the degradation of system is dependent more on the model's ability to reflect the world adequately and less on time passed since the inception of the model. The accuracy of the system remained healthy while the distribution of knowledge in input stream was similar to the knowledge of training data used to create the model. But degradation results in some cases quite rapidly when knowledge obtained from the input stream diverges from the model [12], [15]. This evolution of the world is the major cause of degradation of service.

There are two characteristics of the effect of world evolution on the system and its model.

- First, the fall in accuracy is usually due to a small subset of misclassified new knowledge about the environment. At any given time multiple small subsets may affect the correctness of the model.
- Second, the fall in accuracy is based on evolutionary changes and occur with unpredictable frequency and size. Usually in a large system many small scale changes occur and if self-awareness is only tuned to total system efficiency then these small scale changes go under the radar.

In such systems, we have seen self-optimization [11] or self-healing [14] as a possible solution. However, self-optimization, or self-healing, properties by their very definition are not able to respond effectively to the evolution of the world [4], [16]. Self-healing is the property of system to heal itself against bugs or failures. A self-healing system may be able to identify a drop in efficiency due to partial failure of model. But a deviation of real world from system's world view cannot be categorized as a failure of system specially if individual deviations are too small to warrant action in the total system. Secondly, self-healing techniques try to fix a problem but here the goal is to adjust model to improve efficiency as compared to restoring some property which was lost due to a bug or a failure.

Self-optimization seems like a more appropriate property but self-optimization is not geared to sensing failures to correct but is rather looking for ways to improve performance. Usually self-optimizing systems either try to maintain a threshold of performance or periodically check for an opportunity to improve their performance. For a large scale system where multiple small scale inflow are happening, neither of the methods can capture the error and fix it. If a threshold is being maintained then it is possible that a number of inflow have made the model quite obsolete by the time threshold is reached. This will necessitate a complete or atleast a major relearning of the model. In case periodic opportunity for optimization is checked then it may happen that between two epochs major changes occur and again model is sometimes changed beyond repair. We will discuss these issues in more detail with respect to the system model in section IV.

Our argument is that, since the downgrade of correctness is due to minor changes in some parts of the world, the solution to these shortcomings should also focus on accommodating these changes by minimally tuning the model locally. This will make these adaptations efficient enough to update the model at runtime resulting in a more accurate yet scalable system.

III. Defining Self-calibration

To undo the effects of these evolution-induced changes, we explored calibration of system when the system model and the sensors capturing the state of world do not match.

Calibration is defined as:

Definition 1. *A minimal tuning, filtration, or characterization of raw information about the system's environment to reduce the gap between system's world view and the actual world.*

Calibration is referred to as the process of adjusting system to remove systematic errors in sensor readings. The term has also been used to refer to the procedure by which raw outputs of sensors are mapped to standardized units [7]. Various engineering and professional disciplines use calibration to adjust their instruments' model in a new environment. In essence, when the internal model of the system and it's sensor readings are out of sync, calibrations are done to adjust the differences in model and sensors. In sensor networks, robotics, and computer vision, such calibrations are needed when the system is deployed in a new environment. There are automated methods for calibration but the process is explicitly initiated by the user deploying the system and is done once for an environment [21], [24].

Autonomous systems in contrast very rarely change their physical or operational environment. However, such systems are faced with the problem of an evolving environment where new terms, tags, devices, etc., are added, replaced, updated, and removed frequently. Bychkovskiy and colleagues argued that calibrations can be a good solution for autonomous systems [7]. But, there are two aspects of autonomous systems that contribute in making simple calibration a complex task.

First, for traditional calibration, it is explicitly initiated at a certain time in the life cycle of device. This is not possible for online autonomous systems. Calibration in these systems is a tool to handle changes over time. As we will show in our case studies the need for calibration is unpredictable at design time. Thus we can not define a specific point in time in life cycle of system where calibration will take place. Rather we would have to intelligently identify when we will calibrate the system.

Second, usually there are numerous variables, sensor streams, and tuning parameters in an autonomous system. Deciding how to effectively calibrate the system is a non-trivial task.

To execute effective re-calibration of such system we propose self-calibration as a desired property of time and content driven autonomous and autonomic systems.

We define a self-calibrating system as:

Definition 2. *A system which is able to:*

1) *Identify the need for calibration by intelligently observing relevant raw information and the internal constructed system's world view*
2) *Instrument appropriate calibration actions on the system to minimize the gap between system's world view and the actual world.*

That is, self-calibration is an ongoing process in which system first identifies the point where system model and actual world diverges from each other. It then initiates processes to reduce this gap whenever and however possible. This property as is evident from definition and its application, is not a goal in itself like self-optimization or self-healing. Rather it is an enabling property which supports efficient self-healing and self-optimization by using observational traits of self-healing and self-protection. This ultimately helps in planning and executing a boost in efficiency to provide self-optimization property to the system.

A. Desirous Attributes

Based on the definition of self-calibration we argue that following attributes are required to achieve self-calibration.

1) **Situateability:** First and foremost, the system should be able to situate its data with respect to its model. For autonomous system, the system model is the description of its world. But the real world generating the input stream may have changed since the inception of system model. An autonomous system requires a sensor/analyzer to situate its model with respect to the input stream to identify error in the data-stream, sensor, or the model.
2) **Input Stream Transformation:** System should be able to regulate its input stream to protect its model, correctness or efficiency from temporary events in the world such as data burst or spikes. For example, the personalized spam filter discussed in section VI.2 needs to identify if the updated model is required or not.
3) **Model Transformation:** System should be able to update its model to incorporate changes in the world. This model transformation should reduce the distance between system's internal model and the observations from the world. The model transformation should be minimal to reduce the overhead of self-calibration.
4) **Metrics and Thresholds:** System should have metrics to evaluate data fidelity as a temporary event or a valid change in the world. System should be able to use these metrics and thresholds to trigger input stream transformation, or model transformation. Metrics should also identify when self-calibration will fail and model regeneration will be required.

We define a system that fulfils the above mentioned requirements as a system with self-calibrating property. That is, a system is self-calibrating if it is able to analyze and compare its internal model with the world and tune data and/or model to bridge the gap between system's world view and incoming sensor input stream.

IV. Analytical Model

In this section we describe the abstract system and its property to reason about self-calibration. This will help us define generically, the types of system models which can be made self-managing through self-calibration. We also define boundaries for its usage under the scope of this paper.

First, we build a generic system and world abstraction. Then we will place the performance measurements of interest within the scope of this abstraction followed by definition of self-calibration in this generic world.

A. System Model and Notation

We build on the abstract model proposed by Berns and Ghosh [4]. Whereas they restricted their model to system components, our discussion necessitates abstracting the living and breathing world in which our system operates. We represent this world as R^a. For our system this world produces a set of observations r_t at time t and an output out^a that we can represent as: $[r_t, out^a] \Leftarrow R^a(t)$.

An autonomous system tries to capture these observations and replicate processes in $R^a(t)$ to generate out^a. That is the system is given as: $[out^s] \Leftarrow S(r_t)$. Here S is the autonomous system, r_t are the observations from R^a observed through sensors of S and out^s is the system output to replicate out^a.

Internally S maintains a state X_i. This internal state $S = \{X_i\}$ represents the system and transitions non-linearly over time. X_i has three possibilities: $X_i \in \{C \cup F \cup D\}$. C is the set of sound or correct state, F is the set of faulty state and D is the set of transient state degraded states. These states are roughly modeled on the survey of self-healing system by Ghosh and colleagues [10].

The state in turn is composed of system model M^s and internal state variables x_v. That is: $X_i = \{M^s, x_v\}$. As part of state, the system constructs a model M_t^s which attempts to construct the world R^a using historical trace of observations from some time $t - \gamma$ till t.

$$M_t^s \Leftarrow train(r_{t-\gamma}..r_t)$$

We would now discuss the world R^a and its hypothetical working. It can be argued that R^a builds its output based on some model M^a for every time t. As an external entity, we do not know of this model and can only infer it through learning its historical outputs and observations. Hence we can only approximate M_t^a through some model M_t^s.

This model M_t^s is our best way of capturing the working of world R^a. Online algorithms continually update their model M_t^s based on their observations as is the case of model adaptive control. However, in offline systems this model is not constructed continuously. That is, M^s is created at discrete times and same model is carried over till next model creation is initiated.

Possibly infinite modeling paradigms exist to model the world. However, for the scope of this paper we classify two types of models (M^a) of the world (R^a) which are relevant to the type of systems we are working with. First is a singularity world which is based on a single phenomenon. By singularity we mean phenomenons which represent a single possibly complex system such as controlling an aircraft control planes or forecasting energy load of a system. Self-calibration is well defined under the title of adaptive controls for these systems in literature of control theory and time series analysis. We do not see a need to revisit or reform it to our scope or define such system in our generic definition.

In comparison a constructed world is an aggregation or a composition of multiple sub-components under a single environmental or operational assumption. In such constructed world many components combine or contribute to form R^a. In these systems many independent sub-models or sub-component contribute to form M^a. If we define domain of R^a as the total space world occupies then each component k of the world would represents a sub-space such that:

$$R^a \sum_k {}^k R^a$$

Here superscript prior to variable R represents the k^{th} component of R. When we try to approximate such world model as M_t^s then we see that M_t^s can be in a continuum between generalizational and compositional models which we define as under.

Compositional models try to model the sub-spaces of R^a. These sub-spaces are modeled and combined to form the total system. We can say that the system model M_t^s is sum of j components $M_t^s = \sum_j {}^j M_t^s$ such that a sub-model ${}^j M_t^s$ models one or more components of space R^a. In application parlance, this is akin to clustering of data or maintaining an ensemble of classifiers where a cluster or a single classifier represent j^th component of ${}^j M_t^s$.

In comparison, **generalizational models** are those in which M_t^s is an approximation or generalization of its j components i.e. $M_t^s \approx \forall j {}^j M_t^s$ with some discriminative method to differentiate between ${}^j R^a$ regions. We can also conversely say that ${}^j M_t^s$ is a specialization of a generic model M_t^s where M_t^s represents the total world R^a and ${}^j M_t^s$ represents the k^{th} component of R^a.

It can also be visualized as a system where there are j subsystems with some commonality among them. The total system is an average output of each of the independent subsystem and is constructed in a way so that M_t^s is an average of the entire system.

For these systems When we simulate M_t^s under observations of j^{th} component (${}^j r_t^a$) then it produces output ${}^j out_t^s$ appropriate for the j^{th} component. An example of generalizational model is our spam filtering case study where each mailbox is a component. Global spam filtering constructs a model for all the mailboxes (or senders) through a unified corpus of labeled data. This global spam filter

models all mailboxes to a certain accuracy level.

B. Measurements

It is imperative that the system is observed and measured to verify if it is achieving its intended goals. For this purpose both functional and non functional measures are of interest to ascertain the health and effectiveness of a system. We may evaluate a system through a function ρ which calculates instantaneous performance at time t given as P_t.

$$P_t \Leftarrow \rho(X_i, r_t, out_t^s, out_t^a)$$

This performance is based on overall system state and in most cases the input stream (r_t). There are three important components of P_t that are of interest to us. First and foremost is accuracy that we can quantify as:

$$accuracy = |out^a - out^s|$$

The second measure is of safety and liveness to measure health of the system. We use Alpern and Schneider's definition of the terms [2]. A system may have Y safety and L liveness properties. Informally safety property implies that "bad things never happen to the system" and liveness implies that "good eventually happens".

The third important measure is performance which can be measured as response time of system or as resource requirements to achieve some quality requirement. Measuring performance in abstract system is not feasible due to limitation posed by abstractions. However, we will discuss effect of self-calibration on performance in our case studies.

C. Self-Calibration

We describe self-calibration in our abstract world. A description of other self-* properties in similar context can be found in [4]. We define self-calibration as a function selfCalib of model M^s and input stream $r_{0\cdots t}$. According to our definition of self-calibration, it is the minimal tuning of model or input stream to reduce disparity between system model and actual world. That is:

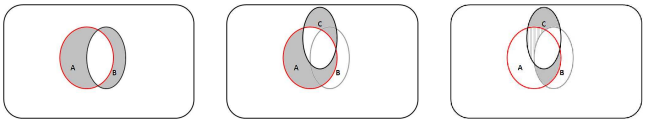
$$(M_{i+1}^s, r_t') \Leftarrow selfCalib(M_i^s, r_t)$$

such that $|M_{i+1}^{s'} - M^a| < |M_i^s - M^a|$. That is, $selfCalib$ is a function that reduces the distance between the real world model and the system's model.

This in itself is not a critical property of a system like self-optimization or self-healing. But as we have shown previously, it enables self-optimization and self-healing in an efficient manner. Next we will discuss how we can engineer self-calibration in the abstract system.

V. Engineering Self-calibration

In the previous section we have described an abstract world for discussion on system and world model and definition of self-calibration. In this section we propose two architectures to implement self-calibration for compositional and aggregational models.



(a) (b) (c)

Figure 1. Venn diagrams showing sets of out^s and out^a for (a) time t (b) time $t+i$ and positive and (c) negative and positive changes from time t to $t+1$. negative is shown colored grey and positive is shaded with vertical lines.

Autonomous data-driven time critical systems create a model to reason about the world around them to make decisions. The goal is to mimic world model M^a as much as possible in the internal model M^s. Though this modeling may not be perfect but correcting this error is beyond the scope of this paper. Our focus in this section are architectures to handle the issue that arises when evolution in world increases the gap between M^a and M^s over time.

We illustrate our point through venn diagrams in figure 1. This system represents a binary classification where set A represents output of M^s and B represents output of M^a at time t in figure 1(a). Without loss of generality we can extend this model to multi-dimensional domain and multivariate decision making.

The error of the system, by set expression is:

$$error_t = (B - A) + (A - B)$$

At some time $t+i$, M^a evolves into C as shown in figure 1(b). the error at $t+i$ is represented as:

$$error_{t+i} = (C - A) + (A - C)$$

Here we are interested in positive change in the error or Δ_{error} which is shown as shaded region in figure 1(c). This in essence is the error that did not exist at time t and is caused due to evolution in R^a. The expression is:

$$\Delta_{error} = \big(C-(A+B)\big)+\big((A+B)-C-(A-B)-(B-A)\big)$$

Self-calibration specifically focus on the harmful effects of the evolution and not consider the positive changes due to it. As can be seen in figure 1(c)), the shaded region is a positive change in system's output due to evolution. If we consider total system performance or accuracy then we may not see much change. But if negative affects of evolution can be captured and fixed as they occur then this can save us from larger and costly wholesale corrections later in the life of system as is the case for self-optimizing tag recommendation [11].

Thus the goal of self-calibration is to reduce Δ_{error}. We propose two architectures which can achieve self-calibration in such systems and result in self-healing or self-optimization of the over all system. The selection of architecture is dependent on the nature of system, existing

autonomous management algorithm and the flavor of self-management required.

There are two tasks of self-calibration, first is to identify when calibration is needed and second is to instrument the calibration. In this discourse we will identify the point at which calibration is needed and object on which calibration needs to be done. The actual method of calibration is domain specific. This will be discussed with more details the in case studies.

A. Architecture 1: Sub-model Self-calibration (SMSC) for Compositional Models

Our first architecture is for systems constructed for a world R^a which is composed of k components such that:

$$R^a = \sum_k {}^k R^a$$

Figure 2 represents such a system. Here R^a is segmented internally into k segments. An autonomous system ideally would identify each region and map it to its internal sub-model ${}^j M_t^s$. Combining all ${}^j M_t^s$ sub-models will cover the domain R^a. This modeling may have error such as incorrect mappings and overlapping regions. resolving this error is beyond the scope of self-calibration.

The goal of sub-model self-calibration (SMSC) architecture is to autonomically calibrate ${}^j M_t^s$ with ${}^k R^a$ as ${}^k R^a$ evolves over time in ways similar to an airplane's controllers adapting to its evolving environment. The goal is to maintain fidelity of of ${}^j M_t^s$ with kR^a.

To identify what and when to calibrate we will look at ways to quantify distance of R^a and M_t^s. For such systems we can say that the distance between M_t^s and M_t^a is equivalent to distance between pairwise sub-models:

$$|M_t^s - M_t^a| \equiv \forall j |{}^j M_t^s - {}^j M_t^a|$$

If measuring ${}^j M_t^s$ is not possible then we can approximate it by measuring the output of sub-components:

$$\forall j |{}^j M_t^s - {}^j M_t^a| = \forall j |{}^j out_t^s - {}^j out_t^a|$$

For systems which transition smoothly we can extend this measurement method of model health to time beyond t. that is:

$$\forall j |{}^j M_t^s - {}^j M_{t+n}^a| = \forall j |{}^j out_{t+n}^s - {}^j out_{t+n}^a|$$

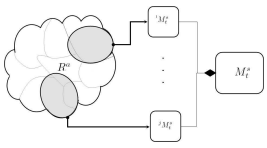


Figure 2. Architectural design of compositional system model. R^a is composed of k components. System model attempts to identify these k regions and model it internally as k sub-models (${}^k M_t^s$). These model compose to form the total system model M_t^s.

Note that this is true for sub-components models but not for entire system as has been discussed above and shown graphically in figure 1. Here we can see that though total error for system may be less at time $t + n$ than at time t. But we can see that the error in sub-components can still be observed and corrected.

We would like to point out that observing total error of system does not provide the insight we need for self-calibration.

$$|M_t^s - M_t^a| \not\equiv |out_t^s - out_t^a|$$

Suppose that gain in accuracy due to evolution is the same as drop in accuracy. This would mean

$$|out_t^s - out_t^a| \equiv |out_{t+i}^s - out_{t+i}^a|$$

This will give the false impression that M_t^s is similar to M_{t+i}^s. However at component level the model would have moved. In case-study 1 we discuss similar system where without self-calibration we were not able to monitor component level degradation. By the time system level degradation was observed the model had already denigrated so much that minor adjustments were not sufficient and wholesale system re-modeling were required.

Through this type of calibration we maintain the safety property that "bad things never happen to the system". Through this system we can capture the system moving towards D state and recover it before it goes to unhealthy state as described in [10].

B. Architecture 2: Reference Model Based Self-calibration (RMBSC)

Reference model based architecture is designed for system models (M_t^s) which are generalization of sub-system models (${}^j M_t^s$). in these models the M_t^s is an average or generalization of the component models ${}^j M_t^s$. Conversely the model (${}^j M_t^s$) is specialization of (M_t^s) which is a generalization of entire system. Systems such as personalized spam filtering, our second case study, is an example of such systems.

Previously generalizational systems used to maintain a single global model. However, recent trend is to increase accuracy by adapting local models for each entity that combines to form the model. Though this increases system accuracy, but it has the overhead of creating j models. This effort if done only once in life cycle may be applicable but many adaptations at runtime are abortively costly.

To implement scalable adaptation of such systems we propose architecture shown in figure 3. This model is similar to MRAC with some differences that we will discuss here. MRAC works on the principle that system maintains a reference model to correct evolution in environment. A controller is designed with certain assumptions and boundaries on the environment. When these boundaries are violated the model updates the controller for the new environment.

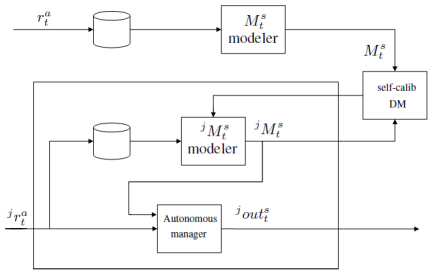


Figure 3. Data flow architecture for Reference Model Based Self-calibration . Data stream r_s^t representing entire system is stored in data store and global model M_t^s is created from it. Self-calibration decision maker (DM) compares this model with j^{th} component's model and provides control signal whether to create a new model for j component at this time. The data stream for j^{th} component is also stored in a local data-store. If self-calib DM provides a positive signal then a new ${}^jM_t^s$ is created and passed to autonomous manager to make run time decisions.

RMBSC core is similar to MRAC. But unlike MRAC our architecture provides a method to train the reference model as well. This way the adaptation itself can be adaptable based on the environment around it. There are two reason for this extension. First, because in some situations it is required. and Second, because we can do it.0.

Consider architecture in figure 3. The architecture represents architecture for j^{th} sub-component. The task of the manager here is to make decisions for j^{th} sub-component using model ${}^jM_t^s$ for incoming dataflow. This model may be an adaptation of the generalized model for all the j components M_t^s. On a global level, a generalized model of entire system containing data from all the j components is maintained. This may be learnt model at time $t+i$ or a reference model depending upon the system. On some specific triggers (e.g. construction of new M_t^s), M_t^s is compared with each of ${}^jM_t^s$. Based on system goals, the distance between ${}^jM_t^s$ and M_t^s is used to initiate an adaptation for selected ${}^jM_t^s$. The adaptation algorithm then adapts the existing M_t^s using the local data of j^{th} component.

According to our assumption, system model M_t^s at its inception is the best approximation of world model M_t^a and no better model can be made with the resources and techniques available. For all practical purposes we can say that if model is created at time t then:

Since M_t^s is a maximal approximation of M_t^a we can write target of self-calibration system as.

$$M_t^s \approx M_t^a$$

According to definition of generalizational model we also know that M_t^s is a generalization of components of S. We can thus say that:

$${}^jM_t^s \sim M_t^s \approx M_t^a$$

If we consider the error for j^{th} component at time t, we can use logic similar to previous case and state that:

$${}^jerror_t = |{}^jM_t^s - M_t^a| \approx |{}^jM_t^s - M_t^s|$$

Using same argument at time $t+i$ if we update our model M_{t+i}^s then we can say that:

$$M_{t+i}^s \approx M_{t+i}^a$$

Using this relationship we can compare the j^{th} component's un-updated model to M_{t+i}^a.

$${}^jerror_{t+i} = |{}^jM_t^s - M_{t+i}^a| \approx |{}^jM_t^s - M_{t+i}^s|$$

This measure of distance of model of j^{th} existing component with the best approximation of world model will give us objective measure to evaluate when and what to calibrate at time $t+i$.

1) When and What to Self-calibrate: There are two variants of calibrating system based on core demands of system. For a system concerned about quality of service (QoS) we can set a threshold of health τ such that when distance of a sub-component breaches this threshold, re-calibration of this component is performed.

$|{}^jM_t^s - {}^jM_t^a| > \tau$

Another possibility is that we can limit the resource utilization by putting a threshold at the number of calibrations that can be done in an iteration. In this method the components are sorted according to Δ_{error} and top x are chosen for calibration where x is dependent upon the available resources.

This type of calibration maintains the safety property that "bad things never happen to the system". Moreover, through this system we can capture the system moving towards D state and recover it before it goes to unhealthy state as described in [10].

VI. Case Studies

In this section we present three case studies of self-calibration from varied domains. The domains vary from automatic tag recommending web 2.0 application to spam filtering to sensor networks. The goal of these case studies is to present exemplar systems where self-calibration can be helpful and provide examples of our proposed architecture.

A. Automatic Tag Recommendation

A folksonomy, or collaborative tagging, is a system of classification of documents through collaboratively creating and managing tags to annotate and categorize content [20]. The system allows its users to assign keywords, or tags, to resources for navigation, finding resources, etc.

We proposed an automatic tag recommendation system for folksonomy based on discriminative clustering [12].A new document is first classified into a specific cluster and the top 5 tags of the cluster are recommended as possible tags for the document. The accuracy of the system however,

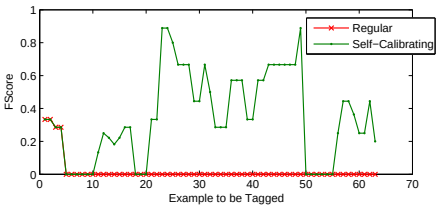


Figure 4. Comparison of cluster accuracy with and without self-calibration. Red line shows regular results when system allows cluster performance to degrade. Green line shows self-calibration results. It can be seen that between example 5 and 10 self-calibration identifies a need for calibration. After the calibration step accuracy of cluster is restored until 50^{th} example when another round of calibration is done

was affected over time. As new ideas and concept emerge the tags and their relationships with documents also change.

To handle this accuracy drop due to distribution drift, we proposed to rebuild the prediction model by re-clustering [11]. To automate the task we proposed a self-managing mechanism for this process. A self-optimizing system though is able to keep a high level of accuracy, but such massive re-modeling step is an overkill. Especially when up to 65% of clusters remain intact and majority of the system is not affected by clustering. This points to the intuitive idea that over time not all information will change but rather some relations between document classification and tags would be re-ordered. What is required is not self-optimization but minimal self-healing of system.

We found that a large number of documents were being clustered correctly but the evolution of tags by users was not represented in the system. We implemented architecture 1 in this scenario. Figure 4 shows resulting f-score with and without self-calibration. The accuracy of cluster was 0.35 before example 5 (Y-axis value). Without self-calibration the average accuracy goes down to 0.0197. However, self-calibration restores the accuracy 0.329. In addition we observed that this drop in accuracy, or its healing, did not affect the global accuracy by much. Without healing global accuracy was 0.1592 and with self-calibration it was 0.16. Such a minor change in accuracy at global level was not observable but at a sub-component level it was observable and correctable. It is evident that self-calibration of system can make the system self-healing.

B. Email Spam Filtering

Filtering spam mail or Unsolicited Commercial Email(UCE) is an important task for mail service providers. The process needs to be fast and effective. Due to this reason spam filtering techniques usually build an offline model on the data collected from users. Emails which are already classified as spam and ham (meaning valid emails) are used to train a classifier. This global model is then used to filter spam emails.

However, as has been shown by Fawcett, spam and ham email patterns shows concept drift [9]. This means that the concept of what is spam "drifts" or changes over time and between users. To show this figure 5 plots class conditional probabilities of spam and ham for two users against the term frequency of spam in the generalized model from ECML data set [5]. Term frequency is the number of times a term appears in an email. Different peaks and troughs in figure 5 points to difference in terms which are considered spam by different users.

To handle this concept drift among users, personalized spam filtering was considered. A global corpus of emails was used to train a global filter and this filter was adapted according to user's own email repository [15], [18]. This results in 5.5% increase for task A and 10% increase for task B in accuracy of spam filtering [15].

But this does not solve the problem of concept drift over time. As can be seen from figure 6. Term frequency for spam and ham changed between two time periods for the same user. The problem can be resolved with repeated construction of the global model and its adaptation for local models but with hundreds of thousands of local models, such adaptations were not possible.

We applied the second self-calibration architecture using the global model as M_t^s and individual mailbox models as ${}^jM_t^s$. This resulted in objective way of identifying the personal mail boxes which were most distant from M_t^s. We applied a threshold such that only the worst 20% of mailboxes were adapted at each iteration. This resulted in minor drop in accuracy while it reduced our computing resource requirement to one fifth.

C. Sensor Networks

Sensor networks present an interesting application for self-calibration. The distributed nature of sensors and adhoc and evolving world they work in makes self-calibration a

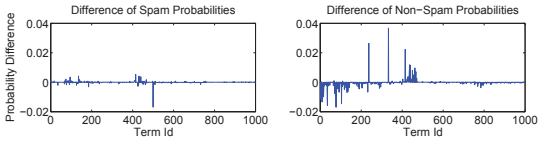


Figure 5. Shift in $p(x|y)$ between training and test data for ECML-A dataset (individual user's e-mails), where $y\epsilon\{spam, nonspam\}$ and x is an email)

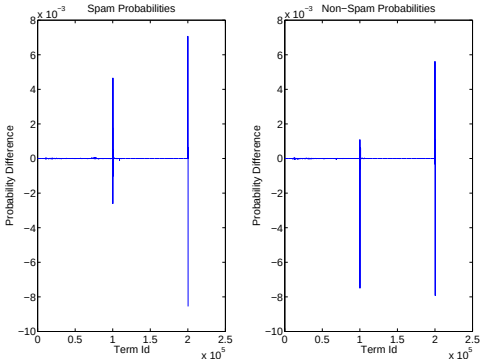


Figure 6. Difference in $p(x|y)$ for e-mails from different time periods for ECUE-1 dataset), where $y\epsilon spam, nonspam$ and x is an email

very interesting application in this domain. The concept of calibration has already been used in sensor networks [7].

Here we will present two different types of sensor networks and relate them with the architectures we presented. Furthermore, our proposed architecture can provide these systems with extending their adaptation. Mainland and colleagues propose Self-Organizing Resource Allocating (SORA) sensor networks [19]. A similar work (DIRL) by Shah and Kumar used an economic model as the global model but had similar architecture [23]. In both systems each sensor can assume different roles in the system. A sensor can be active scanner, passive scanner, data aggregator or a networking node. With a changing world it is impractical to explicitly assign a specific role to each sensor. The proposed systems propagate a central cost or economic model to all sensors. Sensors use this model to bid on different actions and through this auction mechanism each sensor assumes the optimal role according to market forces. Whereas the sensors in this mechanism are self-optimizing the overall emergent behavior is self-healing as well.

Assume that a sensor goes offline. The remaining sensors can then win in auction the responsibility of offline sensors in the most optimal way.

If we consider incorporating the assumptions of DIRL and SORA, we see an application of our second proposed architecture (RMBSC). Both the systems are dependent on a global model which can best utilize the sensor network. This model is based on considering the generalization of sensors to ascertain the most optimal mix of policies. In essence this is creating a M_t^s so that each sensor, or j^{th} component of system can adapt it to its local needs. Although it is assumed that M_t^s will exist but both the system leave its construction as assumption or as future work. RMBSC provides a concrete architecture to incorporate modeling and maintenance of M_t^s and provide tools for its propagation, comparison and utilization.

This global model will require regular evaluation and on each re-evaluation, sensors can decide if they wish to update their role or continue in their current state.

Another flavor of sensor networks works by partitioning the world according to its evolution such as the work of Salazar and colleagues [22]. In this system through a diffusion searching algorithm sensors are arranged in certain configurations. These configurations then internally manage their roles and resources. The configurations in terms of their distribution in space are not fixed and the algorithm allows movements of configurations through sensors. This task however is internal to a configuration as it searches in its neighborhood for appropriate resources and roles for efficient operations. This is an example of localized self-calibrating architecture. As the sensors notice an evolution of the world, they use diffusion algorithm to ascertain change of roles for sensors within the reach of its configuration.

We have been working on various other applications of self-calibration in autonomous systems that cannot be included here due to space limitation.

VII. Future Work

We believe that this work lays the foundation for research and development for self-calibration. There are numerous directions in which we foresee possible progress. Apart from developing techniques and systems to effect self-calibration, some generic engineering questions needs to be answered.

The starting step in this direction are the requirements that gave rise to self-calibration.

Situateability: Situateability is a form of self-awareness where model and world are compared for awareness. We feel that research in self-awareness can sufficiently address this concern. However, mapping of existing self-aware techniques for self-calibration requires further research.

For our case studies, for instance, situating a varying input stream as temporary change in world will be required for spam filtering.

Metrics and Thresholds: Self-calibration maintains data fidelity. However, metrics for validating data fidelity require some in-depth study. Some basic analytical and control theoretic models have been used for other self-* properties but metrics specifically for self-calibration need to be investigated [1], [8], [12].

As we have discussed, metrics to identify when a term in joint distribution of emails crosses the boundary from spam to non-spam is an open problem for live system. Metrics and thresholds for checking cluster fidelity are present [3] but for surgical revision according to an incoming input stream requires further research. Such metrics will be beneficial for smart grid applications and automatic tag recommendation software too. Similarly, metrics for evaluating lighting conditions comparable with a static model are needed for collision avoidance algorithm.

Model transformation: Methods need to researched and defined on how models can be updated to effect self-calibration. Various online algorithms provide an insight on how this can be achieved [6]. However, for systems which are based on offline learning, integration of surgical online fine-tuning will provide interesting research avenues.

Various online algorithms exist [6] for maintaining a model which is always updated. However, we require a model transformation that updates the model in a time-efficient manner without increasing the system turn-around. For this we require model transformations for our clustering approaches in smart grid and automatic tag recommendation systems. We will require different transformation for model transformation of autonomous robot.

Input Stream Transformation: Input stream transformation is most useful in collision avoidance robot and spam filtering. Here we will require metrics which identify input stream variations and correct transformations need to be

applied to input stream so that the model can be efficiently updated.

VIII. Conclusion

For content driven systems where the new information from the outside world is continuously received the constructed model of the world gets obsolete with time. Taking the system to an offline mode to reconstruct the model is not a viable solution for time-critical systems. Reconstruction of the model using self-optimization or self-awareness is also inefficient as the process of reconstruction of the model is expensive and time consuming. However, through self-calibration the model is updated with minimal changes.

In systems that receive a lot of content e.g. spam filtering, tag recommendation etc., much of the information is useless or false positive. Therefore, self-calibration also aims to mitigate the inaccurate information and only accept statistically significant information for the model tuning. We believe that for systems that use AI techniques and are heavily content-driven, self-calibration will provide an important missing property in making these systems self-managing.

References

[1] Abdelwahed, S., Kandasamy, N., and Neema, S. A control-based framework for self-managing distributed computing systems. In *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems (WOSS)* (2004), pp. 3–7.

[2] Alpern, B., and Schneider, F. B. Recognizing safety and liveness. *Distributed Computing 2* (1987), 117–126.

[3] Amigó, E., Gonzalo, J., Artiles, J., and Verdejo, F. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retrieval 12* (2009), 461–486.

[4] Berns, A., and Ghosh, S. Dissecting self-* properties. In *Self-Adaptive and Self-Organizing Systems, IEEE International Conference on* (2009), pp. 10–19.

[5] Bickel, S. Ecml-pkdd discovery challenge 2006 overview.

[6] Blum, A. On-line algorithms in machine learning. In *Online Algorithms*, A. Fiat and G. Woeginger, Eds., vol. 1442 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1998, pp. 306–325.

[7] Bychkovskiy, V., Megerian, S., Estrin, D., and Potkonjak, M. A collaborative approach to in-place sensor calibration. In *Proceedings of the 2nd international conference on Information processing in sensor networks* (Berlin, Heidelberg, 2003), IPSN'03, Springer-Verlag, pp. 301–316.

[8] Diao, Y., Hellerstein, J. L., Parekh, S. S., Griffith, R., Kaiser, G. E., and Phung, D. B. Self-managing systems: A control theory foundation. In *12th IEEE International Conference on the Engineering of Computer-Based Systems (ECBS)* (2005), pp. 441–448.

[9] Fawcett, T. "in vivo" spam filtering: A challenge problem for data mining. *CoRR cs.AI/0405007* (2004).

[10] Ghosh, D., Sharman, R., Raghav Rao, H., and Upadhyaya, S. Self-healing systems - survey and synthesis. *Decis. Support Syst. 42* (January 2007), 2164–2185.

[11] Hassan, M. T., Karim, A., Javed, F., and Arshad., N. Self-optimizing a clustering-based tag recommender for social bookmarking systems. In *International Conference on Machine Learning and Applications (ICMLA)* (2010).

[12] Hassan, M. T., Karim, A., Manandhar, S., and Cussens, J. Discriminative clustering for content-based tag recommendation in social bookmarking systems. In *ECML/PKDD Discovery Challenge Workshop* (2009).

[13] Javed, F., and Arshad, N. Adopt: An adaptive optimization framework for large-scale power distribution systems. In *International Conference on Self-Adaptive and Self-Organizing Systems (SASO)* (2009), pp. 254–264.

[14] Javed, F., and Arshad, N. A penny saved is a penny earned: Applying optimization techniques to power management. In *16th IEEE International Conference on the Engineering of Computer-Based Systems (ECBS 2009), 13-16 April 2009, San Francisco, CA, USA* (2009).

[15] Junejo, K., and Karim, A. Pssf: A novel statistical approach for personalized service-side spam filtering. In *International Conference on Web Intelligence (WI)* (2007).

[16] Kephart, J. O., and Chess, D. M. The vision of autonomic computing. *Computer 36*, 1 (2003), 41–50.

[17] Kivinen, J., Smola, A., and Williamson, R. Online learning with kernels. *Signal Processing, IEEE Transaction on 52*, 8 (Aug. 2004), 2165 –2176.

[18] Kyriakopoulou, A., and Kalamboukis, T. Text classification using clustering. In *In Proceedings of the ECML-PKDD Discovery Challenge Workshop* (2006).

[19] Mainland, G., Parkes, D. C., and Welsh, M. Decentralized, adaptive resource allocation for sensor networks. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation* (Berkeley, CA, USA, 2005), USENIX Association, pp. 315–328.

[20] Peters, I. *Folksonomies. Indexing and Retrieval in Web 2.0*, 1st ed. Walter de Gruyter & Co., Hawthorne, NJ, USA, 2009.

[21] Roth, Z., Mooring, B., and Ravani, B. An overview of robot calibration. *Robotics and Automation, IEEE Journal of 3*, 5 (october 1987), 377 –385.

[22] Salazar, N., Rodriguez-Aguilar, J., and Arcos, J. Self-configuring sensors for uncharted environments. In *Self-Adaptive and Self-Organizing Systems (SASO), 4th IEEE International Conference on* (2010), pp. 134 –143.

[23] Shah, K., and Kumar, M. Distributed independent reinforcement learning (dirl) approach to resource management in wireless sensor networks. *Mobile Adhoc and Sensor Systems (MASS). IEEE Internatonal Conference on* (Oct. 2007), 1–9.

[24] Zhang, S., and Huang, P. S. Novel method for structured light system calibration. *Optical Engineering 45*, 8 (2006).