# LiLoLe—A Framework for Lifelong Learning from Sensor Data Streams for Predictive User Modelling

Mirko Fetter and Tom Gross

Human-Computer Interaction Group, University of Bamberg, Germany
{firstname.lastname}@uni-bamberg.de

**Abstract.** Adaptation in context-aware ubiquitous environments and adaptive systems is becoming more and more complex. Adaptations need to take into account information from a plethora of heterogeneous sensors, while the adaptation decisions often imply personalised aspects and individual preferences, which are likely to change over time. We present a novel concept for lifelong learning from sensor data streams for predictive user modelling that is applicable in scenarios where simpler mechanisms that rely on pre-trained general models fall short. With the LiLoLe-Framework, we pursue an approach that allows ubiquitous systems to continuously learn from their users and adapt the system at the same time through stream-based active learning. This Framework can guide the development of context-aware or adaptive systems in form of an overall architecture.

**Keywords:** Lifelong Learning, User Modelling, Framework.

## 1 Introduction

Context-aware and adaptive systems promise to support users by allowing them to offload tasks to the system. For example context-aware ubiquitous environments automatically execute tasks for the users, by sensing properties about the current situation and inferring the users' needs based on learned models or specified rules. For simple forms of adaptation, such rules can be specified manually with the help of tools [25, 26]. For example the task of automatically muting the phone when arriving at a specific location such as a meeting room can be relatively easily described by a rule. More complex forms of adaptations often already require the application of machine learning algorithms, as a manual specification of rules would be too complex. For example, smartphones are able to distinguish different simple activities like walking, running, or bicycling through activity recognition based on accelerometer data [5] and thus they are able to adapt their user interface to the current activity of the user. In order to enable such devices to recognise specific activities, usually machine-learning classifiers are trained with labelled data collected from a number of users. However, even with these relatively simple machine-learning problems, the generalisability of the learned models to new users who are not part of the group that provided training data, is already a challenge. For example Biao and Intille [5] found that while some of the activities in their work could be identified well with "subject-independent training

S. Sauer et al. (Eds.): HCSE 2014, LNCS 8742, pp. 126–143, 2014.

data“ some of the activities could be detected better with “subject-specific training data”. In order to optimise general activity recognition models for individual users, a number of different approaches have been developed, which try to optimise general models to specific users [23, 35]. The reason for relying on general models, even if they perform worse, is that the final users are not bothered with the effort for training a system before using it.

However, adaptation decisions in ubiquitous environments and adaptive systems are becoming more and more complex, where such simplified approaches based on general models or their optimisation can be hardly applied. They often take a multiplicity of input factors like location, activities, social interactions, etc. that happen simultaneously in physical and electronic space. The input arrives from a plethora of heterogeneous sensors and the adaptation decisions often imply personalised aspects and individual preferences that also can change over time. An illustrative example is the scenario of mediating interruptions by predicting the interruptibility of users [10, 11, 17], where the data of numerous different sensors is brought together with personal interruptibility reports to predict the availability of a user for communication. In such scenarios, general models do not offer enough accuracy and thus fail to offer adequate adaptation support for various reasons, as reflected on by Fetter et al. [10]. In order to overcome such challenges the idea of lifelong learning systems for user modelling [18] was introduced. Thereby the approach is to collect labels for unobservable user states over a prolonged time span to continuously train an adaptive system. The underlying rationale is that the system continuously improves itself, and over time reduces the request for labels, unless novel situations are detected. Hence, lifelong learning aims at improving the prediction quality through learning personalised models while trying to keep the training effort for users low.

In the following we present the LiLoLe framework for lifelong learning from sensor data streams for predictive user modelling. The Framework can guide the development of context-aware or adaptive systems in form of an overall architecture. The Framework thereby provides a foundation for the research and development of systems that deliver more human-centred adaptations, by continuously learning from the user, while keeping the effort for the user low. In the remainder of the paper we motivate the LiLoLe-Framework based on the analysis of the requirements and a look at existing machine learning approaches and the related work. We give a detailed description of the framework and a reference implementation. Finally, we discuss the results of an experiment, which applied the reference implementation to a real world data set and thus is able to predict the availability of individuals for instant messaging (IM) based on 30 different sensors.

## 2 Background and Related Work

In the following we give an introduction to the background of our work, including work that motivated our approach, that is related to our approach, and such that is incorporated in our approach.

### 2.1 A Rationale for Learning Personal Models

While machine learning at first seems an ideal candidate for learning personalised user models, a number of challenges hinder its broad application. As discussed by Webb et al. [30], the main challenges are the need for a huge amount of labelled data for training and the fact that learned concepts may change over time (i.e., concept drift). In user modelling therefore mainly two approaches can be found [37]: content-based, and collaborative learning. In the first approach users' past behaviour is used, to make prediction about their personal future behaviour. In the second approach, the behaviour of a group of similar behaving people is used, to make predictions for an individual user. Yet, both approaches only work if the concept to be learned is not hidden for the learning system, but is available for *in-stream supervision* [18]. For example, for a media player the songs played by a user are observable. Accordingly, based on meta-information like artists and genre (content-based learning) or based on the similar musical taste of other users (collaborative learning), the media player is able to recommend new songs to the user on the basis of the kind of songs the user played before. However, in many cases in ubiquitous computing and context-aware systems the concepts to be learned are hidden from the system (e.g., the users' mood, interruptibility, activity), and labels need to be provided manually. Therefore, in most implementations, where data could not be learned from individual users, the data is collected from a smaller number of users and is used to build general models that later need to fit all users (e.g., in activity recognition, gesture recognition). From a human-centred computing perspective, such approaches do not account for the individual differences of users. Accordingly, often users are forced to adapt to the system instead of the systems adapting to them (e.g., perform an activity in such a way, that the system is able to detect it). Further, as recent work has shown, such general models are not feasible in more complex settings. For example, when Fetter et al. [10] examined the performance of general vs. personal models in the context of predicting Instant Messaging availability of nomadic users based on a number of sensors in mobile settings, they found several aspects that degraded the performance of general models such as: the learned concept (i.e., availability) is highly personalised, and the variation between users was accordingly high; some features that had high predictive power for an individual user had a limited predictive power for other users, and thus were discarded; and some features that worked well in the individual models even had contradictory information for other users when used in general models. Our approach accounts for these challenges.

### 2.2 Lifelong Learning Systems

Only few researchers so far investigated the feasibility of systems that are able to continuously learn personalised models of human situations and adapt the user interface. Kapoor and Horvitz [18] proposed the notion of lifelong learning, where a user trains a machine learning system over a prolonged period based on selective supervision using the experience sampling method (ESM) [16]. However, their approach only was applied in strictly controlled settings, where the knowledge about the feature space was clear in the beginning, and each feature was handcrafted upfront by the system designer. Accordingly this approach only partially adapted to new situations. With Subtle, Fogarty et al. [11] developed the concept of a toolkit to

generate automatic sensor-based statistical models of human situations. The toolkit provided an extensible sensor library, although with a limited number of readily available sensors. It provided support for continuous learning, but only in form of an iterative batch mode, that did not combine the strength of online learning algorithms with selective supervision for choosing best instances to label. Further, it was based on automated feature construction and selection, but did not make use of online feature selection mechanisms, to cope with high dimensional data with a presumably sparse number of features with high predictive power.

While both approaches demonstrated the general feasibility of life-long learning adaptive systems, those ideas have not been further pursued. Though, for a human-centred assessment of these approaches, studies with real-users in real life situations are needed, measuring the users attitude towards such systems. The LILOLE-Framework informs the design of life-long learning systems and promotes implementations, which are robust to deployments in different environments and situations, and thus can be the foundation of further human-centred evaluations.

### 2.3 Data Stream Mining and Active Learning

In the following, the two machine learning approaches that build the foundation of our lifelong learning approach are introduced: data stream mining, and active learning.

The research field of data stream mining is concerned with methods, algorithms, and tools for extracting knowledge from rapid and continuous streams of data. As the development of information and communication technologies make more and more data available (e.g., from sensor networks, network traffic, or human activity data), learning from data streams is an emerging field in the area of machine learning. The aim of the research is to develop new algorithms and approaches that can cope with the special characteristics and requirements of stream-based learning, discussed in the Data Stream Model by Babcock et al. [4] as well as in the work of Bifet [6], Aggarwal [1]), and Gama [12]. So, for example, the data needs to be processed in one-pass and only in small chunks as the amount of data is extremely large, arrives continuously and is potentially infinite [1, 4, 6, 12]. Further, the speed at which data arrives makes processing the data in real-time a prerequisite [4, 6, 12]. Also, a temporal order of the arriving data is not guaranteed [4, 6]. New algorithms furthermore need to be able to deal with temporal locality [2] (i.e., concept drift [31]), as the concepts underlying the data can change over time and thus make past data irrelevant of even contradictory. Finally, the algorithms need to be able to adapt the changing data structures and evolving feature space when old data sources are removed, new ones are added, previous potential features loose their discriminative power, and new features are becoming continually available [1, 4, 6, 12].

Active learning [27] is a recent approach to machine learning. As gathering labelled data mostly is expensive (i.e., causing user effort during the training phase), the idea of active learning is to allow the algorithm to choose the data from which it learns by requesting labels from an oracle (e.g., a human annotator) in form of a query. By using algorithms that only request labels for instances that are close to the decision boundary, the required number of labelled examples is significantly reduced. The main difference between various active learning approaches is the way the algorithms queries for labels. The literature [27] currently distinguishes between

membership query synthesis, pool-based sampling, and stream-based selective sampling. While the first approach synthesises artificial examples based on the provided features and their range of values, the second evaluates the entire collection of unlabelled instances upfront to pick the most promising examples for the query. The last approach, stream-based selective sampling [3], sequentially evaluates the unlabelled instances. For each instance—based on a given querying strategy—the algorithms decides if a label provided by an oracle is likely to improve the overall model quality and accordingly presents a query to the oracle or not.

In the next section we discuss, how the two approaches play together, to support the implementation of a lifelong learning system.

## 3 A Concept for Lifelong Learning of Personalised User Models

Context-aware systems adapt to the users' preferences based on sensing the current context. Therefore they apply machine learning algorithms to the sensor data in form of previously learned models that were trained in a batch learning approach [12]. A sensor can be either a software or a hardware component used to capture data from the users' environment in the digital or physical realm. The underlying idea of batch learning is, that a certain amount of labelled data is available upfront and can be used to train a predictive model (e.g., a gesture recogniser). This trained model then can be applied to new data—with the primary assumption that the training and the new data is independent and identically distributed—in order to make inference on this data. However, the assumption that labelled training data is available at a given time and representative for future data may not hold in any case. When learning personalised user models for building adaptive systems, four influencing factors cause a demand a new machine learning approach:

- Firstly, preferences for context-aware adaptations are often highly personalised and accordingly labelled data is sparse. For example, privacy-based decisions like revealing the current location to others have been found to be personal decisions are influenced by various factors [9].
- Secondly, the data can often only be labelled by the individual and only in the moment when it is experienced; assigning labels later based on a presentation of collected sensor data to the user is often not feasible. For example, the question how interruptible a person was at a given point in time can hardly be answered retrospectively [18].
- Thirdly, the sensor data (like the labels) is highly personalised and grounded in the individual users' daily contexts, hence resulting in an individual evolving feature space based on a continuous sensor data stream. For example, if several people share the preference for muting their work phone when at home, the sensor data that allows localising a person as at home, is individually different (e.g., BSSID of the home network) [10].
- And finally, preferences and contexts can change over time. Accordingly mechanisms are needed, which allow forgetting previously learned concepts, or overwriting them with new concepts, and also allow learning new concepts in new contexts. For example, a changing class schedule after a term break could lead to new preferences for when a phone should be automatically muted.

Implementations of systems should take these factors into account, to provide user-centred adaptations in real-life contexts. In doing so, those systems should be clearly designed to reduce the overall effort for the user. That is, such systems should balance the training effort against the effort of manually adapting the system, and provide an optimised behaviour that greatly reduced the amount of work for the user.

### 3.1 Stream-Based Active Learning

Accordingly, to allow for lifelong learning of personalised user models, new machine learning approaches need to be utilised. We found that the characteristics of two machine learning approaches in their combination promise a solution, able to deal with the above constraints: active learning and data stream mining. As outlined before, active learning [27] departs from the premise that obtaining labelled data for supervised learning can sometimes be very expensive. Hence—instead of presenting a training set with labelled data to the algorithm upfront—the rational behind active learning is that an algorithm that is able to actively select the instances for which it requires labelling by an annotator performs equal or better with fewer labels needed. On the other side, data stream mining subsumes the ideas of learning from extremely large (potentially infinite) amounts of data that continuously arrive at great speed [13] with the possibility of variations in the learned concepts as well as in the underlying data structures. A combination of both, as theoretically explored by Zliobaite et al. [36], promises a solution to the requirements of learning personalised user models.

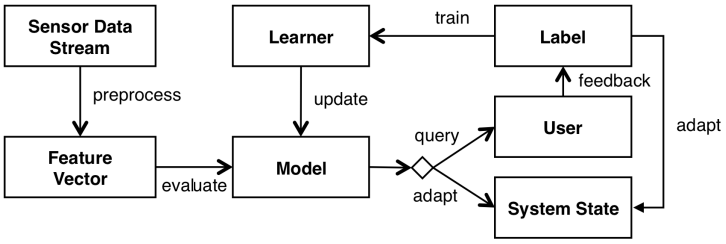


**Fig. 1.** Illustration of the basic principle of stream-based active learning from sensor data streams for lifelong learning

We therefore developed a concept, which utilises a combination of these approaches to enable *stream-based active learning* from sensor data streams, for lifelong learning of personalised user models. **Fig. 1** illustrates the underlying principle of our concept—forming the basis of LiLoLe-Framework.

In a ubiquitous environment, various sensors continuously monitor the users' contexts and activities, capture data from them, and provide this data in form of a sensor data stream. The arriving sensor data is pre-processed and converted to a feature vector that is processable by a machine learning algorithm. Based on this feature vector and a predictive model the system evaluates whether it is certain enough to adapt the system state and does so accordingly, or if it is uncertain

continues training the predictive model. In the second case the user is queried by the system and gives feedback in form of a label. This label is used to train a learner to improve and update the predictive model and to directly adapt the system state.

### 3.2 An Application Scenario

For a more illustrative depiction of the principle we use a simplified, stripped-down scenario of a context-aware system that adapts the Instant Messaging availability status on a laptop computer based on the input of two sensors running on the computer: A *Wi-Fi-Sensor (WF)* and an *Application Focus-Sensor (AF)*. The *Wi-Fi-Sensor* sends a list of all nearby access-points (SSIDs and BSSIDs) together with the respective signal strength (RSSI) every 30 seconds, allowing conclusions on a user's whereabouts. The *Application Focus-Sensor* sends the name of the current application every time the user switches to a new application, allowing conclusions about the user's activity. Whenever a new sensor event arrives in the sensor stream, the application transforms this data into a feature vector and evaluates the data with a decision tree (i.e., the predictive model). When the laptop user starts working with a new application at a familiar location, the model might not be confident enough to adapt the IM status automatically. Accordingly the user is asked about the preferred current availability status in a dialog. The status is adapted based on the user's input, and an incremental training algorithm (e.g., Very Fast Decision Trees (VFDT)) is used to update the decision tree. The next time the user switches to this application at the same location, the status is automatically adapted. While this simplified illustration gives a basic idea of the framework, in order to be flexible and robust for various settings and applicable with current implementations of machine learning algorithms, a few extra steps have to be taken. In the following, we provide further details of the LILOLE-Framework, starting with an overview that extends on the before discussed basic principle. We provide further details on the peculiarities when learning from streaming data in ubiquitous environments and how to integrate real-time feedback by end-users in form of active learning. Subsequently, we distil an algorithm for stream-based active learning from sensor events, which is the foundation of our framework. In the course of this paper we continuously refer to this scenario in form of examples for the different steps, algorithms and approaches.

## 4 The LILOLE-Framework

In the following overview of the LILOLE-Framework (depicted in Fig. 2) we extend on the previously simplified conceptual model of the principle of stream-based active learning from sensor data streams by providing more details on the necessary steps and by proposing adequate methods for each of those steps.

We thereby concentrate on the transformation of the raw sensor data stream into a feature vector as well as the mechanisms of actively querying the user and training a predictive model. Finally, we derive an algorithm formalising the central steps.

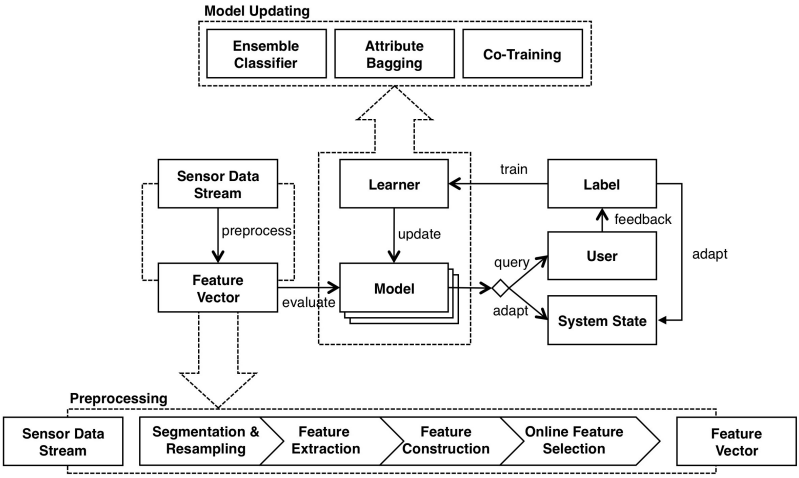


**Fig. 2.** Overview of the LILOLE-Framework

### 4.1 Learning from Streaming Sensor Data

In order to learn from the sensor data stream, several steps need to be completed to preprocess the incoming stream of raw data into a feature vector that is processable by a machine learning algorithm. Each of these steps is now consecutively introduced:

*Resampling & Segmentation.* An incoming sensor data stream consists of a series of repeated measures from multiple sensors that basically can be understood as a multivariate time series (MTS). A MTS represents a series of observations over a time T where each observation at a given point in time t for a specific variables i is denoted. As each sensor can potentially send information at any given point in time, a first step is concerned with resampling the incoming sensor events to discrete timestamps. In our scenario where the WF-sensor was interval-based (i.e., sending an update every 30 seconds) and the AF-sensor is event-driven (i.e., sending an update only when a change occurred), there is a high possibility that for each point in time only data for one sensor is available (cf. Table 1). In order to further process this data, it is necessary to either upsample the data (e.g., through interpolation of the missing values) or downsample the data (e.g., by calculating the average over a sliding window) to a discrete time point resulting in a MTS with a fixed sample rate.

**Table 1.** Example of the sensor data stream from our scenario, showing that data from the interval-based WF-sensor and the event-driven AF-sensor can be available at different time points (*x* denotes an incoming sensor event at a given time $t_n$)

| | $t_1$ (0s) | $t_2$ (11s) | $t_3$ (30s) | $t_4$(60s) | $t_7$(83s) |
|---|---|---|---|---|---|
| **WF-sensor** | x | - | x | x | - |
| **AF-sensor** | - | x | - | - | x |

The right approaches and parameters for resampling should be chosen based on the application scenario and the data. The second step is to cut the MTS into discrete segments of finite length, as the data stream constantly grows, and the MTS cannot be processed in its whole. Again, different algorithms for segmenting time-series are available [19]. The application scenario and the expected data influence the selection of an appropriate algorithm. However, when apriori knowledge about the data is limited, a fixed-size sliding window for segmentation is often a valid first strategy.

*Feature Extraction.* After the data is resampled and segmented, it is ready to be processed by the feature engineering chain. Each incoming segment represents an example (i.e., instance) that is later presented to the predictive model to be classified and to eventually further train the system. Thereby, feature extraction is the first step [20]. The aim of the feature extraction process is to transform the raw sensor data into features that can be interpreted by the machine learning algorithms. This includes a transformation of the data into the data types the machine learning algorithms can process (i.e., numeric and nominal types [32]) and a reduction of the dimensionality to simple key-value pairs of attributes (i.e., features) with the aim to reduce the amount of irrelevant information. Coming back to our scenario, the WF-Sensor for each reading delivers a matrix listing the SSID, BSSID and RSSI of each nearby access-points (cf. Table 2).

**Table 2.** Example of one sensor event of the WF-sensor listing nearby access-points

| SSID | BSSID | RSSI |
|---|---|---|
| UniXXXXXX | 00:1F:45:97:E3:81 | -54 dBm |
| eduroam | 00:1F:45:97:E3:80 | -51 dBm |
| eduroam | 00:1F:45:97:E3:88 | -62 dBm |
| MyNet | 00:4F:81:05:0B:8A | -79 dBm |

Based on this simple raw data, already a plenitude of features can be composed, as for example the following: The number of nearby access points (WF_Sensor_No=4) could indicate how dense the area around the users location is populated while a low average RSSI (WF_Sensor_Avg_RSSI=61,5) could indicate an outdoor usage. The absence, presence, or signal strength of an access point allows to infer the users whereabouts on different granularity levels. Different extractable features could tell that the user is on the University campus (WF_Sensor_SSID_ eduroam_Bool=true), near a specific lecture room (WF_Sensor_BSSID_001F4597E388_RSSI=62) or not at home (WF_Sensor_SSID_ HomeNet_Bool=false) when the history of sensor values is taken into account for feature extraction. The possible combinations are endless, while clearly not every combination does make sense (e.g., WF_Sensor_BSSID_ 001F4597E388_RSSI =”eduroam”) and later can dilute the classification performance. While normally human domain experts guide the feature extraction process, in the case of our framework for lifelong learning, automatic strategies need to be found, as potentially new features are constantly arriving with each new sensor reading. Optimally, different strategies for feature extraction need to be combined here, that allow generating a rich representation of the sensor data. As exemplified above, this can happen through simple functions that calculate mean values from the

sensor values, count the number of occurrences, etc. But also more sophisticated approaches such as Principal Component Analysis are conceivable as *FeatureExtractionStrategies*. If, in the end, not all extracted features meaningfully contribute to the classification result, simple filters or more powerful mechanisms for feature selection can remove those later.

*Feature Construction*. The next feature engineering step is concerned with creating additional features by discovering missing information about the relationship of individual features [20]. While these dependencies of features also can be of multifarious nature and only mechanisms for the automatic feature construction [21] can be taken into account, one aspect that becomes prevalent when using data streams is taking the factor time into account. So, instead of only using features from the last sensor reading, one promising strategy is to build features that take into account a period of time. Coming back to our scenario, features for the last 2, 5, or 15 minutes could be computed to provide additional information. By calculating the fraction of time a given feature was true they could indicate that somebody just arrived at a location in the last 2 minutes (WF_Sensor_SSID_eduroam_Bool_2m =0.2) or is focused on one application for longer (AF_Sensor_MSWord_Bool_15m =0.95). Other options for *FeatureConstructionStrategies* are calculating frequencies for Boolean values or the mean, standard deviation, etc. for numeric values.

*Online Feature Selection*. Feature Selection is the last step in the feature engineering [20] process. Hereby the aim is to reduce the dimensionality of the feature vector, as most classifier perform significantly better on data with a low number of features. The reduction of features is especially necessary, as our approach so far continuously increased the number of features through the strategies of feature extraction and feature construction. Normally, feature selection algorithms select the most predictive features and remove irrelevant and redundant features based on an evaluation process where all features are available upfront as well as labelled data as evaluation criteria. As both prerequisites are not given in our setting of stream-based active learning, special algorithms for online feature selection [29, 33, 34] need to be applied. Conclusively, the process of feature selection constantly can suggest different, improved combinations of features. As each new feature vector would require a new model to be trained, a *FeatureSelectionStrategy* also has to decide, when switching to a new feature vector—and hence to a new model—is beneficial for the overall performance. In the following, more details on how to deal with the resulting multiple models are provided.

### 4.2 Active Learning of User Preferences

Based on the previous steps, the framework is now able to provide the incoming sensor data as single instances with a fixed number of features to the predictive model for evaluation. Based on the update frequency of the sensors, and the chosen parameters for resampling and segmentation a new instance for evaluation is available in a fixed rhythm (e.g., every few seconds). Each time an instance arrives, an *ActiveLearningStrategy* decides whether it uses the instance for adaptation or whether it queries a label from the user. In order to provide an understanding of what a simple

*ActiveLearningStrategy* looks like, we use the example of the *Fixed Uncertainty Strategy* [36]. The *Fixed Uncertainty Strategy* provides the instance to the classifier (i.e., predictive model) in order to obtain a prediction. The strategy then compares the confidence value or the posterior probability of the classifier to a fixed threshold defined e.g. by the developer.. If the classifier is confident enough about the classification, the strategy performs the adaptation based on the classification result, otherwise the user is queried for a label. Of course, more sophisticated active learning strategies are needed, to allow building user-centred systems, not overburdening the user with the training effort. One illustrative example is the introduction of a simple upper-limit on the number of queries a strategy is allowed to pose to the user in a given time frame. Other strategies are decision-theoretic approaches as proposed by Kapoor et al. [16] and also strategies that simply detect novel situations, by applying clustering algorithm on the incoming data, as presented later in our implementation.

The presentation of the query to the user and the feedback by the user (i.e., provisioning of the label) also allows for different approaches. In GUI based systems, an experience sampling based approach [10, 16] is feasible, as exemplified for our scenario in Fig. 3. Of course depending on the application context, multifarious query and feedback mechanisms are conceivable, e.g., a vibration pattern of a smart watch could query the current interruptibility, allowing the user to simply respond by performing a gesture.

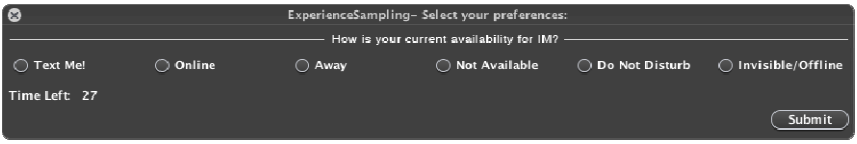


**Fig. 3.** Exemplary experience sampling dialog for our IM scenario

Depending on the provision of a label, the final step now is to update the model through training with the instance and corresponding label. As the training data arrives sequentially, only classification algorithms that allow for incremental learning can be applied in our framework. Examples for such algorithms are incremental decision trees like Hoeffding trees or Very Fast Decision Trees (VFDT), online variants of kernel learners like LASVM or more basic algorithms like an updateable Naïve Bayes classifier.

In respect to our setting that requires learning from data streams under the assumption of concept drift and feature evolution, additional steps need to be taken in the model updating process. As noted in an earlier step, foremost the application of a *FeatureSelectionStrategy* requires flexibility in the model updating process, as most algorithms cannot deal with instances represented by a changing combination of features (i.e., feature evolution). Accordingly, each time the *FeatureSelectionStrategy* suggests a new set of features also a new model needs to be trained. As this would require starting the learning process from scratch, three mechanisms (cf. Fig. 2) allow a smooth transition and preservation of the previously learned concepts. First, the use of ensemble classifiers [22, 24, 28] allows combining several models into one. When a classification is requested, different ensemble strategies (e.g., bagging or boosting)

provide a weighted classification result, computed from the individual models. Ensemble approaches have been demonstrated to work well for learning from concept drifting data streams [22, 24, 28]. Second, attribute bagging [8] (also feature subset ensembles or random subspace method [15]) are specialised ensemble methods for learning from instances with different feature vectors, that can be fitted to tackle the challenges of dealing with an evolving feature space. Third, the application of co-training [7] in both cases allows to train new models separately, before they are added to an ensemble. Co-training simply uses the classification output of the current ensemble, to train new models, with new feature subsets, until a certain training criterion is met. This approach would allow training a new model for a specific period, before it is actively used for predictions in the ensemble.

### 4.3 An Algorithm for Stream-Based Active Learning

In the previous sections we described in detail our concept of the LILOLE-Framework, and provided examples for the implementation including promising strategies and

**Algorithm 1:** Stream-based Active Learning from Sensor Data Streams

**Input:** strategy parameters, strategies
**Output:** $o_t$ as output label for instance $I_t$
**for each** $s_t$ - *incoming SensorEvent*, **do**
  resample $s_t$ to discrete timestamps t' and add to resampled data stream
  segment resampled data stream into segments $S_{t'}$=[ $s_{t'}$, $s_{t'-1}$, $s_{t'-2,...}$ $s_{t'-w}$]
  **for each** *FeatureConstructionStrategy(…)* **do**
    **for each** *FeatureExtractionStrategy(…)* **do**
      *constructFeatures*(*extractFeatures*($S_{t'}$))
    **end for**
  **end for**
  build $I_{t'}$ – *instance* from Features
  **if** *FeatureSelectionStrategy* ($I_{t'}$,…) = **true then**
    start a new classifier $C_n$ with selected features
  **end if**
  **for each** $I_{t'}$ - *incoming instance*, **do**
    **if** *ActiveLearningStrategy*($I_{t'}$,…) = **true then**
      query the true label $a_{t'}$ of instance $I_{t'}$ from the user
      train ensemble E with ($I_{t'}$; $a_{t'}$)
      return true label at' as output $o_{t'}$
    **else**
      classify $I_{t'}$ with ensemble E and obtain the predicted label $p_{t'}$
      return classification result $p_{t'}$ as output $o_{t'}$
    **end if**
    **if** $C_n$ *exists* **then**
      co-train classifier $C_n$ with $o_{t'}$
      **if** $C_n$ *is trained* **then**
        add classifier $C_n$ to Ensemble E
      **end if**
    **end if**
    adapt the system based on $o_{t'}$
  **end for**
**end for**

available algorithms. In order to provide a more formalised documentation of our approach, we propose the following algorithm for the stream-based active learning from sensor data streams to enable lifelong learning systems. In the next chapter we provide insights in an implementation and provide results of a simulation with real user data.

## 5 Implementation and Evaluation of the Framework

In order to put our approach into practise, we build a first reference implementation of our framework, and evaluated the feasibility of the approach in a simulation with real user data. The main aim was to test the end-to-end feasibility of our approach of stream-based active learning. As a basis for our implementation we used Sens-ation [14], an event-based platform for ubiquitous computing, that allows integrating sensors, inference engines (i.e., parameterisable inferencing mechanisms) and actuators. A visual editor for Sens-ation [26] allows us to easily connect sensors, inference engines and actuators in order to layout the event flow (cf. **Fig. 4**.1). We implemented a variety of the proposed strategies in form of parameterisable inference engines (IE) for Sens-ation, which could be configured in the editor (cf. **Fig. 4**.2). However, for some of the strategies we only provide naïve implementations, as the main focus at this state was on the general feasibility of the framework. For all machine learning algorithms, we used the WEKA toolkit [32].

For *Resampling* and *Segmentation* we implemented two separate IEs, each configurable to use one of various strategies. For resampling, we implemented one simple strategy called LastValueStrategy: each time a new sensor event arrives, the value is stored. Depending on a chosen push method, the stored last values for all sensors are either passed on to the next inference engine, each time a new senor event is incoming, or in a configurable time interval. For segmentation, we implemented a simple FixedWindowStrategy, with a configurable window size. While the IE for *feature construction* implements several of the mechanisms proposed above, the IE for *feature extractions* only extracts two simple features, which is the day of the week and the hour of the day. Finally, for the *online feature selection* (OFS), we implemented a naïve random strategy, that randomly selected a percentage of the features for the feature vector. While we were aware that this decision degrades the machine learning performance, the lack of implementations for OFS algorithms at this time in standard machine learning toolkits and the focus of the end-to-end feasibility did not allow for a different solution.

For the active learning IE we implemented two *ActiveLearningStrategies*: a FixedUncertaintyStrategy with a configurable minimum and maximum threshold for the confidence value; a strategy based on the COWEB clustering algorithm that queries for labels, when changes in the underlying data are detected (ClusterStrategy). The IE for the model and the learner (i.e., classifier algorithm) provides the most flexibility by accepting WEKA command-line strings, for setting and configuring a classifier. Of course, the IE accepted only incremental classifiers, that is, all WEKA classifiers that implement the Interface UpdateableClassifier. Furthermore, the use of advanced strategies for model updating like ensemble classifiers and attribute bagging were not applicable, as we had no sophisticated mechanism for feature selection.

Finally, the adaptation mechanisms as well as the label query were implemented as Sens-ation actuators. For the adaptation we implemented two demonstrators: One was able to control the system volume of a PC, the other was able to adapt the Instant Messaging status. For the querying mechanisms, we implemented a simple, XML-configurable ESM-like dialog (cf. **Fig. 3**). The feedback of the user was passed back to the Sens-ation in form of a sensor event.

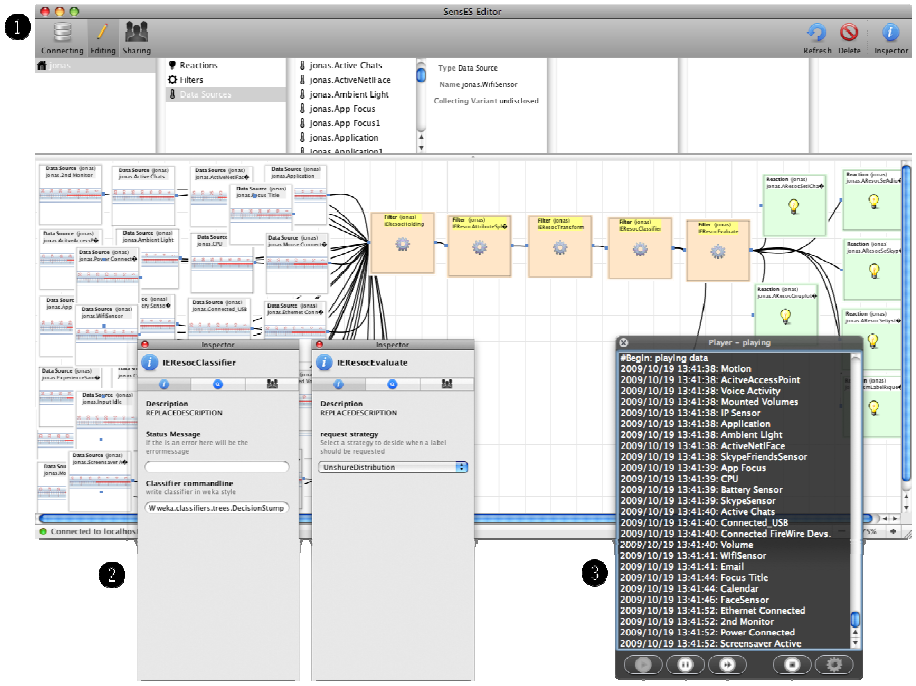


**Fig. 4.** Overview of our setup for the simulation, including the event flow in the visual editor (1) between sensors (white boxes with scale symbol at the left), inference engines (beige boxes with gear symbol in the centre) and actuators (lime boxes with light bulb symbol at the right); two inspector windows, each showing the parameters for an inference engine (2); and the sensor data player (3)

For the evaluation we relied on a dataset from previous work that assessed the predictability of availability for IM [10]. In average an accuracy of 81.35% was reached in this work for this dataset. It needs to be noted that these are the results of an offline, explorative machine learning approach, where step-by-step the best features, algorithms, and parameters where chosen by a human expert based on computational expensive grid search. Accordingly, this accuracy was not considered not as a benchmark but only as a guiding value for our current implementation, relying on naïve implementations and with near real-time constraints.

The data provided a variety of characteristics that deemed it an ordeal for our approach. The dataset, collected by four individuals consisted of several month of sensor data recorded on their laptop computers, together with two estimates of their own availability for IM every hour. 30 different sensors collected data such as nearby Bluetooth and Wi-Fi devices, running applications, connected USB devices, the presence of people, etc., with sampling rates between 30 and 150 seconds and in form of sensor events of varying dimensionality (i.e., sensors that only provide a single value per reading up to sensors that provide a multi-dimensional matrix of values). For the performance evaluation, we implemented a playback application (cf. **Fig. 4**.3), which read the data from the provided XML-dataset files and send it to Sens-ation—practically simulating events from live sensors. For the evaluation, the temporal information in the datasets was omitted, and the data was played back as fast as it could be read from the files. In Sens-ation, all incoming sensor data was pre-processed. However, of course only that data could be evaluated where a label was available as evaluation criteria. For the evaluation, a specially programmed actuator allowed to write out the data received from the evaluation inference engine.

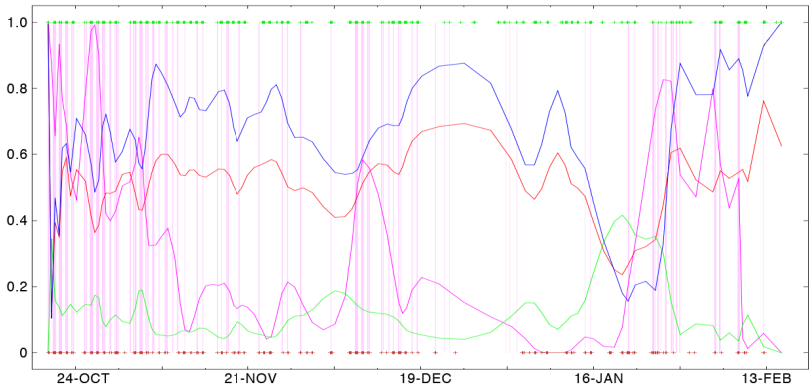


**Fig. 5.** Results of the NaïveBayesUpdateable classifier over time for one dataset. Each red and green cross marks the availability of a label in the data set, that was either predicted right (=1.0) or wrong (=0) by the classifier, whereby each vertical pink line shows a query for a label triggered by the *ActiveLearningStrategy*. The blue line shows the overall accuracy of the classifier over time. (Further: red line = confidence for real class; pink line = confidence for predicted class; green line = normalised absolute difference between real and predicted class).

As many existing metrics for measuring the performance of machine learning (e.g., accuracy, precision or recall) are of limited expressiveness for our setting of lifelong learning, we illustrate our results based on a trend chart over time. **Fig. 5** exemplary shows the results for the dataset of one user over a period of roughly four months (Oct. to Feb). The algorithm used for classification was an incremental Naïve Bayes implementation. A FixedUncertaintyStrategy was applied as the *ActiveLearning Strategy* with a minimum threshold of 0.6 for the confidence value—that is, if the confidence for a prediction falls below the 60% mark, a label is requested. The chart shows the accuracy of the classifier (blue line), quickly goes up and then dithers

around 70% until in the last third—the weeks around New Year— it suddenly breaks down to 20% until it goes up again. This effect is likely caused through the occurrence of a concept or feature drift. In this specific case, probably due to a holiday break of the user that provided the data, which led to different circumstances, a different environment or different availability preferences for this period. Also clearly visible is how the number of queries for labels (vertical pink lines) triggered by the *ActiveLearningStrategy* goes down after an initial intense learning phase. And likewise, how the concept or feature drift around New Year, leads to a more intense training phase at the end of January. While the overall accuracy of the classifier does not reach a performance as demonstrated in [10]—where an average accuracy of 81.35% could be achieved in an offline evaluation with manually crafted feature sets and hand-picked algorithms—it is also true that our implementation does yet not fully exploit the full power of our proposed concept. As in most parts we relied on naïve implementations for the strategies, our main aims was to demonstrate the general feasibility of a lifelong learning system, based on our proposed conceptual framework. The integration of a real feature selection mechanism and the application of more advanced learners like kernel learners or tree ensembles are likely to boost the performance, leading to less queries and a higher accuracy—closer to that of the manual process in [10].

## 6 Conclusions and Future Work

We presented the LILOLE-Framework, a conceptual framework for lifelong learning from sensor data streams for predictive user modelling. The conceptual framework combines the strength of data stream mining and active learning, and thus allows building complex adaptive systems that learn from individual user over a prolonged period of time. We provided detailed insights in the necessary components and building blocks, as well as the underlying processes and data flow, which can guide the engineering process and architectures of similar systems. An end-to-end implementation, partially based on simple and naïve algorithm implementations, already proves that our system is able to learn form an evolving data stream and to adapt to concept drifts. In future work we will extend the number of implemented algorithm, and add more sophisticated mechanisms to the implementation, as already described in the concept.

Mainly, our implementation will ultimately allow us to assess more human-centred aspects of the approach of life-long learning. While previous work assessed the improvement of personal user models through life-long learning only theoretical through offline accuracy and performance measures, our implementation now allows to get real user feedback on the adaptation in field tests in real settings. This includes aspects like the adaptation quality or the satisfaction with the training effort, and allows researching and revealing further human-centred aspects.

**Acknowledgments.** We thank members of the CML, especially Jonas Pencke and David Wiesner.

## References

[1] Aggarwal, C.: An Introduction to Data Streams. In: Aggarwal, C. (ed.) Data Streams - Models and Algorithms, pp. 1–8. Springer, Heidelberg (2007)
[2] Aggarwal, C.: Data Streams: An Overview and Scientific Applications. In: Gaber, M.M. (ed.) Scientific Data Mining and Knowledge Discovery - Principles and Foundations, pp. 377–397. Springer, Heidelberg (2010)
[3] Atlas, L.E., Cohn, D.A., Ladner, R.E.: Training Connectionist Networks with Queries and Selective Sampling. In: Neural Information Processing Systems, Denver, CO, USA, November 27-30, pp. 566–573. Morgan Kaufmann Publishers Inc., San Fransisco (1989)
[4] Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: Proc. of the Twenty-First ACM Symposium on Principles of Database Systems - PODS 2002, Madison, WI, USA, pp. 1–16. ACM Press, New York (2002)
[5] Bao, L., Intille, S.S.: Activity Recognition from User-Annotated Acceleration Data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
[6] Bifet, A.: Adaptive Stream Mining - Pattern Learning and Mining from Evolving Data Streams. IOS Press, Amsterdam (2010)
[7] Blum, A., Mitchell, T.: Combining Labeled and Unlabeled Data with Co-training. In: Proc. of the 11th Annual Conference on Computational Learning Theory, COLT 1998, Madison, WI, USA, pp. 92–100. ACM Press, New York (1998)
[8] Bryll, R., Gutierrez-Osuna, R., Quek, F.: Attribute Bagging: Improving Accuracy of Classifier Ensembles by Using Random Feature Subsets. Pattern Recognition 36(6), 1291–1302 (2003)
[9] Consolvo, S., Smith, I.E., Matthews, T., LaMarca, A., Tabert, J., Powledge, P.: Location Disclosure to Social Relations: Why, When, & What People Want to Share. In: Proc. of the Conference on Human Factors in Computing Systems - CHI 2005, Portland, USA, April 2-7, pp. 81–90. ACM Press, New York (2005)
[10] Fetter, M., Seifert, J., Gross, T.: Predicting Selective Availability for Instant Messaging. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011, Part III. LNCS, vol. 6948, pp. 503–520. Springer, Heidelberg (2011)
[11] Fogarty, J., Hudson, S.E., Akteson, C.G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J.C., Yang, J.: Predicting Human Interruptibility with Sensors. ACM Transactions on Computer-Human Interaction (TOCHI) 12(1), 119–146 (2005)
[12] Gama, J.: Issues and Challenges in Learning from Data Streams. In: Kargupta, H., Han, J., Yu, P.S., Motwani, R., Kumar, V. (eds.) Next Generation of Data Mining. Chapman & Hall/CRC, Tailor & Francis Group, Boca Raton (2008)
[13] Gama, J., Rodriques, P.P.: Data Stream Processing. In: Gama, J., Gaber, M.M. (eds.) Learning from Data Streams - Processing Techniques in Sensor Networks, pp. 25–39. Springer, Heidelberg (2007)
[14] Gross, T., Egla, T., Marquardt, N.: Sensation: A Service-Oriented Platform for Developing Sensor-Based Infrastructures. IJIPT 1(3), 159–167 (2006)
[15] Ho, T.: The Random Subspace Method for Constructing Decision Forests. IEEE TPAMI 20(8), 832–844 (1998)
[16] Horvitz, E., Kapoor, A.: Experience Sampling for Building Predictive User Models: A Comparative Study. In: Proc. of the Conference on Human Factors in Computing Systems, CHI 2008, Florence, Italy, pp. 657–666. ACM Press, New York (2008)
[17] Horvitz, E., Koch, P., Apacible, J.: BusyBody: Creating and Fielding Personalized Models of the Cost of Interruption. In: Proc. of the 2004 ACM Conference on Computer Supported Cooperative Work, CSCW 2004, Chicago, Illinois, November 6-10, pp. 507–510. ACM Press, New York (2004)

[18] Kapoor, A., Horvitz, E.: Principles of Lifelong Learning for Predictive User Modeling. In: Conati, C., McCoy, K., Paliouras, G. (eds.) UM 2007. LNCS (LNAI), vol. 4511, pp. 37–46. Springer, Heidelberg (2007)
[19] Keogh, E., Selina, C., Hart, D., Pazzani, M.: Segmenting Time Series: A Survey and Novel Approach. In: Last, M., Kandel, A., Bunke, H. (eds.) Data Mining in Time Series Databases, pp. 1–22. World Scientific Publishing Co., Singapore (2003)
[20] Liu, H., Motoda, H. (eds.): Feature Extraction, Construction and Selection - A Data Mining Perspective. Kluwer Academic Publishers, Norwell (1998)
[21] Markovitch, S., Rosenstein, D.: Feature Generation Using General Constructor Functions. Machine Learning 49(1), 59–98 (2002)
[22] Opitz, D., Maclin, R.: Popular Ensemble Methods: An Empirical Study. Journal of Artificial Intelligence Research 11, 169–198 (1999)
[23] Reiss, A., Stricker, D.: Personalized Mobile Physical Activity Recognition. In: Proc. of the 2013 International Symposium on Wearable Computers - ISWC 2013, Zurich, Switzerland, September 9-12, pp. 25–28. ACM Press, New York (2013)
[24] Rokach, L.: Ensemble-based Classifiers. AI Review 33(1-2), 1–39 (2010)
[25] Salber, D., Dey, A.K., Abowd, G.D.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In: Proc. of the Conference on Human Factors in Computing Systems- CHI 1999, Pittsburgh, PA, USA, May 15-20, pp. 434–441. ACM Press, New York (1999)
[26] Schirmer, M., Gross, T.: CollaborationBus Aqua: Easy Cooperative Editing of Ubiquitous Environments. In: Proc. of the International Conference on Collaborative Technologies - CT 2010, Freiburg, Germany, July 26-28, pp. 77–84. IADIS Press (2010)
[27] Settles, B.: Active Learning. Morgan & Claypool Publishers, San Rafael (2012)
[28] Wang, H., Fan, W., Yu, P.S., Han, J.: Mining Concept-Drifting Data Streams Using Ensemble Classifiers. In: Proc. of the Ninth ACM International Conference on Knowledge Discovery and Data Mining - KDD 2003, Washington, D.C., USA, August 24-27, pp. 226–235. ACM Press, New York (2003)
[29] Wang, J., Zhao, P., Hoi, S.C.H., Jin, R.: Online Feature Selection and Its Applications. IEEE Transactions on Knowledge and Data Engineering (2013)
[30] Webb, G.I., Pazzani, M.J., Billsus, D.: Machine Learning for User Modeling. User Modeling and User-Adapted Interaction (UMUAI) 11(1-2), 19–29 (2001)
[31] Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23(1), 69–101 (1996)
[32] Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2011)
[33] Wu, X., Yu, K., Ding, W., Wang, H., Zhu, X.: Online Feature Selection with Streaming Features. IEEE TPAMI 35(5), 1178–1192 (2013)
[34] Wu, X., Yu, K., Wang, H., Ding, W.: Online Streaming Feature Selection. In: Procedings of the 27th International Conference on Machine Learning, ICML 2010, Haifa, Israel, June 21-24, pp. 1159–1166. Omnipress, Madison (2010)
[35] Zhao, Z., Chen, Y., Liu, J., Shen, Z., Liu, M.: Cross-People Mobile-Phone Based Activity Recognition. In: Proc. of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI 2011, Barcelona, Catalonia, Spain, pp. 2545–2550. AAAI Press, Menlo Park (2011)
[36] Žliobaitė, I.e., Bifet, A., Pfahringer, B., Holmes, G.: Active Learning with Evolving Streaming Data. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 597–612. Springer, Heidelberg (2011)
[37] Zukerman, I., Albrecht, D.W.: Predictive Statistical Models for User Modeling. User Modeling and User-Adapted Interaction (UMUAI) 11(1-2), 5–18 (2001)