PASIF

A Framework for Supporting Smart Interactions with Predictive Analytics

by

Sarah Marie Matheson

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada
September 2011

Abstract

As computing matures, it is becoming increasingly obvious that a change is necessary for the manner in which web services interact with users. Server-centric models are inconvenient for users. A new paradigm, Smart Interactions, provides a web service architecture which is centered around the user's needs, rather than the simplistic server view currently being used. The system responds to the individual user and is able to adapt to changes to better serve the user. The Smart Internet system helps the user accomplish their tasks efficiently and intuitively.

An important aspect of Smart Interactions is that of cognitive support, which provides enhanced information and guidance to the system or user linked to the current task. This thesis examines predictive analytics and its application to cognitive support in Smart Interactions, and presents and evaluates a framework for using predictive analytic support within the Smart Internet model.

Acknowledgments

I would like to express my sincere thanks to my advisor, Dr. Patrick Martin, for sharing his knowledge, excellent research advice, and patience throughout my time at Queen's. Thank you, too, to Wendy Powley for her academic guidance, long talks on slow afternoons, and being an exceptional mentor.

A fond thank you to my family in New Brunswick for supporting me in my studies with long phone calls, care packages full of chocolate, and endless love and kindness. Finally, thank you to my friends and colleagues, who always knew when it was time to take a break from the books and papers, and have a cup of coffee.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The traditional server-centric paradigm for human-computer interaction, the manner in which many web services continue to interact with users, has limitations which are becoming increasingly apparent [37]. For example, consider a typical e-commerce website where users can shop for clothing, and a typical shopper, Alice. Alice shopped at this online store a few months ago, browsed the collections that were available then, and bought a few items she liked. Alice returns to the store to browse the site again today, hoping to find a product she wants to buy. New items have been added, but Alice still needs to browse through many items she has already seen and dismissed the last time she visited the online store. Some of the clothes she looks at are not available in her size, and some cannot ship to her address. Further, Alice is on a tight budget and does not want to see the very expensive products at the store, but only those which she can afford. An ideal website would analyze current and past behaviours, and immediately suggest to Alice the products that fit her style and are suitable for her, without her needing to do any searching at all [30, 58].

Consider emergency room patient assessments as a second example. When a

patient enters a hospital emergency room, information is immediately gathered about that patient. Hypothetically, their symptoms, along with their demographic and medical history, can be used to make an initial diagnosis and suggest treatment to help identify their priority and help doctors make decisions regarding the patient [1, 36, 46]. However, the current system consists mainly of using paper forms to store information concerning the patient and electronic systems that have no interconnectivity. This results in wasted time collecting redundant information; every time the patient visits a hospital they are asked to repeat much of their medical history and personal information. It also means there is data that is potentially useful in the diagnosis of this patient which has been collected in the past, but is unavailable in this situation.

In these examples, and many others, a paradigm shift is required to move to a user model of the web, where interactions are driven by the needs of the user, rather than by the server. To be fully effective, this interaction paradigm, *Smart Interactions* [37], will require new web infrastructure and new approaches to web services. These will allow more intelligent interactions that centre on user perspective, adapting to the user's needs and providing individualized support.

1.1 Motivation

1.1.1 Smart Interactions

Smart Interactions is an emerging paradigm for human-computer interaction [37]. Traditionally, interactions have been server-centric, where a session consists of a series of requests from the user to a single server and interactions are only supported between

a single user and the server [32].

In the Smart Interaction model, interactions are user-centric [37]. Sessions are focused on user needs, rather than just a series of requests. This may involve synchronous or asynchronous interactions with multiple servers, with support available for both collaborative and collective interactions.

Smart interactions may be useful in many application areas. Presently, the two most prominent are *Smart Internet* and *Smart Environment* [32, 37]. In the Smart Internet archetype, web services are configured and integrated across multiple servers in order to better address the needs of the user. Smart Environment is the branch of Smart Interactions which considers the application of smart techniques in a way that will cause the user's environment to react to their presence and interact with them in an intelligent way. The contributions of this thesis are intended primarily for Smart Internet applications, however, many of the ideas presented can be generalized to Smart Environments.

1.1.2 Predictive Analytics

An important aspect of Smart Interactions is that of providing cognitive support to applications, which consists of helping the user to think about and solve their problems. By providing as much information about the task as possible, the system alleviates the amount of knowledge the user needs to possess in order to successfully accomplish a given task.

Predictive analytics is a method that can be used to assist in decision-making tasks, and is thus a potentially useful tool for the cognitive support of smart interactions. Expanding from business intelligence, predictive analytics attempts to use

business intelligence techniques to form predictions about some future behaviour using data that has been gathered and stored about previous interactions of the system or other data relating to it [4].

Business intelligence (BI) differs from predictive analytics in a few significant ways. BI uses data-mining techniques to find trends in data and linearly project these trends to future behaviours, while predictive analytics uses available data to forecast trends and behaviours and uses these predictions to guide specific decisions [4].

This thesis surveys the areas of Smart Interactions and predictive analytics, and proposes PASIF: **P**redictive **A**nalytics in **S**mart **I**nteractions **F**ramework. PASIF is a framework which allows the integration of predictive analytics techniques to provide cognitive support for Smart Internet applications, and thus support users in carrying out tasks on the Web.

1.1.3 Real-Time Analytics and Concept Drift

Real-time data analytics, also known as real-time data integration and real-time intelligence, use data as it becomes available, along with other available data resources as they are needed, in order to form dynamic predictions and analysis on data as it is being collected [2, 57]. By employing real-time analytics, information can be collected about a user and incorporated efficiently into the current predictive analytics system used by a Web service.

Furthermore, effective real-time analytic systems must be able to adapt to changes in the underlying distribution of data as the system runs. This underlying change is known as *concept drift* [2]. For example, in e-commerce systems this might describe a shift in the products people are interested in buying. In an emergency room scenario,

it could describe the recognition of a new epidemic or a change in the set of available treatments.

Predictive systems that are intended for long-term use, which is often the case for smart internet systems, are prone to concept drift. Thus, effective prediction services will need to adapt to these changes in distribution by incorporating concept drift detection techniques.

1.2 Thesis Statement

This thesis surveys predictive analytics and its application to cognitive support in Smart Interactions, and describes and evaluates a framework for using predictive analytic support within the Smart Internet model. A prediction service for the framework is implemented, analyzed and evaluated. Specifically, the practicality of the modelling approach is determined based on its ability to respond to user interactions in real-time, its reaction to concept drift, and the ability to adapt based on user feedback. Further, the scalability to larger systems is discussed.

1.3 Contributions

The focus of this work is the end-to-end framework which integrates predictive analytics into a web service to provide cognitive support for Smart Internet applications. The framework makes use of the data warehousing routines presented in Bigus et al [11], which provides the foundation for the data-extraction, transformation and loading of data collected online into an efficient and organized mining database.

Predictive models are built offline using a subset of data from the mining database

for training, and are used in conjunction with data collected about current interactions to provide real-time support. As the data is collected, concept drift detection techniques are used to determine if changes have occurred in the way users are interacting with the system. User feedback is used to determine which models and data types best represent the needs of users, using a weighted ensemble modelling technique.

The contributions of this work can be summarized as follows:

- We propose a generic framework for incorporating predictive and real-time analytics in Smart Internet applications, expanding on the CRM framework of Bigus et al [11], an end-to-end framework for developing and deploying prepackaged predictive modelling for businesses.
- We combine ensemble modelling with a weighted scheme based on user feedback, along with offline model building tied to a mining database, to create an online prediction service, and analyze the feasibility of its deployment in real-time applications.
- We incorporate concept drift detection in the modelling process to adapt to changes in the underlying distribution of data as it is collected by the system.

1.4 Organization of Thesis

The remainder of this thesis is organized as follows: Chapter 2 discusses Smart Interactions, predictive analytics, data mining and real-time data analytics, and the leading techniques in these fields. Further it discusses existing real-time analytic frameworks and how they compare to the work presented in this thesis. Chapter

3 presents the generic framework, PASIF, for incorporating predictive analytics into Smart Interactions. Chapter 4 describes the design and proof-of-concept implementation of the predictive service subsystem, and the specific techniques that are used to accomplish real-time and adaptation to concept drift. Chapter 5 presents experimental results for the implementation, and discusses the performance of the framework and its feasibility for being deployed on real systems. Chapter 6 summarizes the contributions, draws conclusions, and lists some possibilities for future work.

Chapter 2

Background and Related Work

2.1 Smart Interactions

Smart Interactions describes a new paradigm of intelligent human-computer interaction, where the system is made to be as user-centric as possible by providing high levels of support to the user and simplifying tasks. This can be separated into two prominent areas of study [37]: Smart Internet, which is the motivation behind this thesis, and Smart Environment, which is concerned with integrating computers seamlessly into everyday life.

Smart Internet differs from the current internet in three main ways [37]. First, the Smart user model for the Web revolves around the user's concerns and cognition, differing from the traditional model which centres on interactions with a server. Second, the concept of a session is altered to centre on the user's perspective rather than the server's perspective. By centering on the user, the system focuses on the specific situation and the tasks the user is trying to perform. In the conventional server model, a session simply consists of a series of requests given to the server by

the user. To achieve this concept of a Smart Internet session, the system must be able to support both synchronous and asynchronous interactions across multiple servers. The third difference is the establishment of dynamic social binding in Web interactions. Previously, sessions were conducted between a single user and a single server. Smart Internet allows effortless collaboration between users and collective interaction with servers.

Many properties of the server-centric model of the Web create inconveniences for the user which the Smart model intends to resolve [37]. The server-centric model lacks integration from the user's perspective, forcing users to seek and access information from multiple independent websites in order to accomplish a single task. Furthermore, all users are treated the same; more support for individualization of web services would add much usability for users. This individualization might be accomplished by having the web service become more aware of the user and adapting to their personal needs. In present web services, service level collaboration (allowing multiple users to collaborate when interacting on the Web) is absent, but is available in the Smart conceptualization of a web session. The Smart Internet additionally provides users the ability to control the layout and content that is displayed on web pages they visit, which is virtually nonexistent in current interactions.

In summary, effective Smart Interactions should provide a number of features including task simplification, cognitive support, context awareness, adaptive aggregation and dynamic collaboration [37]. These assist the user to accomplish tasks efficiently and in a straightforward manner.

2.1.1 Decision-Making Scenarios

In this thesis we propose the PASIF framework, a framework that assists in creating Smart Technologies by providing real-time cognitive support through predictive analytics. In this section we outline some scenarios where real-time decision-making support is needed in order for Smart Interactions to exist.

Personalizing Online Shopping [30, 58] Electronic commerce systems can use Smart technologies to create a more individualized environment for their customers to shop online. Data collected about a client's behaviour, such as their browsing and purchase history, along with demographic and personal information, can be used to recommend services and products which are most likely to be attractive to that customer. This generates an environment where the consumer spends less time searching, as the products they are looking to buy are suggested to them.

Emergency Room Patient Assessment [1, 36, 46] Currently, many emergency rooms use an outdated patient assessment system which consists of using many paper forms to store information concerning the patient (often containing redundant information) and electronic systems that have no inter-connectivity. Information regarding a single patient is scattered across many platforms, and requires doctors and nurses to use numerous access points to obtain all relevant patient information. In Smart Interactions, the system can integrate all information regarding a single patient stored in the hospital's databases and allow the authorized hospital personnel to conveniently obtain it at a single access terminal, saving both time and effort.

Cognitive support might be incorporated into this system as follows. When a patient arrives at an emergency room, the staff inputs an initial profile for the patient

into the system. The system feeds this data into a predictive modeller which uses this new information, along with other data such as the patient's recent history and medical records, records of patients with similar profiles, and information specific to the hospital, in order to guide the medical personnel in assessing the urgency or priority of the new patient, or possibly the suggested treatment to be given to the patient.

Selecting An Investment [34] A user can provide information about their financial goals to an online finance system, which can use this to build an investment profile for the user. Collaborative filtering methods can use the habits of similar profiles, and use the results of their investment decisions, along with other data, to make predictive models and suggestions for what investment decisions the user should consider making in order to best meet their long and short-term financial goals.

Customized Web-Marketing [55] In an electronic commerce environment, a predictive model can make suggestions about the most effective placement for advertisements and the recommended content for the advertisement based on the consumer's purchase and product-browsing history, their user profile and demographic, or the similarity of this user to other clients and the effectiveness of advertisements shown to them in the past.

2.2 Data Mining

Data mining, also known as machine learning and knowledge discovery, is the process of analyzing data sets to extract useful information from them [29, 43]. Given a

large collection of data, one can apply one or more data mining techniques in order to discover previously unknown patterns within the data, or find trends in the data which can then be used to predict future trends or behaviours. Data mining can be divided into two main categories: *supervised* (predictive) and *unsupervised* (descriptive) [43]. These are described in the following subsections.

2.2.1 Supervised Learning

In supervised learning, data is modelled from training data to find patterns within the data which can then be used to predict a label or value, given some set of parameters [43]. The type of data determines whether this is done using a regression or a classification algorithm.

Regression

Regression is a data mining technique used to fit an equation to a dataset [43]. Ideally, such a dataset is continuous and quantitative, rather than categorical. In the latter case, one should opt for classification mining in order to achieve better results. For example, regression could be used to assist in predicting the temperature for a weather forecast by analyzing temperatures patterns of past years, as well as those in recent days and the current air pressure and weather systems.

The simplest form of regression, linear regression, uses the formula of a straight line ($y = mx + b$) and determines the appropriate values for m and b to predict the value of y based on the input parameter, x. Advanced techniques, such as multiple regression, allow the use of more than one input variable and allow for the fitting of more complex models, such as higher order polynomial equations.

Once a reliable equation is found, a prediction for new data can be quickly made through a simple calculation. Additionally, regression is a well-established statistical technique, and therefore many efficient and sound implementations exist.

Classification

Classification is the set of data mining techniques used to fit discrete (categorical) data to a known structure in order to be able to form predictions for the *class label* of unlabelled data [43]. For example, given a set of diagnostic parameters, such as heart-rate, age, sex, and medical test results, this data could be used to classify a patient as "healthy" or "diseased".

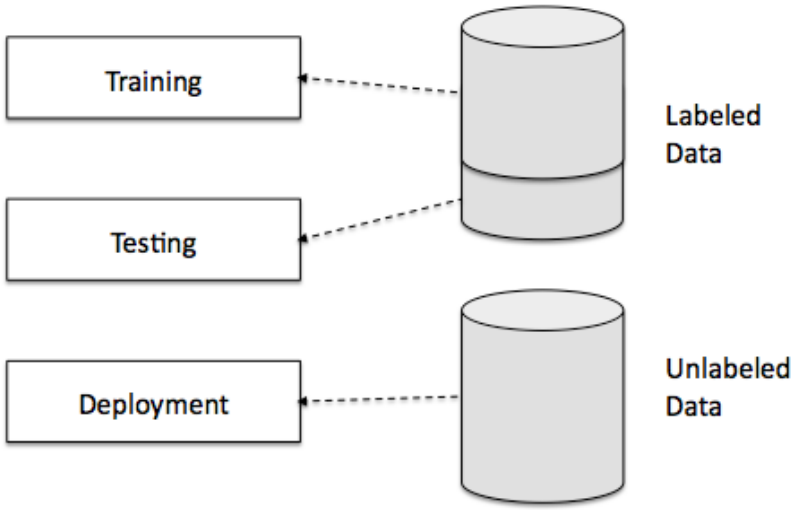


Figure 2.1: Phases of Standard Classification Techniques

Typically classification algorithms are done in three phases, as shown in Figure 2.1. The first two phases, training and testing, use labelled data. That is, data which has known class labels. Training uses a portion of the data to fit a classifying model

to the data. The testing phase then uses the models to try and predict the class labels, and validates the predictions using the actual values in order to determine how accurate the model is. The feedback from this determines how well the models work, and whether new models should be built. Once an acceptable model is built that passes the testing phase, the classifier is deployed on unlabelled data. This is called the deployment phase. Common classification algorithms include Bayesian classification, decision trees, back-propagation and neural networks, and genetic and evolutionary learners [43].

2.2.2 Unsupervised Learning

The goal of unsupervised learning is to understand and discover previously unknown implicit relations or properties. Unlike supervised algorithms, unsupervised do not learn from historical data with known labels, hence, they perform without any supervision [43]. Standard unsupervised techniques include clustering, characterization, association rule mining, and change and deviation detecting techniques [43].

Clustering is a type of algorithm which groups data into smaller sets whose elements share similar characteristics. Characterization, also known as generalization or summarization, attempts to represent the data in a compact form. This might be statistical values, such as averages and variances, in graphical formats, such as histograms or pie-charts, or other human-readable formats. Association rule mining, or dependency modelling, is a technique which searches for notable relationships between items in a data set. Change and deviation detection techniques find the times at which significant changes in the underlying relations have occurred.

2.3 Predictive Analytics

Predictive analytics use data-mining techniques in order to make predictions about future events, and make recommendations based on these predictions [4]. This should not be confused with making predictions using business analytics. Similar to predictive analytics, business analytics use data mining techniques, but rather than modelling or predicting future behaviours, the system finds trends in past data and uses them to anticipate what might happen in the future if the system continues to behave in the same manner [4]. In business analytics, the focus is on finding patterns and tendencies that have occurred in the past and linearly projecting them into the future, without making any hypothesis about why these tendencies exist. The trends in the historical data are simply extended in order to make a prediction, with the assumption that the system will continue to behave exactly as it has in the past.

Predictive analytics perform many of the same tasks as business analytics, and are often built upon business analytic systems. It can be said that predictive analytics are prescriptive, rather than descriptive [11]. For example, if a model is able to predict errors based on a correlation of variables then it should be able to do an analysis and recommend a solution. Additionally, predictive analytics are able to not only deal with continuous changes, but discontinuous changes as well.

Subsection 2.3.1 describes recommendations systems and their role in Smart Interactions. The remaining subsections focus on common techniques for predictive analytics including segmentation-based modelling, cost-sensitive learning, reinforcement learning and collaborative filtering.

2.3.1 Recommendation Systems

A growing issue for computational systems is the problem of information overload [31] that occurs when there is an overwhelming amount of data that is made available to users, making it difficult for them to identify and locate the particular information that is relevant to them. Recommendation systems provide a form of information filtering that removes extraneous data and presents to the user only the small subset of available information that is considered to be most applicable to the current user task [31].

Recommendation systems can be used as a form of predictive analytics. By predicting what information is most valuable to a user, recommendation systems help decide which data to display and how to present it in a useful way. Thus, having at least some form of recommendation or information filtering is essential for many Smart Internet applications.

In order to make recommendations, the system can make use of available statistical data about the user and other users with similar preferences, their profile and demographic information, and the current context of the application environment in order to decide which data is most relevant to the specific user and situation.

Recommendation systems are common in applications involving electronic commerce. Electronic commerce, or e-commerce, consists of the buying and selling of products or services over a computer network, most commonly, the Internet. Personalized recommendations are utilized in these applications to enhance responsiveness of customers and improve relationship marketing [14]. Principally, statistical and knowledge discovery techniques are used to make product recommendations in real-time customer interactions in online shopping environments [44].

Approaches to personalized recommendation systems are classified as either content-based or preference-based [14]. In the content-based approach, techniques such as associative rule mining are applied to a customer's purchase history in order to offer product recommendations to them. In the preference-based approach, product recommendations are made based on other customers who have similar interests and historical data, using techniques such as collaborative filtering.

Using predictive analytic techniques to perform recommendations, rather than the traditional method of having users navigate the database by performing searches, allows for a much more user-centric environment. Traditionally, there are two types of tasks involved in forming recommendations: (i) predicting the order of preference for possible components or actions from the perspective of a given user, and (ii) predicting which action the user will take next [49].

2.3.2 Segmentation-Based Modelling

Segmentation-based modelling is an approach in which data records are partitioned into segments, and separate predictive models are applied individually to each segment. For consumer-based predictive models, the segments might correspond to subpopulations of the customers in the database, or to a subset of transactions that have occurred.

The process of segmentation modelling is traditionally a sequential process. The segmentation of the data records is first performed using a process such as conventional decision tree algorithms, unsupervised clustering algorithms, domain expertise or intuition [6]. After these segments have been defined, predictive models are developed individually for each segment.

A drawback of the sequential segmentation model is that it ignores the strong influence that segmentation exerts on the predictive accuracies of the models within each segment. Some segmentation-based models apply the segmentation and develop the predictive models for each segment simultaneously, avoiding the negative influence present in the sequential approach. A notable example of this is ProbE, which is an autonomous data-mining engine developed for the enhancement of predictive modelling systems [6]. This engine uses a segmentation scheme whose production depends heavily on the predictive models that are generated for use in each of its segments.

The PASIF framework makes use of segmentation-based modelling in order to deal with heterogeneous data sources. However, while the prediction service uses a system consisting of multiple models, the process is better described by the idea of classifier ensembles, which is discussed in Subsection 2.3.6.

2.3.3 Cost-Sensitive Learning

Learning classifiers attempt to create a policy that can make a decision based on environmental data, which will theoretically result in the best outcome. In classical machine learning and data mining applications, the assumption is made that all possible errors made by the classifier are equal. This is often not the case. For example, in medical diagnosis applications, if an error is made that mistakenly diagnoses a healthy patient to be ill then it may have less dire consequences than diagnosing a sick patient as healthy. Cost-sensitive learning strategies account for variations in cost as a consequence of erroneous decisions made by the policy [3, 33].

Boosting learning algorithms are a family of techniques which begin by learning

a prediction model through the combination of weak classification learning rules and then using a sample re-weighting mechanism to emphasize points that are difficult to classify. These can be extended to be cost-sensitive learners by modifying the boosting's loss function so that the two following conditions are met: 1) the expected boosting loss is minimized by the optimal cost-sensitive decision rule and 2) the empirical loss minimization emphasizes a neighbourhood of the target cost-sensitive boundary [33].

Cost-sensitive learning should be supported in a Smart Interaction framework, as many decision-making scenarios require considering varying levels of consequences. Consider the health care scenario as described above, for example, or an investment strategy scenario might value losing money as more expensive than gaining.

2.3.4 Reinforcement Learning

Reinforcement learning is an approach often applied to predictive analytics. The learning environment consists of a discrete set of states and actions, and a set of transitions between them. At each step of the learning algorithm, the environment takes one of several viable actions, receives some positive or negative finite reward and transitions to the corresponding state. The goal is to maximize the total reward accrued over time by finding a policy that will choose the optimal action for any state of the environment. By making observations of the system during experimental actions, the learner attempts to infer this optimal policy.

There are three categories of methods for reinforcement learning: indirect, direct and semi-direct [3]. This categorization describes how the environment is modelled and the learning strategy employed by the system. The remainder of this subsection

discusses each of these categories, and the approaches used by them.

Simulation-Based Reinforcement Learning (Indirect)

Indirect, or "simulation-based", learning methods attempt to create a model of the environment. These models are then used by the learner to experiment with and observe in order to learn and make decisions. Consider, for example, the approach for indirect reinforcement learning presented by Abe et al. [3] in which a Markov decision model (MDP) is built through estimations of transition probabilities and their associated immediate rewards. Using the data generated by this model, the system learns and updates the policy based on current estimates of the value function. By applying an estimated optimal policy, rather than a function approximation, it is possible to introduce new data types as the system learns. Thus simulation-based reinforcement learning may be a good candidate for the prediction service in the PASIF framework, as introducing new data types may be necessary in some cases, as is discussed in Chapter 3.

Batch Reinforcement Learning (Direct)

Direct, or "Batch", learning methods directly estimate the values of possible actions at any given state, rather than trying to create a dependable model of the environment. Often, a value-iteration based supervised learning scheme is used in order to estimate a value function which will be used as the policy in future decision-making. Notable direct methods include the Adaptive Heuristic Critic algorithm, Temporal-Difference Prediction and Q-learning [26].

Temporal-difference prediction methods belong to a class of prediction methods

called incremental learning procedures. The difference between temporally successive predictions is used to update the learner's policies, rather than the conventional method of using only the difference between an initial prediction and actual outcomes [47]. At each step in the algorithm, the prediction of a future event is updated to reflect changes or differences in the environment since the last step. The Adaptive Heuristic Critic, an adaptive policy iteration algorithm [7], uses a similar temporal-difference prediction, but rather than maximizing an immediate reward, the learning attempts to maximize some heuristic value.

Q-Learning [51, 52] is a technique that uses an expected reward function that, given the state of the system, gives the expected value of taking an action and follows a fixed policy thereafter. Consider the Sarsa (State-Action-Reward-State-Action) learning algorithm [42]: an on-policy implementation of Q-learning which interacts with the environment and updates its policy based on an expected immediate reward function by using supervised learning on characterizing features. In the first iteration, a value function is obtained based on the expected reward, and is updated in subsequent iterations to reflect the change in policy.

Semi-Direct Reinforcement Learning

The third category of reinforcement learning, semi-direct learning, combines the strategies of both the direct and indirect methods. Essentially, it is a direct method, but mirrors some indirect aspects such as data-sampling and reward estimations. Abe et al. [3] propose an algorithm in which a selective sampling of data is performed. Only those states which conform to the condition that the action taken in the next step is the best action, with respect to the current estimate of the value function,

are used. In the learning process, the rewards that are estimated by the function are considered, rather than those that are actually observed.

2.3.5 Collaborative Filtering

Collaborative filtering [21] is a class of predictive analytic techniques that focuses on finding resemblances between entities, and uses close relationships to make predictions about them. This is a technique which is commonly used in consumer recommendation systems.

A customer who has a history or demographic that bears a significant resemblance to another's is likely to make similar purchases. Thus, a web service can recommend a product to a customer based on the fact that a very similar customer has purchased it, or a similar product, in the past. Similarly, web pages visited by one may be recommended to the other.

Collaborative filtering can be done in three ways: content-based, memory-based and model-based. Content-based uses profile information about either items or users in order to find similarities about them, without considering any feedback or ratings [35]. Memory-based techniques find user-to-user or item-to-item correlations based entirely on user feedback [12], and requires the use of the entire training set. Model-based systems use a subset of the data to train the system, and create clusters of users or items based on user feedback [5]. Singular Value Decomposition is a successful form of model-based collaborative filtering which stores users or items as vectors, and uses a dot product to calculate correlation [40].

The memory-based approach is ruled out as a candidate technique for an adaptive model for the PASIF framework due to its need for a complete data set. The size

and adaptation requires only subsets of data be used at any given time. For all the techniques, speed is an issue.

Smart interactions require fast reactions from the modelling system, but due to its computational intensity, most forms of collaborative filtering are a weak choice for the predictive framework. Additionally, while current collaborative filtering techniques work well on dense data sets, they are found lacking in sparse settings [5], such as is typically found in data collected in e-commerce and other web services.

Item Set Recommendation

The item set recommendation problem, also known as dependancy modelling, is a particularly well-studied form of the recommendation problem. It concerns the decision of which items should be recommended to a consumer given a set of items that have been added to a virtual shopping cart. A set of items purchased in a single transaction is called a *market basket*.

Before a transaction has been completed, the set of items in a market basket are called a partial basket, and might not be the final basket that is purchased. Memory-based approaches identify similarities between users based on their ratings they have previously assigned to of a set of items [25]. Statistical modelling is a technique that can be used as a possible solution to this problem, such as the approach by Hong et al. [24]. In their approach, recommendations are made using a probability based on partial market baskets (associative buying), as well as an independent recommendation based on the user along with their partial market basket (renewal buying).

Memory-Based Filtering for Item Set Recommendation

Most memory-based collaborative filtering systems use an approach similar to that of Conner et al [15], a nearest-neighbours approach which consists of three phases. First, the users of the system rate items or services that they have previously experienced. The filtering system then matches the user, for whom they are trying to create a predictive model, with other participants who have similar rating patterns. These relations might be obtained through simple statistical correlations.

The closest matches are selected as neighbours of the user, and become the user's *neighbourhood*. Finally, items that the neighbours have rated highly, but that the user has not yet experienced or used are recommended to the user, ranked based on *closeness*, a pre-defined metric which measures item similarity.

Clustering

A common problem in recommendation systems that use collaborative filtering is their inability to scale, due to space and calculation complexities. Clustering is a method that can be used to allow item recommendation using item-based collaborative filtering to have much higher scalability than with standard techniques. Clustering refers to an unsupervised process through which data items are classified into disjoint groups (clusters) based on similarities between them [39].

The k-means algorithm [25] is a basic clustering algorithm that uses an input value, k, to decide how many clusters to create . The first k items encountered by the system, which are typically randomly chosen from the data set, are used as the centers of the clusters, and each remaining item is compared to the closest cluster. The centers are re-computed in each subsequent pass of the algorithm, based on the

comparisons of the added items. It is found that the number of neighbours has a significant effect on the quality of predictions made by the system. Thus, since this algorithm allows control of this value, better prediction models are possible than with traditional collaborative filtering methods.

The k-means algorithm can be expanded to support range attributes [15], allowing the user to make preferences along multiple dimensions. Furthermore, this allows more efficient manipulation of very large data sets in predictive models.

2.3.6 Classifier Ensembles

Classification systems can be built by using multiple classifiers and forming decisions by combining their predictions. The classifiers may implement different classification algorithms, or they may all implement the same with only the data they use differing. This is called the Classifier Ensemble Method [41]. The general structure is shown in Figure 2.2.

For example, a set of four classifiers might be built using four different random samplings of the training data. When deployed, instances are received by the system and relayed to each classifier. Each classifier makes a classification, and the four values are then aggregated in order to form a final decision.

The aggregation might be done using a weighting scheme, based on some metric such as the accuracy of the classifier so far. The aggregation may, on the other hand, only consider the best classifiers of the system, and omit those with poorer performance or accuracy, entirely.

Generation of the data-sets used for training is typically done using bootstrap sampling [13] or boosting [19, 43]. Bootstrap sampling, or "bagging", is done by

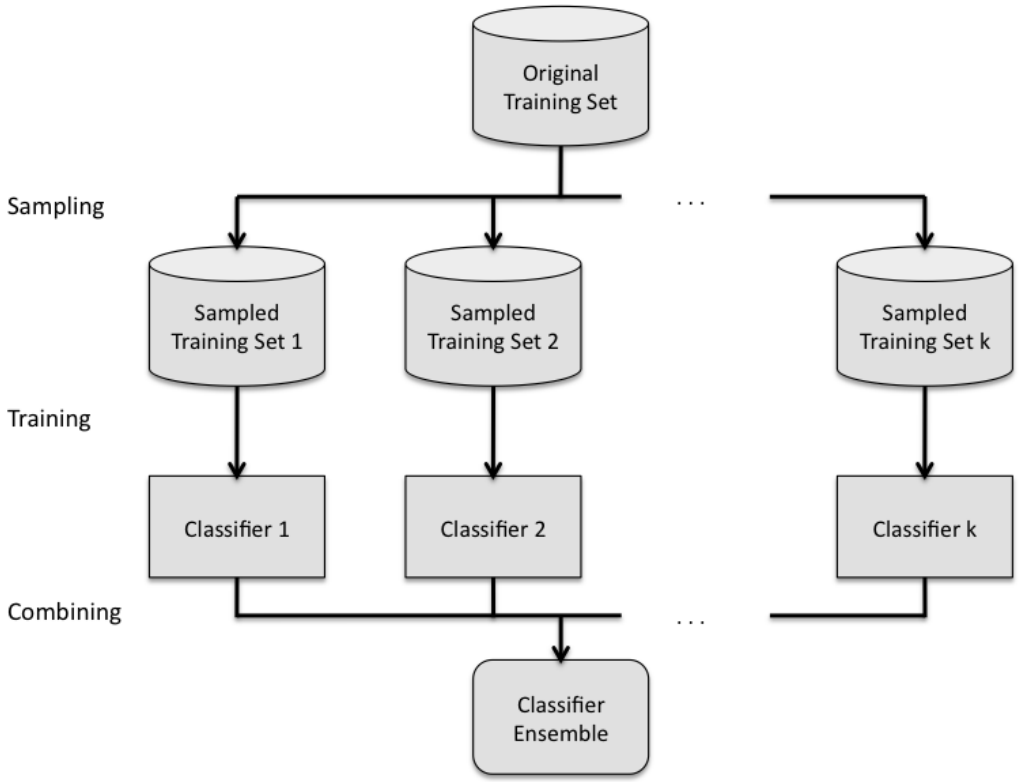


Figure 2.2: Classifier Ensemble Structure

random sampling with replacement. Boosting samples and trains each classifier in succession. The instances are weighted based on the difficulty that previous classifiers had in classifying them, and this weighting is used to derive the data sampling for the datasets.

Boosting maintains weights for records in the training set, such that these weights are updated after each classifier is trained according to the difficulty of classifying that set of records. That is, how difficult it is to create a model that will correctly model the behaviour of that record. The weights are then used to decide which data

instances to focus on when building future classifiers.

By using ensemble-based classifiers, bias caused by the way a given classifier works is avoided, and variance between classifiers of the same type is reduced. Therefore, ensemble classifications better represent the data than a single classifier, and are known to perform better [38].

The Streaming Random Forests framework of Abdulsalam [2] uses an ensemble-based method in order to classify streaming data. The framework proposed in this thesis uses a similar prediction system, but opts for non-streaming data classification in order to form more accurate models.

Type	**Techniques**	**Notable Features**
Recommendation	Content-based and Preference-based	- Optimized for recommendations
Segmentation-based	Sequential, Simultaneous	- Can find multiple distinct trends - Segmentation process can exert strong influence on accuracy within its models
Cost-Sensitive Learning	Boosting	- Account for error cost variations
	Classifier Ensembles	- Allows weighted approach - Can adapt to concept drift
Reinforcement Learning	Indirect, Direct, Semi-direct	- Can introduce new data - Does not require the creation of a dependable system model
Collaborative Filtering	Content-based, Memory-based, and Model-based	- Able to make predictions about an entity for which you have no prior data - Requires complete data set - Computationally intensive – not good candidate for real-time modelling

Table 2.1: Predictive Analytic Techniques

2.4 Real-Time Data Analytics

Real-time data analytics, also known as real-time data integration and real-time intelligence, use data as it is available, along with all other available resources as they are needed, in order to form dynamic predictions and analysis on data [57]. By using real-time analytics, information is kept current in order to form better models of the system, and quick decisions can be made.

For example, the current interactions of an e-commerce customer, such as what they are looking at and the way they are browsing, can be used in real-time to make decisions about what information should be shown to them. Further, the data collected about this interaction can be incorporated quickly into the larger scale models of the site's prediction systems. Real-time data analytics should be able to adapt quickly to changes in the data, as is discussed in Section 2.5.

"Real-time" refers to systems in which the computer responds fast enough so that the user believes it is either immediate, or nearly so. It might also mean that the system reacts in some acceptable amount of time, where this amount of time depends on the specific application.

Consider, for example, the work of Zhang et al [57] on streaming data analytics for health-care scenarios. The framework is intended to be used to help users to monitor and query trends in physiological signals data, with real-time detection and response to adverse trends.

2.5 Concept Drift

Concept drift in data-streams occurs when the underlying trends in the data change over time. Drift detection algorithms have applications in many domains including monitoring and managing network traffic, intrusion detection, and system analysis [2].

Two types of concept changes might be detected in a data set. A sudden short-term change is called a stream burst [59, 60]. Stream bursts are of interest in many applications such as automated stock trading algorithms. Long-term changes in the underlying distribution of the process generating the data is called a distribution change [16, 27, 28, 50]. This might prompt a change in the way the system works or how we model the data.

There are two common approaches to concept drift detection [48]. The first is the sliding window method. The system uses two windows of data: one that represents the current data trends, and a second which represents the most recent underlying distribution. A concept drift is indicated if the conceptual difference between the two windows is greater than some threshold. When such a drift is detected, the reference window is reset to the current one. In the second approach, the system uses the reference data distribution to predict what the distribution of incoming data should be, and uses the difference between the projected and actual distributions as the prediction error.

In both methods, the window size determines what type of concept change are detected. Small windows detect stream bursts, but may not detect slow distribution changes. A larger window detects the distribution changes, but may overlook stream bursts. This is demonstrated in Figure 2.3.

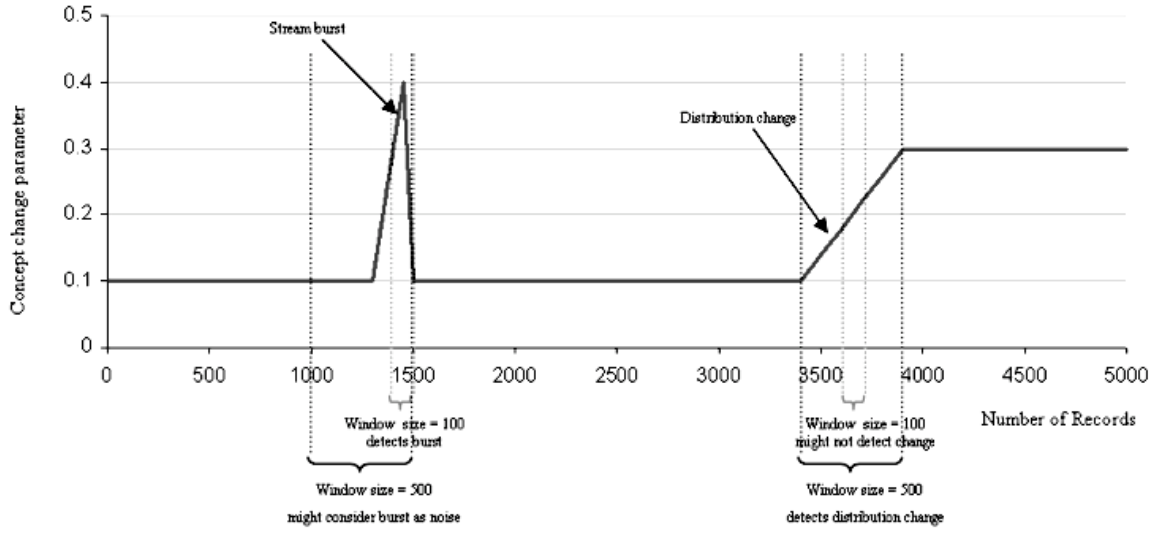


Figure 2.3: Effect of window size in detecting stream bursts and distribution change [2]

Window size also can effect the time delay of finding concept drift, as is demonstrated in Figure 2.4. A window of size N may view up to N records before detecting that a change has occurred, depending on the abruptness of the change.

Our proposed framework makes use of these concept drift detection techniques in order to adapt to changes in user behaviour. The implementation of the framework will be discussed further in Chapters 3 and 4.

2.6 Real-Time Analytic Frameworks

In this section, we discuss frameworks that have been proposed, or implemented, to address real-time data mining in web-service environments, sharing a similar goal with the predictive analytic framework proposed in this thesis.

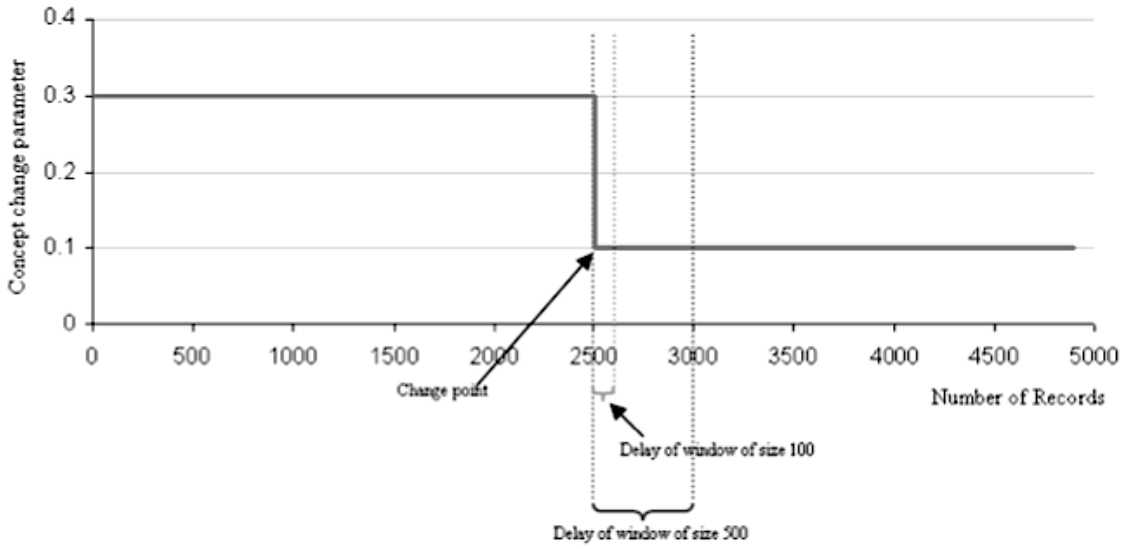


Figure 2.4: Effect of window size on time delays in concept drift detection [2]

2.6.1 CRM Analytic Framework

Bigus et al [11] present a CRM Analytics Framework that provides an end-to-end solution for developing and deploying pre-packaged predictive modelling business solutions, recognizing the fact that useful data may be spread out across data sources. The framework consists of a system that integrates the data based on meta-data, and allows data mining analysis to be executed without manual tuning while utilizing multiple data mining resources. This is done by assessing the relevance of available data for the problem at hand, identifying a set of data fields for use in the mining schema and building ETL scripts for transforming the data to the mining schema. This integration process is shown in Figure 2.5.

It is often the case that a data mining engine is specialized with one or more specific purposes in mind, and being strong at one type of task, they might be inadequate at

another. By creating a standardized data warehouse, Bigus et al create a framework that allows communication with multiple data mining engines, and therefore benefits from each engine's expertise.

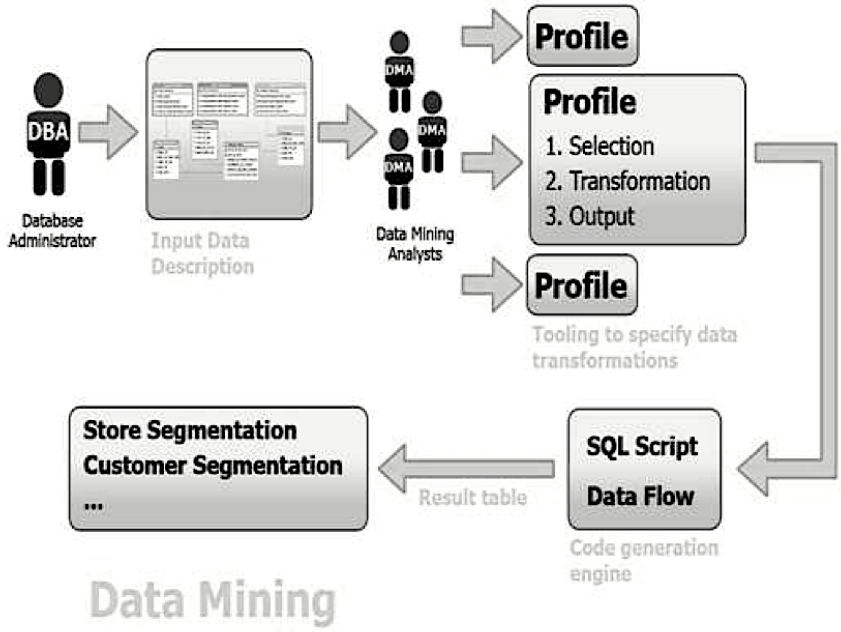


Figure 2.5: Aggregation and data transformation of CRM Analytic Framework [11]

In Chapter 3 we propose an end-to-end predictive analytic framework (PASIF) using an approach similar to the CRM framework's for extracting and aggregating data from heterogeneous sources. We alter the data mining subsystem in order to provide real-time responses to user interactions, concept drift detection and adaptive modelling, creating a user-centric service, which thus fits into the Smart Interaction paradigm.

2.6.2 5-Essentials Framework

The 5-Essentials (5E) Framework proposed by Wong et al [54] is suggested to perform real-time data mining tasks over the Internet. This framework focuses on integrating the following "essential" properties:

- Object-based parallelism
- Selection of a suitable inter-object interaction pattern
- Cut communication overhead by applying a correct programming model (sequential, logically non-distributed, and logically distributed)
- Mobility of program objects
- Detemine which hardware architecture (physically distributed or non-distributed) best suited for timeliness

The 5E project focused on two main objectives. The first was to find the correct pattern for inter-object communications among collaborating mobile data-mining objects. Second, since many parallelization methods exist, those which work best for both passive and active object-based mining needed to be identified. The primary goal of the framework is distributed data-mining over the Internet.

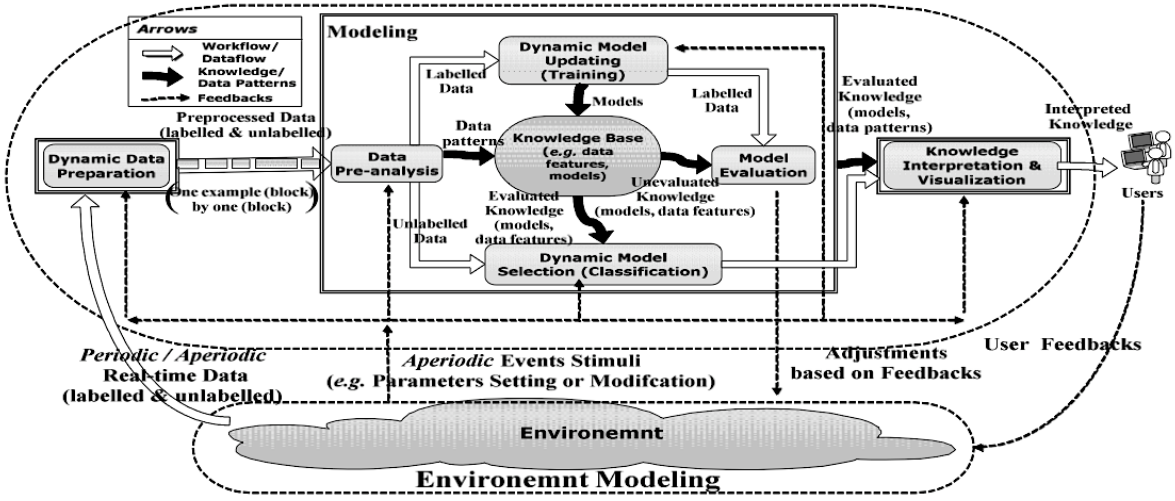
Similar to the 5E framework, we are looking to create a framework which will perform real-time data-mining. Many tools exist for data-mining with efficient, reliable and well-studied algorithms. We choose to make use of these mining engines and attempt to fit them into a real-time system, rather than the less-studied distributed algorithms. Thus, while the 5E framework shares a common goal as the PASIF framework, the approaches used are not useful to our system.

2.6.3 Environment Modelling Framework

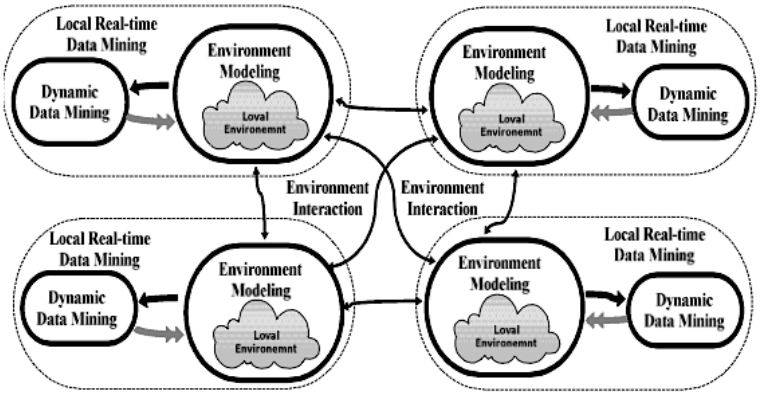
Deng et al [17] propose a framework based on environmental modelling. The goals of the presented framework are similar to our own: enable real-time modelling, support real-time data collection, support continuous feedback, and the detection of concept drift.

Deng et al break modelling into two levels: global and local. The local environment, as shown in Figure 2.6(a), is used for modelling centralized environment elements. The global environment, shown in Figure 2.6(b), is used to support distributed data mining.

We use some similar approaches as the local environmental modelling of Deng et al, such as the sliding window approach for concept drift detection, periodic and aperiodic dynamic data mining, and redundant synchronous model selection and updating.



(a) Supporting framework for local real-time data mining



(b) Supporting framework for global real-time data mining

Figure 2.6: Environment-based real-time data mining framework [17]

Chapter 3

The PASIF Framework

3.1 Incorporating Predictive Analytics into Smart Interactions

Smart Interaction systems must be able to quickly and effectively respond to a user. This involves being able to identify how to best serve a particular user, as well as predict their intended actions, within a reasonable time frame.

A framework which will provide cognitive support for Smart Interactions needs to provide real-time support to users by utilizing data which is being collected during the current interaction, as well as promptly updating predictive models with any new information gathered during these interactions.

In addition, the framework should be able to manage data being collected from several heterogeneous data sources, which may be on different platforms or computers. In order to accomplish this, the system must gather large amounts of data from these sources, and aggregate this data effectively. This includes determining which data

might be beneficial to the system, and deleting that which is not. Extraneous data will make the system slow, will deplete available storage resources, and might lead to inadvertently fitting the data to irrelevant trends. Additionally, data should be removed from the database as it becomes outdated and is determined to no longer be useful to the system.

The data which is kept in the system must then be organized in a way that allows it to be accessed and modified efficiently, thus allowing interactions to be conducted in real-time. With this data, a cognitive support system can be built which attempts to model and predict user behaviour, as well as monitor distributions in the data in order to detect and adapt to changes. The cognitive support for the framework is done through a predictive analytic service.

The main aims of the predictive service are to:

1. Provide online support to users through predictive models, which are built offline and deployed to the system.

2. Gather data about, and respond to, users with personalized interactions in real-time.

3. Maintain a set of models that are weighted based on their performance, which is determined through user feedback.

4. Keep predictive models current by detecting and compensating for concept drift in the underlying system.

In this chapter we describe the proposed framework and how it supports Smart Interactions. Following this, we discuss how the domain of e-commerce can be improved

using Smart technologies, while demonstrating how the framework can effectively be applied in a real application.

3.2 The PASIF Framework

Predictive analytics should provide a good cognitive support tool for Smart Interactions. Figure 3.1 shows an overview of the proposed framework for integrating Smart Interactions and predictive analytics, which we henceforth refer to as PASIF: **P**redictive **A**nalytics in **S**mart **I**nteractions **F**ramework. The PASIF method is similar to that of Bigus et al. [11], which proposes a CRM Analytics end-to-end framework for developing and deploying pre-packaged predictive modelling solutions, as was discussed in Subsection 2.6.1.

The PASIF framework is intended to support real-time interactions and the use of real-time data collected during an interaction to set the context for each session. A set of prediction models are built and updated offline, and used to augment the online interactions.

The implementation of this method consists of the following steps:

1. **Define data warehouse schema.** The heterogeneous data sources which are relevant to the decision-making task are selected and from each source, the attributes that may be useful in building prediction models are identified, and those that are not are omitted. An appropriate database schema is defined to store the necessary data.

2. **Develop data warehouse Extract-Transform-Load (ETL) routines.** Any relevant data from the data sources is retrieved, transformed and loaded into the

data warehouse. These routines can be run periodically to keep the warehouse relatively consistent with data sources.

3. **Define mining database schema.** One or more predictive modelling approaches can be tied to each table in the data warehouse. An appropriate mining schema contains both extracted and predicted attributes, so reporting can be performed on the results of the prediction models.

   The mining database differs from the data warehouse in the way that it organizes the data. It is made with the intention of having information pre-arranged in a manner that is easy to access for a given model. This may mean redundant information exists throughout the database.

4. **Design mining database ETL routines.** Relevant data is retrieved from the data warehouse, transformed and loaded into the mining database. Similar to the data warehouse ETL routines, these can be run periodically to keep data from becoming outdated.

5. **Build prediction models.** One or more prediction models are developed from the data in the mining database in order to form the support required by the system interaction. For a given interaction, input data is collected and fed into the models to produce output that helps in decision-making tasks, and provides cognitive support to the interaction.

6. **Validate and update prediction models.** A closed loop analysis is performed to validate the predictive models by measuring the success of the output they produce. If the degree of success is not sufficient, then the models are updated.

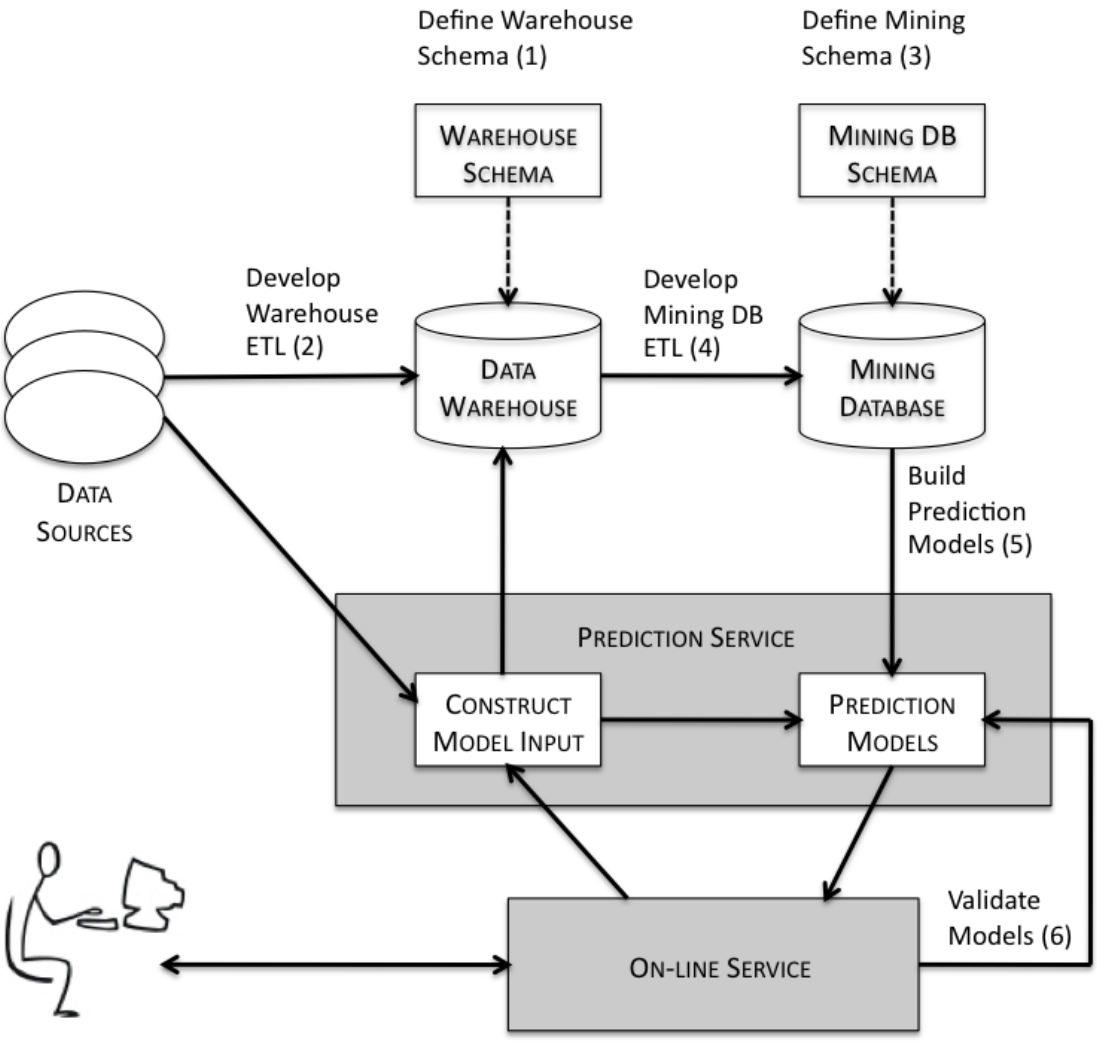


Figure 3.1: PASIF Framework: Incorporating predictive analytics into Smart Interactions

3.3 Real-Time Cognitive Support

One of the main objectives of the PASIF framework is to provide cognitive support to interactions in real-time. Classical data mining is designed to address offline data analytics. While the currently available techniques are reliable and produce accurate analysis of data sets, they typically do not support all the requirements for real-time analytics, such as meeting specific time restraints and adapting to continuous non-stationary data [17]. The PASIF framework uses an adaptation of classical data mining in order to address these issues, while still benefitting from the reliability of the algorithms and existing data mining engines.

Further, the framework provides data extraction and organization capabilities, and a prediction service scheme which facilitates the use of classical data mining algorithms, making them a practical tool for real-time applications. The remainder of this section looks into these, and how they fit into the real-time aspect of the framework.

3.3.1 The Mining Database

One way that the real-time objective is accomplished is by ensuring the database that the prediction service accesses contains only pertinent data, and by keeping that data organized in such a way that it is easy to retrieve and analyze. By keeping a mining database that contains tables which are relevant to specific predictive models and data types, building and updating predictive models is fast and simple.

When data is collected about an interaction, it is added to the data warehouse through a series of Extract-Transform-Load (ETL) routines. During this routine, the relevant data is selected and transformed into standard data types, allowing

heterogeneous types to later be combined. Next, ETL routines move data from the data warehouse to the mining database, organizing the data into pre-defined tables, representing either one type of model or data.

As new data arrives and is added to the system, the prediction service analyses trends in the data to determine if any data records have become outdated. Such data records can be removed from the mining database in order to make room for new data, and to keep models current.

3.3.2 The Prediction Service Subsystem

As new information about the user is gathered by the service, it is immediately processed by the prediction service. Prediction models are built offline using information stored in the mining database and, once built, they are deployed to be used online. The prediction service parses the important information from data collected during an online interaction and inputs this data into the prediction models in order to obtain predictions and decisions relevant to the specific user. The prediction service makes decisions and returns almost immediate responses in order to provide real-time interactions with the user.

Based on feedback from the user or online service, the model's response is validated. Here, a model might be determined to be outdated, or it may be determined that concept drift has occurred, and the predictive service is prompted to adapt. Concept drift detection for the system is described in more detail in Subsection 4.3.5. Adaptation of the predictive service consists of rebuilding or altering the models, or a subset of them, in order to produce better predictions in the future. This is done concurrently in the background, leaving the system functional while the update occurs,

albeit slightly outdated for a short time.

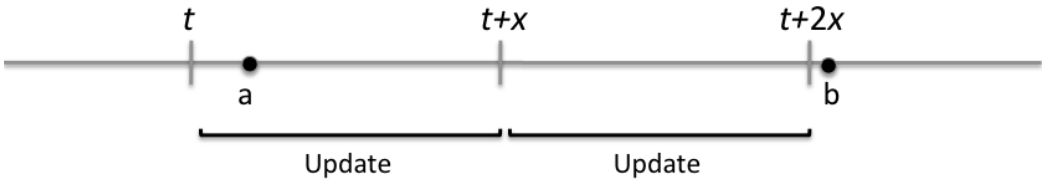
The remainder of this subsection describes the conceptual components of the prediction service subsystem. Further implementation details are provided in Chapter 4.

Redundant Modelling

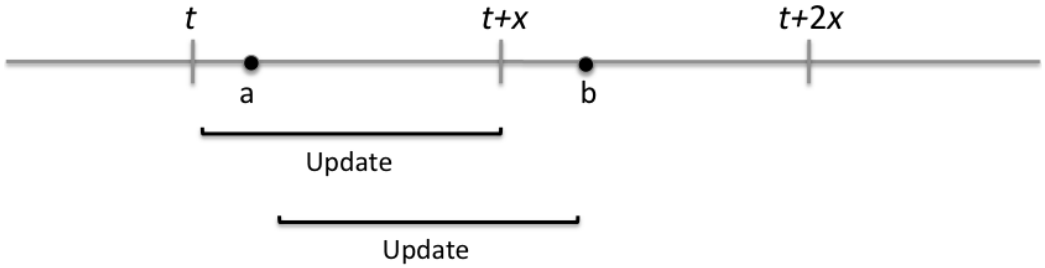
When it is determined that an existing model is outdated and needs to be rebuilt, a new version of the model, that incorporates data that has been collected since its last build, is built offline to replace it. The current model is kept in the system and used until the new model is ready and takes its place in the system. However, if only a single offline model is allowed to be built at a time, any data that is collected during the rebuild is not added to the model which is being built offline, and thus has no effect on predictions until after the model has been rebuilt a second time. Redundant model updates are employed to reduce the information lag of the predictive models.

By allowing redundant models to exist, and staggering the rebuild process, the system is updated more often and is thus kept more current. Consider, for example, a system without this redundancy where each update takes x time, and an update begins at time t. As the system approaches $t+x$, the system still has not incorporated any new information (I) gathered since the last update at t. At this point, the system might try to update again, but the information, I, is not seen in the system until time $t + 2x$, meaning it is possible to have an information lag of twice the build time of the model. That is, a piece of information arriving to the system just prior to an update, at time $t - \delta$, is incorporated in just $x + \delta$ time since its arrival, but one arriving just after, at time $t + \delta$ takes almost $2x - \delta$ time. This is demonstrated in

Figure 3.2 (a), where point a shows where a piece of data is received by the system, and point b shows where the data is incorporated into the models.



(a) Sequential updating



(b) Staggered parallel updating

Figure 3.2: Effect of redundant updates on information lag

If we instead use redundant models, and a staggered update had started at $t+\delta+\tau$, where τ is some small interval, then that piece of information takes only $x+\tau$ time to be included, as shown in Figure 3.2 (b). Given enough redundant model updates, the expected lag is much diminished from the simple approach of using a single sequential update.

It must be considered, however, that each model update requires computation and memory space. Thus, this performance is limited by the available resources to the

system. An upper limit of redundant model updates should be set relatively low in order to preserve resources. For example, in our implementation we set a cap between three and eight per model, depending on the application.

By using redundant modelling, new information is incorporated into the predictive system faster, allowing refinements and re-designs of models to be accomplished closer to real-time.

3.4 Concept Drift

For Smart Interactions, the system should be able to react to user behaviour. Since users are unlikely to consistently behave in one way, it is important for the PASIF framework to detect and react to changes in the data it is receiving. In the proposed framework, it is possible to use either the sliding window method of concept drift detection, discussed in Chapter 2, or the distribution prediction method.

In the sliding window approach, the system uses two windows of data: one that represents the current data trends, and a second which represents the most recent underlying distribution. A concept drift is indicated if the conceptual difference between the two windows is greater than some threshold. When such a drift is detected, the reference window is reset to the current one. We choose to implement this method over the predicted distribution method, since it is simpler to update the windows than it is to calculate the predictions, while producing a similar result.

In the PASIF framework, this can be accomplished by having the modeller calculate the distributions of each attribute in what is considered the current conceptual window. As data comes in, it is added to a queue, whose distribution is compared to

that of the representative model distributions. When drift is detected, this distribution becomes the new representative set, and the model is told to adjust accordingly. This might mean rebuilding or refining certain models which are affected by this change. The implementation of concept drift detection is described in Subsection 4.3.5.

In the distribution prediction method, the system references a window which represents the current distribution of the data. It attempts to predict the distribution of incoming data. Drift is detected when the incoming data does not match the prediction. This is incorporated into the framework in a similar manner as the sliding window approach, only deviating slightly in how the error is calculated.

3.5 Smart Interactions in E-Commerce

Electronic commerce offers a good demonstration of how the PASIF framework might be used in practice. Smart technologies can be applied to an e-commerce system in order to create a more individualized environment for customers shopping online. In this section we briefly consider how Smart technologies can be used to improve e-commerce environments, and how the framework provides the necessary support to do so.

3.5.1 Real-Time Personalization

A major goal in Smart Interactions is to personalize the system's response to a user as much as possible. In Smart e-commerce this means being able to quickly identify what is important to a current customer based on the way they are currently interacting

with the website, and how they have done so in the past. The framework accomplishes this using real-time data processing, as discussed in Section 3.3.

It is often the case that the distribution of items that users are buying varies over time. In the long-term, this can mean products become obsolete, or new users are drawn to the store. Short-term changes may include a new fad item, which is popular for a short time, but the enthusiasm for it subsides after some time. Stores such as sporting goods or clothing stores may also see seasonal changes in the products for which the customers are looking.

Concept drift detection and adaptation, supported by the framework, allow these changes to be accommodated, rather than smoothing all distributions over time. By using a multi-model approach, models whose distributions have changed can be temporarily omitted in predictions and later reinserted once it is determined that the shift was temporary, or they might by dismissed entirely if the distribution never returns to its previous state.

3.5.2 Heterogeneous Data Sources in E-Commerce

In e-commerce environments there are many types of data that can be collected which describe the interactions happening on the website. Often, a customer is required to provide some profile information when they sign up to use a web service. This makes up most of the demographic and personal data, including attributes such as age, gender, address, and personal preferences.

Interaction, or behavioural, data is that which is collected about how a particular customer interacts with the website, and how they have done so in the past. This includes how long a customer stayed on a specific page, click-stream data (what

elements they have hovered the mouse over and which links they have clicked, for example), search terms, and specific events such as when the user has immediately returned to the previous page after viewing a product.

Transactional data consists of all data about product purchases and returns. Typically for a given purchase, information is kept about who bought it, the quantity bought and which other products were bought simultaneously. Additionally, product data about each item is available, such as price, quantity available, quantity sold, its attributes and a set of product categories under which it falls.

With the PASIF framework, recommendations about what products are attractive to a user can be made using data that is collected about the current user. This generates an environment where the consumer spends less time searching, as the products they are looking to buy are automatically suggested to them. The system is able to adapt in order to serve a specific customer individually by immediately deriving patterns from their current behaviour as well as using the data collected as input to models that have been built offline based on this and previous customers.

3.5.3 Company Mergers and Collaborations

Consider a subsidiary, or parent company, which controls multiple company websites, including at least some e-commerce services. It is possible that many users of one of the websites behave in a similar manner as those of another, or that users or products might overlap between the company's Web services. It makes sense, then, that the company might attempt to use models produced from data collected from one website to predict user behaviour of another. Similarly, smaller companies may decide to combine their data with another's in order to improve the capability of their

data mining systems, which might be particularly useful if they have not yet collected much of their own data.

The framework provides a good platform for allowing discrete data sources like these to be used. Individual sets of models can be built for each, as well as models which use the combined data. The ETL routines allow the data to be transformed into compatible tables and types, creating the possibility of aggregate models.

Additionally, if two companies merge, it is unlikely that their data will be compatible. Eventually, they might have enough post-merger data that they can build dependable models without using data collected prior to the merger, but, for at least the early stages of the merger, predictions can be improved by using the integrated data sources.

Again, the ETL routines allow the data to be compatible so that new models can be built comprising of both types using post-merger data. The predictive models can handle multiple model types and can adapt as the system matures. Further, the prediction service is flexible enough to allow new mergers to occur and be incorporated throughout the lifespan of the predictive service.

Chapter 4

The Prediction Service

In this chapter, we describe the design and proof-of-concept implementation of the real-time predictive analytics service for the PASIF framework. An overview of the basic structure is described in Section 4.1. The subsequent sections look into the components of the predictive service in more detail. Throughout the chapter we return to the e-commerce product recommendation example to clarify the ideas presented. Recall that for product recommendation, information is collected about user interactions in order to form a personalized suggestion of a set of products a specific user might like to purchase.

4.1 Structure

This section outlines the main components of the Predictive Analytic decision-making support service. The basic organization is shown in Figure 4.1. The components of the service are the following:

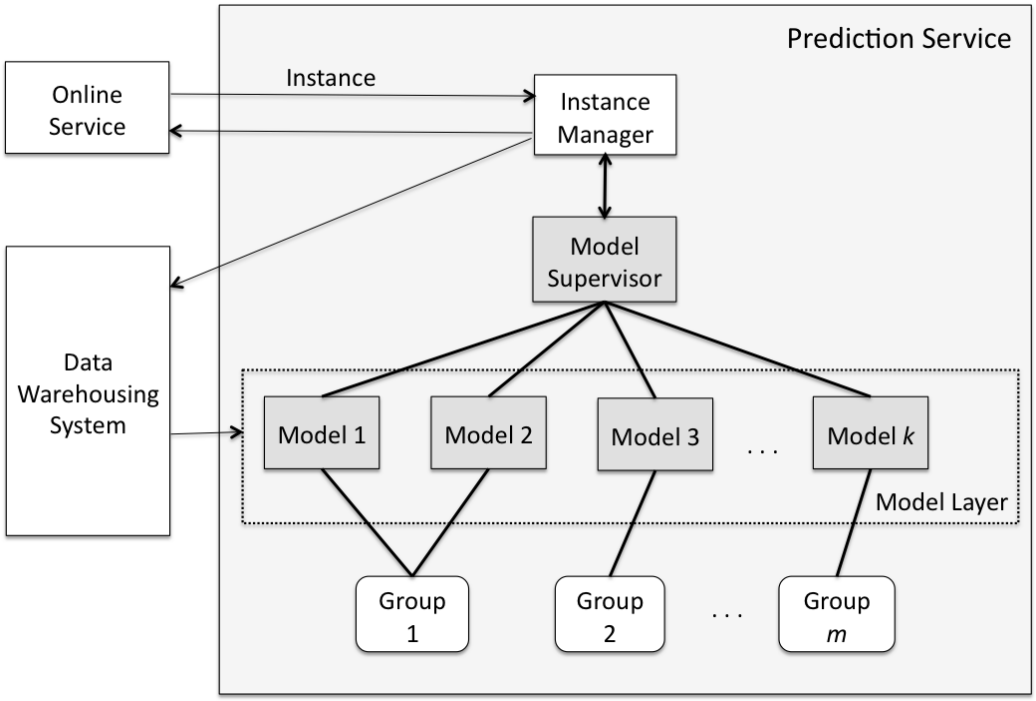


Figure 4.1: The basic organization of the prediction service

Instance An instance is a set of attributes describing a specific interaction. For example, in the e-commerce scenario, an instance might describe a specific customer visiting the website, and the set of items they have in their shopping cart. These are discussed in Section 4.2.

Model Each model is tied to a single group and can be created using one of many possible data mining or predictive analytic algorithms. Given an instance, a model creates a prediction and relays it back to the model supervisor. Upon receiving feedback, the model can alter the level of influence it has on future predictions, by adjusting its weight. The model build and deployment process is discussed further in Section 4.3.

Model Supervisor The model supervisor deploys models and builds new models when concept drift is detected. The model supervisor combines the individual model predictions, as well as relaying feedback to the models. It also controls the deletion of models if either a model is determined to be too poorly fitted to the data, or if inferior models need to be removed in order to make room for new models that might fit better. The model supervisor's role is discussed further in Section 4.3.

Instance Manager The instance manager handles all instances that have been collected by the system. When a new model is initialized, the model supervisor sends a request to this manager for a specific set of instances from which the model will be built. In the full end-to-end system, the instance manager communicates with the mining database.

Group A group is a family of models of the same type. Possible groups for the e-commerce example might include models built using customer, transactional, or interactive and click-stream data.

Simulator Specific to our implementation of the system, the simulator creates a stream of data from a sample data warehouse and online service in order to simulate new instances being sent to the predictor, as well as sending feedback to the system after each prediction. The simulator is explained further in Section 4.4.1, as well as how it fits into the system.

4.2 Online Data Collection

In the end-to-end framework, data is collected from the user in entities called *instances*. An instance is a set of attributes describing a specific interaction between a user and a service. In our example, the service is an e-commerce website for online shopping.

In e-commerce environments, data can be collected during a session and stored in a database, with a session identifier that identifies specific customers. The predictive analytics framework connects to the database and accesses the information relevant to the current session, as information is being gathered. The online service accesses this information using an instance, which contains all data that is available about the current state of the user session, selected from the database. This part of the framework is not implemented, but rather simulated as described in Section 4.4.1.

An instance may contain information from multiple sources. As shown in the example in Figure 4.2, an instance might contain several rows selected about items in a customer's online shopping cart, and data describing their page views.

Figure 4.2: An example instance object

The instance is passed to the instance manager, which transforms the data into

the required format for the prediction modelling subsystem. This is discussed further in Subsection 4.3.3.

4.3 The Modelling Subsystem

4.3.1 Offline Modelling

Models are built offline and then made available in the online service. All models are created in the background, using classical data-mining strategies, rather than using streaming or evolutionary data mining strategies, which update models rather than rebuilding them. We choose to use classical non-distributed data mining approaches, rather than streaming or distributed, as stream and distributed mining approaches only provide approximate results, and classical approaches provide better accuracy [20]. Furthermore, by using this approach, we are able to prune data often, saving disk space and keeping models current. Also, tools for classical mining are readily available.

In our implementation, all models are built using Weka [22, 23], an open source implementation of common machine learning algorithms, as described in Appendix D. The PASIF framework, however, is not limited to the use of a single data mining engine, and the implementation can easily be adapted to simultaneously access various data mining tools. We connect to Weka through Java, and provide it a set of instances, which are converted to the Weka instance format using Java libraries provided by Weka. A model of the data can then be built from the Weka instances. For our testing purposes, we create models of the following Weka model types: bayes network, averaged one-dependence estimator (AODE), k-nearest neighbours, and multi-channel

classification. Given a set of unlabelled attributes, a Weka model can produce a label or prediction, which the Java program can then process and use to respond to the user. For example, given a set of demographic data about a customer who is not yet associated with group of similar customers, or cluster, Weka can assign that customer to a group.

We track the performance of the models and determine when it is necessary to rebuild them, either because the data distribution has changed, or simply because there has been a lot of new information added to the system which should be included in the models in order to attempt to improve the system's performance. When a model is finished being built and tested, it is deployed to the system, and responds to users in an online fashion, as discussed in Subsection 4.3.3.

4.3.2 Ensemble-Based Predictive Analytics

We use a weighted ensemble-based modelling system. Recall from Section 2.3.6 that classifier ensembles can be used to combine multiple data types, or subsets of data to create an adaptive classification system. In weighted ensembles, each model is given a weight that is updated based on the model's performance, in order to give strong models more influence and weaker models less.

We adapt this method to aggregate multiple types of predictive analytic models, rather than exclusively using classifiers. Ensemble-based modelling provides a good platform for asynchronous model updates. That is, models can be added or deleted on the fly. When a new model is added, it is included in the model set with a weight initially set to the average of all current weights in the system. A simple weight adjustment routine is executed which redistributes the weights such that their sum is

```
function MODELLER-ADAPT (feedback, instance)
  inputs:
        feedback, true if positive feedback, else false.
        instance, the instance associated with this feedback

  local variables:
        models, list of predictive models
        groups, list of model groups
        prediction, map from recent instances to prediction made for it


  for: model m: models
    if m.prediction(instance) = this.prediction(instance)
      m.adjust(feedback) // alters weight and expiry as necessary
    else
      m.adjust(!feedback)
    if m is expired
      delete m from models
  if concept drift is detected
    for: group g: groups
      if at maximum number of models
        delete next model to expire
      localmax = index of local max by simulated annealing
      newmodel = new model of type g, using localmax as starting index
      models ← newmodel
  scale model weights to sum to 1
```

Figure 4.3: Model adaptation in the predictive analytic service

one. When a model is deleted, it is removed from the set, and the remaining weights are adjusted.

Further, models are created using varying subsets of the data. The data that comprises these subsets is decided based on where concept drift has recently been detected in the underlying system. This idea extends easily to the sub-data sets used in Ensemble-based modelling [41, 43]. The pseudocode for adapting the set of models is provided in Figure 4.3.

4.3.3 Managing and Reacting to Incoming Instances

When a new set of data attributes, or session instance, is introduced to the prediction service, it is first sent from the online service to the instance manager, as described in Section 4.2. The instance data is relayed to the data warehousing system, which will use extract-transform-load routines to retain the important information, and update the mining database. Simultaneously, the instance manager parses and analyzes the data in the instance in order to quickly respond and react to the user about whom the data is being collected. A reaction might consist of a product recommendation for the user, or layout changes to the website to make browsing more personalized to the user's tendencies, for example.

The parsing includes extracting the information from the instances and writing them into single lines of data, and creating Weka instance objects that are input to models of specific groups in order to create a prediction or classification, as shown in Figure 4.4. For example, given a set of items in a customer's shopping cart, a single line of input for an item set modeller might be a list of integers representing the number of each possible item they have, or a shorter version for sparse data which lists each item in the shopping cart along with their quantity, and assumes all unlisted items to have a quantity set to zero. A second parsed instance could be a set of demographic attributes derived from the user ID using the mining database.

The parsed data is sent to the model supervisor in the form of Weka instances, which then relays the appropriate fragments of relevant data to each model in the set of active models and waits for a response. Each model creates a prediction for the data it has received and responds to the model supervisor with its independent decision (a prediction or classification). The supervisor aggregates the model predictions and

responds to the online service with a final prediction of how the user will behave. In the e-commerce scenario, this could be a prediction of what items the user may want to buy but they may not have seen yet.

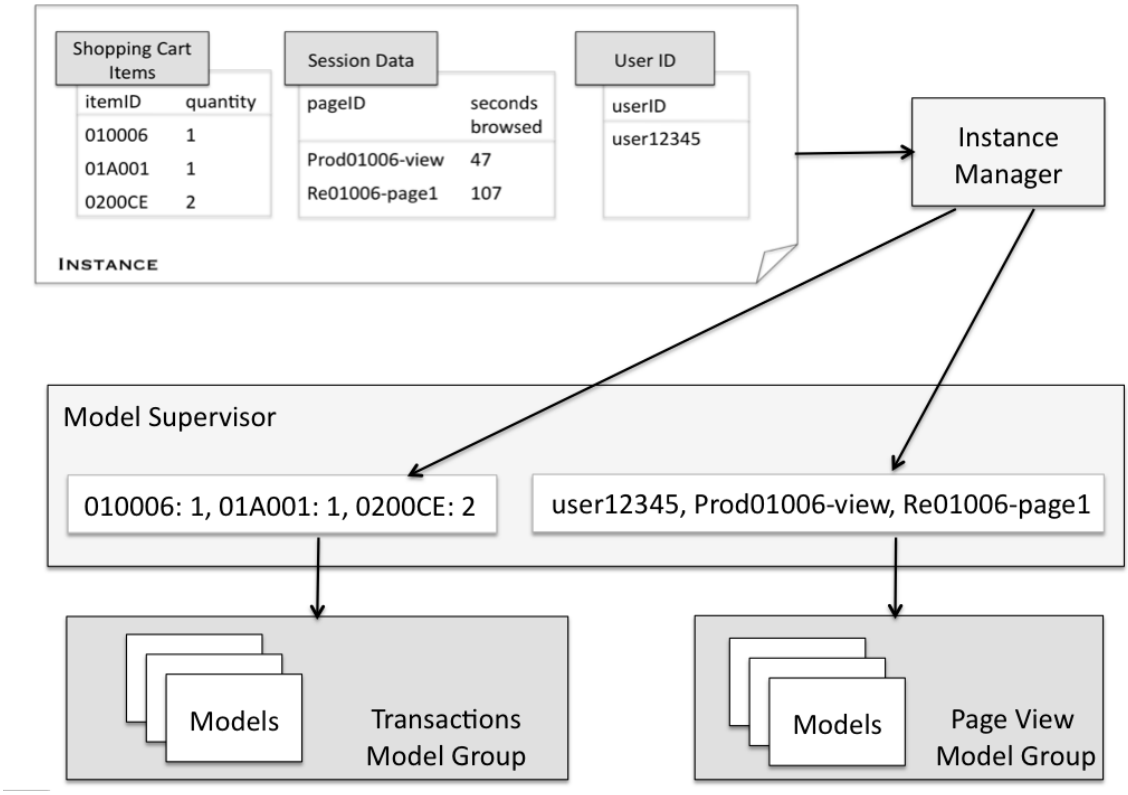


Figure 4.4: Example of instance information parsing by the instance supervisor

4.3.4 Model Validation

During interaction with a user, feedback might be sent back to the model supervisor. The model supervisor then passes along the feedback to the models, as shown in Figure 4.3 and each model adapts itself according to the feedback as shown in Figure 4.5. This may be obtained directly, such as when a user provides a ranking of how

```
function SEND FEEDBACK (feedback)
  inputs:
        feedback, true if positive feedback, else false.

  local variables:
        weight, model's weight
        delta_pos, delta_neg, weight adjustment constants
        expiry the number of iterations left until model expires


  if feedback = true
     weight += delta_pos
  else
     weight -= delta_neg
     decrement expiry
```

Figure 4.5: Model weighting based on feedback

well they think the system has met their needs. Or, it may be derived indirectly based on observations about a user's reactions. For example, in the e-commerce scenario, a customer may be presented with a product determined to be a good match for them by the system. If the user looks further into this product, the system will receive positive feedback, or if the customer ignores the suggestion, the system receives negative feedback.

When positive feedback is received, the model supervisor analyzes which models contributed to the successful interaction, and rewards those by increasing their weight. Those that did not contribute, or negatively influenced the decision, are penalized with a decrease in weight. After the model weights have been adjusted, they are scaled in order to again have all model weights sum to one.

Alternatively, when negative feedback is received by the supervisor, those models which contributed to the failed interaction are penalized, while those that provided a

suggestion which was not accepted as the final decision, keep their previous weight. Again, model weights are then scaled to sum to one.

When a model's weight decreases past a lower threshold, it is deleted from the model set. These weak models would otherwise slow the system's calculations and waste computer storage and memory, while not making significant contribution to the predictive service.

4.3.5 Concept Drift Detection

As feedback is collected, a sliding window approach is used to detect concept drift. Rather than looking for drift in the distributions of data, collected from multiple heterogeneous sources, we use the performance of models and groups of models to determine drift in the underlying system.

Recall the sliding window methods for concept drift detection discussed in Section 2.5. In this approach, two windows of data are considered by the system: a reference window which describes how the system is expected to behave, and a second window describing how the window is currently behaving [48].

A common concept drift parameter is the Shannon's entropy of a specific data attribute. Shannon's entropy [45] is a measure of the randomness associated with a random variable. It is defined as:

$$H(x) = -\sum_{i} p_i \ log_2(p_i),$$

where x is a discrete random variable, and p_i is the occurrence probability of x_i [2]. To calculate the entropy for a data set with a known distribution, the following equation is used:

$$H(x) = -\sum_{x} P(x) \ log_2(P(x)),$$

where $P(x)$ is the probability mass function of x [2]. Since the PASIF framework is incorporating multiple heterogeneous data sources, which may not share random variables, it is necessary to use a different strategy when choosing a detection parameter. Further, Shannon's Entropy is intended for use in streaming algorithms, while we have the additional benefit of stored data and histories. We propose the performance of the model as a drift detection parameter, based on the efficiency of the calculation when checking for drift, as well as the ability to generalize to any data source, without needing to know which attributes are available nor their expected distributions.

We propose the following adaptation to the drift detection technique; for each group of predictive models, a tally is kept of what percent of incoming instances have received positive feedback at each feedback occurrence. To track the distribution, we use the average number of positive feedback occurrences in each window. The current window consists of the most recent feedback, and the reference window is either kept static and updated only when drift is detected, or actively kept at a constant distance from the current window.

Concept drift is determined by the difference between the windows' averages, using a threshold to determine if there is a sufficient change. The threshold can be tuned to define how sensitive the change detections should be; a larger threshold ignores smaller changes in the distribution. The threshold should be set corresponding to the specific scenario for which it is being used and what concept drift detection parameter is being used. If the system begins degrading, it indicates that a change has occurred in the underlying system, and new models should be built using data that reflects the new distribution, omitting data that the system determines to be outdated. The

pseudocode for concept drift detection in our implementation is shown in Figure 4.6.

The window size parameter can be adjusted depending on the sensitivity desired in the detection. A large window ignores burst changes and detects long-term changes, while a small window may miss gradual changes, but finds bursts and long-term changes which occur quickly, as was discussed in Section 2.5.

In order to deal with stream bursts which are large enough to be detected, but only temporarily represent the underlying system, an expiry-based approach is used in lieu of immediately removing the models from the system. Upon creation, each model is assigned an expiry value, and an expiry counter beginning at zero. When a model is determined to be outdated due to concept drift, its expiry count is incremented. Once the expiry counter reaches the maximum assigned value, the model is discarded. If the distribution returns to its previous state before the model expires, then the model is still considered to be useful and continues to make predictions for the system. If it is marked as outdated multiple times during one or more bursts, it is determined to lack the robustness required to handle minor changes to the system.

Data Selection

When a new predictive model is added to the system, it is built using a subset of the data collected by the prediction service that is compatible with that model's group. The subset is chosen based on which data points are determined to best represent the expected distribution of incoming data. Two approaches to do this were considered. In both, the subset is defined by a starting index, which describes from what time in the prediction service's history should data be used. All data from this index up until the most recently collected data constitutes the data subset.

function DETECT-CONCEPT-DRIFT (*model, performanceList, threshold, windowSize, windowInterval*)
returns *true* if drift has been detected, else *false*.
inputs:
model, a predictive model
schedule, a list of performance values; entry i corresponds to the percent of feedback up to feedback occurrence i that has received positive feedback
threshold, minimum difference between windows to flag that drift has occurred.
windowSize, number of data points to include in window
windowInterval, number of data points between current and reference windows
local variables:
currentWindow, the average performance value of the current window
referenceWindow, the average performance value of the reference window
curr, the current index in the performanceList

curr ← *performanceList.size* - 1
if *curr* < (*windowInterval* + *windowSize*) **then return** false
currentWindow ← *model.currentWindow*
referenceWindow ← *model.referenceWindow*

Update window averages:
if (*currentWindow* and *reference window* = *null*) **then**
currentWindow ← average (*performanceList*[(*curr* - *windowSize*) **to** *curr*]
referenceWindow ← average (*performanceList*[
(*curr* - *windowSize* - *windowInterval*) **to** *curr* - *windowInterval*]
else
currentWindow ← *(currentWindow)* +
(*performanceList*[*curr*] - *performanceList*[*curr* - *windowSize*])/*windowSize*
referenceWindow ← *(referenceWindow)* +
(*performanceList*[*curr* − *windowInterval*] -
performanceList[*curr* − *windowInterval* - *windowSize*])/*windowSize*
model.currentWindow ← *currentWindow*
model.referenceWindow ← *referenceWindow*
if | currentWindow - referenceWindow | ≥ *threshold* **then return** *true*
else return *false*

Figure 4.6: Sliding window concept drift detection implementation

In the first approach, when concept drift is detected, the system stores the data index at which the drift occurred. When a new model is built, the data subset consists of all data collected since this index. In the second approach, rather than storing indices, each time a model is built a hill climbing algorithm is executed to locate a starting point for the data subset, which will be described further in this section. A minimum size for the training data in order to create a meaningful model, m, is determined through experimental runs. If the chosen subset is too small, then the last m data points collected are used instead.

The first approach is more computationally efficient, but is lacking in several critical aspects. Most importantly, temporary bursts in the underlying system reset the data starting point each time, omitting any valuable data that has been collected up to that point. Small changes in the system are catered to, and larger trends are ignored.

The hill climbing approach uses momentum and random walk (using both forward and backward moves through the data) to find a local maxima, an adaptation of the simulated annealing algorithm [43] (see Appendix A). At each step of our approach, a synthetic temperature, T, is calculated based on the time, t, where time represents the number of algorithm iterations so far. The formula used in our implementation, based on experimental results, is:

$$T = \frac{1}{10t}$$

If this temperature is negative or zero, the system is determined to be "cooled" and the current index is selected as the starting value for the data subset. If the temperature is positive, the algorithm continues.

The algorithm begins at the most recent data point and walks through the performance data. We define performance as the percent of instances on which the system received positive feedback out of all instances processed up until that time. The next candidate data point is chosen as either the next or the previous datum, with some defined probability, P. P should favour moving uphill through the data, towards points of better performance, but allow for moves to lower performance points. The algorithm should find recent points with the highest performance. A high point of performance suggests that it is the last point at which the system was behaving well, the point after which the classifier began to have poor performance, and thus where concept drift is likely to have occurred. By allowing moves to lower valued points, it is possible to jump smaller local maximas, in order to find larger maximas which reflect larger-scale trends.

We define energy E_i as the performance value of the system up to instance i. An entropy, ΔE, is calculated as the difference between energies of the current point and that of the next.

$$\Delta E = E_{next} - E_{curr}$$

If the entropy between the two points is positive, the candidate data point is accepted as the current value, and the algorithm continues on to its next iteration. However, if the entropy is negative, it only moves with a probability proportional to the Boltzmann factor:

$$P_{move} \propto e^{-\Delta E/T}$$

In Figure 4.7, two concept drift detection points are labelled, A and B, along with a third point that shows the time at which a new model is introduced to the

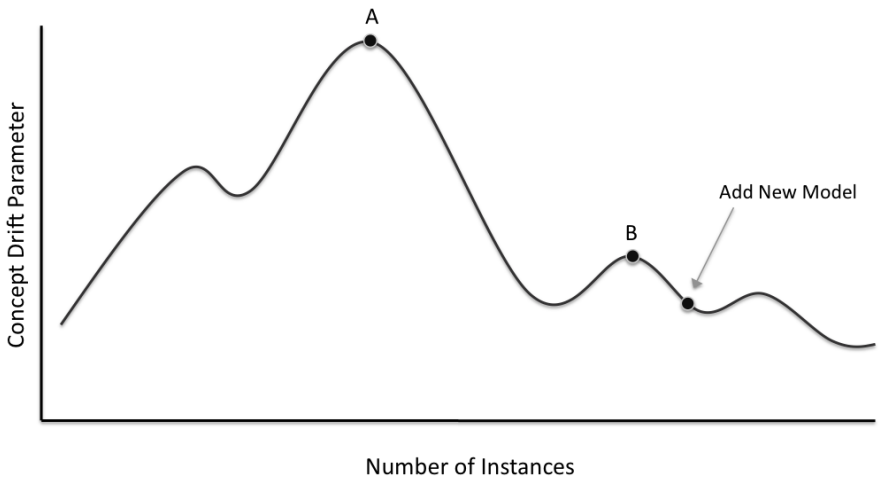


Figure 4.7: Hill climbing approach with random movement for finding data subsets

prediction service. The stored point approach will always create a data set composed of only the data collected since instance A. The hill climbing approach can find either point A or B (or others) and can be tuned to be more sensitive or biased towards recent spikes or overall distributions, depending on the requirements for the specific prediction service.

4.4 Implementation

The real-time predictive analytics service was implemented on a computer running Mac OS X v10.6, with 8 GB of DDR3 RAM and a dual core processor. The prediction service was written in Java 1.6.0, connected to Weka 3.6.3, an open source implementation of common machine learning algorithms [22], and MySQL 5.1.49, an open source relational database management system [53]. Figure 4.8 shows the Java

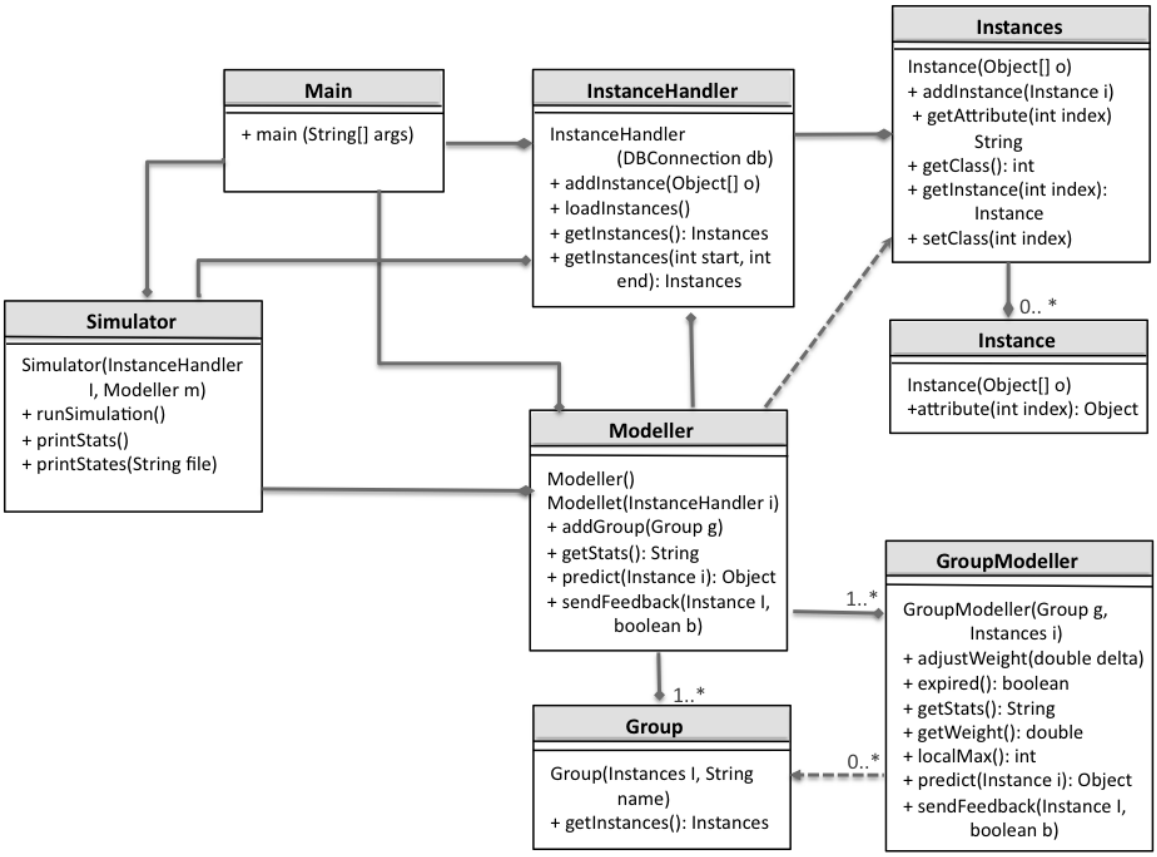


Figure 4.8: Java class diagram for predictive service implementation

class diagram for the implementation.

The major obstacles that were encountered during implementation were in efficiency and heap space, which limited the number of models that can be used at a given time. Initially the system ran with only 1 GB of DDR3 RAM, but would run out of heap space on a single model with a moderately sized data set. To overcome this, more RAM was added. The system continues to be limited in the amount of data that can be used as input to a model. Through experiments we have determined that with our set-up, a classifier-based example with 23 attributes is limited to about 45,000 data records.

4.4.1 Simulating Data Streams

As previously discussed, our implementation consists only of the prediction service for the PASIF framework. The work of Bigus et al [11] provides the groundwork needed for the data warehousing and ming database parts of the framework. Further, in order to test the performance of the prediction service, it is necessary to imitate users accessing and providing feedback to the system.

Thus, in order to effectively simulate a real-time interaction system, we must emulate the following system behaviours and structures.

1. Collection of online streaming data from users about a current interaction.

2. Gathering user feedback to improve and validate the system.

3. Functional data warehouse and mining database.

The adapted implementation is shown in Figure 4.9. Data is provided by a streaming data simulator. All instances are loaded into the simulator upon initialization of

the data, along with the expected responses. The expected results are separated from the attribute data, and stored for use in the feedback system later. A pointer to the current instance is kept which emulates how far into the data stream the program is at a given time. At each simulated interaction, the pointer is moved to the next instance, which is treated as an incoming instance from the online service. It is then processed and relayed through the modelling structures as if it had arrived from an external source.

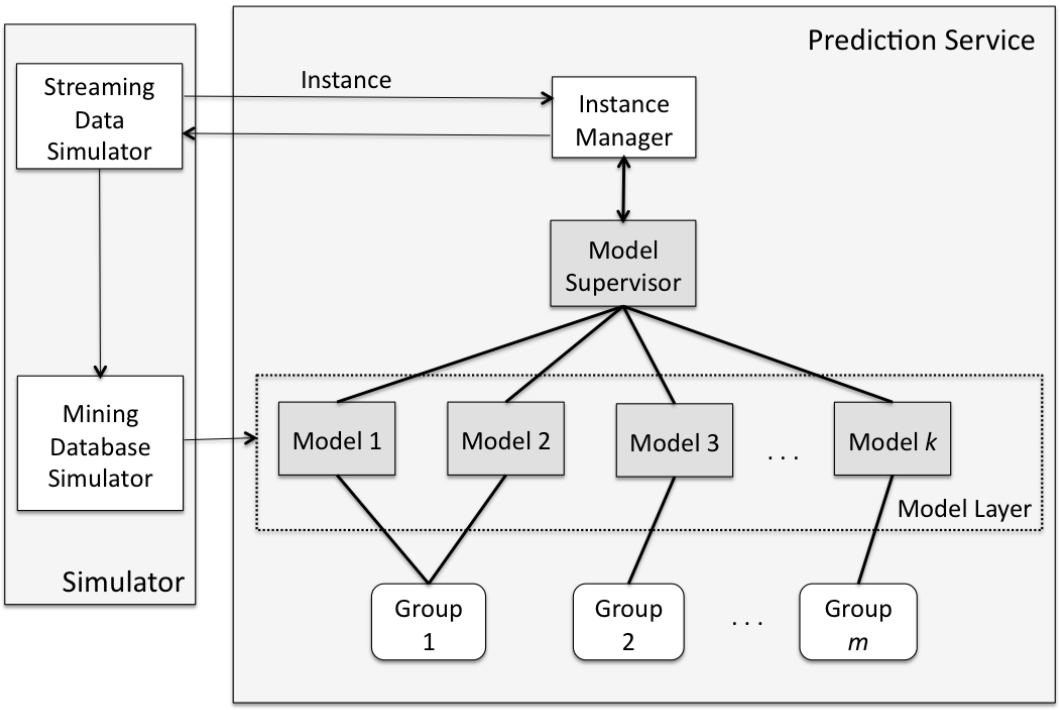


Figure 4.9: Implementation of the prediction service with simulated streaming data and data warehousing

The prediction service sends a response back to the streaming data simulator as if it were replying to the online service. The simulator compares the reply with the stored expected result, and responds with either positive feedback if the prediction

and expected result match or negative feedback if they do not. For example, the prediction service may send a product recommendation for a simulated user, and if the product is contained in the set of items the simulator expects, then it sends a message back to the instance manager conveying positive feedback, using a simple boolean value.

The data warehousing system is simulated by having a pre-defined mining database which communicates directly with the data stream simulator. When instances are used in a simulated interaction, the corresponding data is added to the mining database. The data is pre-converted to the correct form, making the data warehouse and extract-transform-load routines unnecessary for the simulation. Predictive models access the mining database as they would in an implementation of the full end-to-end framework, through the model group layer. Model groups are created upon initialization of the system in order to obtain a connection to the mining database.

Chapter 5

Evaluation

This chapter discusses the evaluation of the prediction service implementation for PASIF. Specifically, we look at the performance of a basic prediction system, the effects of ensemble and adaptive modelling on the performance, and the scalability of the prediction service to larger data sets.

The remainder of this chapter is organized as follows: Section 5.1 describes the data set that is used as input to the prediction service, Section 5.2 describes the experiments that are performed and Section 5.3 discusses the experimental results, and the feasibility of deploying the framework to real applications.

5.1 The Data Set

Evaluation of the prediction service, implemented for the PASIF framework, uses the Agaricus-Lepiota (Mushroom) data set from the UCI Machine Learning repository [18]. This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is

identified as edible, poisonous, or of unknown edibility. The latter class was combined with the poisonous one. It is known that no simple rules exist for determining the edibility of a given mushroom [18].

The data consists of 22 categorical attributes, 8124 instances, and contains missing values. For more detailed information about this data set and its attributes, see Appendix B.

Preferably, a data set directly related to e-commerce would be used to test the predictive service implementation. However, data from a private company was not available. Publicly available e-commerce data sets were found to be limited by their size or sparsity, having too few data points from which to infer meaningful trends.

The key contribution of this thesis is the generic predictive analytic framework, PASIF, which supports the real-time analysis and the evolution of models. Thus, we use the mushroom data set to simulate what is required to evaluate the proposed functionality: the use of multiple data sets, adapting to changes in the underlying data streams and the ability to use incoming data to produce real-time feedback. Multiple data sets are simulated by breaking the attributes of the data set into three subsets, and treating them as distinct data sources, and thus model groups. This is apparent in the mining database schema, as described in Appendix C.

5.2 Experimental Design

Experiments are studied for four cases: a base case, adaptive modelling, ensemble modelling and as a combined system which uses both ensembles and adaptive modelling.

Base Case

The base case experiments are conducted using a single set of the three predictive models, with no drift detection. The models use only one data set for each model group, weighted equally, consisting of all data collected so far in the system at any given point. Models are rebuilt at constant intervals of 400 instances, which was determined through experimental runs to provide low overhead, while providing noticeable improvement in forming accurate predictions.

Online Adaptive Modelling

In the adaptive modelling case, a single model for each model group, equally weighted, is kept within the system. It is rebuilt, at minimum, every 400 instances, or sooner if concept drift is detected. A sliding window of size 20 is used, with a constant distance reference window kept at a distance of 200. We choose the window size of 20 such that is is large enough to hide outlying or atypical results, such as instances that are significantly more difficult to classify than others, while keeping the window small enough that it does not smooth over drifted areas or create a significant detection lag. When models are rebuilt, the hill-climbing method described in Subsection 4.3.5 is used to decide what subset of the data should be used as the training set for the model.

Ensembles for Streaming Data

Ensemble testing is done using an upper bound of 20 models in the system, and no concept drift detection. Consistent with the experiments for the base case, models are rebuilt once for every 400 instances introduced to the database.

Introducing multiple model types in the ensemble case would introduce inconsistencies in the results compared to the two previous cases, which can both only have one model type. For example, consider choosing to use an AODE modeller for the other cases, but also including a Bayesian classifier in the ensemble. If the Bayesian classifier performs better than the AODE for this particular data set, then the ensemble will definitely have better results. To avoid this case, we use a single model type, but use different subsets of the data, chosen randomly with a minimum size constraint.

Combining Online Adaptive Modelling with Ensembles

We combine the second and third test cases to evaluate the system performance as a whole. An upper bound of 20 models is used, but the data subsets are decided using the hill-climbing method. Data models are rebuilt and replaced at least once for every 400 added instances, as well as when concept drift is detected, using a sliding window of size 20, and a constant reference window distance of 200.

5.2.1 Experiments for Performance Analysis

For each case, we look at the performance of the prediction service for the subdivided Agaricus-Lepiota data set before and after concept drift is introduced, as shown in Figures 5.1 and 5.2. For all results, we start with an initial data set of the first 800 data points to build the three initial models (one for each data group), and use the remaining 7324 data points to simulate data that is collected online. These numbers are based on experimental runs used to determine how many data points are needed to build a reasonable model, while leaving as much testing data as possible. The

overall performance, along with average run-times are shown in Table 5.2.

Each result is the average of five runs. For each of the runs, the data has been pre-shuffled at random to produce a unique arrangement. The set of runs are identical for the four cases. For the concept drift scenario, the first 4000 instances are identical to those of the non-drifted data set. The remaining 4125 instances have opposite labels, to simulate a conceptual change in the underlying data. The results for these experiments are discussed in Subsection 5.3.1.

We use the run-time of the experiments to determine the feasibility of using the prediction service and PASIF framework in real-time applications. These results are discussed in Subsection 5.3.2.

5.2.2 Experiments for Scalabilty

To analyze the scalability of the prediction service, we compare the base case to the full combined system. To produce more definitive results, we slow the combined system further by creating an upper bound of 40 models, rather than 20 as done in the previous section. Drift is introduced at intervals of 8000, rather than a single drift at the 4000 mark. The results are discussed in Subsection 5.3.2.

5.3 Results and Discussion

This section discusses the results of the experiments. Section 5.3.1 discusses the performance of each case in the drifted and non-drifted scenarios in terms of their ability to correctly predict labels for the instances, as well as the ability to react and cope with concept drift. Section 5.3.2 discusses the performance of these cases

in terms of run-time and the feasibility of each for deployment to an online service. Section 5.3.2 discusses the scalability of the framework for larger datasets.

5.3.1 Accuracy and Performance

In this section we analyze the performances of each case, including datasets with concept drift and without. We define performance as the percentage of correctly labelled instances, and the ability to deal with concept drift.

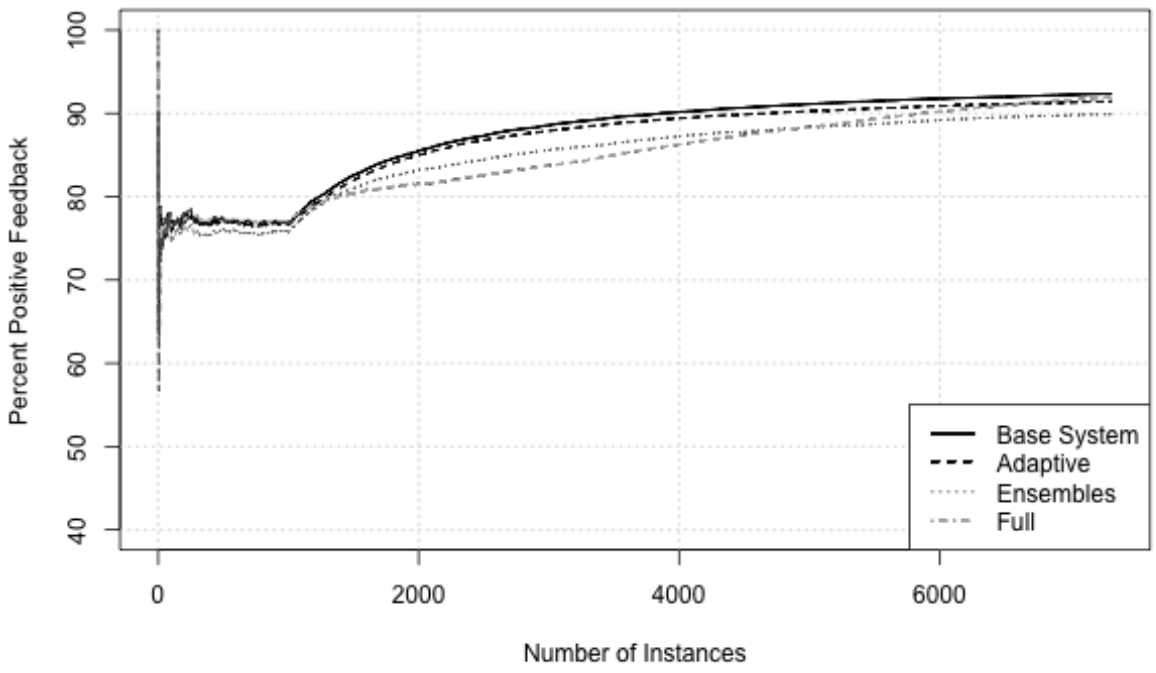


Figure 5.1: Predictive modelling on data without drift

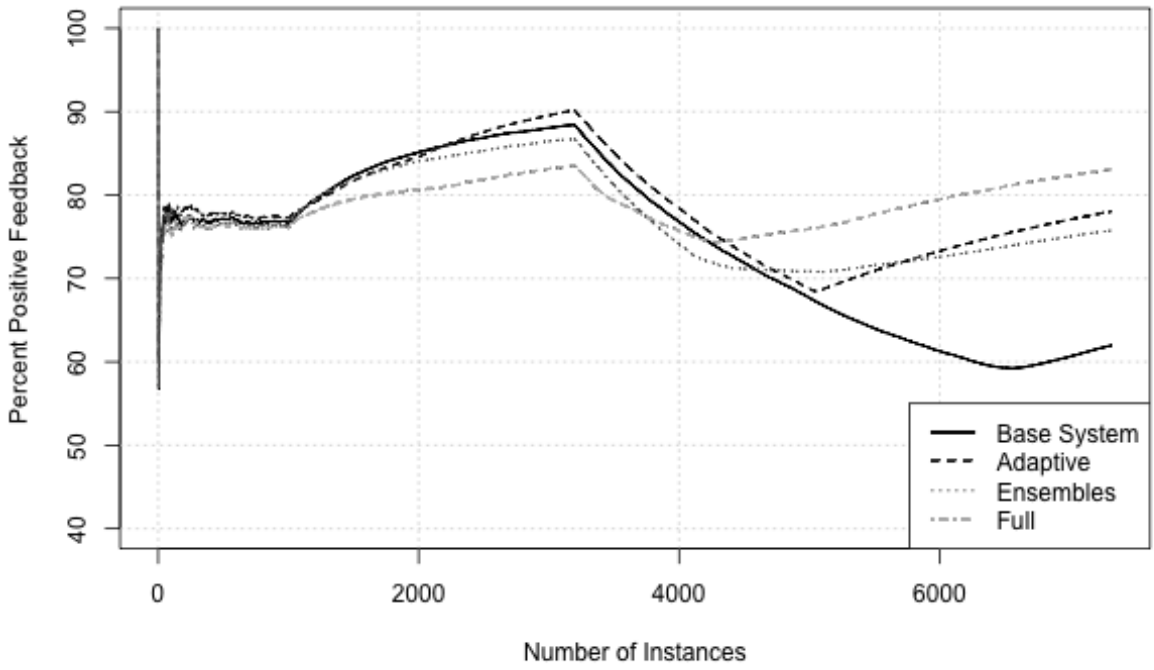


Figure 5.2: Predictive modelling on data with drift

Base Case

In the Base Case, as shown in Figure 5.1, the performance of the system continues to improve as more data is collected when there is no concept drift. However, when the system reaches the point at which concept drift is introduced, the system is unable to recover until the amount of data supporting the new underlying distribution outnumbers that which supports the initial distribution. This is shown in Figure 5.2, where concept drift occurs where performance reaches its maximum, and performance continues to degrade until the amount of data approximately doubles.

Online Adaptive Modelling

For the non-drifted scenario, online adapted modelling performs comparably to the base case. Since no drift has been detected, the adaptive method, like the base case, uses the full data set to create predictive models. Minimal overhead is added by checking for drift, since no rebuild is triggered. When concept drift does take place, the system adapts to use only data since drift has occurred, and out-performs the base and ensemble techniques.

Ensembles for Streaming Data

In the non-drifted scenario, some performance is lost due to incorporating some models without the complete set of all data collected. However, in the drifted data scenario, the aggregate model quickly adapts to changes in distribution by granting more weight to models with better performance. Since models which use more data from the new distribution have significantly better performance, they quickly rise in weight. However, without the drift detection and hill climbing capabilities of the adaptive model, the data set selection is not optimized, and the adaptive method eventually outperforms it.

Combining Online Adaptive Modelling with Ensembles

In the non-drifted scenario, the combined system performs comparable to the other cases. Some performance loss occurs for similar reasons as seen in ensemble modelling. However, as more instances are collected, the prediction service converges to the better models and has an overall performance comparable to the other cases. In the drifted scenario, this modelling case benefits from the quick reaction speed of ensemble

modelling, as well as the overall performance of the adaptive system, producing results with significantly improved accuracy.

	Non-Drifted	
	Accuracy (% correct) $\pm \sigma$	Run-Time (seconds) $\pm \sigma$
Base	92.36 $\pm$ 1.455	8.265 $\pm$ 1.04
Adaptive	91.45 $\pm$ 1.389	10.157 $\pm$ 0.84
Ensembles	89.90 $\pm$ 1.400	9.776 $\pm$ 0.82
Combined	91.96 $\pm$ 2.057	11.574 $\pm$ 1.18

Table 5.1: Summary of performance analysis and run-times, and standard deviation σ, for non-drifted data

	Drifted	
	Accuracy (% correct) $\pm \sigma$	Run-Time (seconds) $\pm \sigma$
Base	61.99 $\pm$ 1.550	8.331 $\pm$ 0.94
Adaptive	78.06 $\pm$ 1.416	12.060 $\pm$ 1.09
Ensembles	75.75 $\pm$ 1.608	9.965 $\pm$ 1.21
Combined	83.07 $\pm$ 2.220	13.115 $\pm$ 1.12

Table 5.2: Summary of performance analysis and run-times, and standard deviation σ, for drifted data

5.3.2 Real-Time Analysis

Table 5.2 summarizes the performance and average run-time for each experimental case for both drift scenarios. Table 5.3 expands this with the slow-down factor of each experiment. We define slow-down as average run-time over the run-time of the base case for that drift scenario.

	Non-Drifted Data		Drifted Data	
	Run-Time (seconds)	Slow-Down	Run-Time (seconds)	Slow-Down
Base	8.265	1.000	8.331	1.000
Adaptive	10.157	1.229	12.06	1.448
Ensembles	9.776	1.183	9.965	1.196
Combined	11.574	1.400	13.115	1.574

Table 5.3: Summary of run-time and slow down of experimental cases, relative to base case

Since both the base case and the ensemble method behave the same regardless of whether concept drift exists, their times do not differ significantly in the two scenarios. The adaptive method has a small overhead to check for concept drift (with a slow-down of 1.229), and has significant overhead caused by the hill-climbing and model rebuilding phases (slow-down of 1.448). The ensemble method has overhead associated with rebuilding and making use of a larger number of models (with a slow-down of 1.183 and 1.196). The combined method of using both adaptation techniques and ensembles, had the combined overhead of both techniques (with slow-down of 1.400 and 1.574).

Recall that the combined method performed, on drifted data, an average of 21.08% better than the base case, 5.01% better than the adaptive method alone, and 7.31% better than ensembles alone. By limiting the amount of data used for model building, and thus keeping the cost of building a single model constant between cases, these factors will scale to a larger system.

Therefore, the combined system might take almost 60% more time than the basic unsophisticated system. But, it will produce results with significant improvement in

performance.

Scaling to Larger Data Sets

This section discusses the scalability of the results seen so far in the chapter to larger data sets, and the feasibility of deploying the PASIF framework to applications that involve large volumes of data.

Figure 5.3 shows the average run-times, taken as the average of 5 runs on distinct data sets. Table 5.4 shows the collected data, and the relative slow-down from the base case to the combined system.

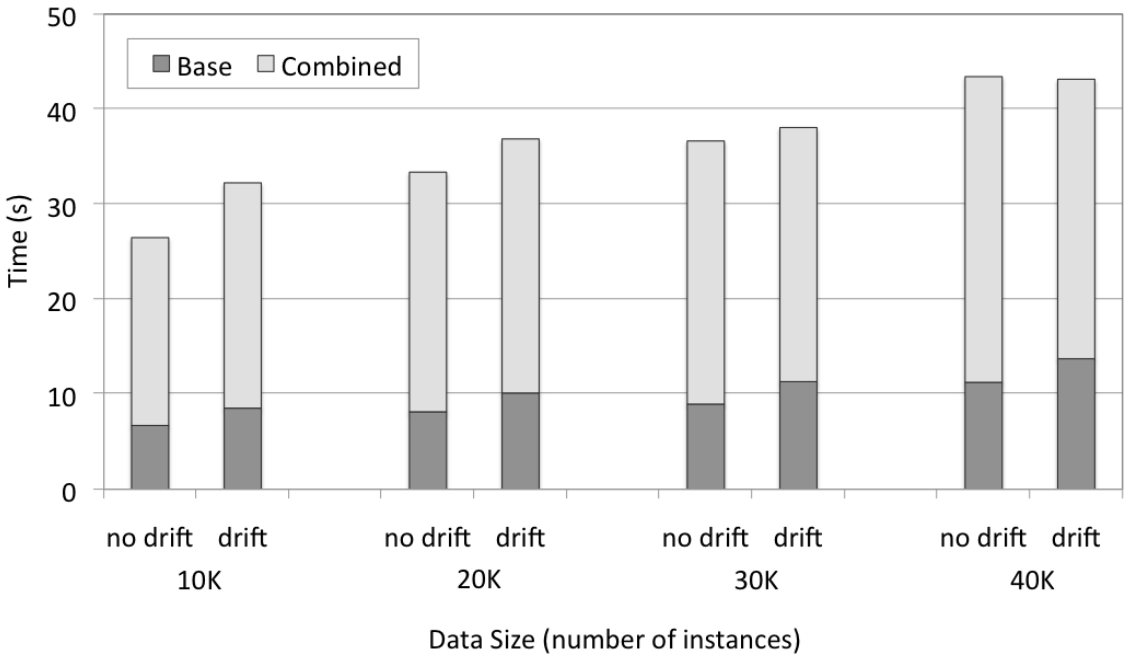


Figure 5.3: Average run-times scaling to larger data sets

Figure 5.4 presents the slow-down factor of the combined system on both the drifted and non-drifted data sets. On non-drifted data, the slow-down stays fairly

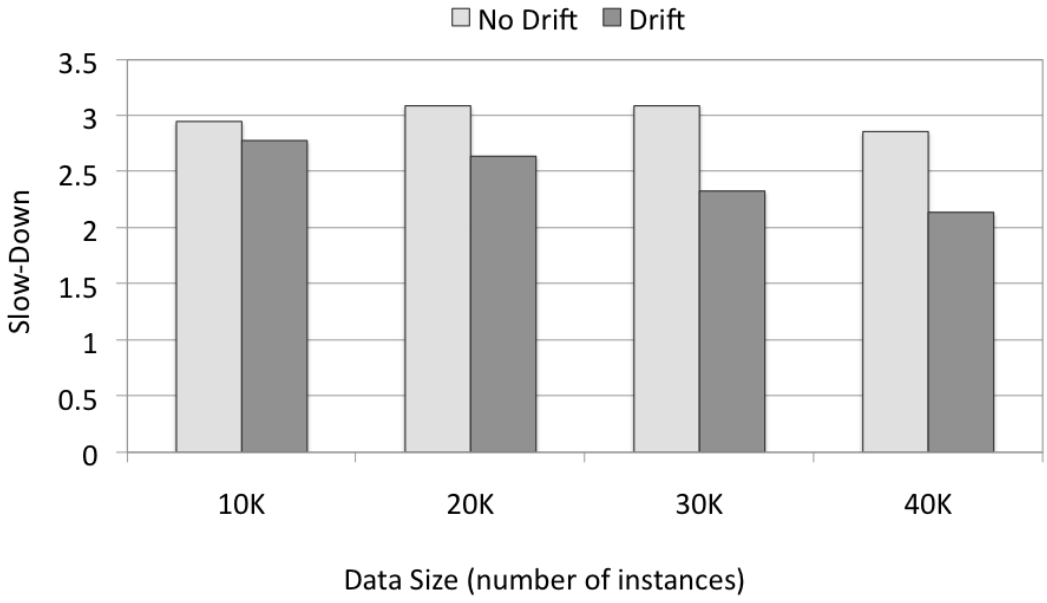


Figure 5.4: Scaling run-time to larger data sets: slow down of combined predictive system, relative to base case

consistent. Both systems work by rebuilding at a constant interval of 400 instances, using all data collected by the system so far. The combined system rebuilds 40 predictive models, rather than the single model used by the base case, which accounts for most of the computational costs, along with the small overhead of checking for drift. On drifted data, the slow-down factor of the combined system actually improves over time, as it uses only the data it deems relevant, and models are built using a smaller data set.

To improve the efficiency in non-drifted data sets, data can be omitted as it gets older. After a point, adding more training data produces insignificant improvement in the accuracy of the predictive models, and a smaller set of training data will suffice. This can be done simply by using only the most recently collected data points.

		Base Run-Time ± σ (seconds)	Combined Run-Time ± σ (seconds)	Slow-Down
No Drift	10,000	6.71 ± 1.36	19.77 ± 1.87	2.95
	20,000	8.15 ± 0.46	25.22 ± 1.20	3.09
	30,000	8.96 ± 1.75	27.69 ± 1.01	3.09
	40,000	11.25 ± 1.92	32.17 ± 1.71	2.86
Drift	10,000	8.53 ± 1.83	23.71 ± 2.34	2.78
	20,000	10.12 ± 2.40	26.74 ± 2.72	2.64
	30,000	11.32 ± 2.78	26.39 ± 3.25	2.33
	40,000	13.73 ± 3.06	29.41 ± 3.51	2.14

Table 5.4: Summary of run-times and standard deviation σ, and slow down of combined case relative to base case

It is therefore determined that the slow-down scales to larger data sets. Further, by changing the maximum number of models, it is possible to trade some accuracy for speed in the combined system.

Chapter 6

Summary and Conclusions

The traditional paradigm for human-computer interactions, where interactions are server-driven rather than user-driven, has limitations that are becoming increasingly apparent [37]. A new paradigm is emerging, Smart Interactions, which focuses on the perspective of the user, and concentrates primarily on their needs.

A major obstacle in achieving Smart technologies is that of providing effective tools for cognitive support. This thesis conjectures that predictive analytics provides a good platform for effective cognitive support for Smart Interactions.

6.1 Thesis Contributions

We have examined key issues in providing cognitive support for Smart Interactions and the excellent potential that predictive analytics has for filling that role. Further, we have proposed how predictive analytics might be implemented as a real-time support tool for Smart applications.

The contributions of this thesis are summarized as follows:

- We have proposed PASIF: a generic framework for incorporating predictive and real-time analytics in Smart Internet applications, expanding on the CRM framework of Bigus et al [11].
- We have combined ensemble modelling with a weighted scheme based on user feedback, along with offline model building tied to a mining database, to create an online prediction service, and analyzed the feasibility of its deployment in real-time applications.
- We have incorporated concept drift detection in the modelling process to adapt to changes in the underlying distribution of data as it is collected by the prediction service.
- We have implemented and examined the proposed prediction service.

6.2 Conclusions

Based on this research, we conclude that:

- Predictive analytics provides a good option for providing cognitive support to Smart Interactions.
- The PASIF framework is a viable solution to incorporating predictive analytics into Smart Interactions.
- The proposed prediction service provides accurate predictions and scalable runtimes, making it an excellent solution for real-time cognitive support in Smart Interactions.

6.3 Future Work

The primary objectives of this work were to propose the generic framework and present a proof-of-concept implementation. To do this, some assumptions were made and some issues overlooked.

Some directions for future work include the following:

First, threads for models can be implemented in the prediction service. This will allow for concurrent model building, rather than the sequential system currently implemented.

Further, an implementation of the full end-to-end framework should be completed and analyzed. The full framework should be tested on data sets for the same experimental results presented in this thesis to examine issues that may arise in the connection between the prediction service and the data warehousing portions of the framework.

Finally, the PASIF framework should be deployed and tested for use in a real application. The performance of the framework should be compared to other real-time analytic frameworks, such as those discussed in Section 2.6.

Bibliography

[1] O. Aase, J. Jonsbu, K. Liestol, A. Rollag, and J. Erikssen. Decision support by computer analysis of selected case history variables in the emergency room among patients with acute chest pain. *European Heart Journal*, 14(4):433–440, 1993.

[2] H. Abdulsalam. *Streaming Random Forests*. PhD thesis, Queen's University, Canada, 2008.

[3] N. Abe, E. Pednault, H. Wang, B. Zadrozny, W. Fan, and C. Apte. Empirical comparison of various reinforcement learning strategies for sequential targeted marketing. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining*, page 3, Washington, DC, USA, 2002. IEEE Computer Society.

[4] L. Agosta. The future of data mining - predictive analytics. *Information Management Magazine*, August 2004.

[5] I. Almosallam and Y. Shang. A new adaptive framework for collaborative filtering prediction. In *Evolutionary Computation, 2008. IEEE World Congress on Computational Intelligence*, pages 2725 –2733, June 2008.

[6] C. Apte, E. Bibelnieks, R. Natarajan, E. Pednault, F. Tipu, D. Campbell, and B. Nelson. Segmentation-based modeling for advanced targeted marketing. In *KDD '01: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 408–413, New York, NY, USA, 2001. ACM.

[7] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *Artificial neural networks: concept learning*, pages 81–93, 1990.

[8] A. Bifet and E. Frank. Sentiment knowledge discovery in Twitter streaming data. In *Proc 13th International Conference on Discovery Science*, Canberra, Australia, pages 1–15. Springer, 2010.

[9] A. Bifet, E. Frank, G. Holmes, and B. Pfahringer. Accurate ensembles for data streams: Combining restricted Hoeffding trees using stacking. In *Proc 2nd Asian Conference on Machine Learning*, Tokyo. JMLR, 2010.

[10] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research (JMLR)*, 2010.

[11] J. Bigus, U. Chitnis, P. Deshpande, R. Kannan, M. Mohania, S. Negi, P. Deepak, E. Pednault, S. Soni, B. Telkar, and W. B. Crm analytics framework. In *Proc. of 15th Int. Conf. on Management of Data (COMAD 2009)*, Mysore, India, December 2009.

[12] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Technical Report MSR-TR-98-12, Microsoft Research, 1998.

[13] L. Breiman. Bagging predictors. *Mach. Learn.*, 24:123–140, August 1996.

[14] H.-M. Chuang, L.-C. Wang, and C.-C. Pan. A study on the comparison between content-based and preference-based recommendation systems. In *Semantics Knowledge and Grid 2008 SKG 08 Fourth International Conference on*, pages 477–480, 2008.

[15] M. Connor and J. Herlocker. Clustering items for collaborative filtering, 2001.

[16] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*, 2006.

[17] X. Deng, M. Ghanem, and Y. Guo. Real-time data mining methodology and a supporting framework. In *Network and System Security, 2009. NSS '09. Third International Conference on*, pages 522 –527, Oct. 2009.

[18] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[19] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *In Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[20] J. Gama and A. Cornujols. Resource aware distributed knowledge discovery. *Ubiquitous Knowledge Discovery*, pages 40–60, 2010.

[21] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The weka data mining software: An update. volume 11, 2009.

[23] G. Holmes, A. Donkin, and I. H. Witten. Weka: A machine learning workbench. In *Intelligent Information Systems,1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pages 357–361, Aug 1994.

[24] S. J. Hong, R. Natarajan, and I. Belitskaya. A new approach for item choice recommendations. In *DaWaK '01: Proceedings of the Third International Conference on Data Warehousing and Knowledge Discovery*, pages 131–140, London, UK, 2001. Springer-Verlag.

[25] Y. Huang. An item based collaborative filtering using item clustering prediction. In *Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium on*, volume 4, pages 54 –56, Aug. 2009.

[26] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4(1):237–285, 1996.

[27] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 180–191. VLDB Endowment, 2004.

[28] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation, and applications. In *Proceedings of the 3rd ACM*

SIGCOMM conference on Internet measurement, IMC '03, pages 234–247, New York, NY, USA, 2003. ACM.

[29] B. Lakshmi and G. Raghunandhan. A conceptual overview of data mining. In *Innovations in Emerging Technology (NCOIET), 2011 National Conference on*, pages 27 –32, Feb. 2011.

[30] N. Leavitt. Recommendation technology: will it boost e-commerce? *Computer*, 39(5):13 – 16, May 2006.

[31] Li-Tung, Y. Xu, and Y. Li. A framework for e-commerce oriented recommendation systems. In *Active Media Technology, 2005. (AMT 2005). Proceedings of the 2005 International Conference on*, pages 309 – 314, May 2005.

[32] P. Martin, M. Matheson, J. Lo, J. Ng, D. Tan, and B. Thomson. Supporting smart interactions with predictive analytics. In *The Smart Internet'10*, pages 103–114, 2010.

[33] H. Masnadi-Shirazi and N. Vasconcelos. Cost-sensitive boosting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PP(99):1 –1, 2010.

[34] M. Mont, Y. Beresnevichiene, D. Pym, and S. Shiu. Economics of identity and access management: Providing decision support for investments. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 134 –141, April 2010.

[35] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital Libraries*, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.

[36] R. Munger. Lights, sirens, and computers: How pen-based computing is changing the way emergency care is conducted and communicated. In *Professional Communication Conference, 1999. Proceedings. IPCC 99. Communication Jazz: Improvising the New International Communication Culture. 1999 IEEE International*, pages 41 –47, 1999.

[37] J. W. Ng, M. Chignell, and J. R. Cordy. The smart internet: transforming the web for the user. In *CASCON '09: Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research*, pages 285–296, New York, NY, USA, 2009. ACM.

[38] K. Nishida, K. Yamauchi, and T. Omori. Ace: Adaptive classifiers-ensemble system for concept-drifting environments. In N. Oza, R. Polikar, J. Kittler, and F. Roli, editors, *Multiple Classifier Systems*, volume 3541 of *Lecture Notes in Computer Science*, pages 509–509. Springer Berlin / Heidelberg, 2005.

[39] G. P. Papamichail and D. P. Papamichail. The k-means range algorithm for personalized data clustering in e-commerce. *European Journal of Operational Research*, 177(3):1400 – 1408, 2007.

[40] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pages 39–42, 2007.

[41] L. Rokach. Ensemble methods for classifiers. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 957–980. Springer US, 2005.

[42] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.

[43] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2nd Edition, 2003.

[44] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic Commerce*, EC '00, pages 158–167, New York, NY, USA, 2000. ACM.

[45] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5:3–55, January 2001.

[46] D. Stetson, R. Eberhart, R. Dobbins, W. Pugh, and A. Gino. Structured specification of a computer assisted medical diagnostic system. In *Computer-Based Medical Systems, 1990, Proceedings of Third Annual IEEE Symposium on*, pages 374 –380, June 1990.

[47] R. S. Sutton. Learning to predict by the method of temporal differences. In *Machine Learning*, volume 3.1, pages 9–44, 1988.

[48] C.-J. Tsai, C.-I. Lee, and W.-P. Yang. An efficient and sensitive decision tree approach to mining concept-drifting data streams. *Informatica*, 19:135–156, January 2008.

[49] K. H. L. Tso and L. Schmidt-Thieme. Evaluation of attribute-aware recommender system algorithms on data with varying characteristics. In *Proceedings*

of the Tenth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pages 831–840. Springer, 2006.

[50] P. Vorburger and A. Bernstein. Entropy-based concept shift detection. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 1113–1118, Washington, DC, USA, 2006. IEEE Computer Society.

[51] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge,England, 1989.

[52] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[53] M. Widenius and D. Axmark. *Mysql Reference Manual*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1st edition, 2002.

[54] A. Wong and R. Wu. 5e: A framework to yield high performance in real-time data mining over the internet. In *High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on*, volume 2, pages 708 – 713, 2000.

[55] R. Yager. Targeted e-commerce marketing using fuzzy intelligent agents. *Intelligent Systems and their Applications, IEEE*, 15(6):42 –45, Nov/Dec 2000.

[56] E. Zhang and M. Mayo. Enhanced spatial pyramid matching using log-polar-based image subdivision and representation. In *Proc International Conference on Digital Image Computing: Techniques and Applications*, Sydney, Australia. IEEE Computer Society, 2010.

[57] Q. Zhang, C. Pang, S. Mcbride, D. Hansen, C. Cheung, and M. Steyn. Towards health data stream analytics. In *Complex Medical Engineering (CME), 2010 IEEE/ICME International Conference on*, pages 282 –287, July 2010.

[58] Y. Zhang, J. Qi, H. Shu, and J. Cao. Personalized product recommendation based on customer value hierarchy. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 3250 –3254, Oct. 2007.

[59] A. Zhou, S. Qin, and W. Qian. Adaptively detecting aggregation bursts in data streams. In L. Zhou, B. Ooi, and X. Meng, editors, *Database Systems for Advanced Applications*, volume 3453 of *Lecture Notes in Computer Science*, pages 985–985. Springer Berlin / Heidelberg, 2005.

[60] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 336–345, New York, NY, USA, 2003. ACM.

Appendix A

Simulated Annealing

Simulated annealing is a version of stochastic hill-climbing search which allows downhill moves [43]. Downhill moves are readily accepted in early iterations of the algorithm, but become less probable with time. Pseudocode for the simulated annealing algorithm is presented in Figure A.1.

The algorithm is analogous to the process of annealing metal [43], where a metal is heated and cooled in order to form some desired underlying structure, namely that which has the lowest internal energy and thus minimal defects. When the metal's atoms are heated, they are released to higher energy states. As they cool, they settle into new configurations. By slowly cooling the atoms, and by allowing the atoms to move to higher energy configurations, it is possible to create configurations with lower energy than in the initial configuration.

In simulated annealing, the downhill moves simulate atoms moving to higher energies, and thus to a less desirable state. By moving from a local maxima, it becomes possible to potentially acquire a state of higher value, and thus climb to a higher maxima.

function SIMULATED-ANNEALING (*problem, schedule*) **returns** a solution state

inputs: *problem*, a problem
schedule, a mapping from time to "temperature"

local variables: *current*, a node
next, a node
T, a "temperature" controlling the probability of downward steps

current $\leftarrow$ MAKE-NODE(INITIAL-STATE[*problem*])
for $t \leftarrow 1$ **to** ∞ **do**
  $T \leftarrow$ *schedule*[t]
  if $T = 0$ **then return** *current*
  next $\leftarrow$ a randomly selected successor of *current*
  $\Delta E \leftarrow$ Value[*next*] - Value[*current*]
  if $\Delta E > 0$ **then** *current* $\leftarrow$ *next*
  else *current* $\leftarrow$ *next* only with probability $e^{\Delta E/T}$

Figure A.1: Simulated annealing [43]

Appendix B

The Agaricus-Lepiota Data Set

The Agaricus-Lepiota (Mushroom) data set is obtained from the UCI Machine Learning repository [18]. This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as edible, poisonous, or of unknown edibility. The latter class was combined with the poisonous one.

The data consists of 22 categorical attributes, ranging from 2 to 12 arities. The data set contains 8124 instances, and contains missing values. Specific attribute information is as follows:

Attribute Information:

1. **Cap-shape**: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
2. **Cap-surface**: fibrous=f, grooves=g, scaly=y, smooth=s
3. **Cap-color**: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y

4. **Bruises**: bruises=t, no=f
5. **Odor**: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
6. **Gill-attachment**: attached=a, descending=d, free=f, notched=n
7. **Gill-spacing**: close=c, crowded=w, distant=d
8. **Gill-size**: broad=b, narrow=n
9. **Gill-color**: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
10. **Stalk-shape**: enlarging=e, tapering=t
11. **Stalk-root**: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
12. **Stalk-surface-above-ring**: fibrous=f, scaly=y, silky=k, smooth=s
13. **Stalk-surface-below-ring**: fibrous=f, scaly=y, silky=k, smooth=s
14. **Stalk-color-above-ring**: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
15. **Stalk-color-below-ring**: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
16. **Veil-type**: partial=p, universal=u
17. **Veil-color**: brown=n, orange=o, white=w, yellow=y
18. **Ring-number**: none=n, one=o, two=t
19. **Ring-type**: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z

20. **Spore-print-color**: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
21. **Population**: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
22. **Habitat**: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

Appendix C

The Mining Database Schema

For the experiments run in this thesis, we simulated three data sets by spliting the Agaricus-Lepiota data set, as described in Appendix B, into smaller subsets. The schema is shown in Figure C.1.

The subsets were determined based on relations between attributes, as well as experimental runs. Groups were determined so that a single group would have good, but not perfect performance (correctly labelling between 60 and 70% of instances on trial runs).

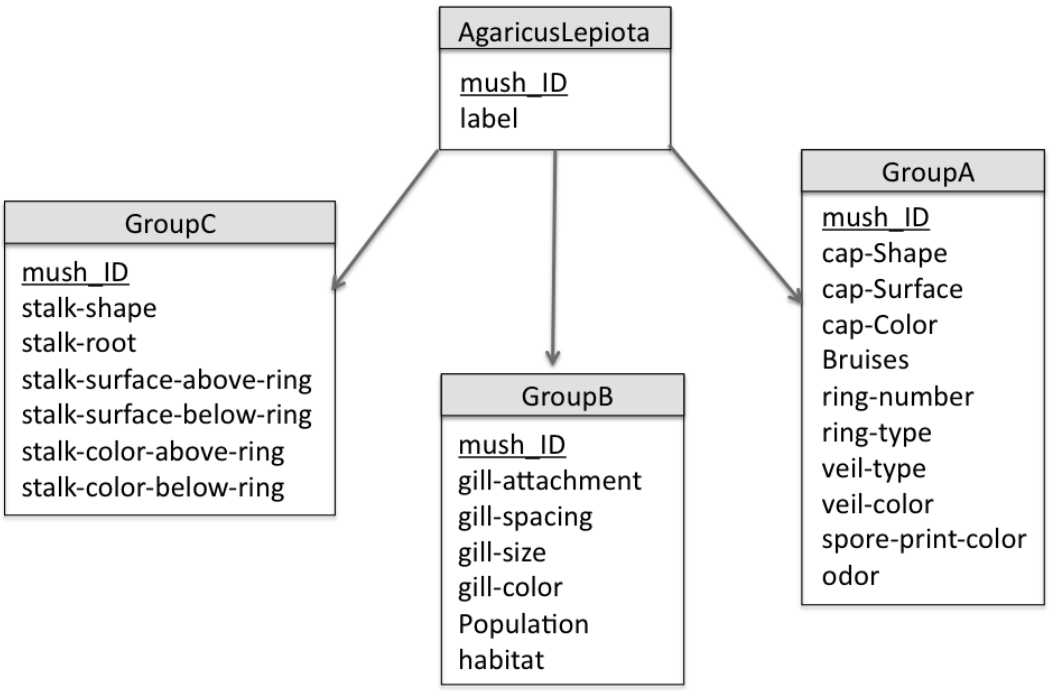


Figure C.1: Mining database schema for experiments

Appendix D

Weka

Weka (**W**aikato **E**nvironment for **K**nowledge **A**nalysis) is a collection of machine learning algorithms, written in Java, for data mining tasks [22]. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization [22, 23].

Weka is open source software issued under the GNU General Public License, developed by Waikato University, New Zealand. The first version (v2.1) was released in 1996 and the most recent stable version (v3.6) was released in 2010.

Noteable applications of the Weka machine learning suite in research include the creation of data stream ensembles for data streams using Hoeffding trees [9], capturing spatial information for object categorization with bag-of-words [56], knowledge discovery in Twitter streaming data [8], and massive online analysis [10].