Chapter 1

Introduction

1.1 Introduction

The study of Adaptive User Interfaces (AUI) is concerned with the intelligent personalisation of software interfaces. The work presented in this dissertation demonstrates how adaptive user interfaces can be applied to mobile phones.

1.1.1 Motivation

Mobile phones have become ubiquitous, outnumbering desktop computers. Also, the computational resources available to such devices continue to improve. This increase in computational power has allowed applications found on desktop computers to be incorporated into mobile phones. However, the question still exists of how to make such applications usable given the limited physical form of mobile phones.

The motivation for incorporating adaptivity into mobile phone user interfaces is to improve their usability. The type of adaptivity presented is based on predicting a user’s future actions, in order to save the user from unnecessary or excessive interactions with a mobile phone.

For an interface to be predictive some form of learning is required. However, the application of traditional machine learning approaches poses several problems in this setting. These problems include:

- A mobile phone’s limited computational and memory capacity which
restricts the application of machine learning.

- A learning environment that exhibits concept drift, e.g. the concept being learned changes as the user’s mode of operation changes.

- A user’s intolerance of any hindrance that machine learning may introduce into the normal operation of a mobile phone.

The difficulties mentioned above are not exclusive to the mobile phone predictive interface setting. For instance, computational efficiency is an important property of any learning algorithm as it dictates an algorithm’s practical usefulness. However, given the large gap in computational resources between the average mobile phone and the modern personal computer, computational efficiency is all the more important in the mobile phone setting. Furthermore, the issue of computational efficiency is not only a concern with mobile phones but also with any other computing device that has similar computational limitations.

The problem of concept drift is present in any machine learning setting where the learned concept can change. In the mobile phone predictive interface setting concept drift is of interest because the rate at which it occurs is relatively high when compared with other learning settings. The mobile phone predictive interface setting is not the only setting characterised by a high rate of concept drift. In general, learning for any user interface adaptation will exhibit high degrees of concept drift, due to the many short-lived functions a user can perform on an interface.

The mobile phone predictive interface setting has the potential to introduce a number of problems that a user will not tolerate. These problems are lag in interface responsiveness, poor performance in predicting the user’s wants or needs, and a sense of interface indeterminism which can ultimately lead to a sense of loss of control for the user. These problems are not restricted to the mobile phone predictive interface setting and can arise in any predictive interface setting. However, given that a user may have had previous experience with telephony products that have historically been relatively reliable and simple to use, he or she may demand that a mobile phone be similarly reliable and simple to operate.
1.1.2 Thesis Contributions

This dissertation offers several solutions to the problem of incorporating adaptivity into the user interfaces of mobile phones. These solutions form the main contribution, and are as follows:

• The primary contribution is the proposal, implementation and evaluation of a predictive menu as a means of incorporating adaptivity into a mobile phone interface.

• A novel learning algorithm that is suited to the unique requirements of the menu prediction environment is presented. This algorithm employs a dynamic windowing strategy to address concept drift in the menu prediction environment. The dynamic windowing strategy of the learner is evaluated and compared with other menu prediction learning approaches, using both simulated and real world data. A preliminary theoretical justification for the dynamic windowing strategy is provided.

• An information-theoretic approach for determining concept drift in training examples is presented. The aim of this approach is to overcome the need to specify the hypothesis space a priori. The information-theoretic approach is compared with the previous approach that specified hypotheses a priori. A comparison is made between the two approaches using real world data.

• Motivated by the benefit of menu prediction on a mobile phone interface, a method for inducing shortcuts on a mobile phone interface is presented. The novel approach is conceptually split into two stages. The first stage identifies relevant action sequences. A regular expression is used to specify and hence identify relevant action sequences. The second stage attempts to predict which action sequence to present to the user as a shortcut. Several different shortcut prediction approaches are presented, including simple prediction strategies, probabilistic models, and decision tree learning. The shortcut prediction approaches are compared and evaluated using real world data.
Finally, a novel criterion for evaluating shortcut prediction is presented. The criterion combines a measure of a shortcut’s stability with the efficiency it brings about. Several methods of combining measures of efficiency and stability are evaluated and discussed.

The first part of this thesis is concerned with improving the predictive accuracy of user interface prediction. Several approaches are presented and evaluated. The second part considers the real aims of prediction in a user interface. When concerns such as a user’s comprehension of prediction approaches are taken into account, the conclusion is that relatively simple learning approaches outperform more complicated approaches.

The publications produced during the course of this PhD include:


1.1. INTRODUCTION

1.1.3 Overview of Thesis

The body of this dissertation is organised as follows:

- Chapter 2 discusses and reviews previous work related to the use of intelligence and adaptivity in user interfaces. The chapter provides an important perspective on the goals and problems of creating adaptive user interfaces. Closely related to the work presented in this dissertation is the review of interface adaptivity on mobile computing interfaces. A review is also conducted into the user modelling and machine learning approaches commonly used to achieve adaptivity.

- In Chapter 3, an approach for incorporating adaptivity on a mobile phone interface is presented. The approach is based on predicting a user’s next menu selection. The approach attempts to increase interface efficiency while maintaining interface predictability. A novel learning approach is presented which overcomes the problems of learning in this setting, and this learning approach is evaluated from both a theoretical and experimental perspective.

- A deficiency with the approach presented in Chapter 3 is the need for the learning approach to have its hypothesis space manually specified. In Chapter 4, an approach is presented that removes this limitation. A rule learner is used to generate hypotheses online. An information-theoretic approach is used to evaluate each proposed hypothesis. The evaluation method determines when the concept being learned has changed. The evaluation criterion acts as a controller to the rule learner, and determines which training examples should be learned over. The hypotheses learned by the rule learner form the hypothesis space used by the learning approaches presented in Chapter 3. The information-theoretic approach is compared experimentally with the approach introduced in Chapter 3.

- In Chapter 5, menu prediction is extended to the prediction of action bi-grams. The actions that a user intends to make are predicted and then presented as a shortcut. Compared to menu prediction, this
approach can be applied more widely, allowing for greater interface adaptivity and therefore greater savings in interface efficiency. Several learning approaches are considered and evaluated experimentally. Apart from predictive accuracy, these approaches are evaluated with regard to another criterion that is specific to the interface adaptivity setting. This criterion measures the rate at which predictions change. Several methods for combining predictive accuracy with this measure are discussed.
Chapter 2

Background

This chapter provides a background on research conducted into intelligent and adaptive user interfaces. Given the volume of material available on intelligent and adaptive interfaces, only a review of past developments relevant to the work presented in the following chapters is provided. There is a particular focus on a discussion of machine learning techniques used in adaptive user interfaces, and the application of adaptive user interfaces to mobile computing devices. This background motivates the novel approach to user interface adaption on mobile devices presented in later chapters.

2.1 Intelligent User Interfaces

The central goal of human computer interaction (HCI) is to better facilitate the interaction between computer systems and their users. Within HCI, intelligent user interface (IUI) research is made distinct by the use of artificial intelligence techniques to improve user interfaces. According to Miller et al. [79], the need for IUI arises from the fact that neither artificial intelligence nor good interface design alone can overcome serious interface problems. Miller et al. notes that a well-designed interface can communicate effectively with its users, but if it does not understand its users’ needs, it will be unsuccessful. On the other hand, artificial intelligence techniques allow an interface to use reason in addressing its users’ needs, but it will be unsuccessful if it does not effectively communicate with its users. Together, good interface design and
artificial intelligence complement one another.

The exact characteristics that make an interface intelligent are open to debate. One reason for this lack of consensus is that the field spans a broad number of research areas. Instead of tackling IUI as a whole, researchers have focused on the following areas: dialogue understanding, error remediation, tutoring, intelligent help, and user modelling. Despite this division, Chin [19] states the one essential characteristic of an IUI must be its ability to take the initiative in response to users’ commands and actions. Waern [109] provides an alternative definition by discussing what cannot be classified as an IUI. Firstly, the notion that any intelligent system or system with a good interface is an IUI is too broad. Many systems can claim to be intelligent or having a good interface design without intelligence being exhibited in the user interface. Alternatively, the notion that human dialogue is required for a IUI is too restrictive. However, dialogue-based interfaces provide a good example of how intelligence can be applied to an interface, and were the basis of many early IUIs. The advent of direct manipulation graphical interfaces [102] has expanded the possibilities of IUIs beyond that of solely dialogue. Finally, it is not necessary for an IUI to maintain and adapt a model of its user. An interface can act intelligently without the need to capture a user’s plans, goals and beliefs in a user model.

Debate also exist as to whether IUIs should be structured as an agent or as a tool with intelligently organised direct manipulation features [19]. An agent perspective seems a natural choice, since both rational agents and IUIs need to act in a way that optimises some preferred outcome in their environment. An agent perspective also affords the advantage of applying the notions of rational agency to IUIs. The intentional stance of using the mental attitudes of belief, desire and intention have long been used to characterise the properties of practical reasoning agents. Naturally, this intentional stance is useful in describing and representing the properties of an IUI. Furthermore, issues such as handling uncertainty in an environment and bounding rationality have also been dealt within agent research and can be directly applied to IUIs. However, the agent perspective fails when considering the independent and autonomous nature of agents. An IUI that acts purely as an agent could exhibit unexpected behaviour to a user. An alternative to the agent
2.1. INTELLIGENT USER INTERFACES

perspective is the idea of an interface in which all interactions have direct and perceptible results for the user. From this perspective, an IUI would be a means of directly and intelligently manipulating a computer system [102].

2.1.1 Adaptive User Interfaces

Within IUI research is the area of adaptive user interfaces (AUI). An AUI can be broadly defined as an interface that can be modified based on the characteristics of its user. Edmonds [33] identifies three ways in which adaptation can take place: at the request of the user; by prompting the user to change the interface; or by automatically adapting the interface. The difference between these methods is that the user is required to intervene in the adaptation process. This difference is also reflected in the meaning of adaptive and adaptable user interfaces. Adaptive user interfaces (AUI) usually refer to interfaces that automatically adapt to a user, while adaptable user interfaces rely on the user choosing the interface’s characteristics. Adaptable user interfaces are defined by Kantorowitz and Sudarsky [55] as being able to support more than one dialogue mode, allowing a user to switch smoothly between dialogue modes at any time, and allowing a user to easily learn the different dialogue modes.

A more precise definition of an AUI is given by Grace de la Flor [26], and it separates the notion of a user model from an AUI. In this definition, the user model is the component that implements algorithms to capture and express personalisation information. The AUI takes a human perspective, and is the graphical user interface through which personalisation information can be accessed and used. AUIs can also be defined by their components. Benyon [12] describes the need for an AUI to have a user model, task model and a interaction model. Similar to Grace de la Flor’s definition, the user model captures the user’s personalisation information. A task model captures the semantics of the tasks that the system can perform and the interaction model captures the ways in which the user can interact with the system. Together these three components allow an interface to reason about how best to adapt to a user. Another definition, provided by Langley [67], states that “An adaptive user interface is a software artifact that improves its ability to in-
teract with a user by constructing a user model based on partial experience with that user.” Several components can be identified in this definition: a performance component; a data acquisition component; and a user model component. The performance component attempts to reduce user effort; the data acquisition component acquires information on past user interactions; and the user model component uses past interactions to improve the performance component.

It is interesting to note that the benefit of AUIs was discussed as early as 1960 by Licklider [69]. Licklider describes the notion of enabling “men and computers to cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs”. Similarly, Feeney and Hood [35], in a later paper noted “that a computer can and should adapt itself to the individual using it”.

One of the main motivations of AUI research is to allow a wide range of users to interact efficiently and effectively with a computer system. In many cases, interfaces are developed for a prototypical user and do not address differences amongst users. An example of how user differences influence interactions was noted by Egan [34], who in one experiment found that some users could take up to 30 times longer than others to complete a computer-based task. This is due to differences such as:

- Physical Ability
- Cognitive Skill
  - Learning style
  - Spatial ability
  - Perception
  - Deductive reasoning ability
- Personality

Users can differ along a number of lines. For instance, Iseki and Schneiderman [49], note that experienced users often prefer a more efficient interface, based upon command-line input, while novice users often prefer menu-based
2.2. SYSTEMS WITH ADAPTIVE USER INTERFACES

interfaces. One explanation is that menu-based interfaces present operations in a structured way, and allow a user to complete tasks without having to remember a set of predefined commands. A further difference is that novice users are often goal-oriented in their interactions, while their more experienced counterparts are focused on completing tasks efficiently. One application of an AUI is to accommodate users with different levels of experience. However, this task is made difficult by the fact that a user’s experience level is not constant, but increases as he or she interacts more with a system. A further discussion of user differences, and how these differences can be accommodated using AUIs, can be found in [12] and [89].

Kuhme and Schneider-Hufsmidt [63] identify two other motivations for AUIs. Both are related to a changing system environment. If users have to operate many different facets of a computer system, or are required to operate many different computer systems, they are faced with a changing system environment. In both cases an AUI can provide users with feedback on the current context of the system and provide a consistent interface across the different system environments.

A key argument for AUIs is that users are generally unwilling to change their behaviour to suit an interface’s interaction style. Instead, the interface should adapt to a user’s characteristics. The aim is to increase the performance of a task by matching the capabilities of the computer with the cognitive skill and behaviour strategies of the user. A brief description of a selection of systems that have implemented and evaluated AUIs is now given.

2.2 Systems with Adaptive User Interfaces

We now discuss several systems that have incorporated AUIs. We begin by considering the classification of AUI systems into categories.

There are many different ways of classifying AUIs [27]. One classification discussed by Langley [66], is by the role the interface plays for the user. In this regard two basic categories can be identified. The first category consists of interfaces that can adapt by filtering information for a user (informative AUIs). The second category consists of interfaces which generate new knowledge structures, such as plans or rules, that can be applied to assist the user.
(generative AUIs). A description is now provided of several informative and generative AUIs.

2.2.1 Information-Filtering Adaptive User Interfaces

Users are often overloaded by the amount of information available via the Internet. This problem has motivated the development of AUIs that filter information. The aim of these systems is to filter or recommend information based on a user’s preferences and requirements. Montaner et al. [83] identifies five components that such system should have: a user profile to capture a user’s preferences; a method of generating an initial user profile; a method of incorporating feedback from the user on what they find relevant; a method of learning from user feedback; and a means of ensuring that the user profile is up-to-date with the user’s current interests.

Apart from these five components, we also need to consider how the user profile is applied to identify relevant information. In this respect, AUIs that filter information have some similarities with information retrieval systems. Firstly, both attempt to find information that closely matches what the user wants. In the case of an AUI, what the user wants is specified by a user profile, while in information retrieval it is based on a user’s query. Secondly, both require a means of specifying the degree of certainty or uncertainty with which to classify new information. Finally, both are concerned with retrieving information with a high degree of precision and recall. However, information retrieval systems sometimes use collaborative or stereotypical filtering techniques, in which information is filtered based on the preferences of other similar users. AUIs differ in this regard by using only the user’s profile to filter information.

Given the components of an information-filtering AUI, we can now describe several systems in terms of how these components are implemented.

**WWW Recommendation**

**Personal WebWatcher** was created by Mladenic [81] and is an agent that assists in browsing the World Wide Web (WWW). It is derived from WebWatcher, which was created by Armstrong et al. [5]. **Personal Web-**
2.2. SYSTEMS WITH ADAPTIVE USER INTERFACES

Watcher provides interactive assistance by highlighting hyperlinks that will lead users to the information they require. The web pages a user views are considered interesting by the system and are used as training examples by a learning algorithm. Interesting web pages are converted to a TF-IDF (Term Frequency Inverse Document Frequency) vector representation [51]. To reduce the number of features passed to the learning algorithm, only those terms that make a good distinction between interesting and uninteresting web pages are used. Mladenic evaluates both a k-nearest neighbour and Naïve-Bayes learning approach, concluding that there is little difference in predictive accuracy between the two. Figure 2.1(a) shows the Personal WebWatcher homepage, where the user can enter a topic of interest. After entering a topic of interest, the user is returned to the web page from which they entered Personal WebWatcher. Figure 2.1(b) shows the following modifications Personal WebWatcher makes to this web page:

- A list of commands are displayed at the top of the page.
- The topic of interest is used as a search term on the pages WebWatcher has visited. The results of this search are displayed below the list of commands.
- Hyperlinks which are deemed useful according to the user’s topic of interest are highlighted.

Letizia [70] is another interface agent that attempts to predict topics of interest on the WWW. When requested by the user, Letizia recommends a list of future hyperlinks to follow. Unlike Personal WebWatcher a learning algorithm is not used. Instead, simple heuristics are used to infer the user’s interests. For example, saving a reference to a web page is assumed to indicate a strong interest in the information on that page. The use of heuristics allows Letizia to explain its recommendations to the user. Letizia avoids the issue of recommending too few or too many hyperlinks by allowing the user to specify a percentage of the hyperlinks to recommend.

The Syskill & Webert agent by Pazzani and Billsus [86] is yet another interface agent that predicts web pages of interest for a user. This agent differs from Personal WebWatcher and Letizia by requiring a user to
Figure 2.1: (a) The main web page of **Personal WebWatcher**. From this page the user enters a topic of interest. (b) The web page from which the user entered **Personal WebWatcher**. **Personal WebWatcher** modifies this web page according to the topic of interest entered by the user.
2.2. SYSTEMS WITH ADAPTIVE USER INTERFACES

rate web pages. A learning algorithm is used to generate a representation of the user’s preferences from the rated web pages. Pazzani and Billsus show a Naïve-Bayes classifier to be the most appropriate learning algorithm for this task. Unlike Personal WebWatcher and Letizia, The Syskill & Webert agent is not limited to recommending hyperlinks, and can also filter the results from a search engine.

Filtering and re-ranking search engine results is also the basis of a system described by Pretschner and Gauch [90]. In this system, previously visited web pages are represented using a TF-IDF vector model. These web pages are categorised according to a concept hierarchy that contains 4,300 nodes. A user’s preference or “browsing behaviour” is inferred from the time spent viewing a web page versus the web page’s length. The top five categories viewed by a user are weighted according the “browsing behaviour” measure, and become the user profile. Search engine results are filtered and re-ranked according to this user profile.

Balabanovic [8] presents a system that searches the Internet for web pages that are of interest to a user. This interface agent performs a four-step cycle: It searches the web using a search heuristic (in a bounded amount of time); a number of pages are then selected from the search results using a heuristic; relevance feedback is obtained through user rating of the pages presented; and finally the heuristics are updated based on the feedback. Similar to the systems mentioned previously, web pages and the user profile are represented as a TF-IDF vector model. The search heuristic scores each web page by taking the dot product of the vector representation of the web page and the user profile. Users assign a real number to weight each web page they view. The user profile is updated by simply adding the weighted web pages to the existing user profile.

Amalthaea [84] differs from the previously discussed systems in a number of ways. Firstly, it consists of a collection of agents that perform information filtering and information discovery tasks on the WWW. Secondly, it uses an evolutionary approach to learn a user profile. Information-filtering agents request web pages based on the user profile, and information discovery agents take such requests and access search engines for relevant web pages. Users provide feedback on suggested web pages, which results in credit being
Figure 2.2: The agents and communication processes within the Amalthaea system

assigned to the agents responsible for the suggestions. An agent’s fitness is related to the amount of credit it receives. The entire process can be seen in Figure 2.2. The evolutionary approach weeds out agents with poor fitness and creates new agents from the cross-over of agents with good fitness.

Internet News and Usenet

Information-filtering interfaces have also featured in more specialised applications on the Internet, such as filtering news gathered from the WWW and from Usenet groups.

Anatagonomy [54] and its predecessor Krakatoa [53] attempt to personalise newspaper articles from the WWW. Anatagonomy can infer a user profile from either explicit or implicit feedback from a user. Implicit feedback is obtained by how the user interacts with a news article. Operations that suggest a user has read an entire article, such as scrolling, are used to assign high scores to an article. The user profile is represented as a TF-IDF vector model and is generated from high scoring articles. Articles that contain words from the user profile vector model are suggested to the user. Another aspect of Anatagonomy is its ability to personalise the layout and format
2.2. SYSTEMS WITH ADAPTIVE USER INTERFACES

of news articles presented to the user.

**NewsWeeder** [64] also provides a personalised news-filtering service. It differs from **Anatagonomy** by not inferring feedback via passive means. Instead, it relies on a user to explicitly rate articles on a scale from one to five according to interest. Two methods were considered for filtering articles. The first involved representing articles in each rating category using a TF-IDF vector model. An average is taken to create a prototype representation for each category. A cosine similarity metric is then used to categorise new articles. The second approach uses an Minimum Description Length [40] (MDL) approach to determine a probabilistic distribution of terms (tokens) in each rating category. Informally, the similarity between an article and a rating category is inversely proportional to the number of bits required to encode the article with the category’s probabilistic model.

**Browse** [50] also requires a user to provide active feedback by accepting or rejecting suggested articles. The user model is represented as a neural network. In particular, each word in an article represents a node in the network, and is assigned an energy corresponding to the word’s frequency in the article. The links between nodes in the network are assigned a strength according to the co-occurrence frequency of words in the article. A key benefit of the neural network model, when compared with simple keyword matching techniques, is the ability to identify relevant articles without requiring all keywords to be contained in the article.

**PSUN** [104] differs from systems previously discussed in terms of the representation and learning technique used. Given an initial set of articles determined to be of interest to the user, fixed length sequences of \( n \) characters are extracted. These sequences are organised into a network with weighted links that represent the co-occurrence of different words. These n-gram models are refined into a user profile by means of a genetic algorithm. A k-lines-like method is used to evaluate the fitness of competing n-grams.

One problem faced by all these information-filtering AUIs is that a user’s interests change over time. This means that a user profile needs to adapt and stay current. **PSUN** maintains a current user profile by using a genetic algorithm which removes old and irrelevant agents via natural selection. Gradual forgetting is another technique, and is used by **ifWeb** [6] and
the similar SiteIF [105] systems. Gradual forgetting places more weight on recent examples of user behaviour when generating a user profile.

2.2.2 Generative Adaptive User Interfaces

AUIs are not restricted to information filtering and recommendation tasks. Many AUIs attempt to assist a user by completing or organising their intended tasks. Providing assistance of this nature requires an AUI to capture and reason with knowledge about the user and task domain. Some of the areas in which generative AUIs exist include programming-by-demonstration, scheduling, and intelligent tutoring or help systems. We now present and discuss several systems.

Action Prediction

Davison and Hirsh [24] investigated predicting user actions in a command-line interface. Based on the assumption that users’ command-line actions tend to repeat themselves, Davison and Hirsh describe a well suited online learning algorithm. The algorithm makes the assumption that an action depends only on the previous action performed. A table is used to store the frequency and co-occurrence of commands. Given a previous command, the most probable future command can be determined. Korvemaker and Greiner [61] extended Davison and Hirsh’s work by considering the prediction of entire command-line sequences, including command parameters. Other predictive features were also considered. These included the error code returned from the last command, time-of-day, and day-of-week. Gorniak and Poole [39] also attempt to predict a user’s future actions. However, they assume that action prediction can be applied to an application without modifying its normal operation. A user is viewed as an agent whose interactions with an application can be observed. From these interactions the agent’s decision-making policy may be inferred. An interaction history is used to assign probabilities to future actions. Probabilities are based on a k-nearest neighbour technique that determines similar sequences in the interaction history.

Action prediction has also been investigated within graphical user interfaces (GUI). The goal of action prediction on GUIs is generally to reduce
the number of mouse-clicks that a user needs to perform. Bao et al. [9] present FolderPredictor, a system that predicts which folder in a GUI folder hierarchy a user will access. The system applies a cost-sensitive prediction algorithm to previous folder accesses, which have been grouped according to a “task” context. Experimentally, the system reduces the cost of accessing a folder in terms of mouse-clicks by approximately 50 percent.

**Scheduling Assistance**

Jourdan et al.’s [52] calendar management apprentice (CAP) provides decision-making advice on the location and duration of meetings to be scheduled. CAP contains a knowledge base of calendar events and hand-written rules that govern how advice is supplied. Using each specific calendar event, generalisations are made about an event’s duration and location parameters. Two inductive learning approaches were considered, a neural network back-propagation approach and a decision tree approach. CAP addressed two important issues. Firstly, to prevent old and irrelevant events from influencing the generalisations, only the previous 180 calendar events were considered at any point-in-time. Secondly to allow newly learned rules to be incorporated into an existing rule base, statistics were kept on the number of correct and incorrect predictions each rule made, and were used to rank the salience of each rule. Figure 2.3(a) shows the components of the CAP systems, and Figure 2.3(b) shows CAP suggesting a meeting length of 60 minutes, but the user has changed this to 120 minutes.

**Intelligent Help and Tutoring**

The LUMIERE [47] project attempts to provide assistance to a user in the same way as a butler provides assistance, that is by anticipating a need and intervening non-intrusively to address it. When embedded in an application, LUMIERE used a Bayesian network to infer a user’s goal from observing a user’s actions. The appropriate action to achieve this goal was then suggested. LUMIERE served as the basis for the Microsoft Office™ Assistant.

A Bayesian network was also used in the DeepListener project [48] to facilitate reasoning with uncertainty. Again, a Bayesian network was used
Figure 2.3: (a) **CAP** components. (b) The **CAP** interface. A meeting length of 60 minutes is suggested, the user has changed this value to 120 minutes.
to infer a user’s goal. However, in this case, inference was derived from a spoken command instead of observed actions.

**Programming-by-Demonstration**

A generative AUI can increase an interface’s efficiency by automating repetitive procedures. With a similar goal, programming-by-demonstration aims to allow a user to specify a procedure via demonstration alone, without needing a procedure to be explicitly programmed. Automating a procedure can increase an interface’s efficiency, reduce the possibility of user-induced errors and allow the user freedom to perform other tasks.

An early programming-by-demonstration system was Witten’s [117] predictive calculator. User actions were stored in fixed length-k tuples. Whenever input actions matched all but the last element in a tuple, the last action in the tuple became the predicted action. This process is the basis of the length-k modelling technique.

**EAGER** [23], another programming-by-demonstration system, attempts to alleviate the problem of performing repetitive tasks in an application. **EAGER** searches for iterative patterns in a user’s interactions. When a pattern is found, **EAGER** uses this to predict the user’s next actions by highlighting either a menu item, button or text. Patterns are identified by searching for loops in the history of events. Whenever **EAGER** receives an event from an application, it searches for a similar event in the stored history of events. For text-based events, any common subsequence of events in the history is considered similar. Numeric events are considered similar if they are identical, consecutive, or follow some linear sequence. If the same event occurs more than once in the event history, then a loop has been discovered. All events within the loop are considered to be part of an event sequence. Events are described with enough semantic detail to avoid meaningless or redundant events from being included in an event sequence. For instance, if an item was dragged onto a desktop before being dropped onto another application, then the redundant desktop move event would not be considered.

**SMARTedit** by Lau et al. [68] also attempted to automate repetitive tasks through the use of a macro. When users want to automate a repetitive
task, they encode the necessary actions into a macro. Macros allow a user to record a sequence of actions that can be played back at a later time. However, macros can be inflexible, and editing them requires knowledge of the macro language. **SMARTedit** overcomes these problems by inferring macros from a user’s actions in a text editing application. A version space algebra learning technique is used to assist in generating macros. This technique only considers useful or likely macros, and therefore reduces the number of possible macros to be considered. Actions are represented as text-editing functions that change an application from one state to another. As a user performs actions, the set of possible functions is reduced by eliminating all those functions inconsistent with the action. In this way a macro can be inferred from very few observations.

**Segmentation and Data Mining Sequential Patterns**

An important element of automatic macro generation is the identification of patterns in a user’s behaviour. Chi Keun Wong and Keung Hui [74], in their personalised AUI system, describe this as a problem of determining user behavioural patterns which are hidden in a stream of user events. Dietterich [28] provides a review of the many techniques for extracting homogeneous segments from sequential and time-series data.

In addition to the problem of determining user behavioural patterns, there is the problem of determining if the pattern is relevant and of interest. One approach commonly used in the data mining field is to regard a pattern as interesting if it recurs with a high frequency. Often this frequency is specified as a threshold of support parameter.

Himberg et al. [46] describe a system for recognising context in mobile devices by segmenting time series data. Himberg et al. suggest this time series data could consist of sensory data such as noise level, luminosity, humidity or even acceleration. A dynamic programming technique is used to find segments that minimise intra-segment variance. Of considerable concern was the computational expense of minimising this intra-segment variance. Their results show that such an approach can identify some contextual phenomena in their time series data.
2.2.3 Mobile-Device Adaptive User Interfaces

AUIs have also been applied to mobile computing devices. An early example is the Names++ application by Schlimmer [101], which attempted to impart adaptivity to an Apple Newton™. This system was built upon previous work performed by Hermens and Schlimmer [45] and focused on interface adaptivity in an address book application. Three intelligent interface elements were evaluated: handwriting recognition, adaptive menus, and predictive text fill-in. For instance, in text entry fields on the device, the adaptive menu displayed the four most recently entered values for the field. Predictive text fill-in automatically filled in text entry fields using the values entered in previous similar fields. It was shown that by using predictive text fill-in or an adaptive menu, the user interface could be up to twice as fast as typing on screen when entering text.

The nature of mobile computing devices makes them a suitable platform for investigating interfaces that adapt based on their context [39]. The goal of these context-aware interfaces is to provide customised services and information based on a user’s context. A user’s context might be defined by their current activities. Eagle and Pentland [32] describe a system for sensing a user’s social context on a mobile device. A user’s context may also be defined by their location. The CRUMPET project [88] investigated the use of an agent-based approach for providing tourists with relevant information. Agents are an obvious means for implementing mobile-based assistants. Koch and Rahwan [60] review a number of existing attempts to build agent-based mobile assistant applications.

2.3 Machine Learning for User Modelling

User modelling is an important part of an AUI. It allows an interface to adapt to a user’s personal preferences and requirements. One method for generating a user model is by inferring it using a machine learning technique [111]. Some of the machine learning techniques used to infer a user model include linear models, TF-IDF vector-based models, Markov models, neural networks, rule-induction methods, unsupervised classification methods, and
Bayesian networks.

Linear models are a relatively simple approach to user modelling. The value of an unknown quantity can be predicted using the weighted sum of similar quantities. The learning task consists of determining the weights to assign to the known quantities.

The TF-IDF vector-based model is used by many of the information-filtering AUI systems, including Personal WebWatcher [81], Anagnostomy [54], and NewsWeeder [64]. The terms within a document are represented using a vector space model [51]. The vector of terms in a document are weighted according to a TF-IDF measure. The similarity between two documents can be determined from the cosine of the angle between the vector representation of the two documents. A user model can be learnt by identifying the best possible TF-IDF vector-based model to represent the user.

Given some observed event, the Markov model approach predicts an event based on the probability distribution of events that follow on from the observed event. The Markov model assumes that events follow the Markov property, in that any future state depends only on the current state. In this case, the learning task is to determine the appropriate probability distributions.

A neural network user model is represented by means of a network of nodes, where each node can represent some function. These functions can have non-linear activation thresholds. The edges between nodes can also be assigned differing weights. A neural network representation can be learnt by adjusting the threshold on functions and the edge weights so that the overall network models the user's observed actions. A neural network approach was used in the Browse information-filtering AUI.

The rule-induction methods, which include decision-tree learners, represent a user model as a set of rules that when applied to an event separates it into one of a number of classes. Each observed event is represented as an instance, or more specifically as a vector of attribute-values. The classification rules dictate which class an event belongs to, based on the value of its attributes. Classification rules are learnt by observing events from each class, and then generalising about the attribute values that separate each
class. The Syskill & Webert system evaluated a decision-tree learner for generating a user model.

Unsupervised classification methods differ from the rule-induction methods in that the class to which an observed event belongs is unknown. One unsupervised classification technique is to cluster events into groups that have similar attributes. In Mobasher et al.'s [82] system, URLs were clustered to create a personalised news filtering adaptive user interface.

Bayesian networks represent probabilistic modelling. Each node in a Bayesian network contains a conditional probability distribution over its possible values. This distribution is conditioned for all possible values of its parent nodes. The conditional probabilities of a Bayesian network can be derived from the frequency of observed values. The LUMIERE [47] project took advantage of a Bayesian network to provide intelligent help to a user.

According to Webb et al. [111] there are four issues involved with the application of machine learning to automated user modelling: the need for a large training data set; the need for a labelled training set; computational complexity; and concept drift. We now discuss these issues in more detail.

### 2.3.1 Obtaining Sufficient Labelled Training Data

When a machine learning technique is used to infer a user model, feedback on the user model's predictions provides the training data for the learner. The feedback, be it a numeric rating or an accept or reject value, provides the label for the data. Many machine learning techniques improve their predictive accuracy when given more training data. However, obtaining a user model with high predictive accuracy is difficult in many applications, since the amount of feedback and thus training data is limited. Supervised machine learning techniques require that training data be labelled, that is a user must provide some feedback on a user model's predictions. User feedback can be obtained either by the user explicitly rating predictions or by inferring the prediction's usefulness from the user's behaviour. Often explicit feedback is difficult to obtain since users often do not wish to be burdened with the task of rating predictions. If a user does not rate predictions, then feedback must be obtained implicitly. Kelly and Teevan [57] review a number of im-
explicit feedback techniques. They note that inferring feedback from a user’s behaviour is problematic because observed behaviour does not necessarily reflect a user’s intentions.

### 2.3.2 Computational Complexity of User Modelling

The effectiveness of a system that uses machine learning to infer a user model is strongly influenced by the computational complexity of the learning technique. For an adaptive user interface to be effective it needs to be responsive to the user. A computationally expensive user model learning technique can introduce a time-delay into an adaptive user interface, which interferes with the normal human-computer interaction. In many cases simple learning techniques outperform more sophisticated techniques both in terms of accuracy and efficiency. This is illustrated in Davison and Hirsh’s [24] command-line prediction algorithm. Their IPAM algorithm, which predicts command-line actions based on the last command-line entered, is efficient and simple, and provides higher predictive accuracy than the more complex C4.5 [92] decision-tree approach.

A further issue is the comprehensibility of the decisions an adaptive interface makes. Some machine learning techniques use knowledge representations that are incomprehensible to a user. For instance, the LUMIERE [47] project represents its knowledge via a Bayesian network, which can be difficult to represent in a concise and user-understandable way. Alternatively, Lieberman’s Letizia [70] system uses a set of heuristics and as such can explain its reasoning process to a user in a straightforward manner.

### 2.3.3 Concept Drift

Over time users may change their preferences or requirements. If a machine learning technique is used to capture a user’s preferences or requirements then such changes represent concept drift for a learner. An early attempt to address concept drift was the STAGGER [100] system. The STAGGER system adjusted to changes in what was being learnt by revising its learnt concepts. The trigger for revision depended upon a measure of logical sufficiency and dependency between the learnt concept description and new
training instances. The time taken for such measures to change, meant that changes in a target concept could not be identified immediately by STAGGER.

The FLORA [62] learning system and its subsequent versions handled concept drift by selecting relevant instances on which to construct a target concept. The method for determining relevancy was based on selecting training instances bounded by a window into the past. The use of a windowing method can be seen in the original FLORA system and in the PECS [99] system. Adaptive window size methods have also been considered [59, 115]. Depending on the window size chosen, such learning systems have the potential to track rapid changes in a target concept. However, in some situations, such as the user interface domain presented here, target concepts will repeat themselves according to some possibly hidden context. It is therefore desirable in terms of efficiency for a learning system to be able to recognise and reuse recurring target concepts. Salganicoff’s PECS and the FLORA3 [116] and FLORA4 [113] learning systems can take advantage of recurring target concepts.

Another approach to concept drift was Widmer’s [114] METAL(B) and METAL(IB) systems. A two-level learner was used in these systems. The role of the meta-level learner was to learn to detect the contextual clues associated with a particular target concept and to focus the base-level learner accordingly. Identifying regions in which a concept remains stable was also attempted by Harries, Sammut, and Horn in their SPLICE [41] system. Given training examples provided in batch, contextual clustering was used to identify the clusters in which a concept remained stable.
Chapter 3

Predictive Menu on a Mobile Phone

3.1 Introduction

The first part of this chapter details the benefit of an adaptive user interface (AUI) on a mobile phone. In particular, we focus on adaptive menus on mobile phones. An adaptive menu predicts a user's menu selections. Several learning approaches for predicting these selections are presented. Also, a general discussion of the issues surrounding learning in the mobile phone environment is provided.

The second part of this chapter formalises the learning setting. Several learning approaches are detailed and their merits and shortcomings discussed. A novel learning approach is presented that is ideally suited to this learning setting. A theoretical analysis is provided of each learning approach. Real-world data is used to evaluate the effectiveness of these learning approaches. Also, the differences between the empirical results and the theoretical results are discussed.

3.2 Adaptive Menu-Based Interface

Mobile phones are often difficult to use because of their restricted physical size. A restricted physical size means that mobile phones are limited in the
possible input and output facilities they can possess. For instance, many mobile phones have only a numeric keypad for user input. A full alphabetic keyboard or a full-sized positioning/pointing device would be impractical. In addition, physical size also dictates a mobile phone’s screen dimensions. For instance, the screens on mobile phones are often a fraction of the size and resolution of their desktop counterparts. The limited input and output facilities on mobile phones severely restricts their use.

The application of an adaptive user interface offers the potential to improve the useability of a mobile phone. The interface of a mobile phone can be adapted to suit its user’s needs. This adaptation could improve the efficiency and accuracy of a user’s interaction with the device. However, there are several issues that must be considered if an adaptive user interface is to be applied to a mobile phone. These include: can adaptivity increase the efficiency of a mobile phone’s interface; how should adaptivity be applied to a mobile phone’s interface; and what are the implications for a learning-based approach to adaptivity. These issues are now discussed in detail.

### 3.2.1 Interface Predictability and Efficiency

Interface efficiency is not the only requirement when designing a user interface. Another important requirement is interface predictability. Schneiderman [103] highlights this point by advocating that interface stability is required for effective human-computer interaction. With regard to mobile computing devices, Lindholm and Keinonen [71] state that interface efficiency should not come at the cost of interface predictability. The importance of interface predictability arises from the fact that users need to know what effects their actions will have on an interface. If the effects of actions are known, users can plan which actions to take to achieve their goals. Ensuring that an interface is both predictable and efficient is a difficult task. Interface efficiency can be gained by enabling an interface to adapt. However, if users are not aware of how and why an interface adapts, they lose their ability to predict what effects their actions will have on an interface.

Two methods have been discussed for adapting an interface while at the same time maintaining its predictability. Weld et al. [112] describes an ap-
3.2. **ADAPTIVE MENU-BASED INTERFACE**

A new approach that separates the dynamic and static areas of an interface, called *Partitioned Dynamicity*. With this approach, static areas of an interface retain the functionality a user expects, while dynamic areas facilitate interface efficiency through adaptivity. Weld et al. cite the news section of the Yahoo web site as an example of partitioned dynamicity. The layout of the web page remains static, but the news stories are dynamic. Alternatively, McGrenere et al. [76] presents an approach that employs a dual interface. One interface is adaptable and is personalised by the user, while the other interface contains all the standard and default features. The user has a choice between the two interfaces. The adaptable interface contained only a subset of the features available in the default interface. For instance in the adaptable interface, drop-down menus contained only some of the menu items available in the default interface. To evaluate the dual interface approach, McGrenere et al. [76] asked a number of participants to use a dual interface version of Microsoft Word™. The following hypotheses were posed about the dual interface: that dynamic features would be used; that the concept of a dual interface would be easily understood by users; that users would be more satisfied with the dual interface than with the original interface; that users would find it easier to learn new features, and that users would find the dual-interface easier to navigate and control. Quantitatively, these hypotheses were validated by the majority of participants.

The approach described in this chapter does not separate the adaptive and static aspects of an interface. Instead, adaptivity is achieved by changing the default behaviour of menu artifacts in an interface. Instead of the first menu item being highlighted by default, a learning approach predicts which menu item the user is going to select. The menu item predicted by the learning approach is then highlighted. Predictability is maintained by keeping the content and ordering of menu items constant. Efficiency is achieved by predicting which menu item a user will select. Predicting which menu item a user will select reduces the number of scrolling key presses a user has to make. Furthermore, since highlighting is the only action taken, the cost of an incorrect prediction is minimal. An overview of the menu prediction approach is shown in Figure 3.1. In this simple overview, the learner is presented with attributes describing the current state of the interface. Based
on these attributes, the learner predicts which menu item the user will select and highlights it after the menu is opened.

Figure 3.1: Menu prediction on a mobile phone interface. Whenever a user opens a menu, the current state of the interface is sent to a learner. The learner then determines which menu item the user will select and highlights it.

To illustrate the possible concepts that could be learnt we now describe one possible scenario. Supposing a user wants to call a person whose phone number they do not have, they first obtain the person’s number, enter it into their mobile phone then call the number. These actions could be represented by a concept that states that whenever the user has entered a new number into their mobile phone, the next action they will take is to dial the number. The menu prediction approach would attempt to learn this concept and whenever the user has entered a new phone number, the Call menu item is highlighted.

3.2.2 Predictive Menu Issues

Computational Overhead

Any learning approach that predicts menu selections on a mobile phone must be computationally efficient. Computational efficiency is required for two reasons. Firstly, mobile phones have limited memory and processing capacity. Secondly, the responsiveness of a mobile phone’s interface is dictated by the computational burden placed on the mobile phone. Maintaining interface responsiveness is critical, especially given that it is often the most significant factor in determining user productivity [31].
3.2. **ADAPTIVE MENU-BASED INTERFACE**

**Limited Training Examples**

In the menu prediction setting, the user will provide only a limited number of menu selections as examples to the learner. If a learner is to be useful in this setting, it must be able to learn effectively from relatively few examples.

**Comprehensibility**

User comprehensibility is a desirable characteristic of a learner in the menu prediction setting. For a learner to be comprehensible, it must be able to express its decision-making process in a way that a user understands. The importance of comprehensibility is that if a user understands the reasoning behind a learner’s menu prediction then the predictability of the mobile phone’s interface is maintained. Although user comprehensibility is a desirable characteristic, it may not be feasible to convey this information given a mobile phone’s restricted screen size and resolution.

**Concept Drift**

Another concern is how the problem of concept drift is addressed in the menu prediction setting. Generally, concept drift occurs when the concept being learnt by the learner changes. Addressing concept drift is especially important in the menu prediction setting. For instance, a learner must induce a concept that describes a user’s menu selections. However, many factors will change the way a user makes menu selections. The user may become more adept at using the menu, or a different user altogether may interact with the menu. In either case, the concept describing a user’s menu selections will change over time.

To address concept drift in this setting, the learner must induce *local* concepts. The learner must be able to recognise when a local concept is no longer relevant and induce a new local concept (i.e. identify when a concept changes, and then relearn it).
3.2.3 Addressing Predictive Menu Issues

The learning approach presented in this chapter relies upon two strategies to address the predictive menu issues introduced in the previous section.

The first strategy is the use of a highly restricted and tailored hypothesis space. Restricting the representational form of the hypothesis space saves the learner from searching an extensive hypothesis space, while tailoring the hypothesis space allows only likely hypotheses to be represented. Also, tailoring the hypothesis space allows the learner to take advantage of prior knowledge of the problem domain.

The second strategy is to take advantage of the characteristics of concept drift. One characteristic is that concepts are often locally stable; that is, concepts can be assumed to stay the same for at least some period of time. Given this assumption the learning approach can apply a strategy that uses only the most recent examples of a user’s menu selections to determine the current concept. Another characteristic of concept drift is that concepts often recur. That is, concepts that were previously relevant can be assumed to become relevant again in the future. Given this assumption, previously learnt concepts can be stored and checked to see if they have become relevant again, saving the learner from having to induce the concept “from scratch”.

Together, a highly restricted and tailored hypothesis space and recurring concepts reduce the menu prediction problem to one of tracking which concept is relevant in the given context. The issues surrounding menu selection prediction on a mobile phone are discussed more formally in the following sections.

3.3 The Learning Setting

We now describe formally the role of learning in menu prediction. The notation used throughout this chapter is presented in Table 3.1.

In order for a learner to predict which menu item the user will select, certain attributes of the learner’s environment must be predictive of the menu selection. In the mobile phone setting it was assumed that a set of attributes describing the user’s actions and the interface’s current state would
3.3. THE LEARNING SETTING

Table 3.1: Notation needed for describing the menu prediction learning setting

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition/Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{X}$</td>
<td>the set of <em>examples</em>, where each example is represented by $m$ attributes.</td>
</tr>
<tr>
<td>$\mathcal{Y}$</td>
<td>the set of $n$ labels, ${1, \ldots, n}$, where each label represents the position or index of a menu item in a menu.</td>
</tr>
<tr>
<td>$\mathcal{A}$</td>
<td>the set of labelled <em>examples</em>, where $\mathcal{A} = \mathcal{X} \times \mathcal{Y}$.</td>
</tr>
<tr>
<td>$\mathcal{F}$</td>
<td>the set of functions of the form $f : \mathcal{X} \to \mathcal{Y}$.</td>
</tr>
<tr>
<td>$\mathcal{H}$</td>
<td>the set of hypotheses.</td>
</tr>
<tr>
<td>$\mathcal{C}$</td>
<td>the set of concepts.</td>
</tr>
<tr>
<td>$T$</td>
<td>the time index that parameterises $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{F}$.</td>
</tr>
</tbody>
</table>

be predictive of the menu selection. The values that these attributes take on represent an example to the learner.

On every menu selection made by the user, the learner receives an example, $x_t \in \mathcal{X}$, and the menu item selected by the user, $y_t \in \mathcal{Y}$. Note, the subscript $t$ is used as the index for time parameterisation, whereas, $i$ and $j$ are used to denote non-time indexes. Together, the pair $(x_t, y_t)$ form a labelled example at time $t$. After $t + 1$ menu selections the learner will have observed a sequence of labelled examples called a *sample*, i.e. $\text{sample}_t = ((x_0, y_0), (x_1, y_1), \ldots, (x_t, y_t))$.

The objective of a learning system is to predict which menu item to highlight $y_t \in \mathcal{Y}$ when given some example $x_t \in \mathcal{X}$ and a sample up until time $t$, $\text{sample}_{t-1}$. This is achieved by discovering a function $f_t : \mathcal{X} \to \mathcal{Y}$ based on the $\text{sample}_{t-1}$ that is then used to predict the menu selection at time $t$. If the prediction is successful, then $y_t = f_t(x_t)$.

It is also assumed that the attributes describing an example would be discretely valued and that the values each attribute could take would be relatively small and known *a priori*, ensuring that the set $\mathcal{X}$ is finite. Given that both the set $\mathcal{X}$ and $\mathcal{Y}$ are finite, the set of functions $\mathcal{F}$ is also finite. However, in the menu prediction setting not all functions are considered equally likely to occur. Moreover, some functions in $\mathcal{F}$ represent highly unlikely mappings. To take advantage of this prior knowledge within the problem domain, only those functions in $\mathcal{F}$ that were considered likely to occur were provided to
the learner. These likely functions represent a hypothesis space $\mathcal{H}$ (where $\mathcal{H} \subset \mathcal{F}$) which is provided to the learner.

Two characteristics dictated the representational scheme used to specify hypotheses. Firstly, hypotheses needed to be amenable to efficient computational evaluation, and secondly, it was desirable that they be comprehensible to a user. Representing hypotheses as decision lists accommodates both requirements. The form of decision list chosen is shown below.

"if $l_1$ then $y_1$, else if $l_2$ then $y_2$, ..., else $y_n$" where each $l_i$ is a CNF expression and each $y_i \in \mathcal{Y}$.

The decision list representation also ensured that each hypothesis was a function.

If a hypothesis $h_t \in \mathcal{H}$ correctly predicts $y_t \in \mathcal{Y}$ when evaluated on $x_t \in \mathcal{X}$ then $h_t$ is said to agree with the labelled example $(x_t, y_t)$. If a hypothesis $h_t \in \mathcal{H}$ predicts incorrectly, then $h_t$ is said to disagree with the labelled example $(x_t, y_t)$. A hypothesis is considered consistent if it agrees with all labelled examples in a sample.

### 3.3.1 Loss Model

When the learner selects a hypothesis that disagrees with an example, the learner has in effect highlighted a menu item the user does not want to select. The user must then scroll to the correct menu item. The cost associated with predicting the wrong menu item can be measured in a number of ways. One way is to determine the number of scrolling key presses a wrongly highlighted menu item would cause the user. If each $y_t \in \mathcal{Y}$ is an index into a list of menu items, then an absolute loss function captures this measurement, i.e.

$$L(h_t(x_t), y_t) = |h_t(x_t) - y_t|. \quad (3.1)$$

The absolute loss function allows the notion that a wrongly highlighted menu item located further away from the intended menu item results in a larger cost to the user. However, if the user is only concerned as to whether the correct menu item was highlighted, then the discrete loss function is suitable, i.e:
3.3. THE LEARNING SETTING

\[
L(h_t(x_t), y_t) = \begin{cases} 
0, & \text{if } h_t(x_t) = y_t \\
1, & \text{otherwise.} 
\end{cases} \tag{3.2}
\]

A discrete loss function also allows the menu-item prediction problem to be viewed as a binary-concept learning problem. The labelled example \((x_t, y_t)\) can be viewed as a possible “course-of-action” for the learner. From this perspective, the discrete loss function determines whether a “course-of-action” is appropriate. Given some hypothesis \(h_t\) and some “course-of-action” \((x_t, y_t)\), the discrete loss function defines a boolean-valued concept, i.e. “\((x_t, y_t)\) is an appropriate course-of-action given \(h_t\)”. More formally, the discrete loss function is a characteristic function on each \(h_t \in \mathcal{H}\). We let a concept \(c_t\) be the extension of a hypothesis \(h_t\), i.e \(c_t = \text{ext}(h_t)\) and let \(\mathcal{C}\) be the set of possible concepts.

An assumption was made that any observed “course-of-action” \((x_t, y_t)\) agreed with at least one hypothesis. In other words, for any \((x_t, y_t)\) observed, there existed a hypothesis such that \(h_t(x_t) = y_t\). Therefore, the set of possible concepts was restricted such that \(\mathcal{C} \subseteq \mathcal{H}\).

In the setting just described, the learning problem becomes one in which the learner must determine which “course-of-action” is appropriate (appropriate being defined as a “course-of-action” that produces a loss of 0). For the learner to determine which “course-of-action” is appropriate, it needs to know which \(h \in \mathcal{H}\) is being used to assess “appropriateness”. In other words, the learner needs to find the hypothesis which captures the concept of “appropriateness”. This concept is called the target concept and is denoted as \(c^*\).

3.3.2 On-line Model

For menu-item prediction to be useful it needs to operate every time a user opens a menu in an interface. As such, the learning approach for menu-item prediction must operate over a series of trials. In this regard, the learning approach for menu-item prediction operates in an on-line and supervised setting. A general model of this type of on-line and supervised prediction setting is provided by Littlestone [72].
The objective of the learner in this on-line and supervised prediction setting is to reduce overall loss by improving its predictive accuracy after each trial. Since accuracy relies on finding the target concept, the learner must find the hypothesis that captures the target concept after as few trials as possible.

### 3.3.3 Concept-drift Model

If a hypothesis agrees with every example in a sample (a consistent hypothesis), then the hypothesis provides an accurate mapping between some set of attribute values and the menu item a user will select in the sample. In this case, the hypothesis captures the target concept in the sample. In the menu prediction setting, the target concept can be considered a model of how a user interacts with a menu. Finding a hypothesis that captures the target concept is similar to determining how a user interacts with a menu.

It is conceivable that the model by which a user interacts with a menu may change over time. There are a number of reason why such a model may change: a different person may become the user, a user may become more adept at using the interface, the user may operate the device in a different way, or the environment may not provide enough information for which to build an accurate model in the first place. The result of a changing model of user interaction is a changing target concept. For instance, at some trial $t$ the user selects menu item $y_t$ given the attribute values $x_t$. However, at some later trial $t + d$, the user selects a different menu item $y_{t+d} \neq y_t$ when given the exact same attribute values $x_{t+d} = x_t$ (the target concept thus changed at some point from trial $t$ to trial $t + d$).

Although there may not be a single global target concept, it may be that a set of local concepts can be composed to form a global target concept, where each local concept is only relevant for a given period of time.

In the menu prediction setting it was assumed that the context was defined by the task the user was performing. For instance, while the user was performing one task their interactions could be captured by the hypothesis $h_i$. When the user changed tasks their interactions could then be captured by a different hypothesis $h_j$, where $i \neq j$. As a result the target concept
3.3. **THE LEARNING SETTING**

would change whenever the user changed tasks.

Figure 3.2(a) shows two local concepts, each represented by a different hypothesis. Each concept becomes the target concept only in a given context. It was assumed that the context was determined by some attribute not present in the example space. Figure 3.2(b) shows concept drift in an on-line setting. In this case, the context is inherently time dependent.

![Diagram](image)

Figure 3.2: (a) Two local concepts, hypothesis \( h_i \) is the target concept in one context and hypothesis \( h_j \) the target concept in another context. (b) Concept drift in an on-line setting, in which the context is time dependent.

We define a simple model to describe how the target concept changes. We assume that at any trial \( t \) there is target concept \( c_t \). We assume that with probability \( \lambda \) the target concept changes from one trial to the next, whereas with probability \( 1 - \lambda \) we assume the target concept does not change. When the target concept does change, the current target concept cannot be reselected, although it may be reselected in latter trials. Finally, it is assumed that \( \lambda \) remains stationary over all trials.

If concept drift occurs in an on-line learning model, the learner has to find the hypothesis that captures the target concept on the current trial. The learner has available to it only previously observed examples.
3.4 Menu Prediction Approaches

A number of learning approaches could be conceived for the menu prediction setting. A description of seven approaches, together with a discussion of their theoretical properties is now provided.

3.4.1 First Menu-Item

The First Menu-Item approach does not alter the default ordering of menu items. The first menu item is always highlighted. The First Menu-Item approach is formally described in Algorithm 3.4.1. This approach represents the way in which most menus operate and is important because it serves as a basis for comparing other menu prediction approaches.

**Algorithm 3.4.1 First Menu-Item algorithm**

1: Given: a list of \( n \) menu items \((1, 2, ..., n)\).
2: In each trial \( t \geq 1 \):
3: predict menu item \( \hat{y}_t = 1 \).

If we assume that there is some distribution over the menu items that are selected in any trial sequence, and that the ordering of menu items is random, then each menu item is equally likely to be the first menu item. The probability of selecting a menu item other than the first menu item is our expected discrete loss and is given by:

\[
\frac{n - 1}{n}.
\]

(3.3)

The expected absolute loss is:

\[
\frac{n - 1}{2}.
\]

(3.4)

As expected such a system would produce a “poor” result. Clearly there is opportunity for improvement.
3.4.2 First Menu-Item (frequency ordered)

The First Menu-Item (frequency ordered) approach assumes a fixed non-arbitrary ordering of menu items. Menu items are ordered according to their likelihood of being selected. The menu item most likely to be selected appears at the top of the menu, while the least likely to be selected appears at the bottom of the menu. This approach assumes that the selection likelihood of menu items is known \textit{a priori} and does not change. The first menu item is always highlighted. The First Menu-Item (frequency ordered) approach determines whether a different ordering of menu items (other than the default one specified by the interface designer) would be more appropriate for a particular user.

Algorithm 3.4.2 First Menu-Item (frequency ordered) algorithm

1: Given: a list of \( n \) menu items (1, 2, ..., \( n \)), ordered according to their selection likelihood.

2: In each trial \( t \geq 1 \):

3: predict menu item \( \hat{y}_t = 1 \).

The First Menu-Item (frequency ordered) approach assumed that the selection frequency of menu items was known \textit{a priori}, and that their ordering was based on this selection frequency.

Some probability theory formalisms are now introduced to help understand the effectiveness of this approach. These are also used in later sections.

We define a probability space \((\Omega, P)\), where \( \Omega = \mathcal{Y} \). We assume the probability measure \( P \) is known \textit{a priori} and can be defined by a probability mass function:

\[
    f(i) = \begin{cases} 
        p_i, & \text{if } i \in \mathcal{Y} \\
        0, & \text{otherwise}
    \end{cases}
\]

(3.5)

where \( p_i \) is the probability menu item \( i \) is selected. Note, that \( \sum_{i=1}^{n} p_i = 1 \).

From this we can calculate the expected discrete loss:

\[
    \sum_{i=2}^{n} p_i.
\]

(3.6)
Given the menu items are sorted by frequency we assume \( p_1 \geq p_2 \geq ... \geq p_n \). However, in the worst case, where \( p_1 = p_2 = ... = p_n = \frac{1}{n} \), the expected discrete loss would be \( \frac{n-1}{n} \). On the other hand in the best case, where \( p_1 = 1, p_2 = 0, ..., p_n = 0 \) the expected discrete loss would be 0.

The expected absolute loss can also be evaluated by:

\[
\sum_{i=1}^{n} (i-1)p_i.
\] (3.7)

If we assume the worst case, the expected absolute loss will be \( \frac{n-1}{2} \) and if we assume the best case the expected absolute loss will be 0. Notice that in the worst case, the expected discrete and absolute loss is the same as the First Menu-Item approach.

Of course, it is unlikely that the probability mass function \( f \) would be at either extreme. However, if we assumed that menu item selections were similar in nature to utterances in a natural language, then Zipf’s law could be used [118].

Informally, Zipf’s law is the observation that the rank of a menu item is inversely proportional to the fraction of times it is selected. For instance if a menu item’s relative selection frequency is \( \frac{1}{2} \), then its rank is 2; since \( \frac{1}{2}^{-1} = 2 \). Zipf’s law can be used to justify ranking menu items according to their frequency of use. More formally, Zipf’s law can be described by a power-law distribution. In particular, the zeta distribution [2].

Zipf’s law predicts that from a population of \( n \) elements, the relative frequency of element \( k \) is:

\[
f(k; s; n) = \frac{\frac{1}{k^s}}{\sum_{i=1}^{n} \frac{1}{i^s}}.
\] (3.8)

The classic version of Zipf’s law has \( s = 1 \) in which case the frequency probabilities become:

\[
f(k; n) = \frac{\frac{1}{k}}{\sum_{i=1}^{n} \frac{1}{i}}.
\] (3.9)

If we assume the selection frequency for menu items does follow this distribution and therefore \( p_i = f(i; n) \), then the expected discrete loss of the First Menu-Item (frequency ordered) approach would be:
3.4. MENU PREDICTION APPROACHES

\[
1 - \frac{1}{\sum_{i=1}^{n} \frac{1}{i}}.
\]  
(3.10)

So for example if there were 14 menu items \(^1\), \(n = 14\), then the expected discrete loss would be 0.692. This is considerably better than the worst case where the expected discrete loss would be 0.929.

The expected absolute loss would be:

\[
\frac{n}{\sum_{i=1}^{n} \frac{1}{i}} - 1.
\]  
(3.11)

Again, in the case of \(n = 14\) we would observe an expected absolute loss of 3.31, or put another way we would have to make on average 3.31 key presses to select the menu item we want. This is considerably better than the worst case of 6.5 key presses.

If Zipf’s law did apply to menu selections, then this means any learning system would need to perform well to do better than this simple frequency reordering approach.

### 3.4.3 Menu-Item Last Selected

The Menu-Item Last Selected does not alter the ordering of menu items, instead it alters the menu item that is highlighted. The Menu-Item Last Selected approach represents a relatively simplistic approach to menu-item prediction; it highlights the menu item that was last selected by the user. A description of this approach is given in Algorithm 3.4.3.

**Algorithm 3.4.3 Menu-Item Last Selected algorithm**

1: Given: a list of \(n\) menu items indexed by \((1, 2, ..., n)\).
2: In each trial \(t \geq 1\):
3: \quad predict the menu item \(\hat{y}_t = y_{t-1}\).
4: \quad receive and retain the correct menu item, \(y_t\).

If we let \(S\) be the probability that the user selects the same menu item

\(^1\)The Nokia\(^\text{TM}\) Series 60 user interface that we evaluate later in this Chapter has 14 menu items.
between trials, e.g. \( y_{t-1} = y_t \) at any given time \( t \), then the expected discrete loss for the *Menu-Item Last Selected* approach is just \( 1 - S \).

If the user selects menu items according to some distribution, then the expected discrete loss of this approach would be worst when the user selects from \( n \) menu items uniformly at random. e.g. \( p_1 = p_2 = \ldots = p_n = \frac{1}{n} \). In the case when a user selects a menu item uniformly at random, \( S = \frac{1}{n} \) and the expected discrete loss would be \( \frac{n-1}{n} \), because there is a \( 1 \) in \( n \) chance they selected the previous example. In the best case, where the user selects the same menu item \( i \) with certainty, \( p_i = 1 \), then the expected discrete loss would be \( 0 \). Note, we must also take into account the loss associated with making a mistake on the first trial.

To understand the expected absolute loss associated with the *Menu-Item Last Selected* we again consider the situation in which the user selects from \( n \) menu items uniformly at random. To obtain the final result we examine several particular cases.

If the first menu item is highlighted then the expected absolute loss is:

\[
\frac{1}{n} \sum_{i=1}^{n-1} i.
\]

If the second menu item is highlighted then the expected absolute loss is:

\[
\frac{1}{n} \left( \sum_{i=1}^{n-2} i + \sum_{i=1}^{n-1} i \right).
\]

Therefore, when the \( j \)-th menu item is highlighted the expected absolute loss is:

\[
\frac{1}{n} \left( \sum_{i=1}^{n-j} i + \sum_{i=1}^{j-1} i \right).
\]

The expected absolute loss when each menu item is selected uniformly at random is:

\[
\frac{1}{n} \sum_{j=1}^{n} \left( \frac{1}{n} \left( \sum_{i=1}^{n-j} i + \sum_{i=1}^{j-1} i \right) \right).
\] (3.12)
3.4. MENU PREDICTION APPROACHES

\[ = \frac{1}{n} \sum_{j=1}^{n} \left( \frac{1}{n} \left( \frac{(n-j)(1+n-j)}{2} + \frac{(j-1)(j)}{2} \right) \right) \]

\[ = \frac{1}{n^2} \sum_{j=1}^{n} \left( \frac{(n-j)(1+n-j)}{2} + \frac{(j-1)(j)}{2} \right) \]

\[ = \frac{1}{2n^2} \sum_{j=1}^{n} \left( (n + n^2) + j(-2n - 2) + 2j^2 \right) \]

\[ = \frac{1}{2n^2} \left( n(n + n^2) + \frac{-2n^3 - 4n^2 - 2n}{2} + \frac{4n^3 + 6n^2 + 2n}{6} \right) \]

\[ = \frac{n^2 - 1}{3n}. \]

However, the nature of this approach means that the user selecting menu items uniformly at random is not the worst case scenario for expected absolute loss. If the user selected menu items at opposing indexes in the menu with equal probability, e.g. \( p_1 = 0.5, p_2 = 0, \ldots, p_n = 0.5 \), then a higher expected absolute loss could be achieved. Furthermore, if the user did not select menu items according to some distribution but instead adopted an adversarial strategy, an absolute loss of \( n - 1 \) could be achieved on each trial.

3.4.4 Most Common Menu-Item

The Most Common Menu-Item approach does not alter the ordering of menu items. The menu item highlighted is the menu item selected most by the user up to that point. A description of this approach is given in Algorithm 3.4.4.

The Most Common Menu-Item approach is very similar in nature to the First Menu-Item (frequency ordered) approach. Both approaches predict the menu item that is selected with the highest relative frequency. As such, if the Most Common Menu-Item approach knows \textit{a priori} the menu item that will be selected most often, it will have the same expected discrete loss. Although similar in terms of discrete loss, the two approaches will differ in terms of expected absolute. Importantly, the Most Common Menu-Item approach does not attempt to minimise absolute loss by ordering menu items according to their relative frequency. However, for small \( n \) the Most Common Menu-Item approach is not at a great disadvantage, especially if the menu item
Algorithm 3.4.4 Most Common Menu-Item algorithm

1: Given: a list of menu items indexed by \( (1, 2, \ldots, n) \).
2: Associate a counter with each menu item, \( B = (b_1, b_2, \ldots, b_n) \).
3: Initialise these counters to zero.
4: In each trial \( t \geq 1 \):
5: predict the menu item associated with highest valued counter.
6: \( \hat{y}_t = \text{argmax}_{i \in \{1,\ldots,n\}} b_i \uparrow \).
7: receive the correct menu item, \( y_t \).
8: update the counter associated with the correct menu item:
9: \( b_{y_t} \leftarrow b_{y_t} + 1 \).

\( \uparrow \) ties are resolved by choosing the menu item with the lowest index.

highlighted is towards the centre of the menu. For instance, if the middle menu item is highlighted, we can be assured a maximum absolute loss of \( \frac{n^2}{2} \).

The major difference between the Most Common Menu-Item approach and the First Menu-Item (frequency ordered) approach is that the former does not know the relative selection frequency of menu items \textit{a priori}. Instead, the Most Common Menu-Item approach uses observed menu selections to approximate the actual relative selection frequency of menu items.

The disadvantage of all the approaches introduced so far is that the strategy used to predict the menu item is fixed. As a consequence we are assuming the target concept \( c^* \) is also fixed. Furthermore, we are assuming that the target concept can be captured by a relatively simple strategy. In the real world it is unlikely that both these assumptions hold. However, an advantage of the approaches introduced so far is their relatively small computational requirements. Furthermore, it may be that such simple strategies will be accurate enough to produce a good result.

### 3.4.5 Most Common Hypothesis

The menu prediction approaches that we are about to introduce differ from those presented in that they utilise hypotheses. In this setting, hypotheses form functions that map some previous observed parameters to a particular menu item. The ability of a hypothesis to capture the context in which a
3.4. MENU PREDICTION APPROACHES

user selects a menu item is its major advantage in prediction. For instance, a user may select a particular menu item with high certainty in some specific context. The context could be defined by any number of things; such as the time-of-day; the previous menu items selected; or some external event. Hypotheses allow us to capture information about the user, and use it to make better predictions. However, the performance of a hypothesis-based approach depends on there being a sufficient number of hypotheses that approximate the actual behaviour of the user which can be found and applied in a timely manner.

The Most Common Hypothesis approach highlights the menu item according to a hypothesis. The hypothesis chosen is the one that has been consistent with the most menu-selection examples made by the user. This hypothesis is evaluated to predict which menu item to highlight. A description of this approach is given in Algorithm 3.4.5.

**Algorithm 3.4.5 Most Common Hypothesis algorithm**

1. Given: a list of menu items indexed by (1, 2, ..., n).
2. a set of hypotheses, \( \mathcal{H} \).
3. Associate a counter for each of the hypotheses in \( \mathcal{H} \), \( B = (b_1, b_2, ..., b_{|\mathcal{H}|}) \).
4. Initialise these counters to zero.
5. In each trial \( t \geq 1 \):
6. receive a state of the environment \( x_t \).
7. obtain a set of menu-item predictions \( \{h_1(x_t), h_2(x_t), ..., h_{|\mathcal{H}|}(x_t)\} \), by evaluating each hypothesis in \( \mathcal{H} \).
8. predict the menu item using the hypothesis with the highest valued counter:
\[
\hat{y}_t = h_{\text{argmax}_{i \in \{1,...,|\mathcal{H}|\}} b_i(x_t)} \dagger.
\]
9. receive the correct prediction, \( y_t \).
10. update the counters associated with the correct hypotheses:
\[
b_i \leftarrow \begin{cases} 
  b_i + 1, & \text{if } h_i(x_t) = y_t \\
  b_i, & \text{otherwise}.
\end{cases}
\]

\( \dagger \) ties are resolved by choosing the menu item with the lowest index.

Hypotheses are evaluated in terms of how much they disagree with a sample of user behaviour. After \( t \) trials, a sample of user behaviour is given
as a sequence of examples \(((x_0, y_0), \ldots, (x_{t-1}, y_{t-1}))\). The disagreement of a hypothesis \(h\) can be determined as follows\(^2\):

\[
disagreement(h, ((x_0, y_0), \ldots, (x_{t-1}, y_{t-1}))) = \sum_{i=0}^{t-1} L(h(x_i), y_i),
\]

where \(L\) is the discrete loss function (3.2). In other words, disagreement corresponds to the number of times the wrong menu item was predicted.

Given this definition of disagreement we can formalise the objective of the Most Common Hypothesis approach as finding the hypothesis that minimises disagreement over a sample of user behaviour:

\[
\arg\min_{h \in \mathcal{H}} \, disagreement(h, ((x_0, y_0), \ldots, (x_{t-1}, y_{t-1}))).
\]

### 3.4.6 Fixed Window

The Fixed Window approach does not alter the ordering of menu items. The menu item highlighted is determined by the hypothesis that has been consistent with the most recent menu-selection examples made by the user. Menu selection examples are added to the window; if the window is full, the oldest menu selection example is removed. The hypothesis consistent with the most menu selection examples in the window is evaluated to predict which menu item to highlight. The window size must be given as a parameter into this approach. This approach is formally described in Algorithm 3.4.6.

Hypotheses are evaluated in terms of how much they disagree with a fixed size sample of user behaviour. The sample size is fixed, since only a certain number of recent examples are included in the sample. The key advantage of the Fixed Window approach is the use of a fixed size sample. The benefit of restricting a sample to only a certain number of recent examples becomes apparent if the concept being learnt changes. In the presence of concept drift, the examples being observed cannot be guaranteed to have been generated by the one concept. However, if we assume that more recent examples are

\(^2\)Given a highly restricted hypothesis space, it is feasible to determine the disagreement of each hypothesis with respect to a given sample.
Algorithm 3.4.6 Fixed Window algorithm

1: Given: a list of menu items indexed by $(1, 2, ..., n)$.
2: a set of hypotheses, $\mathcal{H}$.
3: a window $w$ that holds a fixed size array of examples, $w[0], ..., w[i]$.
4: Associate a counter for each of the hypotheses in $\mathcal{H}$, $B = (b_1, b_2, ..., b_{|\mathcal{H}|})$.
5: Initialise these counters to zero.
6: In each trial $t \geq 1$:
7: receive a state of the environment $x_t$.
8: obtain a set of menu-item predictions $\{h_1(x_t), h_2(x_t), ..., h_{|\mathcal{H}|}(x_t)\}$, by evaluating each hypothesis in $\mathcal{H}$.
9: predict the menu item using the hypothesis with the highest valued counter:
10: $\hat{y}_t = h_{\arg\max_{i \in \{1, \ldots, |\mathcal{H}|\}} b_i(x_t)}\dagger$.
11: receive the correct prediction, $y_t$.
12: update the window array accordingly:
13: add $(x_t, y_t)$ to element $w[ t \mod i ]$ of $w$ array.
14: initialise the counters associated with each hypothesis to zero.
15: update hypothesis counters accordingly
16: for each $(x, y)$ in $w$
17: $b_i \leftarrow b_i + 1$, if $h_i(x) = y$

$\dagger$ ties are resolved by choosing the menu item with the lowest index.
more likely to have been generated by the current concept, then we should remove older examples from consideration. One approach of forgetting older examples is to use a fixed size window of examples. If the size of the window is set correctly, then the Fixed Window approach is still able to identify the current concept, even in the presence of concept drift.

After \( t \) trials a fixed sample of user behaviour is given as a sequence of examples \( \{(x_{\max\{0,t-w\}}, y_{\max\{0,t-w\}}), \ldots, (x_{t-1}, y_{t-1})\} \), where \( w \) is the size of the window. Similarly to the Most Common Hypothesis approach the disagreement of all hypotheses is determined. The disagreement of a hypothesis \( h \) can be determined as follows:

\[
disagreement(h, \{(x_{\max\{0,t-w\}}, y_{\max\{0,t-w\}}), \ldots, (x_{t-1}, y_{t-1})\}) = \sum_{i=\max\{0,t-w\}}^{t-1} L(h(x_i), y_i),
\]

where \( L \) is the discrete loss function (3.2).

The hypothesis that \textit{minimises disagreement} for this fixed window approach, over the fixed sample of user behaviour is selected, i.e:

\[
\arg\min_{h \in \mathcal{H}} \disagreement(h, \{(x_{\max\{0,t-w\}}, y_{\max\{0,t-w\}}), \ldots, (x_{t-1}, y_{t-1})\}).
\]

3.4.7 Most Recent Correct Hypothesis

The Most Recent Correct Hypothesis approach does not alter the ordering of menu items. The menu item highlighted is determined by the hypothesis that has been consistent with the most recent menu-selection examples made by the user. Instead of a fixed size window of examples, the hypothesis that is consistent for the longest run of menu selection examples is chosen. This hypothesis is then evaluated to predict which menu item to highlight. A description of this approach is given in Algorithm 3.4.7.

The Most Recent Correct Hypothesis approach addresses the problem of concept drift through the strategy it uses for evaluating hypotheses. More precisely, hypotheses are evaluated against all previous examples, and the
3.4. MENU PREDICTION APPROACHES

Algorithm 3.4.7 Most Recent Correct Hypothesis algorithm

1: Given: a list of menu items indexed by \( (1, 2, \ldots, n) \).
2: a set of hypotheses, \( \mathcal{H} \).
3: Associate a counter for each of the hypotheses in \( \mathcal{H} \), \( B = (b_1, b_2, \ldots, b_{|\mathcal{H}|}) \).
4: Initialise these counters to zero.
5: In each trial \( t \geq 1 \):
6: receive a state of the environment \( x_t \).
7: obtain a set of menu-item predictions \( \{h_1(x_t), h_2(x_t), \ldots, h_{|\mathcal{H}|}(x_t)\} \), by evaluating each hypothesis in \( \mathcal{H} \).
8: predict the menu item using the hypothesis with the highest valued counter:
9: \[ \hat{y}_t = h_{\arg\max_{i \in \{1, \ldots, |\mathcal{H}|\}} b_i(x_t)} \]
10: receive the correct prediction, \( y_t \).
11: update each counter accordingly:
12: \[ b_i \leftarrow \begin{cases} b_i + 1, & \text{if } h_i(x_t) = y_t \\ 0, & \text{otherwise.} \end{cases} \]

\( ^\dagger \) ties are resolved by choosing the menu item with the lowest index.

A hypothesis that has no disagreement for the longest consecutive number of past examples is selected. The assumption behind this approach is that not all examples should be used to evaluate hypotheses. Instead, only recent examples should be considered, in particular recent examples that meet the criteria that they are all consistent with some hypothesis.

The Most Recent Correct Hypothesis algorithm represents a novel approach for addressing concept drift. As presented in Section 3.10, many of the techniques for dealing with concept drift have employed a strategy of favouring more recent examples over older ones. This has been accomplished by either forgetting examples older than a fixed period, similar to our Fixed Window approach, or by discounting examples relative to their age. The Most Recent Correct Hypothesis approach employs neither of these strategies. Instead, it uses the hypotheses themselves to restrict the examples under consideration.

Naively, the Most Recent Correct Hypothesis algorithm requires that every hypothesis be evaluated against all previous examples on every iteration.
CHAPTER 3. PREDICTIVE MENU ON A MOBILE PHONE

However, a more efficient implementation requires only that a small amount of information regarding the state of each hypothesis is maintained between iterations. For example, Algorithm 3.4.7 implements the algorithm efficiently by associating a counter with each hypothesis. A counter records the number of consecutive examples the hypothesis has been consistent.

**Hypothesis Selection**

The fundamental difference between the *Most Common Hypothesis* approach, the *Fixed Window* approach, and the *Most Recent Correct Hypothesis* approach is the method they use for determining which previous examples of user behaviour to include in a sample. After \( t \) trials, a user’s observed behaviour is given as a sequence of examples \(((x_0, y_0), \ldots, (x_{t-1}, y_{t-1}))\)

1. The *Most Common Hypothesis* approach selects a hypothesis by minimising disagreement over all examples.

2. The *Fixed Window* approach selects a hypothesis by minimising disagreement over a recent fixed period of observed user behaviour. (i.e. the hypothesis that has minimal disagreement over the last \( w \) examples.)

3. The *Most Recent Correct Hypothesis* approach selects a hypothesis by minimising disagreement over some recent varying period of observed user behaviour. (i.e. the hypothesis that has no disagreement over the longest run of examples into the past.)

We now discuss the effect that different sampling methods have when minimising disagreement.

### 3.5 Minimising Disagreement

The objective of minimising disagreement is to determine a hypothesis that performs well when predicting a user’s future behaviour. If we let \( D_X(x) \) denote the frequency that a certain \( x \) is observed, the performance of a hypothesis can be evaluated as follows:
3.5. MINIMISING DISAGREEMENT

$$\text{error}_{c^*}(h) = \sum_{x \in X} D_X(x)L(h(x), c(x)).$$

Using the above definition, two types of hypotheses can be defined:

An $\epsilon$-bad hypothesis is defined as:

$$\text{error}_{c^*}(\epsilon\text{-bad hypothesis}) > \epsilon.$$  

And an $\epsilon$-good hypothesis is defined alternatively as:

$$\text{error}_{c^*}(\epsilon\text{-good hypothesis}) \leq \epsilon.$$  

In both definitions, the error term $\epsilon$ represents the degree of error between the hypothesis and the actual target concept. The use of the error term allows us to define the difference between a “good” and “bad” hypothesis.

Assuming a finite hypothesis space, we can bound the probability that an $\epsilon$-bad hypothesis is consistent (agrees) with a sample of size $m$. It can be shown [80] that the probability that at least one $\epsilon$-bad hypothesis is consistent with all examples in the sample $m$, is at most:

$$|\mathcal{H}| e^{-\epsilon m}. $$

Therefore, if $|\mathcal{H}|$ remains constant and the target concept $c^*$ is fixed, increasing the amount of user behaviour available to the learner (i.e. increasing the sample size $m$), means it is less probable that at least one $\epsilon$-bad hypothesis is consistent with the sample. In other words, the larger the sample size $m$, the greater the chance of finding only an $\epsilon$-good hypothesis.

It would seem that this reasoning favours the Most Common Hypothesis approach as it uses all of a user’s observed behaviour (i.e. it uses the largest sample size of the three approaches). However, it is also assumed that the learners are situated in an environment that exhibits concept switching. Concept switching implies that a user’s behaviour cannot be explained by just one hypothesis. We now describe how concept switching affects the process of minimising disagreement.
3.5.1 Minimising Disagreement and Concept Switches

Concept switching poses the following problems to a learner that minimises disagreement:

1. If the target concept is allowed to switch, then it is likely that sections of a user’s past behaviour will be produced according to differing target concepts. Minimising disagreement over all observed behaviour will, over time, produce the hypothesis representing the target concept that was most prevalent in the sample. The hypothesis identified, and the target concept it represents, may not be descriptive of the user’s current desired behaviour.

2. Supposing a learner can identify a sample of user behaviour that is generated by the current target concept, for instance using only recent user behaviour. Minimising disagreement over this sample will, over time, produce the hypothesis representing the current target concept. However, whenever a context switch occurs the learnt hypothesis becomes obsolete. The situation is made worse by the fact that the learner does not know when a concept switch has occurred.

The first problem might be naively avoided by restricting the amount of observed behaviour available to the learner. For instance, sampling only recent examples, as intuitively this behaviour is more likely to have been generated by the current target concept. However, there are two problems with this naive approach. Firstly, the learner will still need a sample large enough to eliminate all $\epsilon$-bad hypotheses. Secondly, a learner cannot predict when a concept switch will occur, therefore even if an $\epsilon$-good hypothesis is found, it may be rendered obsolete by a concept switch.

Ideally, a solution to both these problems would be to ensure that the sample size is large enough to eliminate all $\epsilon$-bad hypotheses, while at the same time ensuring that it is not so large as to include examples produced by a previous target concept. Figure 3.3 illustrates such a sampling strategy.
3.5. MINIMISING DISAGREEMENT

Figure 3.3: Solution: minimise disagreement over sample large enough to eliminate \( \epsilon \)-bad hypotheses, but small enough to exclude examples from previous target concepts.

### 3.5.2 Hypothetical Construct - Being-the-Leader

To achieve the ideal solution we consider a hypothetical version of the three learning approaches. The hypothetical version of the problem domain is called *Being-the-Leader*. The *Being-the-Leader* versions of the three learning approaches know what menu item the user will select before the user selects it. Using the on-line model presented earlier, the *Being-the-Leader* versions operate as follows:

The learner is presented with a series of trials \( T = \{0, 1, 2, \ldots, n\} \). In each trial \( t \in T \):

1. The learner observes some state of the environment, \( x_t \)
2. The learner knows which menu item will be selected, \( y_t \)
3. The learner stores the observed example, \( (x_t, y_t) \)
4. The learner uses some subset of examples \( ((x_0, y_0), \ldots, (x_t, y_t)) \) to predict which menu item will be selected \( \hat{y}_t \)
5. The learner determines the loss associated with its prediction:

\[
L(y_t, \hat{y}_t) = \begin{cases} 
0, & \text{if } y_t = \hat{y}_t \\
1, & \text{otherwise.}
\end{cases}
\]

\(^3\)If *Being-the-Leader* were feasible, an obvious solution would be to use a sample that consisted of just \( (x_t, y_t) \) to minimise disagreement. Although this approach would determine a hypothesis that would agree with the menu selection (example) on the current trial, if the hypothesis were used on any other future example its predictive accuracy would be poor, since a sample of size of at least \( m \) is needed to find a hypothesis with a probable \( \epsilon \)-good performance.
CHAPTER 3. PREDICTIVE MENU ON A MOBILE PHONE

The name Being-the-Leader comes about as follows: if a learner chooses the hypothesis $h$ that has minimised disagreement on some observed examples, including the current example, then $h$ is the leader up to that point.

Alternatively, the original versions of three learning approaches can be considered to Follow-the-Leader, since they do not know the current example, they can only attempt to follow the leader $h$.

Figure 3.4 shows the difference between Follow-the-Leader and Being-the-Leader.

![Figure 3.4: Follow-the-Leader vs. Being-the-Leader. The Follow-the-Leader has available examples $((x_0, y_0), ..., (x_6, y_6))$ to use to predict $y_7$. The Being-the-Leader has available examples $((x_0, y_0), ..., (x_7, y_7))$ to use to predict $y_7$.](image)

In an on-line setting it is impossible to know for certain which menu item a user will select. As such, Being-the-Leader is impossible to implement. We now give two reason for considering the hypothetical Being-the-Leader learners:

1. The difference in loss between Following-the-Leader and Being-the-Leader is the same for each of the three learning approaches. Given that the difference in loss between Following-the-Leader and Being-the-Leader can be bounded, then we can analyse the Being-the-Leader versions of the three learning approaches by considering how they handle the second problem of concept switching, i.e. how to exclude examples that have originated from a previous target concepts.

2. The Being-the-Leader learners can be guaranteed that the example $(x_t, y_t)$ comes from the current target concept. Using the definition of $\lambda$ in Section 3.3.3, the second last example $(x_{t-1}, y_{t-1})$ comes from
the target concept with probability $1 - \lambda$ (since $1 - \lambda$ represents the probability that the target concept did not switch between trials). If we apply this probabilistic approach to all examples in a learner’s sample we can determine the probability that the examples in a learner’s sample are produced from the current target concept. Furthermore, we can determine a sampling strategy that is in some sense optimal in excluding examples that are produced from a previous target concept.

In the next section we analyse how concept drift affects the examples observed by the learner. In particular, how the characteristics of concept drift can be used to determine the proportion of examples in a sample that were produced by the current target concept. The Being-the-Leader hypothetical version of the problem domain is used for ease of analysis. We show how the Being-the-Leader versions of the Most Common Hypothesis, the Fixed Window and the Most Recent Correct Hypothesis approaches differ in relation to the examples over which they choose to minimise disagreement.

**Being-The-Leader with Increasing Window Size**

In the previous section it was shown that increasing the sample size provided to a learner improves the probability of finding a $\epsilon$-good hypothesis. However, if the sample size is increased by widening the window size of a learner, the greater the likelihood that examples from previous target concepts will be included. Including examples from previous concepts to a sample will introduce an effect similar to classification noise$^4$. Figure 3.5 shows examples, going backwards in time, for a Being-the-Leader learner.

Using the model of concept switching introduced early, going back in time, the probability that a concept switch occurred from trial $t$ to $t - 1$ is given by $\lambda$. Conversely, the probability that a concept switch did not occur is $1 - \lambda$.

If a concept switch is represented by the symbol 1 and a non-concept switch by the symbol 0, we can describe the possible outcomes going back from $t$ to $t - 2$ as follows:

---

$^4$Note, other than this effect, we assume our data is noise-free in the following sections.
Figure 3.5: An example \((x, y)\) is classified at trial \(t\) according to a target concept \(c^*\). There is a \(1 - \lambda\) probability that the target concept was used to classify the example at \(t - 1\) and a \(\lambda\) probability that it was not used.
3.5. MINIMISING DISAGREEMENT

1. 00 - There is a $(1 - \lambda)^2$ probability that no concept switches occurred.

2. 01 - There is a $(1 - \lambda)(\lambda)$ probability of the concept not switching than switching.

3. 10 - There is a $(\lambda)(1 - \lambda)$ probability of the concept switching than not switching.

4. 11 - There is a $\lambda^2$ probability that two concept switches occurred.

According to this model of concept switching the probability of exactly $m$ concepts switches in the $n$ trials going back from $t$ to $t - n$ is given by a binomial distribution:

$$P(m|n) = \binom{n}{m}\lambda^m(1 - \lambda)^{n-m}.$$

The probability of a concept switch determines whether the concept at $t - 1$ is different from the target concept at $t$. However, the probability of a concept switch does not determine whether the concept at $t - 2$ is different from target concept at $t$. There is a chance that the target concept switches from $t$ to $t - 1$ and is then reselected from $t - 1$ to $t - 2$.

Figure 3.6 shows a concept class $C = \{a, b, c\}$. The concept before any concept switch has occurred is the target concept. In this case $c^* = a$. After 1 concept switch the target concept cannot be present. After 2 concept switches there is a chance that the target concept has been reselected.

3.5.3 Markov Processes

We now combine the model of concept switching shown in Figure 3.5 and Figure 3.6 by introducing a state-transition diagram. The state-transition diagram describes the probability of being in either one of two states after each trial. The two states correspond to either the target concept being present or some other concept being present.

As before, we let $\lambda$ be the probability of a concept switch occurring on each trial. Furthermore, we introduce $\alpha$ as the probability of the target concept being selected from the set of concepts when a concept switch does
Figure 3.6: Before any concept switch, the target concept is present. After one concept switch, the target concept cannot be present. After two concept switches the target concept can be present.

occur. Note, in the model of concept switching introduced earlier and shown in Figure 3.6, we do not allow the target concept to be reselected on consecutive concept switches. Given these parameters, Figure 3.7 describes the probability of transitioning between each state on each trial.

Figure 3.7: State transition probabilities for moving from a state in which the target concept \(c^*\) is present to one in which some other concept is present on each trial, and \textit{vice versa}.

We now define a probabilistic framework of being in either of the two states after \(n\) trials.
3.5. MINIMISING DISAGREEMENT

Preliminaries

We define the probability space \((\Omega, P)\), where \(\Omega = \mathcal{C}\). We define \(P\) as a probability measure on the powerset of \(\Omega\). Let \(C_n\) be a discrete random variable that defines the probability of the target concept being present after \(n\) trials:

\[
C_n(\omega) = \begin{cases} 
1, & \omega = c^* \text{ after } n \text{ trials} \\
0, & \text{otherwise}.
\end{cases}
\]

The set of outcomes for which \(C_n = 1\) is the event such that \(\{\omega \in \Omega : C_n(\omega) = 1\}\), which is the event \(\{c^*\}\). The probability of this event is given by \(P(C_n = 1)\).

We now describe how modelling concept switching using a Markov process can determine the probability of the target concept being present after \(n\) trials. That is, determining the \(P(C_n = 1)\) and \(P(C_n = 0)\).

If we assume that the concept selected after a trial is dependent only on the present concept, then the selection of concepts follows a Markov process. Using the notation introduced we can more formally define the Markov property, which is the independence of conditional probabilities.

\[
P[C_n = j_n | C_{n-1} = j_{n-1}] = P[C_n = j_n | C_{n-1} = j_{n-1}, ..., C_0 = j_0]
\]

Assuming state transitions abide by the Markov property we can model the probability of being in either of the two states using a Markov process. We begin by defining the state-space, shown in Figure 3.7, by the set \(\{0, 1\}\). Let the first element 0 represent the state in which the target concept is not present after \(n\) trials and let the second element 1 represent the state in which the target concept is present after \(n\) trials. Note, we have previously defined \(C_n\) to be a random variable that represents the probability of the target concept being present after \(n\) trials. In the context of \(C_n\), the state-space represents the two possible values \(C_n\) can take.

We now define the probability of transitioning between states in Figure 3.7 using a transition matrix \(P\). More formally, the \((i, j)\)-th entry of \(P\) is given
as \( p_{(i,j)} = P[C_n = j|C_{n-1} = i] \). Since the state-space contains only two elements, we can fully describe \( P \) as follows:

\[
P = \begin{pmatrix}
p_{(0,0)} & p_{(0,1)} \\
p_{(1,0)} & p_{(1,1)}
\end{pmatrix}
\]

Using the values shown in Figure 3.7, we can define \( P \) in our setting as:

\[
P = \begin{pmatrix}
1 - \lambda \alpha & \lambda \alpha \\
\lambda & 1 - \lambda
\end{pmatrix}
\]

Assuming this transition process is stationary, (i.e. the values of \( P \) do not change over time) then the matrix \( P \) is relevant for all trials.

**Transition Matrix**

There exists a connection between Markov chain theory and matrix theory [36], that allows the overall transition probabilities after \( n \) trials to be determined. Assuming the transition matrix \( P \) is irreducible and aperiodic, then it can be shown [36] that \( P \) after \( n \) trials is given by:

\[
\left( \begin{array}{cc}
1 - \lambda \alpha & \lambda \alpha \\
\lambda & 1 - \lambda
\end{array} \right)^n = \frac{1}{\alpha + 1} \left( \begin{array}{cc}
1 & \alpha \\
1 & \alpha
\end{array} \right) + \frac{(1 - \lambda \alpha - \lambda)^n}{\alpha + 1} \left( \begin{array}{cc}
\alpha & -\alpha \\
-1 & 1
\end{array} \right)
\]

(3.13)

**Steady State Analysis**

As a notational shortcut, let \( \pi_{n,j} = P[C_n = j] \).

At \( n = 1 \), \( \pi_{1,0} \) can be determined as:

\[
\pi_{1,0} = \pi_{0,0} p_{(0,0)} + \pi_{0,1} p_{(1,0)}
\]

At \( n = 1 \), \( \pi_{1,1} \) can be determined similarly:

\[
\pi_{1,1} = \pi_{0,0} p_{(0,1)} + \pi_{0,1} p_{(1,1)}
\]
3.5. MINIMISING DISAGREEMENT

In matrix form, this operation can be represented as:

\[
(\pi_{1,0}, \pi_{1,1}) = (\pi_{0,0}, \pi_{0,1}) \begin{pmatrix} P_{(0,0)} & P_{(0,1)} \\ P_{(1,0)} & P_{(1,1)} \end{pmatrix}
\]

We define a vector, containing row 0 and row 1 at \(n\) as: \(\pi_n = (\pi_{n,0}, \pi_{n,1})\). The above matrix operation can now be given by \(\pi_1 = \pi_0 P\). Furthermore, we have shown that when \(P\) is independent of \(n\) (time homogeneous), the transition probabilities after \(n\) trials is given by \(P^n\). Therefore, the row vector after \(n\) trials is given by:

\[
\pi_n = \pi_0 P^n
\]

Therefore, knowing the initial probabilities of each state and the stationary transition matrix \(P\), we can determine the probability of being in each state after \(n\) trials.

For Being-the-Leader, we know that before any trial has occurred, the target concept is present. In other words, at \(n = 0\) we know \(P(C_0 = 1) = 1\) and conversely, \(P(C_0 = 0) = 0\). Therefore, \(\pi_0 = (\pi_{0,0} = 0, \pi_{0,1} = 1)\).

Furthermore, using Equation 3.13 we know \(P^n\). We can now determine the probabilities of being in either state, and therefore, the probability of the target concept being present after \(n\) trials.

An important property of a regular Markov chain is that the two rows of \(P\) tend towards the same vector as the number of trials increases [36], such that as \(n \to \infty\):

\[
(P(C_\infty = 0), P(C_\infty = 1)) = \left( \frac{1}{\alpha + 1}, \frac{\alpha}{\alpha + 1} \right)
\]

(3.14)

The resulting vector represents a unique stationary distribution (the state probability row vector at steady-state). Note, \(\lambda\) does not factor in the limit and if we consider how \(P(C_\infty = 1)\) changes with \(\alpha\), then as expected the probability of \(P(C_\infty = 1)\) grows with an increasing \(\alpha\).

Also, as \(\alpha \to 0\) the probability of the target concept appearing goes to 0. Whereas, as \(\alpha \to 1\) the probability of the target concept appearing goes
to a $\frac{1}{2}$, this is because our model of concept switching insists that a different concept is selected whenever a concept switch occurs.

We now discuss several properties of an increasing window size with the \textit{Being-the-Leader} approach.

### 3.5.4 Expected Probability of Target-Concept

For \textit{Being-the-Leader}, on the $n$th trial, the probability that the $n$th example is classified by the target concept present on the $n$th trial is $\pi_{n,1} = 1$.

On the $n$-th trial we have a sequence of probabilities $\pi_{0,1}, \ldots, (\pi_{n,1} = 1)$. These are the probabilities that the examples received on the trials $0, \ldots, n$ were classified by the target concept present on the $n$-th trial.

The mean of this sequence gives us the expected probability that an example selected uniformly at random from this sample was classified by the target concept present on the $n$-th trial. That is:

$$
\overline{\pi}_{i,1} = \frac{1}{n+1} \sum_{i=0}^{n} \pi_{i,0} P(0,0)^i + \pi_{i,1} P(1,1)^i
$$

$$
= \frac{1}{n+1} \sum_{i=0}^{n} P(1,1)^i \text{ where } (\pi_{0,0} = 0, \pi_{0,1} = 1)
$$

$$
= \frac{1}{n+1} \sum_{i=0}^{n} \frac{\lambda\alpha}{\lambda\alpha + \lambda} + \frac{\lambda(1-\lambda\alpha-\lambda)^i}{\lambda\alpha + \lambda}
$$

$$
= \frac{1}{n+1} \sum_{i=0}^{n} \frac{\alpha + (1-\lambda\alpha-\lambda)^i}{\alpha + 1}
$$

$$
= \frac{1}{(n+1)(\alpha + 1)} \left( (n+1)\alpha + \frac{(1-\lambda\alpha-\lambda)^{n+1} - 1}{(1-\lambda\alpha-\lambda) - 1} \right)
$$

$$
= \frac{\alpha}{\alpha + 1} + \frac{1 - (1-\lambda\alpha-\lambda)^{n+1}}{(n+1)\lambda(\alpha+1)^2}.
$$

Note, as $n$ goes from 0 to $\infty$, $\overline{\pi}_{i,1}$ ranges (as expected) from 1 to $\frac{\alpha}{\alpha+1}$. The formula may be used to determine the number of correct (or erroneous) examples passed into a learning system.

Conversely, the expected probability that an example selected uniformly at random was classified by some concept other than the target, is:
3.5. MINIMISING DISAGREEMENT

\[
\pi_{i,0} = \frac{1}{\alpha + 1} - \frac{1 - (1 - \lambda \alpha - \lambda)^{n+1}}{(n + 1)\lambda(\alpha + 1)^2} = 1 - \pi_{i,1}
\] (3.21)

3.5.5 Similarity Between Differing Concepts

For Being-the-Leader, we know the current example is classified according to the target concept. Furthermore, we know that the last example is classified by the target concept with probability \(1 - \lambda\), and classified by some other concept with expected probability \(\lambda\).

However, there is a chance that although classified by a different concept, the last example has the same classification which would have been given to it by the target concept. We define the similarity between some other concept and the target concept as the probability that the two concepts disagree on a random example [42]. If for each element in \(\{c \in \mathcal{C} : c \neq c^*\}\), \(P_{x \in \mathcal{D}_x}(c(x) \neq c^*(x)) \leq \Delta\), then the probability that the last example is misclassified is at most \(\lambda \Delta\). A classification different than that given by the target concept is considered classification noise.

The probability of a classification different than that given by the target concept on the \(n\)-th trial is less than or equal to:

\[\pi_{n,0} \Delta\]

Note, with a noise-rate \(\lambda \Delta\), it can be shown [4] that a sample size of \(m\) is needed to pac-identify the target concept when \(\Delta = 1\). Where \(m\) is given by:

\[m \geq \frac{2}{\epsilon^2(1 - 2\lambda)^2}ln \left(\frac{2|\mathcal{C}|}{\delta}\right),\] (3.22)

where with probability \(1 - \delta\) a hypothesis within \(\epsilon\) error is found. Note that \(\Delta = 1\) is the worst case for \(\Delta\).

However, in our case the noise-rate is not constant, and in a sample of this size there will be an even higher noise-rate, which in turn will require a larger sample size.
CHAPTER 3. PREDICTIVE MENU ON A MOBILE PHONE

Expected Probability-of-Disagreement

Given Equation 3.21, the expected probability that an example selected uniformly at random from an \( n \)-sample will disagree with the target concept is less than or equal to:

\[
\left( \frac{1}{\alpha + 1} - \frac{1 - (1 - \lambda \alpha - \lambda)^n}{n\lambda(\alpha + 1)^2} \right) \Delta. \tag{3.23}
\]

Note the formulation of this equation includes sampling up to and including the \( n - 1 \)-th trial.

3.5.6 Most Common Hypothesis: Being-the-Leader

The expected probability that the \( n \)-th example is classified by some concept other than the target concept is governed by \( \pi_{n,0} \). As \( n \) approaches infinity, \( \pi_{n,0} \) converges to the stationary distribution \( \pi_{\infty,0} \), where:

\[
\pi_{\infty,0} = \frac{1}{\alpha + 1}.
\]

Therefore, as \( n \) approaches infinity, the probability that the examples in the \( \text{sample}_{\infty} \) are classified by some other concept is \( \frac{1}{\alpha + 1} \), and the expected probability-of-disagreement for the hypothesis capturing the target concept in the \( \text{sample}_{\infty} \) is less than or equal to \( \frac{1}{\alpha + 1} \Delta \).

The expected probability-of-disagreement of the Most Common Hypothesis: Being-the-Leader is less than or equal to:

\[
\frac{1}{\alpha + 1} \Delta.
\]

Note, if \( \alpha < 1 \) (i.e. it is not certain that we will transition from some other concept to the target concept), then in the limit the majority of the sample will be classified by concepts other than the target concept. The hypothesis capturing the most prevalent concept will minimise disagreement over this sample. The learner will be consistent (a Bayes classifier) as its expected performance (in terms of probability-of-disagreement) will converge to the performance of the best hypothesis as the number of trials \( t \) approaches infinity.
3.5. MINIMISING DISAGREEMENT

3.5.7 Fixed Window: Being-the-Leader

A Being-the-Leader version of the Fixed Window approach will use a fixed window of size \( w \) of previous examples, i.e. from trial \( t \) back to trial \( t - w \).

The expected probability that the \( n \)-th example is classified by some concept other than the target concept is governed by \( \pi_{n,0} \). Therefore, on every trial, the probability of some concept other than the target moves further away from the initial state and further towards the steady state-distribution, i.e.:

\[
P(C_0 = 1) = 0 \rightarrow P(C_\infty = 1) = \frac{1}{\alpha + 1}.
\]

The expected probability-of-disagreement of the Fixed Window: Being-the-Leader is less than or equal to:

\[
\left( \frac{1}{\alpha + 1} \frac{1 - (1 - \lambda \alpha - \lambda)^{w+1}}{(w + 1)\lambda(\alpha + 1)^2} \right) \triangle.
\]

As long as the window does not include all observed trials, the expected loss (disagreement) of the Fixed Window: Being-the-Leader is less than that of the Most Common Hypothesis: Being-the-Leader. Furthermore, the rate at which the loss of the Fixed Window: Being-the-Leader approaches that of the Most Common Hypothesis: Being-the-Leader approach is governed by:

\[
\frac{1 - (1 - \lambda \alpha - \lambda)^{w+1}}{(w + 1)\lambda(\alpha + 1)^2}.
\]

The significance of this equation is that as the window size \( w \) increases the faster the Fixed Window: Being-the-Leader approaches the Most Common Hypothesis: Being-the-Leader in terms of expected probability-of-disagreement. It also provides a justification for the use of the Fixed Window: Being-the-Leader approach over the Most Common Hypothesis: Being-the-Leader approach.
3.5.8 Most Recent Correct Hypothesis: *Being-the-Leader*

One issue with the most recent correct approach is that when a concept shift occurs, new examples may be generated from the new concept that are also consistent with the previous concept. As expected this will mean that the most recent correct approach will continue to induce the hypothesis for the previous concept. Thus the sample will include examples from both the new concept and the old, and hence some of these examples will contribute to the probability-of-disagreement. However, as the number of examples for the new concept increases, the probability that they are all consistent with the previous concept rapidly approaches 0. A maximum bound on this expected probability-of-disagreement due to this concept shift effect is now formulated.

Let \( n \) denote the number of examples that have been generated from the new concept. If for each element in \( \{c \in C : c \neq c^*\} \), \( P_{x \in D_X}(c(x) = c^*(x)) = \mu \). The probability that all of these will be consistent with the old concept will be:

\[
\mu^n.
\]

If all the examples of the new concept are consistent with the previous concept then the expected probability-of-disagreement will be:

\[
\frac{\frac{1}{\lambda}(1 - \mu)}{n + \frac{1}{\lambda}}.
\]

Whereas if any of the examples from the new concept do not agree with the previous concept then the hypothesis from the previous concept is discounted, since the hypothesis for the new concept will be a longer consistent hypothesis. Assuming a hypothesis for the new concept is embraced, then the probability-of-disagreement will be 0.

Thus the expected probability-of-disagreement at a particular point in the sequence will be:

\[
\mu^n \frac{\frac{1}{\lambda}(1 - \mu)}{n + \frac{1}{\lambda}}.
\]
Given that we expect any particular concept to go for \( \frac{1}{\lambda} \) examples, the expected probability-of-disagreement can be averaged over these cases. This average is given by:

\[
\frac{\left(\sum_{n=1}^{\frac{1}{\lambda}} \mu^n \frac{1}{n+\frac{1}{\lambda}} \right)}{\left(\frac{1}{\lambda}\right)} = \sum_{n=1}^{\frac{1}{\lambda}} \frac{1}{n+\frac{1}{\lambda}} \\
\leq \frac{1-\mu}{1+\frac{1}{\lambda}} \sum_{n=1}^{\frac{1}{\lambda}} \mu^n = \frac{\mu}{1+\frac{1}{\lambda}} \left(1 - \mu^\left(\frac{1}{\lambda}\right)\right).
\]

As \( \lambda \) decreases the expected probability-of-disagreement bound will also decrease. Also if \( \mu = 0 \) or \( \mu = 1 \) then the bound will be 0. Note this bound is confirmed by the simulated evaluation.

To compare the Being-the-Leader versions of the Most Recent Correct Hypothesis approach and the Fixed Window approaches a simulation was created which generated a sequence of runs. These sequences were used to measure the expected probability-of-disagreement, together with the number of training examples provided to the different approaches. The parameters for these simulations were, \( \lambda = 0.1 \), 100 different concepts, and concepts were uniformly chosen when a concept switch occurred. The approaches were run over a sequence of 20,000 trials and the results were averaged over the entire sequence. The probability that different concepts had overlapping examples was the same between all concepts. However, we varied this probability between 0 and 1, shown by the \( x \)-axis in the graphs in Figures 3.8 and 3.9.

As shown in Figure 3.8 the expected probability-of-disagreement of the Most Recent Correct Hypothesis: Being-the-Leader approach is 0 when there is no overlap between concepts and also when there is complete overlap between concepts. This is to be expected since the “Most Recent Correct” sample will never have an example generated by a concept other than the target when concept overlap is 0, and will never have an example that would be classified differently than the target concept when concept overlap is 1.

Figure 3.8 also shows that the expected probability-of-disagreement of the Most Recent Correct Hypothesis: Being-the-Leader approach is greater than the Fixed Window: Being-the-Leader approach (with a window size of 2), when overlap between concepts is greater than 0.4. Although the approach performs worse, it must be noted that the difference in performance
CHAPTER 3. PREDICTIVE MENU ON A MOBILE PHONE

Figure 3.8: The expected probability-of-disagreement for the Most Recent Correct Hypothesis: Being-the-Leader and the Fixed Window: Being-the-Leader approaches, with varying amount of overlap between hypotheses.

Figure 3.9: The average window size used by the Most Recent Correct Hypothesis: Being-the-Leader and the Fixed Window: Being-the-Leader approaches, with varying amount of overlap between hypotheses.
3.6. COMPARISON

is relatively small, and when overlap is at 0.4 the average window size of the Most Recent Correct Hypothesis: Being-the-Leader approach, shown in Figure 3.9, is actually five times greater than the Fixed Window: Being-the-Leader approach. Having a greater number of examples ensures that the Most Recent Correct Hypothesis approach will have more examples in which to eliminate an $\epsilon$-bad hypothesis.

We can see from Figure 3.8 that the Fixed Window: Being-the-Leader approach can only reduce its expected probability-of-disagreement by reducing its window size. However, by reducing its window size the approach severely limits the number of examples from which to select a hypothesis. In the implementable Follow-the-Leader versions of these approaches it is imperative to have as many correct examples (examples generated according to the target concept) as possible. We can see that there is a trade-off between ensuring that every example in a sample is “correct” and maximising the sample size.

3.6 Comparison

We now compare the Being-the-Leader versions of the Most Common Hypothesis approach, the Fixed Window approach, and the Most Recent Correct Hypothesis approach in terms of expected probability-of-disagreement.

Firstly, since the expected probability-of-disagreement of the Most Common Hypothesis: Being-the-Leader approach is at most:

$$\frac{1}{\alpha + 1} \Delta,$$

and the expected probability-of-disagreement of the Fixed Window: Being-the-Leader approach is at most:

$$\left( \frac{1}{\alpha + 1} - \frac{1 - (1 - \lambda(\alpha - \lambda)^{w+1})}{(w + 1)(\alpha + 1)^2} \right) \Delta,$$

it is clear that the Fixed Window: Being-the-Leader approach will perform better on average than the Most Common Hypothesis: Being-the-Leader approach. Furthermore, since the difference between Being-the-Leader and
Follow-the-Leader is the same for both approaches, we can also say that the Fixed Window approach will perform better than the Most Common Hypothesis approach.

Secondly, when the degree of overlap between concepts is below a certain value $\mu$, the Most Recent Correct Hypothesis: Being-the-Leader will perform better on average than the Fixed Window: Being-the-Leader approach. When the degree of overlap between concepts is above this value, the difference in expected probability-of-disagreement is relatively small and the Most Recent Correct Hypothesis: Being-the-Leader will have a larger sample size than the Fixed Window: Being-the-Leader approach. When we consider the implementable versions of these approaches, it is reasonable to expect that the Most Recent Correct Hypothesis approach will perform better from having a larger sample size than the Fixed Window: Being-the-Leader approach.

### 3.7 Discussion

The benefit of the first four approaches: the First Menu-Item approach; the First Menu-Item (frequency ordered) approach; the Menu-Item Last Selected approach; and the Most Common Menu-Item Selected approach is their simplicity and that they follow an intuitively good strategy for predicting the correct menu item. However, such approaches are not adaptive and do not learn, and therefore cannot be assured of a high level of predictive accuracy across different users and different contexts.

The performance of the last three approaches: the Most Common Hypothesis approach; the Fixed Window approach; and the Most Recent Correct Hypothesis approach was analysed in terms of loss (expected probability-of-disagreement). We considered a hypothetical problem domain called Being-the-Leader. We bounded the difference, in terms of loss, between the hypothetical Being-the-Leader and the actual Follow-the-Leader problem domains. We then determined the expected probability-of-disagreement of each of the Being-the-Leader versions.

Intuitively, the Most Recent Correct Hypothesis used a windowing strategy that was a balance between being large enough to eliminate $\epsilon$-bad hypotheses, while at the same time being small enough to exclude examples
classified by previous target concepts.

3.8 Experimental Evaluation

We now present an experimental evaluation of each menu-item prediction approach. The experimental evaluation serves to back the theoretical results presented in the previous section.

Each menu prediction approach was initially evaluated using a simulated data set. The simulated data set was constructed using parameters similar to those found in the actual mobile phone learning setting. The simulated data set conforms to the assumptions made about the learning problem, i.e. target concepts change with a fixed probability $\lambda$ and that the set of possible concepts is contained within the hypotheses provided, $\mathcal{C} \subseteq \mathcal{H}$. The simulated data set provides a justification for the Most Recent Correct Hypothesis approach.

In addition to the simulated data set, each menu prediction approach was evaluated on real-world data collected from several mobile phone users. Each menu prediction approach was evaluated on two sets of real-world data. The first set consisted of data collected from two users unobtrusively over a period from one to two months. The second set consisted of data collected from five users each performing the same scripted scenario.

3.8.1 Smartphone Platform

The Nokia™Series 60 smartphone was chosen as the platform to investigate menu-item prediction. The main menu in the Contacts address book application on the Series 60 phone was used as the basis for investigating menu-item prediction. The main menu in the Contacts address book application consists of 14 menu items ordered: Open, Call, Create message, New contact, Edit, Delete, Duplicate, Add to group, Belongs to group, Mark/Unmark, Send, Contacts info, Help and Exit. A modified version of the Contacts address book application was implemented on a Nokia N-Gage QD™Series 60 phone. The modified Contacts application recorded each menu item a user selected together with 3 associated attributes. Each menu selection made by
the user represented a training example for the learner.

The example space of the menu selection environment was defined by the following 3 attributes:

- The last menu item selected by the user.
  \( \text{last-menu-item-selected} \in \{ \text{Open, Call, Create message, New contact, Edit, Delete, Duplicate, Add to group, Belongs to group, Mark/Unmark, Send, Contacts info, Help and Exit} \} \).

- Whether the user has scrolled to a person in the \( \text{Contacts} \) address book.
  \( \text{scrolled} \in \{ \text{true, false} \} \).

- To which groups the person selected in the \( \text{Contacts} \) address book belonged. The number of values this attribute could take was assumed to be small and known \( \text{a priori} \) to learning, e.g.
  \( \text{contact-belongs-to-group} \in \{ \text{Work, Family, Friends} \} \).

(Note, these attributes were chosen as they represent features that are easy to observe and would intuitively be predictive of the user’s menu selection in the \( \text{Contacts} \) application.) Each instance was assigned a label based on which menu item the user selected:

- \( \text{menu item} \in \{ \text{Open, Call, Create message, New contact, Edit, Delete, Duplicate, Add to group, Belongs to group, Mark/Unmark, Send, Contacts info, Help and Exit} \} \).

Hypothesis Space

The hypothesis space used, \( \mathcal{H} \), consisted of a set of functions. Each hypothesis, \( h \in \mathcal{H} \) was represented as a \textit{decision list}. When evaluated, each \textit{decision list} predicted a natural number from the set \( 0, 1, 2, ..., 13 \). The predicted number represented the index of a menu item in the \( \text{Contacts} \) address book menu.

A condition in a \textit{decision list} consisted of any test or conjunction of tests on the attributes in the example space. The hypothesis space was restricted to represent only those concepts that were likely to occur in the menu-item prediction setting. Table 3.2 through to Table 3.9 shows 8 basic \textit{decision}}
lists] that comprise the highly restricted hypothesis space that was used in
the evaluation. These hypotheses were chosen as they represent common
patterns a user might perform while operating the Contacts application. The
complete hypothesis space contained 14 repetitions of each of the decision
lists shown. Each decision list was repeated with a different menu item for
the default rule (we show only those decision lists with the default rule Call).

3.9 Results

3.9.1 Simulation

A simulated data set was used to evaluate the menu-item prediction ap-
proaches. The data set attempted to model the likely data a user would
generate when interacting with an address book application on a mobile
phone. We used the Contacts address book application described previously,
as the basis for the parameters used to create the data set. The parameters
are as follows:

Data Set Parameters

The three attributes recorded each time a user selected a menu item are: the
last menu item selected by the user; whether the user has scrolled to a person
in the Contacts address book; and which groups the person selected in the
Contacts address book belonged too.

The last menu item selected attribute could take on 14 values, the scrolled
attribute was boolean valued and the groups that a contact belonged to could
take on three values (e.g. Work, Family, Friends)\(^5\). The number of attribute-
value combinations was 84, i.e: \(|\mathcal{X}| = (14 \times 2 \times 3) = 84\). The user could select
one of 14 menu items, i.e. \(|\mathcal{Y}| = 14\). The learner could take 1176 possible
“courses-of-action”, i.e. \(|\mathcal{Z}| = |\mathcal{X} \times \mathcal{Y}| = (84 \times 14) = 1176\). There are
\(2^{1176}\) possible ways of assigning a \(\{0,1\}\)-discrete loss function (that defines
a concept “appropriateness”) to the set of possible “courses-of-action”, i.e.
\(|\mathcal{C}| = 2^{1176}\).

\(^5\)Although the number of groups could vary from person to person, it was assumed that
this number was small and each group was known a priori.
Table 3.2: A decision list in the hypothesis space that represents the concept of a user creating new contacts and then adding the contacts to groups.

| IF last-menu-item-selected = New Contact AND scrolled = true |
| THEN New Contact |
| ELSE IF last-menu-item-selected = New Contact AND scrolled = false |
| THEN Add to group |
| ELSE IF scrolled = false |
| THEN New Contact |
| ELSE (DEFAULT) Call |

Table 3.3: A decision list in the hypothesis space that represents the concept of a user creating new contacts and then editing the new contacts’ details.

| IF last-menu-item-selected = New Contact |
| THEN Open Contact |
| ELSE IF last-menu-item-selected = Open Contact |
| THEN Edit Contact |
| ELSE IF last-menu-item-selected = Edit Contact |
| THEN Open Contact |
| ELSE (DEFAULT) Call |

Table 3.4: A decision list in the hypothesis space that represents the concept of a user viewing a contact’s details and then editing the details.

| IF last-menu-item-selected = Open Contact |
| THEN Edit Contact |
| ELSE (DEFAULT) Call |

Table 3.5: A decision list in the hypothesis space that represents the concept of a user editing a contact’s details and then viewing the details.

| IF last-menu-item-selected = Edit Contact |
| THEN Open Contact |
| ELSE (DEFAULT) Call |
3.9. **RESULTS**

Table 3.6: A *decision list* in the hypothesis space that represents the concept of a user creating a message to send to contacts who belongs to group $X$, e.g. $X = \{\text{Work, Family, Friends}\}$.

\[
\text{IF contact-belongs-to-group} = X \\
\text{THEN Create message} \\
\text{ELSE (DEFAULT) Call}
\]

Table 3.7: A *decision list* in the hypothesis space that represents the concept of a user editing contacts who belongs to group $X$, e.g. $X = \{\text{Work, Family, Friends}\}$.

\[
\text{IF contact-belongs-to-group} = X \\
\text{THEN Edit Contact} \\
\text{ELSE (DEFAULT) Call}
\]

Table 3.8: A *decision list* in the hypothesis space that represents the concept of a user deleting contacts who belongs to group $X$, e.g. $X = \{\text{Work, Family, Friends}\}$.

\[
\text{IF contact-belongs-to-group} = X \\
\text{THEN Delete Contact} \\
\text{ELSE (DEFAULT) Call}
\]

Table 3.9: A very simple *decision list* in the hypothesis space that represents the concept that all the user wants to do is call a contact.

\[
(\text{DEFAULT}) \text{ Call}
\]
The simulated data set was based on 112 different hypotheses. This number is based on the eight decision lists shown in Tables 3.2-3.9, each with 14 different default rules\(^6\), i.e: \(8 \times 14 = 112\). Each hypothesis represents a possible concept, i.e: a subset of the possible “courses-of-actions” that are “appropriate”. The 112 concepts defined by the 112 hypotheses, is vastly smaller than the possible number of concepts that could be defined over the possible “courses-of-actions”. The reason for choosing a small number of hypotheses was to mirror the assumption that only a very small percentage of the set of possible “courses-of-action” would be meaningful to a user.

Each of the 112 concepts was represented by a different random mapping of the attribute-value combinations to a menu item, i.e: a random subset of the possible “courses-of-action”. To model concept change, a sequence of concepts was generated. The sequence was 1000 concepts long. A uniform distribution was used to select the first concept; then for each iteration the current concept was kept with probability \((1 - \lambda) = 0.9\). If the concept changed, a new concept was reselected with respect to the uniform distribution. The sequence was intended to represent a user selecting menu items according to a concept, with the user changing to a new concept 10 percent of the time.

Each concept in the 1000-long sequence was used to generate a training example. A uniform distribution was used to select a random attribute-value combination, and this was then classified according to the given concept to create a training example. For instance, given the concept: “the user always chooses the *Open* menu item after they have scrolled, otherwise they choose the *Edit* menu item” remains true for the first 15 elements of the concept sequence. Then the first 15 training examples would represent examples, where if scrolling is true then the menu item attribute equals *Open* otherwise the attribute equals *Edit*. The same sequence of training examples was provided to all of the menu prediction approaches described in the previous section. The hypothesis-based menu prediction approaches were given the

\(^6\) Actually, only 13 different default rules could be used on the decision list in Table 3.4-3.8. If the leaf nodes on these decision lists were the same, e.g. “IF last-menu-item-selected = Edit Contact THEN Call ELSE (DEFAULT) Call, then they would represent the same concept captured in Table 3.9, e.g. ELSE (DEFAULT) Call
112 possible concepts that the training examples were generated from.

Each menu-item prediction approach attempted to predict the next menu item the user would select. After making a prediction each approach received the next training example. This training example informed the approach on what the user did select. The aim of the simulation was to evaluate how well each menu prediction approach performed against a data set that modelled the expected concept-drift characteristics of a real user. The simulation was repeated 100 times.

**Measuring Discrete Loss**

The average number of mispredictions for each menu-item prediction approach is shown in Figure 3.10. The average number of mispredictions is equivalent to the average loss for each menu-item prediction approach, when loss is measured using the discrete loss function $L$. The discrete loss function is defined by Equation (3.2). Note, the standard error on the estimates of all these means was below 0.01.

The first four menu prediction approaches - First Menu Item, First Menu Item (freq. reordered), Last Menu Selected and Most Common Menu Item - have the same amount of discrete loss. This is to be expected given the way the simulated data set was generated. The menu items in the simulated data set were chosen uniformly at random. That is, every menu item would occur in the simulated data set with equal probability. If $Y$ is a random variable representing the 14 possible menu items in $\mathcal{Y}$, then on every trial the probability that the menu item $y$ would be selected was $P(Y = y) = \frac{1}{14}$. The first four menu-item prediction approaches predict the same menu item on two consecutive trials. That is, the menu item predicted on trial $t$ will be the same as the menu item predicted on trial $t + 1$ ($y_t = y_{t+1}$). The probability that the menu item $y_t$ is different from the menu item $y_{t+1}$ is the probability that a menu item other than $y_t$ is selected. The probability of this event is $P(y_t \neq y_{t+1}) \equiv P(X \neq y_t) \equiv \frac{13}{14}$. Since a misprediction results in a loss of one, the expected loss over a series of trials is $\frac{13}{14}$, which can be seen in Figure 3.10.

The *Most Common Hypothesis* approach should also have a loss similar
Figure 3.10: The number of mispredictions each menu prediction approach makes in per trial, over 1000 trials, averaged over 100 runs using simulated data. The mean number of mispredictions is equivalent to the mean discrete loss of each approach.

to the first four menu prediction approaches. However, it will take more trials for *Most Common Hypothesis* to approach this loss. This can be seen in Figure 3.10, where the approach produces slightly less loss than the first four approaches. The reason for this is the way the simulated data set is generated. Although menu items are assigned to attributes uniformly at random, there are only 84 values attributes can take on in each hypothesis. The small number of attributes means that the distribution of menu items in each hypothesis only roughly approximates uniformity. The non-uniformity in the mappings of each hypothesis means that over a small number of trials a non-uniform distribution of menu items will exist in the data set. The *Most Common Hypothesis* can identify the hypothesis producing this pattern, giving it an advantage in predicting menu selections.

At one extreme we have the expected loss when randomly predicting a menu item from a uniform distribution, i.e. the expected loss equals $\frac{13}{14}$. At
the other extreme, if the learner is 100 percent accurate, it will only produce loss due to concept drift ($\lambda = 0.1$), i.e. the expected loss is less than or equal to 0.1. The next five approaches lie somewhere between these extremes.

The Fixed Window approach with a window size of one, is interesting. Having one example allows the approach to do better than random. In particular, one example allows it to drop its expected loss from 0.92 to 0.82, a reduction of approximately 12.5 percent. In other words, one example allows us to eliminate 12.5 percent of the bad hypotheses in the hypothesis space. Having a window size of two or three reduces the loss even further. However, a window size of four increases loss. This increase can be explained by the effect of concept drift and the resulting presence of examples in the window that were classified by concepts other than the target concept (noisy examples).

The Most Recent Correct Hypothesis approach outperforms all the fixed window approaches. It does this by using a window size that will always contain all the examples of the current target concept and unlikely to contain examples form a different concept.

**Measuring Absolute Loss**

Although discrete loss provides a simple means of measuring the effectiveness of the prediction approaches, the actual aim of menu selection prediction should be to minimise the number of scrolling key presses required by a user. The average number of scrolling key presses for each approach is shown in Figure 3.11. Since each of the 14 menu items represent an index in the menu, each menu item can be represented by an integer, e.g. 0 - Open, 1 - Call, ..., 13 - Exit. The class label is therefore an integer between zero and 13 (inclusive). The number of scrolling key presses is equivalent to the distance between the menu item predicted and the menu item desired by the user. The average number of scrolling key presses is equivalent to the average loss, when loss is measured using the absolute loss function, see Equation (3.1). Note, the standard error on the estimates of all these means was below 0.08.

Since the menu items are distributed uniformly at random in the simulated data set, each menu item will occur with equal probability. If $X$
CHAPTER 3. PREDICTIVE MENU ON A MOBILE PHONE

Figure 3.11: The number of scrolling key presses each menu-item prediction approach makes in per trial, over 1000 trials, averaged over 100 runs using simulated data. The mean number of scrolling key presses is equivalent to the mean absolute loss of each approach.

is a random variable representing the 14 possible menu items in \( \mathcal{Y} \), then on every trial the probability that the menu item \( y \) would be selected was \( P(X = y) = \frac{1}{14} \). The First Menu Item approach predicts the first menu item on every trial. With 14 menu items to select, a misprediction with this approach will result in an absolute loss between one and 13. The absolute loss associated with mispredicting a menu item \( y \) is equal to the value of \( y \) itself (the value of \( y \) is its index in a menu). Over a series of trials the expected loss is equivalent to the index of the expected menu item. Using Equation (3.4) the expected absolute loss is 6.5. The average mean loss of the First Menu Item approach, shown in Figure 3.11, agrees with the calculated expected absolute loss.

Since menu items are distributed uniformly at random in the simulated data set, reordering menu item according to their selection frequency will not improve the absolute loss. Figure 3.11 shows this, with the First Menu Item
3.9. RESULTS

(freq. reordered) doing no better than the First Menu Item approach.

The Last Menu Selected approach does better than the first two approaches. The reason for this is that the last menu selected does not necessarily highlight the top-most menu item. For instance, the last menu item may have been \( y = 7 \) (which is also the menu item highlighted), if so the absolute loss for a misprediction could only be between zero and seven. Using Equation (3.12) the expected absolute loss is 4.64.

Furthermore, since menu items are distributed uniformly at random in the simulated data set, each menu item is also equally likely to be the most common menu item. As such, the Most Common Menu Item approach will incur a loss similar to the Last Menu Selected approach. Figure 3.11 shows the similarity in loss between these two approaches.

The hypothesis based approaches: Most Common Hypothesis, Fixed Window and Most Recent Correct Hypothesis contain hypotheses that are equally likely in predicting anyone of the 14 menu items. If these approaches chose hypotheses at random, then they would incur a loss similar to the Most Common Menu Item and the Last Menu Selected approaches. However, if they chose hypotheses according to a non-random strategy they would incur a loss less than these approaches. It is interesting to note that these approaches can perform worse than random. In fact, if they were 100 percent accurate at choosing the menu item furthest away from the menu item selected they could incur an absolute loss greater than 6.5. Figure 3.11 shows that the hypothesis-based approaches perform well when compared to the other approaches, suggesting that they can indeed learn menu predictions. The Most Recent Correct Hypothesis approach performs well when compared with the Most Common Hypothesis and Fixed Window approaches.

Figure 3.12 illustrates why the Most Recent Correct Hypothesis approach works well when compared with the other hypothesis-based approaches. Given a sequence of training examples one through four in Figure 3.12, hypotheses that are consistent with the current example are marked by a tick, and those that are not with a cross. In Figure 3.12 we can see that when deciding on example five, the hypothesis-based learning approaches will choose different hypotheses. The Most Common Hypothesis approach will choose the hypothesis that has been correct most often. In this case it will be Con-
Figure 3.12: A sequence of four examples are presented to three hypotheses. Ticks represent those examples consistent a hypothesis, while crosses those that are inconsistent. The target concept changes from a concept that is represented by hypothesis A to one represented by hypothesis C after example two. Note, a hypothesis may not represent the underlying concept yet still be consistent with an example.

cept A. The Fixed Window approach with window size one will tie between hypotheses A, B and C. With a window size of two, hypothesis C will be correctly chosen. With a window size of three, a tie between hypothesis A and C exists, and with a window size of four hypothesis A is chosen. The Most Recent Correct Hypothesis approach, like the Fixed Window approach with window size two, will correctly choose hypothesis C. However, unlike the Fixed Window approach with window size two, the Most Recent Correct Hypothesis will chose hypothesis C for example six (not shown). Note, the Most Recent Correct Hypothesis approach is not limited by a set window size of examples when evaluating hypotheses.

Following-the-Leader vs. Being-the-Leader Simulation

In Section 3.5.2 we provided an analysis of how the window size would effect the menu prediction learning approaches. In particular, we examined the difference between Being-the-Leader and Follow-the-Leader variants.

We now show how window size affects Follow-the-Leader and Being-the-Leader approaches using the simulated data set from the previous Section. Figure 3.13, graphs the discrete loss of both approaches using a fixed window size that ranges from one to 25.

The Follow-the-Leader approach with a window size of one will produce
3.9. RESULTS

A mean discrete loss around 0.8, this is also seen in Figure 3.10. As the window size increases the mean discrete loss of Follow-the-Leader drops to around 0.25. However, as the window size increases beyond three, the mean discrete loss increases. Note, as the window size increases the mean discrete loss converges around 0.8. This effect would be predicted by the theory.

With a window size of one, there are not enough examples to eliminate $\epsilon$-bad hypotheses. As the window size increases, more $\epsilon$-bad hypotheses will be eliminated, while not including noisy examples from previous concepts. However, at a certain point the window will grow too large and the effect of concept switching will introduce examples labelled by previous concepts. As the window size grows larger it will produce the same mean discrete loss as the Most Common Hypothesis or the Best Hypothesis in Hindsight when Following-the-Leader.

The Being-the-Leader approach with a window size of one makes no mis-predictions. It is also apparent that as the window size increases, the mean discrete loss increases. This increase can be explained by the fact that as the window size grows, more examples labelled by previous target concepts will exist. Furthermore, as the fixed window size grows the mean discrete loss increases until it is equivalent to selecting the Most Common Hypothesis or the Best Hypothesis in Hindsight when Being-the-Leader.

Appendix C shows the mean discrete loss for both Follow-the-Leader and Being-the-Leader approaches using a fixed window size that ranges from one to 1200.

3.9.2 Real World Data

We now present an evaluation of the menu prediction approaches using data collected from users.

Two Users

The menu interactions of two users using the Contacts address book application on their Nokia N-Gage QD™series 60 mobile phones were recorded. The Contacts address book application provides the main point of access to the personal contact details stored on the user’s Nokia N-Gage QD™Series
60 mobile phones. The application can be used to call and SMS contacts, as well as to store other details such as addresses, birthdays, etc.

Each menu interaction a user made in the Contacts address book application was recorded using a modified version of the application. The modified Contacts address book application was identical to the original application, but allowed the user’s interactions to be recorded. The users were both experienced with the use of the Contacts address book application, and were asked to use the application as they normally would.

The modified Contacts application recorded which menu item the user selected. The menu items a user could select included (in this order): Open, Call, Create message, New contact, Edit, Delete, Duplicate, Add to group, Belongs to group, Mark/Unmark, Send, Contacts info, Help and Exit.

The modified Contacts application also recorded seven attributes every time the user selected a menu item. The seven attributes recorded were:

- The name of the person selected in the Contacts application, when the menu item was selected.
3.9. RESULTS

- The group to which the person selected in the Contacts address book belonged. The number of values this attribute could take, \( k \), was assumed to be small and known \textit{a priori} to learning, e.g. \( \text{contact-belongs-to-group} \in \{ \text{Work, Family, Friends} \} \).

- The last menu item selected by the user.
  \( \text{last-menu-item-selected} \in \{ \text{Open, Call, Create message, New contact, Edit, Delete, Duplicate, Add to group, Belongs to group, Mark/Unmark, Send, Contacts info, Help, Exit and NoLastCommand} \}. \text{NoLastCommand} \) indicates that application has been restarted, and no last command has been recorded since then.

- Whether the user has scrolled to the person selected in the Contacts address book.
  \( \text{scrolled} \in \{ \text{true, false} \} \).

- The location of the phone, in terms of Cell-ID, when the menu item was selected.

- The time when the menu item was selected.

- The day of the week when the menu item was selected.

Every time a menu item is selected a formatted data record is stored. An example of such a record is given below.

\begin{verbatim}
Bec|Friend|Create message|scrolled|NoLastCommand|14171|12:05 PM|Thu 18th Nov '04
\end{verbatim}

Note, each attribute is delimited by the | symbol, and the third attribute represents the menu item selected. Each menu selection made by the user represented a training example. Table 3.10 summarises the training data set for each user.

The training data for each user was presented to each of the menu prediction approaches introduced and discussed in Section 3.4. On each trial, each menu prediction approach was given the seven attributes and asked to predict the menu item the user would select. The loss associated with each
Table 3.10: Information on the training data sets used to evaluate the menu prediction approaches.

<table>
<thead>
<tr>
<th>No.</th>
<th>User Type</th>
<th>Data Collection Period</th>
<th>Data Set Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>experienced, light user</td>
<td>30 days continuous</td>
<td>42</td>
</tr>
<tr>
<td>User 2</td>
<td>experienced, heavy user</td>
<td>52 days continuous</td>
<td>147</td>
</tr>
</tbody>
</table>

approach’s prediction was determined. Each approach was then shown the menu item the user selected.

The effectiveness of the menu prediction approaches over the training data in terms of discrete loss and absolute loss is shown in Figure 3.14 and Figure 3.15 respectively.

We now discuss these results.

Discussion

First Menu Item: The ordering of menu items in our version of the Contacts application is identical to the ordering used by Nokia™. Menu items are ordered (Open, Call, Create SMS, New, etc). If menu items were selected uniformly at random by a user, then we would expect the absolute loss to be 6.5. For both users the First Menu Item approach requires substantially less key presses than the expected average of 6.5 key presses, suggesting Nokia™ has chosen a menu ordering that is in line with the selection frequency of menu items made by an average user.

First Menu Item (frequency re-ordered): This approach improved on the First Menu Item approach. As expected, customising the menu ordering for a particular user will improve the number of scrolling key presses that user needs to make. However, the ordering was derived by inspecting all training instances beforehand. Knowing the selection frequency of menu items beforehand is an unrealistic assumption.

Last Menu Selected: This approach performed well on the data from the second user, but not on data from the first user. This can be explained by the fact that the second user’s data was characterised by long sequences in which they exclusively selected the SMS menu item, whereas the first user’s
Figure 3.14: The discrete loss for each user evaluated on the different menu prediction approaches. The bar indicates the standard error for each approach.

Figure 3.15: The absolute loss for each user evaluated on the different menu prediction approaches. The bar indicates the standard error for each approach.
data did not include long sequences in which they consecutively selected the one menu item.

**Most Common Menu Item:** This approach exhibited the same behaviour between the two users as the *Last Menu Selected* approach. Again, the second user used the *Contacts* application predominantly to send SMS messages, selecting the SMS menu item 62 percent of the time. Always predicting the most commonly selected menu item was advantageous for the second user.

The introduction of hypotheses as shown in the last six approaches in Figure 3.15, allowed the average number of key presses to remain consistently low for both users. The use of a hypothesis space allows the learner to capture concepts exhibited by each user. However, the *Most Recent Correct Hypothesis* approach did not out-perform the other hypothesis-based approaches. Instead, all the hypothesis-based approaches performed similarly. One reason for this is that evaluating the hypothesis-based approaches using data that has been collected over a long period may not provide an accurate comparison between approaches. Data collected over a long period will contain situations where the one concept remains stable for a period of time, such as adding a new contacts to the address book. But these situations may be outweighed by situations where the concept changes after every menu interaction. For instance, a user may use the phone to make a call, then leave the phone for a few hours before looking up someone’s work number. In this case, only one example is provided for the concept representing “making a call” before it is changed to a concept representing “looking up work number”. Any learner will require a concept to remain stable for more than one training example, and therefore all the hypothesis-based approaches will require that a concept remains stable for more than one menu interaction.

**Scenario**

A problem with comparing the menu prediction approaches using the data collected from the two users was the vastly different tasks each user performed. To compare the menu prediction approaches in a more controlled fashion, we gathered data from five different users performing a similar set of tasks in the *Contacts* application. Each user was asked to perform a set
of tasks specified by a written scenario. The scenario-driven data created a situation in which concepts remained stable for more than one menu interaction and allowed for a standardised comparison to be made between the approaches.

The scripted task asked each user to perform the following sequence of actions:

1. Create 10 new contacts.
2. Assign four of the contacts to one group and six to another group.
3. Edit the four contacts who belong to a single group.
4. Send an SMS to the four contacts in a single group.
5. Delete two contacts.
6. Create one more contact and assign it to a group.

Note, the user interface allowed different ways to accomplish each of these tasks. Therefore, these tasks could be accomplished by differing menu item selection sequences.

The first two users of the five were User 1 and User 2 from the previous section. While the first two users were familiar with the Contacts application on the Nokia N-Gage QD™Series 60 mobile phone, the last three were not. Except for User 1 and User 5, all were heavy mobile phone users. Table 3.11 summarises the training data set for each user.

Table 3.11: Information on the scenario-driven training data sets used to evaluate the menu prediction approaches.

<table>
<thead>
<tr>
<th>No.</th>
<th>Mobile phone usage</th>
<th>Contacts app. experience</th>
<th>Data Set Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>light user</td>
<td>yes</td>
<td>48</td>
</tr>
<tr>
<td>User 2</td>
<td>heavy user</td>
<td>yes</td>
<td>42</td>
</tr>
<tr>
<td>User 3</td>
<td>heavy user</td>
<td>no</td>
<td>57</td>
</tr>
<tr>
<td>User 4</td>
<td>heavy user</td>
<td>no</td>
<td>54</td>
</tr>
<tr>
<td>User 5</td>
<td>light user</td>
<td>no</td>
<td>58</td>
</tr>
</tbody>
</table>
Similar to the two user evaluation, each menu selection made by the user represented a training example for the learner. The training data for each user was presented to each of the menu prediction approaches introduced and discussed in Section 3.4. On each trial, each menu prediction approach was given the seven attributes and asked to predict the menu item the user would select. The loss associated with each approach’s prediction was determined. Each approach was then shown the menu item the user selected.

The effectiveness of the menu prediction approaches in terms of discrete loss and absolute loss are shown in Figure 3.16 and Figure 3.17 respectively.

**Discussion**

It is apparent from Figure 3.17 that the *First Menu Item* approach can be improved upon, in some cases to the extent of saving the user three key presses per menu selection. Although a saving of three key presses does not sound significant, over the entire run of examples our *Most Recent Correct Hypothesis* approach, when compared with the *First Menu Item* approach, provided a saving of 144, 148, 164, 111 and 143 key presses for users one, two, three, four and five respectively.

The variability exhibited by the *First Menu Item* (frequency re-ordered), *Last Menu Selected*, *Most Common Menu Item* and *Most Common Hypothesis* approaches suggests that they are not general solutions to menu selection prediction for all users. We also see that no one approach performs the best for all users. This could suggest the need to customise the approach for the user.

All four *Fixed Window* approaches and the *Most Recent Correct Hypothesis* approach performed well on the data. Furthermore, the average number of key presses was consistently low for each user with these approaches. This suggests that not only can identifying a concept be useful, but that some method of handling concept changes is required.

The performance of the four *Fixed Window* approaches and our *Most Recent Correct Hypothesis* approach was similar for each user. However, we would have expected our *Most Recent Correct Hypothesis* approach to perform better than the *Fixed Window* approaches. The *Fixed Window* ap-
3.9. RESULTS

Figure 3.16: The discrete loss for each user evaluated on the different menu prediction approaches using scenario-driven training data. The bar indicates the standard error for each approach.

Figure 3.17: The absolute loss for each user evaluated on the different menu prediction approaches using scenario-driven training data. The bar indicates the standard error for each approach.
approaches consider all previous examples, of a fixed window size, relevant when evaluating a hypothesis space. As discussed in Section 3.7, the deficiency of a fixed window is apparent when concept changes occur, as can be seen in Figure 3.12. The deficiency of using a fixed window size was not obvious in Figure 3.16 and Figure 3.17. This may have been due to the fact that the concepts exhibited in the data remained stable for a few menu interactions, and therefore were able to be tracked by the small window sizes we used. Despite this, a fixed window size will invariably result in too many or not enough training examples being kept. Our Most Recent Correct Hypothesis approach overcomes this inherent problem by, in effect, assigning to each hypothesis an adaptive sized window. Whenever a hypothesis becomes inconsistent with a new example, its window is cleared of examples. The rationale for the Most Recent Correct Hypothesis approach is to prevent the consideration of examples generated from a previous concept when evaluating the hypothesis space.

3.10 Related Work

The approach presented in this chapter is closely related to various work associated with learning in the presence of concept drift. In the area of predicting from expert advice, an initial result was provided by Littlestone and Warmuth [73]. Littlestone and Warmuth modified a version of the Weighted Majority algorithm, called WML, that tolerated a shifting best expert. The modification was based on placing a lower bound on the weight of any particular expert. The minimum weight modification prevented the weight of any expert from becoming too low, and hence allowed any expert to recover more readily from a shifting concept. Herbrster and Warmuth [43] modified the goal of Littlestone and Warmuth by seeking to design a master algorithm that essentially “tracks” the performance of the best sequence of experts. They were able to bound the additional loss of this approach over the best off-line partition of experts for any arbitrary sequence of examples.

The minimum weight modification has also been analysed by Auer and Warmuth [7] when learning shifting disjunctions. Furthermore, Mestererham [78] applied the minimum weight modification to the Winnow algorithm and
3.10. RELATED WORK

gave mistake-bounds for learning any linear-threshold function that is allowed to shift. Other linear-threshold functions for tracking shifting concepts have been studied by Herbster and Warmuth [44] and by Kivinen et al. [58]. In a similar way to the minimum weight modification approaches, the Most Recent Correct Hypothesis approach presented in this chapter attempts to allow any hypothesis the opportunity to recover from a concept switch. Applying the hypothesis that has been consistent for the longest run into the past ensures that any hypothesis performing poorly can recover quickly in the event of a concept switch.

In the area of computational learning theory, Helmbold and Long [42] presented a simple tracking algorithm that found the hypothesis which minimised disagreement over a suffix of previous examples. Helmbold and Long [42] proved theoretical limitations on the rate of concept drift that could be tolerated for a class of target concept, measured by its VC-dimensions. In particular, they presented an algorithm that is proven to tolerate concept drift rates of \( \delta \leq c_1 \frac{\epsilon^2}{d \ln \frac{1}{\epsilon}} \), using a window size \( m \geq \frac{64d}{\epsilon} \ln \frac{64}{\epsilon} \), where \( d \) is the VC-dimension of the concept class. Long [75], continued their work, deriving bounds on the rate of concept drift sufficient for agnostic learning. Further analysis has been conducted into concept drift by Case et al. [17], who presented a method for measuring the quality of predictions under concept drift. Optimal bounds on the necessary and sufficient permanence of concepts for certain concrete classes were derived. In addition, Bartlett et al. [10] derived a sufficient condition on the guaranteed estimability of a family of sequence of functions. The model of concept drift presented by Helmbold and Long [42] is similar to the model of concept drift presented in this chapter. Furthermore, the Most Recent Correct Hypothesis approach employs a similar minimising disagreement approach for hypothesis selection. Both our Most Recent Correct Hypothesis approach and the initial algorithm presented by Helmbold and Long suffer from the problem of finding an efficient means of performing the minimising disagreement step. However, our Most Recent Correct Hypothesis approach does differ from that of Helmbold and Long by not analysing the approach in terms of a “fixed” suffix of previous examples.

Other related work includes Blum and Chalasani [14] and Ben-David and Dichterman [11], who both considered learning in the presence of concept
drift as learning in the presence of some hidden variable, which determined concept drift. Blum and Chalasani studied the use of a finite set of deterministic concepts to probabilistically model concept drift. Their work constrained the number of concepts visited and the frequency of concept switching, a constraint that the work presented in this chapter also employed. Ben-David and Dichterman addressed the inherent hidden variable in concept drift by learning a function that projected examples onto some observable space.

Separate from the machine learning issues surrounding concept drift is the problem of evaluating mobile phone interfaces. A model-based evaluation of mobile phone interfaces was performed by St. Amant et al. [3]. The Fitt’s law model (see appendix A.1), which predicts the amount of time a user will take to interact with an interface, in conjunction with the GOMS model (see appendix A.2) for describing menu interactions, was shown to closely align with real user behaviour on a mobile phone interface. The conjunction of the Fitt’s law model and the GOMS model can be shown to be equivalent to the average number of key presses metric that we used in evaluating our predictive menu approaches. This equivalence provides justification for the use of the average number of key presses as a measure of evaluation.

3.11 Conclusion

This chapter introduced an approach for adapting the menu artifacts on mobile phone interface. Standard mobile phone menus highlight the first menu item by default, requiring the user to scroll down to find and select their desired menu item. The approach proposed in this chapter keeps the list of menu items static but simply starts the user off at the predicted menu item, saving the user a number of key presses if predicted correctly. Although this change is subtle, it has the potential to save users an enormous number of key presses over time. These savings are clearly seen in the experimental evaluation of this chapter.

To achieve menu-item prediction a novel machine learning technique, the Most Recent Correct Hypothesis, was used. The Most Recent Correct Hypothesis learning approach used a small and highly biased hypothesis space
to meet the computation limitations of a mobile phone. Furthermore, it employed a dynamically sized windowing approach to address concept drift. The results of an experimental evaluation indicated the benefit of the Most Recent Correct Hypothesis learning approach. Importantly, they showed the benefit of using a dynamically sized windowing approach over a fixed sized windowing approach.
Chapter 4

Menu Prediction: A Minimum Description Length Approach for Inducing Hypotheses

4.1 Introduction

In the previous chapter, menu prediction was evaluated in the setting of the Contacts address book application on a Nokia™ Series 60 smartphone. In that setting, three menu prediction approaches that relied on a small set of hand-crafted hypotheses were introduced and evaluated. These were: the Most Common Hypothesis approach, the Fixed Window approach, and the Most Recent Correct Hypothesis approach. The hypotheses used by these approaches were hand-crafted to capture the possible concepts a user might adopt in that setting. However, if menu prediction is to be applied to different menus of different applications on a mobile phone, then the burden of eliciting and specifying hypotheses may be too great. Furthermore, it may not always be possible to know what possible concepts are suitable ahead of time. To address these problems, this chapter extends the menu prediction approaches presented previously by replacing the hand-crafted hypotheses with ones induced by an online learner.

Any learner attempting to induce hypotheses on a mobile phone will face the issues discussed in Section 3.2.2 on page 32. Briefly, these include:
CHAPTER 4. MDL APPROACH FOR INDUCING HYPOTHESES

computational and memory limitations of the mobile phone platform, limited training data from which to learn, and concept drift. In the previous chapter, hand-crafted hypotheses were used since these removed the computational burden and the training data requirements of inducing hypotheses. Furthermore, this allowed the possible concepts a user might adopt to be accurately specified reducing the problem of learning under concept drift to one of tracking the current concept. In replacing the hand-crafted hypotheses with ones induced by a learner we do not address the computational or training data requirements but instead focus on the problem of learning under concept drift. We introduce an approach that allows us to generate hypotheses under concept drift. The approach presented is based on an information theoretic approach for identifying concept drift.

The approach consisted of generating a hypothesis after every menu interaction. These hypotheses were evaluated using a Minimum Description Length (MDL) technique, which is based on an approach introduced by Rissanen [98]. The MDL technique was used to assess the complexity of the hypotheses produced. If the complexity of the hypotheses produced remained consistent or reduced as more menu interactions were trained over, then the target concept was assumed to be stable. However, if the complexity increased it was assumed the target concept had changed. An increase in hypothesis complexity was used to identify the end of one concept and the start of another. We now discuss the MDL technique in more detail.

4.2 Minimum Description Length Principle

According to Grunwald [40], the Minimum Description Length Principle is an inductive inference approach that solves the problem of model selection. Informally, the MDL principle suggests that the best hypothesis for a given data set is the one that encodes the data set with the smallest total code length. This can be derived from the fact that generally the goal of learning is to find the most probable hypothesis for some given data. A formulation of the Bayes rule can be used to determine this maximum \textit{a posteriori} hypothesis:
4.3. THE LEARNING SETTING

\[ P(H|D) = \frac{P(H)P(D|H)}{P(D)}. \]

Where \( P(H|D) \) is the posterior probability of the hypothesis given the data \( D \). According to information theory we are assured that the shortest expected coding length for an event with probability \( P \) is \(-\log(P)\) bits. This property allows the problem of finding the maximum a posteriori hypothesis to be cast as one of finding a hypothesis which minimises the code length needed to describe the hypothesis and the data given the hypothesis [40].

4.3 The Learning Setting

We now describe the role of the MDL principle in discovering hypotheses in an environment where the target concept can change over time. We present the MDL principle as an addition to the menu prediction approaches described in Chapter 3. In particular, the MDL principle is presented as an alternative to the hand-crafted hypotheses required by the menu prediction approaches presented in Chapter 3.

4.3.1 MDL Principle and Concept Drift

A learning algorithm needs some means of dealing with an environment in which the concept being learnt can change over time. In Section 2.3.3 on page 26 we discussed how several learning algorithms handle environments with concept drift. We now present an approach for learning in an environment where the concept being learnt can switch over time. The approach is based on applying the MDL principle to hypotheses generated in an online fashion. The MDL of consecutive hypotheses is monitored and used to determine when a concept switch occurs. We now describe the approach in more detail.

The implementation of our MDL approach consisted of three parts:

- The first part consisted of a rule learner which was used in an online fashion to induce a decision list after every menu interaction. We
investigated the use of Clark and Niblett’s CN2 rule induction algorithm [21].

• The second part consisted of calculating the codelength of each decision list produced. The coding scheme used to calculate the codelength is described in the Section 4.3.2.

• The third part monitored the codelength of consecutive hypotheses. The codelength of a hypothesis was considered to be relative to its complexity. If the complexity of consecutive hypotheses decreased or remained the same, this suggested that the hypotheses being generated were consistent with a sequence of examples produced according to the same underlying concept. The reason for this is that a hypothesis of similar complexity would be able to encode more examples if those examples were all consistent with the same underlying concept. Note, we are assuming that the hypotheses generated are able to accurately model the underlying concept. However, if the complexity of consecutive hypotheses increased, then it suggested that the underlying concept had changed. The reason is that the example observed after a concept change would have to be encoded as an exception by the next hypothesis generated. This would result in a more complex hypothesis being generated and an increase in codelength between consecutive hypotheses.

4.3.2 Codelength Calculation using the MDL Principle

The codelength of a decision list was calculated based on a method developed by Quinlan and Rivest [93]. The structure of the decision list was encoded as a binary decision tree. The tree structure was encoded by representing leaf nodes by a binary 0 and internal nodes by a binary 1. Each leaf node was followed by its default categorisation. Each internal node was followed by an encoding of the node’s condition and by the encoding of its children, in order. For instance, given three conditions X, Y and Z, and three categorisations A, B and C, the binary decision tree in Figure 4.1 would be represented by the string 1 X 1 Y 0 A 0 B 1 Z 0 C 0 A.
4.3. THE LEARNING SETTING

The cost of encoding each internal node’s condition was determined by taking the log\(^1\) of the number of possible attributes and the log of the possible of values for that attribute. Both discrete and real-valued attributes can be encoded in this way since the value of the attribute or cut point of the attribute can be represented. The cost of encoding each internal node’s condition was determined by taking the log of the number of possible categorisations.

Given a set of objects, the cost of a miscategorisation also needs to be encoded. The cost of encoding a leaf node was taken as the cost of specifying those objects that were exceptions to the leaf-node’s default categorisation. Suppose the left-most leaf node in the binary decision tree in Figure 4.1 categorised four objects; the first object is correctly categorised as A, the second object is correctly categorised as A, the third object is miscategorised as A when it is in fact B, and the fourth object is miscategorised as A when it is in fact C. The categorisation of this leaf could be represented by the string “0 0 B C”, where the first two characters represent the default categorisation of the leaf node and the last two characters represent the exceptions. Given this string the following method is used to determine the cost of encoding the categorisation of a leaf node[93].

\[^1\]All logarithms in this chapter are base 2.
$L(n; k_1, k_2, ..., k_t) = \log \left( \left( \frac{n + k - 1}{k - 1} \right) \cdot \left( \frac{n}{k_1, k_2, ..., k_t} \right) \right),$

where $n$ is the number of characters in the string, $k_1$ objects are of category 1, $k_2$ objects are of category 2, ..., $k_t$ objects are of category $t$, and $k = k_1 + ... + k_t$. This encoding method ensured that leaf nodes that categorised fewer objects incorrectly would be encoded more compactly.

Since we were concerned only with the relative differences between successive codelengths we did not implement the more efficient incremental coding scheme suggested in Wallace and Patrick [110].

### 4.4 Menu Prediction Approaches

In Chapter 3, three menu prediction approaches were presented that relied on a highly restricted and tailored hypothesis space, the Most Common Hypothesis, the Fixed Window, and the Most Recent Correct Hypothesis approaches. The hypothesis space of these approaches consisted of hand-crafted hypotheses. We now present a modification to these approaches that attempts to generate their highly restricted and tailored hypothesis space using the MDL approach described in the previous section. The three approaches are listed below.

- The **Most Common Hypothesis - MDL** (a description of this approach is given in Algorithm 4.4.1).
- The **Fixed Window - MDL** (a description of this approach is given in Algorithm 4.4.2).
- The **Most Recent Correct Hypothesis - MDL** (a description of this approach is given in Algorithm 4.4.3).

Each approach is initialised with an empty hypothesis space. After each trial a hypothesis is induced, and if this hypothesis has a greater codelength than the hypothesis induced on the previous trial, it is included in the hypothesis space. In every other regard, these approaches are identical to the
4.4. MENU PREDICTION APPROACHES

Most Common Hypothesis, the Fixed Window, and the Most Recent Correct Hypothesis approaches presented in Chapter 3.
4.4.1 Most Common Hypothesis - MDL

Algorithm 4.4.1 Most Common Hypothesis - MDL technique

1: Given: a list of menu items indexed by $(1, 2, ..., n)$.
2: a set of hypotheses, $\mathcal{H} = \{\}$.
3: a variable $j$.
4: Associate a counter for each of the hypotheses in $\mathcal{H}$, $B = (b_1, b_2, ..., b_{|\mathcal{H}|})$.
5: Initialise these counters to zero and initialise $j = 1$.
6: Define a codelength function for the hypothesis $hyp$ generated on trial $t$: $\text{codelength}(hyp_t)$, where $\text{codelength}(hyp_0) = 0$.
7: In each trial $t \geq 1$:
8: receive a state of the environment $x_t$.
9: induce a hypothesis $hyp_t$ from learner based on examples $e_j, ..., e_t$.
10: if $hyp_t \neq hyp_{t-1}$ then let $j = t$, add $hyp_{t-1}$ to $\mathcal{H}$ and associate a counter for $hyp_{t-1}$ in $B$.
11: obtain a set of menu-item predictions $\{h_1(x_t), h_2(x_t), ..., h_{|\mathcal{H}|}(x_t)\}$, by evaluating each hypothesis in $\mathcal{H}$.
12: predict the menu item using hypothesis with the highest counter:
13: $\hat{y}_t = h_{\arg\max_{i \in \{1, ..., |\mathcal{H}|\}} b_i(x_t)}$.
14: receive the correct prediction, $y_t$.
15: update the counters associated with the correct hypotheses:
16: $b_i \leftarrow \begin{cases} b_i + 1, & \text{if } h_i(x_t) = y_t \\ b_i, & \text{otherwise.} \end{cases}$

$\dag$ ties are resolved by choosing the menu item with the lowest index.
4.4. MENU PREDICTION APPROACHES

4.4.2 Fixed Window - MDL

Algorithm 4.4.2 Fixed Window - MDL technique

1: Given: a list of menu items indexed by $(1, 2, \ldots, n)$.
2: a set of hypotheses, $\mathcal{H} = \{\}$.
3: a window $w$ that holds a fixed size array of examples, $w[0], \ldots, w[i]$.
4: a variable $j$.
5: Associate a counter for each of the hypotheses in $\mathcal{H}$, $B = (b_1, b_2, \ldots, b_{|\mathcal{H}|})$.
6: Define a codelength function for the hypothesis $hyp$ generated on trial $t$: $codelength(hyp_t)$, where $codelength(hyp_0) = 0$.
7: Initialise these counters to zero and initialise $j = 1$.
8: In each trial $t \geq 1$:
9: receive a state of the environment $x_t$.
10: induce a hypothesis $hyp_t$ from learner based on examples $e_j, \ldots, e_t$.
11: if $hyp_t > hyp_{t-1}$ then let $j = t$, add $hyp_t$ to $\mathcal{H}$ and associate a counter for $hyp_{t-1}$ in $B$.
12: obtain a set of menu-item predictions $\{h_1(x_t), h_2(x_t), \ldots, h_{|\mathcal{H}|}(x_t)\}$, by evaluating each hypothesis in $\mathcal{H}$.
13: predict the menu item using hypothesis with the highest counter:
14: $\hat{y}_t = h_{\text{argmax}_{i \in \{1, \ldots, |\mathcal{H}|\}} b_i(x_t)}$.
15: receive the correct prediction, $y_t$.
16: update the window array accordingly:
17: add $(x_t, y_t)$ to element $w[t \mod i]$ of $w$ array.
18: initialise the counters associated with each hypothesis to zero.
19: update hypothesis counters accordingly
20: for each $(x, y)$ in $w$
21: $b_i \leftarrow b_i + 1$, if $h_i(x) = y$

† ties are resolved by choosing the menu item with the lowest index.
**Algorithm 4.4.3** Most Recent Correct Hypothesis - MDL technique

1. Given: a list of menu items indexed by \((1, 2, \ldots, n)\).
2. a set of hypotheses, \(\mathcal{H} = \{\}\).
3. a variable \(j\).
4. Associate a counter for each of the hypotheses in \(\mathcal{H}\), \(B = (b_1, b_2, \ldots, b_{|\mathcal{H}|})\).
5. Define a codelength function for the hypothesis \(hyp\) generated on trial \(t\):
   
   \[ \text{codelength}(hyp_t), \text{ where codelength}(hyp_0) = 0. \]

6. Initialise these counters to zero and initialise \(j = 1\).
7. In each trial \(t \geq 1\):
   
   8. receive a state of the environment \(x_t\)
   9. induce a hypothesis \(hyp_t\) from learner based on examples \(e_j, \ldots, e_t\).
10. if \(hyp_t > hyp_{t-1}\) then let \(j = t\), add \(hyp_{t-1}\) to \(\mathcal{H}\) and associate a counter for \(hyp_{t-1}\) in \(B\).
11. obtain a set of menu-item predictions \(\{h_1(x_t), h_2(x_t), \ldots, h_{|\mathcal{H}|}(x_t)\}\), by evaluating each hypothesis in \(\mathcal{H}\).
12. predict the menu item using hypothesis with the highest counter:
   
   \[ \hat{y}_t = h_{\arg\max_{i \in \{1, \ldots, |\mathcal{H}|\}} b_i(x_t)} \uparrow. \]

13. receive the correct prediction, \(y_t\).
14. update each counter accordingly:
   
   \[ b_i \leftarrow \begin{cases} 
   b_i + 1, & \text{if } h_i(x_t) = y_t \\
   0, & \text{otherwise.} 
   \end{cases} \]

\[ \uparrow \text{ties are resolved by choosing the menu item with the lowest index.} \]

**4.5 Experimental Evaluation**

We now present an experimental evaluation of the MDL versions of the menu prediction approaches: the *Most Common Hypothesis* approach, the *Fixed Window* approach, and the *Most Recent Correct Hypothesis* approach.

The experimental evaluation was based on the same real-world data sets presented in Section 3.9.2 on page 85.
4.5. EXPERIMENTAL EVALUATION

The most recent version of the CN2 rule induction algorithm was used [20] and the following parameters were used:

- Ordered rule algorithm.
- Laplacian error estimate.
- Size limited set (Star-size) of 7.
- All other parameters used their default values.

The decision lists produced after each iteration of the CN2 algorithm were stored in a data structure that allowed each decision list to be evaluated on the data sets.

4.5.1 Two Users

As presented in Section 3.9.2 on page 85, a modified version of the Contacts address book application was used to collect data on two users. Data was collected continuously over a period from one to two months. Table 3.10 on page 88 summarises this training data set. Using this training data, the effectiveness of the menu prediction approaches using the MDL technique was determined.

Figure 4.2 compares the mean discrete loss for User 1 using the MDL technique and the specified (hand-crafted) hypotheses technique from Section 3.9.2. Figure 4.3 compares the absolute loss of the two techniques.

Figure 4.4 compares the mean discrete loss for User 2 using the MDL technique and the specified (hand-crafted) hypotheses technique from Section 3.9.2. Figure 4.5 compares the absolute loss of the two techniques.

Figure 4.6 and Figure 4.7 shows the code length of the hypotheses produced after every example for User 1 and User 2 respectively. In addition, the size of the restricted hypothesis space used by the menu prediction approaches is overlayed on these graphs.
Figure 4.2: A comparison between the MDL technique and the specified hypotheses technique, in terms of discrete loss for User 1. The bar indicates the standard error for each approach.

Figure 4.3: A comparison between the MDL technique and the specified hypotheses technique, in terms of absolute loss for User 1. The bar indicates the standard error for each approach.
4.5. EXPERIMENTAL EVALUATION

Figure 4.4: A comparison between the MDL technique and the specified hypotheses technique, in terms of discrete loss for User 2. The bar indicates the standard error for each approach.

Figure 4.5: A comparison between the MDL technique and the specified hypotheses technique, in terms of absolute loss for User 2. The bar indicates the standard error for each approach.
Figure 4.6: The relative code-lengths of the hypotheses produced online and the size of the restricted hypothesis space used by the menu predictions approaches for User 1.

Figure 4.7: The relative code-lengths of the hypotheses produced online and the size of the restricted hypothesis space used by the menu predictions approaches for User 2.
4.5. EXPERIMENTAL EVALUATION

4.5.2 Two Users - Discussion

The mean discrete loss and absolute loss exhibited by User 1 in Figure 4.2 and Figure 4.3 respectively shows that the MDL technique did not perform as well as the specified hypotheses technique. This is to be expected given that the specified hypotheses technique has available to it hypotheses that were hand-crafted to capture the possible concepts a user might adopt. However, with the exception of the Most Common Hypothesis approach in Figure 4.2, all the menu prediction approaches using the MDL technique performed better than the standard First Menu Item menu prediction approach for both users.

The largest difference in terms of discrete and absolute loss between the MDL and specified hypothesis techniques occurred with the Most Common Hypothesis approach. The large difference between the two techniques is to be expected given the way in which the Most Common Hypothesis approach operates. The Most Common Hypothesis approach performs best when there exists one hypothesis that is consistent with a majority of the training examples. In the case of the MDL technique it is unlikely such a hypothesis will be found since the aim of the MDL technique is to learn individual concepts, which are often represented by short runs of training examples when concept drift exists. Although the MDL technique performs poorly with the Most Common Hypothesis approach, the difference between the two techniques is reduced in the Fixed Window approaches (using a window sizes of 2, 3, and 4) and the Most Recent Correct Hypothesis approach.

One reason for the difference between the MDL and the specified hypotheses techniques is the disadvantage the MDL technique has when compared to the specified hypotheses technique. The MDL technique has the disadvantage of having to induce hypotheses before they can be applied for menu prediction. In effect, the MDL technique will never be able to apply a concept when it is first encountered, and must therefore rely on concepts recurring in the sequence of training examples. The training examples generated from User 1 did in fact exhibit recurring concepts. Figure 4.6 shows that out of the 35 hypotheses generated, 10 hypotheses were unique. Recurring concepts can also been seen for User 2 in Figure 4.7. Out of the 72 possible hypotheses generated, 14 hypotheses were unique.
The results presented for User 2, shown in Figure 4.4 and Figure 4.5 are quite different from those presented for User 1. The first difference can be seen in Figure 4.4. Unlike User 1, the performance of the MDL technique is slightly better than that of the specified hypotheses technique. If the difference between the two techniques was significant, it would suggest that the MDL technique induced hypotheses that were more accurate than those specified. However, the difference in performance of the MDL technique and the specified hypotheses technique is not mirrored when absolute loss is measured, as seen in Figure 4.5. One explanation for this difference is that on average a misprediction by the MDL technique resulted in more key presses (absolute loss) than a misprediction made by the specified hypotheses technique. This could be explained by the MDL technique performing better in terms of discrete loss but worse in terms of absolute loss. To illustrate this point, we could imagine the case where the user’s actions could be modeled by relatively simplistic hypotheses. In such cases the MDL technique would be able to induce these simplistic hypotheses, and therefore predict the correct menu item most of the time. However, in those situations where these simplistic hypotheses fail to predict the correct menu item, they may instead predict a menu item that results in a large number of key presses for the user (absolute loss). The problem is that the menu prediction approach has to choose a hypothesis to predict a menu-item. If the hypotheses available to it are poor, then in certain situations the menu-item predictions could result in an absolute loss greater than the loss of selecting menu items at random. This problem does not affect the specified hypotheses technique to the same degree. The specified hypotheses technique consists of hand-crafted hypotheses and, if chosen correctly, will contain hypotheses that accurately model both the simplistic and complex behaviour of a user.

4.5.3 Scenario

The data presented in Section 3.9.2 on page 90, was used to evaluate the menu prediction approaches using the MDL technique. As discussed in Section 3.9.2, the scenario-driven collection of data was used to compare the menu prediction approaches in a more controlled fashion.
4.5. **EXPERIMENTAL EVALUATION**

Data was gathered from five different users performing the same scenario in the *Contacts* application. The scenario asked each user to perform the following tasks in the Nokia™ *Contacts* application:

1. Create 10 new contacts.
2. Assign four of the contacts to one group and six to another group.
3. Edit the four contacts who belong to a single group.
4. Send an SMS to the four contacts in a single group.
5. Delete two contacts.
6. Create one more contact and assign it to a group.

As noted in Section 3.9.2, the user interface allowed completely different ways to accomplish the same task, therefore these tasks could be accomplished by different key press sequences.

Using the data collected from five users who performed the scenario, the effectiveness of the menu prediction approaches using the MDL technique was determined. Figure 4.8(a), 4.8(b), 4.8(c), 4.8(d), and 4.8(e) compares the mean discrete loss between the MDL technique and the specified hypotheses technique for each user. Figure 4.9(a), 4.9(b), 4.9(c), 4.9(d), and 4.9(e) shows the mean absolute loss for each user.

Figure 4.10(a), Figure 4.10(b), Figure 4.10(c), Figure 4.10(d) and Figure 4.10(e) shows the codelength of the hypotheses produced after every example for the five users. In addition, the size of the restricted hypothesis space used by the menu prediction approaches is overlayed on these graphs. These graphs show how the codelength of successive hypotheses change over a series of examples and also how the restricted hypothesis space grows over a series of examples.

4.5.4 **Scenario - Discussion**

Generally, the differences between the MDL technique and the specified hypotheses technique for the scenario-driven data were similar to the differences
CHAPTER 4. MDL APPROACH FOR INDUCING HYPOTHESES

Menu Prediction Approach

(a) User 1

Menu Prediction Approach

(b) User 2

Menu Prediction Approach

(c) User 3
Figure 4.8: Menu prediction approaches evaluated on five users performing the scenario in the Nokia™ Contacts application. A comparison in terms of discrete loss is made between a technique that learns hypotheses online using a MDL technique and one that employs a small set of specified hypotheses.
CHAPTER 4. MDL APPROACH FOR INDUCING HYPOTHESES

Menu Prediction Approach

(a) User 1

(b) User 2

(c) User 3
Figure 4.9: Menu prediction approaches evaluated on five users performing the scenario in the Nokia™ Contacts application. A comparison in terms of absolute loss is made between a technique that learns hypotheses online using a MDL technique and one that employs a small set of specified hypotheses.
CHAPTER 4. MDL APPROACH FOR INDUCING HYPOTHESES

(a) User 1

(b) User 2

(c) User 3
Figure 4.10: The relative codelengths of the hypotheses produced online and the size of the restricted hypothesis space used by the menu predictions approaches for the five users performing the scenario in the Nokia™ Contacts application.
between the MDL technique and the specified hypotheses technique for Users 1 and 2. In particular, in Figures 4.8(a), 4.8(b), 4.8(c), 4.8(d), and 4.8(e) the menu prediction approaches using the MDL technique did not perform as well as those using the specified hypotheses technique. This is also reflected in the absolute loss shown in Figures 4.9(a), 4.9(b), 4.9(c), 4.9(d), and 4.9(e).

Although the menu prediction approaches do not perform as well as those using the specified hypotheses, they do perform better than the standard First Menu Item approach for all users. Furthermore, for User 3 in Figure 4.8(c) and Figure 4.9(c) the menu prediction approaches using the MDL technique perform better than the menu prediction approaches that do not employ a learning system, i.e. the First Menu Item, the First Menu Item (frequency ordered), the Menu-Item Last Selected, and the Most Common Menu-Item approaches.

Although the MDL technique in general performed poorly, there were two users in which the performance difference between the MDL technique and the specified hypotheses technique was noteworthy. The first was User 2 and the second was User 3.

For User 2 the MDL technique approaches performed much worse than the specified hypothesis approaches, as can be seen in Figure 4.8(b) and Figure 4.9(b). The Most Common Hypothesis MDL technique had approximately 29 percent more mispredictions and required 0.5 more scrolling key presses than the specified hypotheses technique. The Fixed Window MDL technique had greater than 50 percent more mispredictions and required two more scrolling key presses than the specified hypotheses technique for all window sizes. The Most Recent Correct Hypothesis MDL technique performed worst, with approximately 60 percent more mispredictions and requiring 2.5 more scrolling key presses than the specified hypotheses technique.

The training examples generated by User 2 can explain the large difference between the MDL and specified hypotheses techniques. The training examples User 2 produced showed that the user completed the scenario by selecting the same menu item in contiguous sequences. For instance, the first two tasks of the scenario were to create 10 new contacts and add four of them to one group and six to another group. User 2 accomplished this
task by selecting the *new contact* menu item 10 times, then selecting the *add to group* menu item 10 times. The MDL technique, given such a sequence, would find that the hypothesis it induced after every trial would decrease in codelength. This effect can be seen in Figure 4.10(b), where the MDL codelength decreases in consecutive sections. After the 10 *new contact* menu item examples have been observed the codelength would increase due to the first *add to group* menu item being observed. The increase in codelength would cause the newly induced hypothesis to be added to the hypothesis space. Since the scenario did not call for any more contacts to be created, this newly induced hypothesis would not be useful for any future menu-item predictions. One possible modification to address the situation is to add the induced hypothesis after its codelength has decreased for a certain period of time. Adding the induced hypothesis in this way would allow the hypothesis to be employed before a concept change occurred.

For User 3 the MDL technique performed similarly to the specified hypotheses technique in all of the menu prediction approaches, as can be seen in Figure 4.8(c) and in Figure 4.9(c). The *Most Common Hypothesis* MDL technique, the *Fixed Window* MDL technique (all window sizes), and the *Most Recent Correct Hypothesis* MDL technique all performed slightly better than their specified hypothesis variants in terms of absolute loss. In terms of discrete loss the *Fixed Window* MDL technique (with window size 1) and the *Most Recent Correct Hypothesis* MDL technique performed slightly better than the specified hypothesis variants.

The training examples generated by User 3 can explain the similarity between the two techniques. User 3 accomplished the scenario by interleaving the tasks of the scenario. For instance, the task of creating 10 new contacts and adding four of them to one group and six to another was completed by interleaving the selection of the *new contact* menu item and the *add to group* menu item. The interleaving of tasks allowed the MDL technique to induce a hypothesis and apply it for future menu-item predictions. The effect of interleaving tasks can be seen in Figure 4.10(c), where the size of the restricted hypothesis space increases within the first half of examples then remains constant. Table 4.1 shows one of the hypotheses correctly captured by the MDL technique.
Table 4.1: One of the hypotheses produced by the MDL technique from User 3’s training examples. This hypothesis captures User 3 performing the task of adding new contacts and assigning them to groups in the Nokia\textsuperscript{TM} Contacts application.

\begin{tabular}{l}
\textbf{IF} selected-contact-belongs-to \textbf{= No Group} \\
\textbf{THEN} Add to group \\
\textbf{ELSE} New contact
\end{tabular}

Another interesting observation was the variation in loss exhibited between users. This variation in loss could be accounted for by the many ways in which the tasks could be accomplished and by the many different hypotheses that could be induced. Furthermore, three of the users were not experienced with the Contacts application and could have made mistakes while completing each task, therefore reducing the ability to induce consistent hypotheses.

Although the MDL technique shows promise as an alternative to specifying hypotheses \textit{a priori}, it does not achieve the same level of discrete and absolute loss as the specified hypotheses technique. This can be seen from the evaluation on both real-world and scenario-driven data, and is to be expected for a number of reasons. Firstly, the decision rules produced by the rule learning algorithm may not match the accuracy of hand-crafted decision rules. Secondly, as mentioned before, there is an underlying assumption that concepts will recur and that induced decision rules can be applied in the future.

One area of further research would be to investigate combining the MDL technique and the specified hypotheses technique. Specified hypotheses could be used to seed the MDL technique. The seeded hypotheses would ideally represent general concepts that were applicable for all users. Initialising the MDL technique with specified hypotheses could reduce the initial loss of the menu prediction approaches.
4.6 Conclusion

Motivated by the problem of menu prediction presented in the previous chapter, this chapter introduces an approach for menu prediction based on an information theoretic approach to hypothesis selection. The aim of the approach is to overcome the requirement of eliciting and specifying hypotheses \textit{a priori}, which was required by the learners presented in the previous chapter.

The basis of the approach presented in this chapter was the use of the minimum description length principle (MDL) to guide hypothesis induction in the presence of concept drift. The approach consisted of an online rule learning algorithm that induced a hypothesis for every new training example it received. The MDL principle was used to evaluate each hypothesis induced by means of a codelength measure. The codelength measure was used to indicate the over-fitting of a hypothesis. The approach was based on the assumption that over-fitting would be most obvious when a change in concept occurred. Although the computational overhead required by the MDL technique renders it impractical as a means for predicting menu items on a mobile phone, it was studied as means of comparing possible improvements to the approach presented in the previous chapter.

The MDL technique for inducing hypotheses was evaluated on the same data set and against the same learning approaches used in the previous chapter. An evaluation on both the real world and scenario-driven data indicated that the MDL technique did not perform as well as the approaches in which hypotheses were specified \textit{a priori}. Although the MDL technique did not perform as well as the specified hypothesis technique, it did perform better than the standard \textit{First Menu Item} approach that does not employ any form of menu prediction. The reason for the poor performance of the MDL technique compared with the specified hypotheses technique was discussed, and several methods for improving its performance suggested.
Chapter 5

Inducing Action Sequences on a Mobile Phone

5.1 Introduction

As discussed in Chapters 3 and 4, menu prediction provides a means for incorporating adaptive elements into a mobile phone user interface. In this chapter, a more extensive form of user interface adaptation is investigated. In particular, the approach investigated is based on inducing action sequences on a mobile phone interface. The basis of this approach is to identify those action sequences commonly repeated by a user. Once identified, a learning approach is used to predict which action sequence to present to a user in the form of an action-sequence shortcut. An action-sequence shortcut is defined as a macro or script that automates a sequence of commonly repeated actions or commands. As with the menu prediction approach, the aim of inducing action sequences is to adapt a user interface in a manner that reduces the effort required to interact with a mobile phone.

Sections 5.1 to 5.5 of this chapter focus on inducing action sequences on a mobile phone. A general discussion is provided of the issues surrounding action-sequence induction on a mobile phone, after which the action-sequence induction process is described in detail. Several learning approaches for inducing action sequences are introduced and discussed. These approaches are evaluated using data collected from several mobile phone users. The results
of this evaluation are then presented and discussed.

Section 5.6 of this chapter presents and discusses two criteria by which action-sequence shortcuts can be evaluated. In particular, we evaluate action-sequence shortcuts in terms of both efficiency and stability on data collected from several mobile phone users. These two measures are based on the notion that users prefer both predictability and stability in an interface.

Finally, in Section 5.7, a novel criterion for evaluating action-sequence induction is outlined. The novel criterion is based on combining measures of efficiency and stability. Several methods for combining these two measures are discussed. A comparison is made between two of these methods using data collected from several mobile phone users.

5.2 Predicting User Action Sequences

As discussed in the previous chapter, applying adaptive elements to a mobile phone user interface has the potential to improve the interface’s usability. The method of adaption presented in the previous chapter was based on predicting menu selections. A disadvantage of menu prediction is that it can be applied only to the menu artifacts on a mobile phone interface. To overcome this problem, this chapter extends the idea of predicting menu selections to the prediction of entire action sequences.

The goal of menu prediction was to ensure that an interface was both efficient and predictable. The action-sequence induction approach also intends to increase the efficiency of a user interface while maintaining its predictability. Efficiency is achieved by potentially saving a user from having to execute a sequence of actions. Predictability is achieved by augmenting the interface with action-sequence shortcuts. Furthermore, we restrict the space of possible action-sequence shortcuts that can be induced and present only one action-sequence shortcut at a time to a user.

The action-sequence shortcut induction approach can be conceptually split into two parts. The first part identifies those action sequences which are candidates to be automated by an action-sequence shortcut. Identifying action-sequence shortcut candidates involves identifying those action sequences that frequently recur and when automated would benefit a user.
5.2. PREDICTING USER ACTION SEQUENCES

The second part applies a learner to predict which action-sequence shortcut candidate to present to a user. Predicting which action-sequence shortcut candidate to present involves learning which action sequence a user will perform next. Predicting only one action-sequence shortcut was considered essential, since presenting all candidate action-sequence shortcuts would not be practical on a mobile phone screen. Furthermore, predicting the action-sequence shortcut removes the need for the user to decide between alternative candidate shortcuts.

In summation, the action-sequence shortcut induction approach investigated in this chapter has the following benefits:

1. An action-sequence shortcut allows a user to perform a sequence of actions without having to execute each action individually.

2. Inducing action-sequence shortcuts removes the need for a user to specify them manually.

3. Induced action-sequence shortcuts can adapt, removing the need for a user to maintain existing action-sequence shortcuts.

4. The context in which an action-sequence shortcut is used can be learnt, allowing an action-sequence shortcut to be presented only when it is needed.

Although inducing action-sequence shortcuts has the potential to save a user considerable effort, a number of issues need to be considered. These include identifying candidate action sequences to be automated via shortcuts; representing action-sequence shortcuts in an efficient yet expressive manner; learning which action-sequence shortcut candidates to present to a user; and evaluating different action-sequence shortcut induction approaches. These issues are now discussed in further detail.

5.2.1 Action-Sequence Shortcut Induction Issues

Identifying Action-Sequence Shortcut Candidates

Two broad approaches can be taken to gather data on a user’s actions. The first requires a user to specify the actions he or she performs. Many
programming-by-demonstration approaches require the user to specify their actions, and examples of such approaches are given in Section 2.2.2 on page 21. The second approach continuously and passively records which actions a user makes, inferring the user’s intentions. Note that these two approaches are similar in nature to the intended and keyhole notions of plan recognition described by Cohen et al. [22]. Although the second approach has the benefit of not requiring direct user involvement it does introduce several difficulties.

The first difficulty is inferring the user’s intention from their actions. If actions are observed via low-level events such as key presses, then without knowing the context in which these actions are made, it may be ambiguous as to what high-level actions a user intends. The second difficulty is determining which action sequences made by a user should become an action-sequence shortcut. In the area of plan recognition, a similar problem of identifying possible plans is often solved by determining the probabilities that certain action sequences will be observed and then selecting candidate plans based on these probabilities. An example of action-sequence probabilities for plan recognition in a user interface is presented in Waern [108]. Further approaches that use the probability of action sequences are presented in Section 2.2.2 on page 18.

Related to the issue of identifying candidate action sequences is the issue of determining which action sequences would make for “good” candidate action-sequence shortcuts. For instance, candidate action-sequences identified for an application that is rarely used would be of little value to the user. Furthermore, some action sequences could represent tasks that the user would never want automated. An example might be a deletion task, where the user would always want to manually confirm a deletion. One means of determining “good” candidate action sequences is by applying Blackwell’s attention investment model [13], which bases programming decisions on the cost, benefit and risk of “attentional” units to the user. Applying this model could provide a means of evaluating possible candidate action-sequence shortcuts.
Representing Action-Sequence Shortcuts

If a user manually defines an action-sequence shortcut, it can be expected that the user comprehends what the action-sequence shortcut will accomplish. However, the aim of action-sequence shortcut induction is to remove the need for the user to specify an action-sequence shortcut and instead induce it from observation. As such, it is not possible to assume that a user will necessarily understand what an induced action-sequence shortcut will accomplish. Therefore conveying the meaning of an induced action-sequence shortcut becomes important.

Since an action-sequence shortcut automates a sequence of actions, a relatively simple representation scheme is to list a shortcut’s actions. However, such a scheme suffers from two problems. Firstly, this representation scheme is not compact. Secondly, determining how to represent actions may be difficult. If actions are represented at the lowest level, such as key presses, then a user has to infer what the key presses will accomplish. Alternatively, if actions are represented at a higher level, such as tasks, then it requires the induction process to have some means of inferring such high-level tasks from key presses. The aim of many shortcut representation schemes is to employ a programming language that is easy for the user to comprehend. In particular, textual and visual programming languages have been studied as representational schemes. Burnett [15] discusses the strategies and issues of visual programming, as well providing examples of several techniques. Textual representation schemes have also been employed, spanning the variety of programming languages available. A discussion of the merits and drawbacks between textual and graphical programming languages has been widely explored [87, 85, 106].

An alternative representational scheme is to represent a shortcut by the state it intends to bring about. Such representation schemes are often used in the planning domain, in which a plan is represented by its goal state. However, with this representation scheme there is still the difficulty of determining how to represent the goal state of a shortcut. Furthermore, by describing only the goal state, information regarding possible side-effects of achieving the goal state may not be conveyed to a user.
Predicting Action-Sequence Shortcuts

Apart from identifying action-sequence shortcut candidates and representing them, a further issue involves the presentation of action-sequence shortcuts to a user. Supposing there are many action-sequence shortcut candidates, it may not be feasible to visually present all action-sequence shortcut candidates to a user. Furthermore, only one action-sequence shortcut will be appropriate to a user at any one time. The action-sequence shortcut induction approach investigated in this chapter employs a learner to predict which action-sequence shortcut candidate to present to a user. In this regard, the action-sequence shortcut induction process can be seen as an approach for predicting which action sequence a user will perform. Similarities exist in the plan recognition domain, in particular, the plan matching stage. In traditional plan recognition, the plan matching phase attempts to identify and explain a user’s current actions by matching their actions with a set of known plans. Various approaches have been employed to reason about which plans match a user’s intentions, including deductive approaches [56], probabilistic approaches [18] or a combination of both [91].

As with menu prediction, there exist several issues with employing a learner in this setting, including the computational burden that a learning approach would introduce on a mobile phone, the limited training examples provided by a user, the comprehensibility of the decisions produced by the learner, and concept drift. These matters are described in more detail in Section 3.2.2 on page 32.

Evaluating Action-Sequence Shortcut Prediction

As discussed in Section 3.2.1 on page 30, both efficiency and predictability are important characteristics of any adaptive user interface approach. These characteristics also apply to action-sequence shortcut induction. For instance, even though an action-sequence shortcut induction approach may increase the efficiency of an interface, if the action-sequence shortcut presented alters frequently then the unpredictable nature of the approach represents a cost to the user.
5.2.2 Addressing Action-Sequence Induction Issues

The action-sequence induction approach presented in this chapter is tailored for the mobile phone setting. We therefore do not address all of the action-sequence shortcut induction issues discussed in the previous section. We now discuss the issues that were addressed and the strategies used to address them.

The first strategy employed was the use of a regular expression language to identify candidate action sequences. In this approach, any action sequence that matched a given regular expression was considered a candidate action sequence. A regular language provided a simple and compact means of specifying a set of candidate action-sequences.

The second strategy employed was the use of a learning system to predict which action-sequence shortcut candidate to present to a user. The learning system attempted to identify the context in which a given action-sequence shortcut was applicable. In removing irrelevant action-sequence shortcuts, the learning system addressed the issue of presenting action-sequence shortcuts within the limited screen space of a mobile phone.

To address the issue of inducing efficient and stable action-sequence shortcuts, both efficiency and predictability measures were used to evaluate action sequence induction approaches. Efficiency was measured in terms of keypress savings for a user, while predictability was measured by the number of times over successive predictions that the induced action-sequence shortcut remained the same. In addition, several approaches for combining these measures were investigated.

5.3 The Learning Setting

In the setting of the mobile phone, we restricted our focus to inducing action sequences that would automate the making of outgoing calls and messages (SMS and MMS) on a mobile phone. Inducing action-sequence shortcuts for calls and messages was chosen as these represented the primary role and the most used functions of a mobile phone. In addition, the actions associated with making an outgoing communication can be observed passively and
directly, by recording the key-presses a user makes.

To make an outgoing communication a user must perform two basic actions - entering a recipient’s phone number and selecting the type of outgoing communication to be made. There are several types of outgoing communication that can be made on a mobile phone, such as a video-call, voice-call or messaging (SMS and MMS). Due to the relative simplicity of the candidate action-sequence shortcuts being considered, we represented action-sequence shortcuts as action-pairs. The approach presented is not restricted to action-pair shortcuts, with any length action sequence being applicable.

Figure 5.1 demonstrates the action-sequence shortcut induction approach when applied to predicting an outgoing communication.

![Figure 5.1: Action-sequence shortcut induction on a mobile phone interface. The action-pair of calling the contact named Adam is captured. This action-pair is presented to the user when the learner predicts the user intends to call Adam.](image)

### 5.3.1 Notation and Approach

We now describe the action-sequence shortcut induction approach in detail.

In the mobile phone environment, we assumed that there were a finite number of actions a user could make. These were denoted by the set $\mathcal{A}$.

Over time, a user’s interactions with a mobile phone would produce a sequence of actions. The sequence of actions made by a user is denoted $\mathbf{S} = (\alpha_1, \alpha_2, \ldots, \alpha_N)$, where $\alpha_i \in \mathcal{A}$. The actions in the sequence $\mathbf{S}$ are ordered according to the time they occurred. We denote $\alpha_i$ occurring before $\alpha_j$ as $\alpha_i < \alpha_j$. The length of a sequence was denoted $|\mathbf{S}|$ is the number actions in the sequence.
5.3. THE LEARNING SETTING

A special type of subsequence was considered, called an n-gram. An
n-gram of the sequence $S = (\alpha_1, \alpha_2, ..., \alpha_N)$, is an n-long subsequence of
consecutive actions. The $i$-th $n$-gram of the sequence is $(\alpha_i, \alpha_{i+1}, ..., \alpha_{i+(n-1)})$.
The total number of $n$-grams in the sequence $S$ is given by $\max(0, N - n + 1)$.

**Identifying Action Sequences**

Given a sequence of user actions, the action-sequence shortcut induction ap-
proach investigated in this chapter was concerned with identifying those sub-
sequences associated with making an outgoing communication. In the mobile
phone setting, making an outgoing communication requires the following two
high-level actions.

1. Entering or finding the recipient’s phone number.

2. Selecting the type of outgoing communication, i.e. voice-call, video-
call, SMS or MMS.

Given a sequence of user actions $S$, those $n$-grams where $n = 2$ called
*bi-grams*, needed to be identified.

Since the set of possible actions a user can make on a mobile phone is
given by $\mathcal{A}$, the set of possible bi-grams, is given by $\mathcal{B} = \mathcal{A} \times \mathcal{A}$. Given a
sequence of actions $S = (\alpha_1, \alpha_2, ..., \alpha_N)$, there are $N - 1$ bi-grams. Given
the $N - 1$ bi-grams in $S$, the action-sequence induction approach was only
concerned with those bi-grams that represented the actions associated with a
making an outgoing communication. We now present a means of identifying
these bi-grams.

**Constraining Possible Bi-grams**

The set of bi-grams in any sequence of actions was given by $\mathcal{B}$. To identify
a subset of $\mathcal{B}$ associated with making an outgoing communication, a regular
expression $\mathcal{RE}$ was used. The bi-grams in $\mathcal{B}$ matching the pattern $\mathcal{RE}$ was
denoted by $\mathcal{B}_{\mathcal{RE}}$. The use of regular language as a constraint specification tool
in pattern mining was first proposed in the SPIRIT system by Garofalakis
et al. [38].
The regular expression used to identify outgoing communication bi-grams is specified in the following derivation rules:

```
<outgoing communication> ::= <phone number> <communication type>
<communication type> ::= <video call> | <voice call> | <SMS> | <MMS>
```

This regular expression basically finds those action bi-grams whose first action is the selection of a phone number and whose second action is the selection of an outgoing communication type.

### 5.3.2 Action-Sequence Prediction as a Classification Task

Given a set of valid bi-grams \( B_{\text{RE}} \), the problem of action-sequence induction is one of predicting which valid bi-gram to present to the user as an action-sequence shortcut.

In order for a learner to predict which bi-gram to present, certain attributes of the learner’s environment must indicate which outgoing communication a user will perform. As with the menu prediction approach presented in the previous chapters, it was assumed that attributes describing a user’s actions and the interface’s current state would indicate the actions a user would perform. The values that these attributes take represent an example to the learner.

On every outgoing communication made by the user, the learner receives an example, \( x_t \in \mathcal{X} \), and the outgoing communication bi-gram made by the user \( b_t \in B_{\text{RE}} \). Together, the pair \( (x_t, b_t) \) form a labelled example at time \( t \). After \( N \) examples the learner will have observed a sample, where \( \text{sample}_N = ((x_0, b_0), (x_1, b_1), ..., (x_N, b_N)) \).

The objective of a learner is to predict which outgoing communication bi-gram \( b \in B_{\text{RE}} \) to present to the user when given some example \( x_t \) and a sample up until time \( t \), \( \text{sample}_{t-1} \). This is achieved by discovering a function \( f_t : \mathcal{X} \rightarrow B_{\text{RE}} \) based on the \( \text{sample}_{t-1} \), which is then used to predict the outgoing communication bi-gram at time \( t \). If the prediction is successful, then \( b_t = f_t(x_t) \).
5.3. THE LEARNING SETTING

Given that both the set $\mathcal{X}$ and $\mathcal{B}_{\text{RE}}$ are finite, the set of functions $\mathcal{F} = [\mathcal{X} \to \mathcal{B}_{\text{RE}}]$ is also finite. The functions $\mathcal{F}$ represent a hypothesis space $\mathcal{H}$ provided to the learner.

If a hypothesis $h \in \mathcal{H}$ is evaluated using $x \in \mathcal{X}$ and correctly predicts the outgoing communication bi-gram $b \in \mathcal{B}_{\text{RE}}$, then $h$ is said to agree with the example $(x, b)$. If a hypothesis $h \in \mathcal{H}$ predicts incorrectly, then $h$ is said to disagree with the example $(x, b)$. A hypothesis is considered consistent if it agrees with all examples in a sample. The objective of the learner is to find a hypothesis which is consistent.

5.3.3 Gain Model

When the learner selects a hypothesis that disagrees with an example, the learner has presented an action-sequence shortcut that will not save a user any effort. However, if the learner selects a hypothesis that agrees with an example, the learner has presented an action-sequence shortcut that can save the user a number of key presses. Note, to simplify the model it was assumed that there was no cost associated with presenting the wrong action-sequence shortcut. The function $G$ represent the gain associated with presenting an action-sequence shortcut, i.e:

$$G((x, b), h) = \begin{cases} Saving(b), & \text{if } b = h(x) \\ 0, & \text{otherwise}. \end{cases}$$

Where, $Saving$, is determined by the number of key presses an action-sequence shortcut saves a user, i.e. $ Saving : \mathcal{B}_{\text{RE}} \to \mathbb{N}$.

On-line Model

For action-sequence shortcut induction to be useful, it needs to predict an action-sequence shortcut every time a user intends to make an outgoing communication. As such, action-sequence shortcut prediction can be said to operate over a series of trials. In this regard, action-sequence shortcut induction operates in an on-line and supervised setting. This setting is described in more detail using the on-line prediction model introduced by Littlestone [72].
The objective of the learner in this on-line and supervised prediction setting is to maximise overall gain by improving predictive accuracy after each trial. Since accuracy relies on finding a consistent hypothesis, the learner must find a consistent hypothesis after as few trials as possible.

5.4 Action-Sequence Prediction Approaches

A number of learning approaches could be conceived for predicting which candidate action-sequence shortcut to present. A description of seven approaches is now provided, with a discussion of their properties.

5.4.1 No Shortcut

The No Shortcut approach does not induce any action-sequence shortcuts. This is important because it serves as a control for comparing the other action-sequence shortcut induction approaches. The No Shortcut approach is formally described in Algorithm 5.4.1.

<table>
<thead>
<tr>
<th>Algorithm 5.4.1 No shortcut algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Given: a set of unique and valid bi-grams $B_{RE}$.</td>
</tr>
<tr>
<td>2: In each trial $t \geq 1$:</td>
</tr>
<tr>
<td>3: do not present any shortcut.</td>
</tr>
</tbody>
</table>

5.4.2 Last Performed

The Last Performed approach presents as an action-sequence shortcut the last outgoing communication action bi-gram performed. If the last outgoing communication was text-messaging one’s wife, then the Last Performed approach would present an action-sequence shortcut representing the bi-gram of entering the wife’s phone number, and selecting text-message as the communication type. The Last Performed approach is formally described in Algorithm 5.4.2.

This approach uses a fixed strategy. Given a sequence of observed outgoing communication bi-grams $(b_0, ..., b_t)$, the fixed strategy at $t + 1$ is defined
Algorithm 5.4.2 Last Performed algorithm
1: Given: a set of unique and valid bi-grams $B_{RE}$.
2: In each trial $t \geq 1$:
3: present as a shortcut the bi-gram $\hat{b}_t = b_{t-1}$
4: receive bi-gram representing the outgoing communication made, $b_t$.
5: retain the bi-gram, $b_t$.

by the following function:

$$f((b_0, ..., b_t)) = b_t.$$

5.4.3 Most Frequent

The Most Frequent approach maintains a counter for each element in $B_{RE}$. Whenever an outgoing communication is made, the bi-gram matching the actions performed is incremented. The bi-gram with the highest counter is presented as an action-sequence shortcut. The Most Frequent approach is formally described in Algorithm 5.4.3.

Algorithm 5.4.3 Most Frequent algorithm
1: Given: a set of unique and valid bi-grams $B_{RE}$.
2: Associate a counter $c_b$ for each element of $b \in B_{RE}$.
3: Initialise these counters to zero.
4: In each trial $t \geq 1$:
5: present as a shortcut the bi-gram associated with highest valued counter.
6: $\hat{b}_t = \arg \max_{b \in B_{RE}} c_b$ †
7: receive bi-gram representing the outgoing communication made, $b_t$.
8: update the counter associated with the received bi-gram:
9: $c_{b_t} \leftarrow c_{b_t} + 1$

† ties are resolved by the index order.

Similar to the Last Performed approach this approach also uses a fixed strategy to predict a bi-gram. The fixed strategy at $t + 1$ is defined by the following function:
\[ f((b_0, ..., b_t)) = b_{\text{max}_{\text{counter}}} \]

A disadvantage of the Last Performed and the Most Frequent approaches is that the strategy used to predict bi-grams is fixed. As a consequence we are assuming the target concept \( c^* \) is also fixed. Furthermore, we are assuming that the target concept can be captured by a relatively simple strategy. In the real world it is unlikely that both these assumptions hold. However, an advantage of both approaches is their relatively small computational requirements. Furthermore, it may be that such simple strategies will be accurate enough of the time to produce a good result.

A problem specific to the Most Frequent approach is that the bi-gram predicted is based on observed frequency. However, if limited training data is available, the observed frequency of a bi-gram may be a poor approximation of its actual probability of selection.

### 5.4.4 Contingency Based

The Contingency Based approach determines whether a statistically significant dependency exists between the two actions in a bi-gram. A 2x2 contingency table is used to analyse the relationship between the two actions in a bi-gram. In particular, we want to know if the association between columns and rows is more than would be expected by chance. More formally, it is the task of testing the null hypothesis that the joint distribution of the cell counts in a 2-dimensional contingency table is the product of the row and column marginals [1]. This is the so-called model of independence.

There are many methods for testing the model of independence in a 2-dimensional contingency table. However, we assume a small sample size, and as such the Fisher’s exact test is used\(^1\). The bi-gram with the highest statistical significance of dependency will be presented as an action-sequence shortcut to the user.

As an example of how this approach operates, consider that the most frequently occurring action sequence consists of voice-calls to the contact name “John”, and the second most frequent action sequence consists of sending text-messages to “John”. Due to a large number of both voice-calls and

\(^1\)When the sample size is small, there is much scientific disagreement as to which method is most appropriate for testing the model of independence.
text-messages to the contact “John”, no dependency would exist between “John” and either of type of communication, i.e. voice-call or text-message.

We now discuss the Contingency Based approach in more detail.

**Contingency Table**

To represent a valid bi-gram (an element of $B_{RE}$) in the form of a statistical model, the features of the bi-gram are mapped to random variables. Since a bi-gram consists of two actions, a bi-gram can be represented by two variables $X$ and $Y$.

For each valid bi-gram a $2 \times 2$ contingency table is constructed. For the valid bi-gram $b_{RE} \in B_{RE}$, we have two actions $\alpha_1$ and $\alpha_2$. The actions $\alpha_1$ and $\alpha_2$ of $b_{RE}$ are mapped to two random variables $X$ and $Y$. In this case, $X$ and $Y$ are binary-valued, where the value of $X$ is determined by the presence of absence of the action $\alpha_1$ and the value of $Y$ is determined by the presence of absence of the action $\alpha_2$. The $2 \times 2$ contingency table is:

\[
\begin{pmatrix}
  n_{00} & n_{01} \\
  n_{10} & n_{11}
\end{pmatrix}
\]

Given a data sample, in our case a sequence of observed examples $((x_0, b_0), \ldots, (x_t, b_t))$, we construct a set of observed bi-grams $B$. For each $b \in B$, a valid bi-gram $b_{RE}$ contingency table is calculated as follows:

1. The cell $n_{00}$ represents the number of times $\alpha_1$ does not equal the first action in $b$ and $\alpha_2$ does not equal the second action in $b$.

2. The cell $n_{01}$ represents the number of times $\alpha_1$ does not equal the first action in $b$ and $\alpha_2$ equals the second action in $b$.

3. The cell $n_{10}$ represents the number of times $\alpha_1$ equals the first action in $b$ and $\alpha_2$ does not equal the second action in $b$.

4. The cell $n_{11}$ represents the number of times $\alpha_1$ equals the first action in $b$ and $\alpha_2$ equals the second action in $b$.

For instance, given the set of observed bi-grams:
Table 5.1: A Contingency table constructed from the example observations in $\mathcal{B}_{\text{example}}$.

<table>
<thead>
<tr>
<th></th>
<th>CALL</th>
<th>not CALL</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAM</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>not ADAM</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

$\mathcal{B}_{\text{example}} = \{\text{ADAM-CALL, SAM-SMS, ADAM-CALL, SAM-CALL, BOB-CALL, ADAM-SMS, BOB-SMS}\}$

The values of a $2 \times 2$ contingency table for the valid bi-gram $b_{RE} = \{\alpha_1 = \text{ADAM, } \alpha_2 = \text{CALL}\}$, based on the observed bi-grams $\mathcal{B}$ is shown in Table 5.1.

The joint frequency distribution of $X$ and $Y$ is described by the counts $\{n_{ij}\}$ for the data sample in the contingency table. The marginal distribution of $X$ and $Y$ are the row and column totals obtained by summing the joint frequencies.

**Sampling Plan**

The observed examples $((x_0, b_0), ..., (x_t, b_t))$, are assumed to be independently and identically distributed. On each iteration, the number of previously observed examples is fixed, therefore on each iteration we have a multinomial sampling\(^2\). The previously observed examples are classified according to the two variables $X$ and $Y$.

**Model**

A probabilistic model is used to describe the distribution of the population from which the data sample was drawn. We assume a population where there is no association between the two actions of a valid bi-gram $b$. Under this assumption, the joint distribution of the cell counts is the product of the row and column marginals. For instance, in Table 5.1, $P(X = 0, Y = 0) =$

\(^2\)A multinomial sampling is one in which the total number of sampled subjects is fixed.
5.4. ACTION-SEQUENCE PREDICTION APPROACHES

\[ P(X = 0)P(Y = 0) \]. The hypothesis of no association between \( X \) and \( Y \) is called a “model of independence”.

The “model of independence” serves as a null hypothesis. The “model of independence” hypothesis is tested by evaluating the fit of the model to the randomly drawn data sample. The measure of the model’s fitness is assessed to see if it is statistically significant.

Significance of Sampled Contingency Table

A Fisher’s exact test \([77]\) was used to evaluate the “model of independence” hypothesis. That is, all tables with the same row and column totals have their probability of occurrence calculated according to a probability distribution known as the hypergeometric distribution. The resulting tables are listed in order according to this probability. The probabilities are summed from each end of the list to the observed table. The smaller sum is the one-tailed \( P \) value. The two-tailed \( P \) value is the sum of the probabilities, under the null hypothesis, of all tables having a probability of occurrence no greater than that of the observed table.

A number of characteristics of the sampled data favoured the use of Fisher’s exact test over a goodness-of-fit statistic. Firstly, a large data sample was not always available. Secondly, the number of actions \( \alpha_1 \) could take (recipients’ phone numbers) was much larger than the number of actions \( \alpha_2 \) could take (type of outgoing communication). This difference between the two actions would result in a highly unbalanced contingency table. The advantage of Fisher’s exact test is its ability to give reliable results regardless of the distributional characteristics of the data sample. The implementation of the Fisher’s exact test used \([25]\), was based on the published algorithm by Mehta and Patel \([77]\).

The Contingency Based approach is formally described in Algorithm 5.4.4.

The Contingency Based approach is based on the idea that the actions in a bi-gram should be related in some way. This approach was considered since it was assumed that the bi-gram with the highest dependency between its two actions would not alter often. However, high dependency between two actions does not guarantee that the two actions are frequently used.
Algorithm 5.4.4 Contingency Based algorithm
1: Given: a set of unique and valid bi-grams $\mathcal{B}_\text{RE}$.
2: In each trial $t \geq 1$:
3: for each unique and valid bi-gram.
4: let $X$ be the first action and $Y$ the second action of the bi-gram.
5: construct the following 2x2 contingency table:

<table>
<thead>
<tr>
<th></th>
<th>$Y$</th>
<th>not $Y$</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>$a$</td>
<td>$b$</td>
<td>$A$</td>
</tr>
<tr>
<td>not $X$</td>
<td>$c$</td>
<td>$d$</td>
<td>$B$</td>
</tr>
<tr>
<td></td>
<td>$R$</td>
<td>$S$</td>
<td>$N$</td>
</tr>
</tbody>
</table>

7: there are $N$ bi-grams in the observed sequence $S$.
8: let $a,b,c$ and $d$ be the absolute frequency resulting form the double dichotomy of $N$ bi-grams according to the properties $X$ and $Y$.
9: given the null hypothesis (i.e. no association between $X$ and $Y$).
10: calculate the significance of the observed data (the total probability of observing data as extreme or more extreme if the null hypothesis is true)\(^\dagger\).
11: if statistically significant bi-grams exist
12: present as a shortcut the bi-gram with lowest total probability (dependent bi-gram) $\hat{b}_t$\(^\ddagger\).
13: else
14: do not present any shortcut.

\(^\dagger\) Fisher’s exact test was used to evaluate the null hypothesis given the observed data [77].
\(^\ddagger\) ties are resolved by the index order.
Therefore, this approach may produce stable action-sequence shortcuts, but may not provide great savings in terms of key presses when compared to other action-sequence shortcut prediction approaches.

5.4.5 C4.5 Based

Given a data sample, \(((x_0, b_0), ..., (x_t, b_t))\), the C4.5 Based approach constructs a C4.5 decision tree after every example. Whenever a user attempts to make an outgoing communication, the C4.5 decision tree is evaluated using the current example. The result of this evaluation is a classification, which in our setting is the prediction of a bi-gram. The predicted bi-gram is presented as an action-sequence shortcut. The C4.5 Based approach is formally described in Algorithm 5.4.5.

\begin{algorithm}
  \caption{C4.5 Based algorithm}
  \begin{algorithmic}[1]
  \item Given: an empty set of training examples.
  \item In each trial \( t \geq 1 \):
  \item construct a C4.5 tree classifier using training examples.
  \item receive current state of the environment \( x_t \).
  \item using \( x_t \), evaluate C4.5 tree classifier to predict bi-gram.
  \item present as a shortcut the predicted bi-gram \( \hat{b}_t \).
  \item receive outgoing communication bi-gram and the current state of the environment, \((x_t, b_t)\).
  \item add training example \((x_t, b_t)\) to set of training examples.
  \end{algorithmic}
\end{algorithm}

Using C4.5, action-sequence shortcut prediction becomes a multi-class classification problem. Each element of \( B_{RE} \) is a possible class and each element of \( X \) is a possible set of attribute values.

After every trial, the sequence of observed examples \(((x_0, b_0), ..., (x_t, b_t))\) are used to construct a C4.5 decision tree classifier. Given an example, the most recently induced decision tree is evaluated. The classification given to the example is used to determine the bi-gram to present as an action-sequence shortcut.
Advantages of C4.5 for Bi-gram Prediction

Learning models can be characterised along a number of lines [65]. We now discuss the elements of a decision tree classifier that are advantageous for bi-gram prediction. An important characteristic of a decision tree classifier is its ability to perform successfully without the need for many input parameters. This is desirable in the mobile phone setting, as the user is not expected to be aware of the mechanics of the learning approach. Furthermore, a decision tree classifier is generally computationally faster at evaluation time, since only a small subset of features need to tested.

Disadvantages of C4.5 for Bi-gram Prediction

Although decision tree classifiers have a number of advantages for bi-gram prediction, there are also a number of deficiencies with this approach. One important problem concerns the online nature in which training data is received. Although online decision tree classifiers exist [107], decision tree learning is usually performed on batches of training data. Moreover, the implementation of the C4.5 approach considered here requires a decision tree to be re-learnt, using the entire training data set, whenever a new training example arrives. The construction of a new decision tree for every new training example represents a large computational expenditure. Another issue regarding the use of decision tree classifiers is the incorporation of cost-sensitive learning and the handling of unbalanced classes. The handling of cost-sensitivity and unbalanced classes would seem essential in the action-sequence shortcut prediction setting, but it is not obvious how best to incorporate these features into a decision tree classifier.

5.4.6 Naïve-Bayes Based

Given a data sample, $((x_0, b_0), ..., (x_t, b_t))$, the Naïve-Bayes Based approach constructs a probabilistic model of the training data. In particular, a Naïve-Bayes probabilistic model is constructed. Given an example, the Naïve-Bayes probabilistic model is used to determine the bi-gram with the highest probability. This bi-gram is presented as an action-sequence shortcut. The
Naïve-Bayes Based approach is formally described in Algorithm 5.4.6.

**Algorithm 5.4.6 Naïve-Bayes Based algorithm**

1: Given: an empty set of training examples.
2: In each trial \( t \geq 1 \):
3: train Naïve-Bayes classifier using training examples.
4: receive current state of the environment \( x_t \)
5: using \( x_t \), evaluate classifier to determine probability of each bi-gram.
6: present as a shortcut the bi-gram with the highest probability \( \hat{b}_t \).
7: receive outgoing communication bi-gram and the current state of the environment, \((x_t, b_t)\).
8: add training example \((x_t, b_t)\) to the set of training examples.

\( \dagger \) ties are resolved by the index order.

The Naïve-Bayes approach is based on the use of a Naïve-Bayes classifier for action-sequence shortcut prediction. Briefly, the Naïve-Bayes classifier is a Bayesian network, in which the root node represents the class variable and the leaf nodes represent the attribute variables. Directed arcs from each leaf node to the root node represent the conditional probability of the class value, given a particular attribute value. There are no arcs between attribute nodes, since the attribute are considered to be conditionally independent of each other. The hypothesis space consists of the possible parameters of the model. The “best” hypothesis is found by calculating the most likely model given the observed data. This is known as the maximum a posteriori decision rule. Further details of Naïve-Bayes classification is given in Duda and Hart [30].

The Naïve-Bayes approach is similar to the C4.5 Based approach in that they are both multi-class classification problems, where each element of \( \mathcal{B}_{RE} \) is a possible class and each element of \( \mathcal{X} \) is a possible set of attribute values. The Naïve-Bayes approach also operates in a similar fashion to the C4.5 Based approach. After every trial, the sequence of observed examples \(((x_0, b_0), \ldots, (x_t, b_t))\) are used to train a Naïve-Bayes classifier. Given an example, the most recently trained classifier is used to determine the probability of each class, i.e. the probability of each bi-gram. The bi-gram with the highest probability is presented as an action-sequence shortcut.
Advantages of Naïve-Bayes for Bi-gram Prediction

A number of advantages can be identified with the use of a Naïve-Bayes classifier, many of which are due to its probabilistic output. Unlike the discriminant output of a decision tree classifier, a Naïve-Bayes classifier provides a posterior probability of class membership. A class posterior probability distribution has many advantages. In particular, uncertain predictions can be identified, allowing the option for the classifier to forgo making a prediction\(^3\). Furthermore, using the class posterior probability distribution, class membership can be ranked. Another advantage of a standard Naïve-Bayes classifier is its ability to train very quickly, requiring only a single pass on the training data set in order to compute probability mass or density functions.

Another advantage is the high-bias and low variance of a Naïve-Bayes classifier, that is, the assumption that the data set can be summarised by a single probabilistic distribution and that this is sufficient to discriminate between classes. This characteristic is especially important in the action-sequence shortcut prediction setting where few training examples are available.

Disadvantages of Naïve-Bayes for Bi-gram Prediction

A number of problems can be identified with the use of a Naïve-Bayes classifier in the action-sequence shortcut prediction setting. Firstly, the approach relies on the assumption that the attributes describing an outgoing communication are conditionally independent of one another, an assumption that cannot be guaranteed in the action-sequence shortcut prediction setting. However, attribute independence may not be a likely issue, as Domingos and Pazzani [29] observe that Naïve-Bayes outperforms several other classification approaches, even in domains where there is substantial attribute dependence. More importantly, they show that Naïve-Bayes does not need attribute independence to be optimal under zero-one loss.

Another issue regarding the use of Naïve-Bayes classifiers is the discretisation of the class posterior probability distribution. In the action-sequence shortcut prediction setting, only one prediction is given, and it is not obvious

\(^3\)Note, we do not allow this “reject” option in the action-sequence shortcut prediction setting.
5.4. ACTION-SEQUENCE PREDICTION APPROACHES

how best to decide this prediction, especially when the multiple classes have similar posterior probabilities.

5.4.7 Hybrid Approach

The Hybrid Approach combines the Most Frequent approach and the Naïve-Bayes Based approach. The Naïve-Bayes Based approach is used the majority of the time. However, if the probability of the bi-gram predicted is below a given threshold, the Most Frequent approach is used. In effect, the Naïve-Bayes classifier is invoked only if it is confident in classifying which bi-gram to present as an action-sequence shortcut.

The Hybrid Approach combines the Most Frequent and the Naïve-Bayes based approaches. It operates in the same way as the Naïve-Bayes based approach. However, if the outgoing communication with the highest probability has a probability below a given threshold, then the Most Frequent approach is used for prediction.

Advantages of Hybrid Approach for Bi-gram Prediction

The main advantage of the hybrid approach is its ability to employ the best characteristics of the Most Frequent and the Naïve-Bayes based approaches. In particular, the Naïve-Bayes approach is used only if it can predict an action-sequence shortcut with high probability. That is, the Naïve-Bayes is used only when it is highly confident of which action-sequence shortcut will be used. If the Naïve-Bayes is not confident, then the Most Frequent approach is used.

Disadvantages of Hybrid Approach for Bi-gram Prediction

A major disadvantage of the hybrid approach is the additional complexity it introduces. Not only does a Naïve-Bayes classifier have to be trained after every new action-sequence shortcut example, but a frequency count needs to be maintained on all possible action-sequence shortcuts. However, if implemented correctly, such additional complexity can be eliminated since the Naïve-Bayes approach will inherently capture the frequency of action-
Algorithm 5.4.7 Hybrid Approach algorithm

1: Given: a set of unique and valid bi-grams $B_{RE}$.
2: an empty set of training examples.
3: a threshold probability.
4: Associate a counter $c_b$ for each element $b \in B_{RE}$.
5: In each trial $t \geq 1$:
6: train Naïve-Bayes classifier using training examples.
7: receive current state of the environment $x_t$
8: train Naïve-Bayes classifier using training examples.
9: using $x_t$, evaluate classifier to determine probability of each bi-gram.
10: if the bi-gram with the highest probability, has a probability $\geq$ threshold
11: then present as a shortcut the bi-gram with the highest probability $\hat{b}_t$.
12: else
13: present as a shortcut the bi-gram associated with highest valued counter.
14: $\hat{b}_t = \arg\max_{b \in B_{RE}} c_b$
15: receive outgoing communication bi-gram and the current state of the environment, $(x_t, b_t)$.
16: update the counter associated with the received bi-gram:
17: $c_{b_t} \leftarrow c_{b_t} + 1$
18: add training example $(x_t, b_t)$ to set of training examples.

$\dagger$ ties are resolved by the index order.
5.5 EXPERIMENTAL EVALUATION

sequence shortcuts. However, even without additional complexity, it is not obvious where the cutoff point should be between the two approaches.

5.5 Experimental Evaluation

A mobile phone (smartphone) platform was used to investigate action sequence induction. In particular, the Nokia™ Series 60 smartphone platform was used. We restricted our focus to inducing action sequences that would automate the making of outgoing calls and messages (SMS and MMS) on a mobile phone. To make an outgoing communication on the Series 60 platform, two high-level actions (or bi-gram) can be identified. The first action consists of entering a recipient’s phone number, a task that can be performed in many ways. For instance, the phone number can be entered, selected via a speed-dial feature or looked up using the mobile phone’s Contacts address book application. The second action consists of selecting the type of outgoing communication to be made. Both actions require many key presses to complete. However, both actions can be observed and captured programmatically on the Series 60 platform. Figure 5.2 shows the process of making an outgoing communication on a Series 60 mobile phone platform. Figure 5.2(a) shows the first action, entering the recipient’s phone number. Figures 5.2(b) and 5.2(c) show the second action, either placing a call or writing a text-message.

5.5.1 Real World Data

We now present an evaluation of the action-sequence shortcut prediction approaches using data collected from several users.

Mobile Communication and Context Data Set

We used data collected and made publicly available by Raento, et al. [94] as part of the ContextPhone [96] project. The ContextPhone consisted of a prototype platform that allowed contextual information to be gathered on Nokia™ Series 60 mobile phones. Briefly, the information collected by this
application included: a user’s location based on cell tower ID; the active profile on a user’s phone; and the time, contact name, contact number, duration of all incoming and outgoing communications made on the phone. Data from 15 anonymous users was made available. These users were assigned identities from 1-6, 11-14, and 21-25. We used 13 of the 15 users, ignoring User 3 due to missing data and User 4 due to the small size of its data set. The time period over which the data was collected varied for each user. A full description of the data collected and the collection process is provided by Raento [95].

Data Set Characteristics

We now describe in more detail the data recorded for each user.

Two logs were kept for each user. The first log recorded information regarding a user’s location and their mobile phone profile. The second log recorded information regarding a user’s communication on their mobile phone. The logs contained stop and start markers, indicating when the data gathering software had been restarted.

The location and profile log recorded the following properties:

- network operator name
- location-area-code (LAC)
5.5. EXPERIMENTAL EVALUATION

- cell tower identifier (Cell-ID)
- profile identification: General, Silent, Meeting, Outdoor, Pager, Offline

The communication log recorded the following properties:

- communication type: call, sms
- communication direction: incoming, outgoing
- communication duration
- communication phone number (anonymised)
- communication contact name (anonymised)

Appendix B shows an excerpt from the location and profile log for User 1 and an excerpt from the communication log for User 1.

**Processed Data Set**

To evaluate the action sequence induction process, the location, profile and communication logs were processed in order to produce a suitable training data set. The combination of the communication contact name and communication type attributes served as a single outgoing communication bi-gram. An outgoing communication bi-gram became the class attribute in a training example.

Using the date-time attribute, the location and profile log was aligned with the communication log. The location and profile attributes occurring before an outgoing communication became additional attributes of a training example. In addition, attributes describing the last communication made were also included as attributes.

The format of a processed training example is shown below:

```
3, 3, 13, 12, 512, 448, 215, call, outgoing, ?, ?, ?, 200 – call
```
CHAPTER 5.  INDUCING ACTION SEQUENCES

Note, each attribute is delimited by a comma, the ? symbol represents a missing attribute value. The last attribute represents the outgoing communication bi-gram. The attributes in the training example (from left-to-right) are described below:

- communication date: day-of-the-week
- last communication made date: day-of-the-week
- communication date: hour
- last communication made date: hour
- last communication duration
- last communication phone number (anonymised to a number)
- last communication contact name (anonymised to a number)
- last communication type: call, sms
- last communication direction: incoming, outgoing
- location-area-code (LAC)
- network operator name
- cell identifier (Cell-ID)
- communication contact name (anonymised to a number) - communication type: call, sms

Table 5.2 summarises the training data set for each user. The users are re-labelled one through to 13, the Raento No. gives the original label from the Raento, et al. data set.
Table 5.2: Information on the training data sets used to evaluate the action-sequence shortcut prediction approaches.

<table>
<thead>
<tr>
<th>No.</th>
<th>Raento No.</th>
<th>Data Collection Period</th>
<th>Data Set Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>1</td>
<td>305 days</td>
<td>879</td>
</tr>
<tr>
<td>User 2</td>
<td>2</td>
<td>148 days</td>
<td>99</td>
</tr>
<tr>
<td>User 3</td>
<td>5</td>
<td>149 days</td>
<td>369</td>
</tr>
<tr>
<td>User 4</td>
<td>6</td>
<td>112 days</td>
<td>314</td>
</tr>
<tr>
<td>User 5</td>
<td>11</td>
<td>140 days</td>
<td>1402</td>
</tr>
<tr>
<td>User 6</td>
<td>12</td>
<td>112 days</td>
<td>112</td>
</tr>
<tr>
<td>User 7</td>
<td>13</td>
<td>110 days</td>
<td>204</td>
</tr>
<tr>
<td>User 8</td>
<td>14</td>
<td>104 days</td>
<td>249</td>
</tr>
<tr>
<td>User 9</td>
<td>21</td>
<td>93 days</td>
<td>285</td>
</tr>
<tr>
<td>User 10</td>
<td>22</td>
<td>93 days</td>
<td>598</td>
</tr>
<tr>
<td>User 11</td>
<td>23</td>
<td>88 days</td>
<td>176</td>
</tr>
<tr>
<td>User 12</td>
<td>24</td>
<td>88 days</td>
<td>429</td>
</tr>
<tr>
<td>User 13</td>
<td>25</td>
<td>93 days</td>
<td>122</td>
</tr>
</tbody>
</table>

5.5.2 Results

To evaluate each action-sequence shortcut prediction approach we replayed each user’s action sequence and simulated how each approach would have behaved on an actual mobile phone. Beginning with no examples of outgoing communications, the sequence of outgoing communication training examples is played. At each point-in-time the approach must guess an action-sequence shortcut. If it guesses correctly then a saving in key presses is made (we assume the bi-gram is used if possible). Clearly, the approach may only use examples before the current point in the played sequence.

Using the Contacts address book application, we determined that a voice-call action-sequence shortcut represented an average saving of seven key presses and a text-message action-sequence shortcut an average saving of 10 key presses.
\[ \text{Saving}(b) = \begin{cases} 7, & \text{if the second action in } b \text{ is a voice-call.} \\ 10, & \text{if the second action in } b \text{ is a text-message.} \end{cases} \]

Table 5.3 shows the percentage of key presses each action-sequence shortcut prediction approach could save each user. Note, the probability threshold for the Hybrid Approach was set to 0.9. This relatively high probability was selected to ensure that the predictions from the Naïve-Bayes classifier were used only if the classifier could discriminate significantly between possible outgoing communication bi-grams.

### 5.5.3 Discussion

In all cases, except for users 1, 2, 6, and 13, the Last Performed approach produced the highest saving in key-presses. However, even for users 1, 2, 6, and 13 the Last Performed approach performed comparatively well, as seen in Figure 5.3. Averaged over all users the Last Performed approach saved approximately 24 percent of the total number of key presses made. The Last Performed approach shows that the last communication made by a user has a strong bearing on the contact name and communication type a user will make next.

The Most Frequent approach performed relatively well for all users. However, except for User 2, Figure 5.3(b), it did not perform better than the Last Performed approach or the learning based approaches, the C4.5 Based approach and the Naïve-Bayes Based approach. This indicates that there is some value to employing more sophisticated strategies for predicting outgoing communication shortcuts.

The two approaches that employed a learning system, the C4.5 Based and the Naïve-Bayes Based approaches, followed closely behind the Last Performed approach, and for many users outperformed the Most Frequent approach. It is interesting to note that the performance of both approaches was comparable for all users, this is most obvious in Figure 5.3(e). This indicates that neither learning approach, decision-tree learner or probabilistic classifier, had a significant advantage in this setting. Furthermore, given that neither approach performed significantly better than the Last Performed
5.5. EXPERIMENTAL EVALUATION

Table 5.3: Percentage of key presses saved if action-sequence shortcuts are used when making outgoing communications.

<table>
<thead>
<tr>
<th>User</th>
<th>No Shortcut</th>
<th>Last Performed</th>
<th>Most Frequent</th>
<th>C4.5 Based</th>
<th>Naïve-Bayes Based</th>
<th>Contingency Based</th>
<th>Hybrid Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>28.43</td>
<td>17.88</td>
<td><strong>29.85</strong></td>
<td>28.53</td>
<td>22.09</td>
<td>25.65</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>39.55</td>
<td>44.49</td>
<td>44.49</td>
<td>43.50</td>
<td>0</td>
<td><strong>49.44</strong></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td><strong>19.08</strong></td>
<td>13.75</td>
<td>13.79</td>
<td>11.97</td>
<td>1.43</td>
<td>14.26</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td><strong>29.90</strong></td>
<td>21.78</td>
<td>22.58</td>
<td>21.70</td>
<td>2.52</td>
<td>24.52</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>13.56</td>
<td>6.31</td>
<td>12.18</td>
<td>10.26</td>
<td>3.43</td>
<td>7.34</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td><strong>30.62</strong></td>
<td>28.92</td>
<td>26.73</td>
<td>29.77</td>
<td>6.08</td>
<td>32.32</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td><strong>24.65</strong></td>
<td>15.77</td>
<td>20.85</td>
<td>22.72</td>
<td>5.79</td>
<td>18.34</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td><strong>19.72</strong></td>
<td>8.87</td>
<td>11.93</td>
<td>15.65</td>
<td>3.57</td>
<td>12.03</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td><strong>19.89</strong></td>
<td>8.39</td>
<td>17.74</td>
<td>17.35</td>
<td>8.34</td>
<td>11.86</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td><strong>16.53</strong></td>
<td>9.64</td>
<td>10.41</td>
<td>12.85</td>
<td>0.77</td>
<td>11.25</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>17.07</td>
<td>8.42</td>
<td>12.25</td>
<td>11.58</td>
<td>1.977</td>
<td>9.67</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td><strong>21.10</strong></td>
<td>11.12</td>
<td>18.24</td>
<td>15.13</td>
<td>0.67</td>
<td>12.68</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>20.24</td>
<td>20.68</td>
<td><strong>23.09</strong></td>
<td>14.55</td>
<td>0</td>
<td>19.15</td>
</tr>
</tbody>
</table>
CHAPTER 5. INDUCING ACTION SEQUENCES

(a) User 1

(b) User 2

(c) User 6
Figure 5.3: Percentage of key presses saved if action-sequence shortcuts are used when making outgoing communications for users 1, 2, 6, 13, and the average for all users.
approach, it would indicate that the outgoing communication made by the user was not highly dependent on the attributes considered.

The Contingency Based approach performed considerably worse than all the other approaches, as can be seen in Figure 5.3(e). The main disadvantage of this approach is the requirement that the actions in the outgoing communication bi-gram need to be statistically dependent on one another for a shortcut to be presented. Therefore, a certain amount of data needs to be observed before it is possible for this approach to predict a shortcut, and if actions are not dependent on one another then the approach will never produce a shortcut prediction. Although this approach performed poorly, it is interesting to note that for User 1, the approach performed comparatively well, as seen in Figure 5.3(a). This indicates that for certain users the approach can be of some merit. On the basis of how the Contingency Based approach works, these users could be characterised as making outgoing communications to a relatively small set of phone numbers, and consistently making the same type of outgoing communication to a particular phone number.

The Hybrid Approach performed consistently well for all users. However, if we compare it to the Naïve-Bayes Based approach by itself, we see that it did not perform significantly better or worse, as can be seen in Figure 5.3(e). This indicates that deferring to the Most Frequent approach (when the Naïve-Bayes Based approach fails discriminate highly between shortcuts) does not provide a significant advantage. However, like the Contingency Based approach, the Hybrid Approach performed well for one user, User 6, as seen in Figure 5.3(c). This suggests that the approach may be of merit to particular users, especially those for whom the Most Frequent approach performed well.

5.6 Evaluating Action-Sequence Induction

In the previous section, the action-sequence shortcut prediction approaches were empirically evaluated using real world data. Primarily, the predictive accuracy of these approaches was considered. If action-sequence shortcut prediction is viewed as a binary classification task, then we can evaluate the approaches in terms of sensitivity, specificity and positive predictive value. We now discuss these performance measures in more detail.
5.6.1 Binary Classification Task

Every time a user makes an outgoing communication, the action-sequence shortcut that represents it, is labelled positive. Implicitly, all those action-sequence shortcuts that do not represent a user’s action are labelled as negative. When the action-sequence shortcut induction process predicts an action-sequence shortcut, it predicts that this action-sequence shortcut will be positive, and that all other action-sequence shortcuts will be negative. This process can be viewed as a supervised binary classification task.

A confusion matrix (contingency table) can be constructed to evaluate the performance of an action-sequence shortcut prediction approach. Table 5.4 shows a generic contingency table for a binary classification task.

Table 5.4: Contingency table for a binary classification task.

<table>
<thead>
<tr>
<th></th>
<th>Predicted</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td>Actual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Positive</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>Negative</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

If the action-sequence shortcut induction approach predicts an action-sequence shortcut, and it is actually used, we say the prediction was a true positive. Implicitly, all action-sequence shortcuts not predicted are labelled negative. If the predicted action-sequence shortcut was a true positive, then all negative labelled action-sequence shortcuts are true negatives (i.e. we correctly predicted that they would not be used).

Alternatively, if the action-sequence shortcut induction approach predicts an action-sequence shortcut that the user does not use, we say the prediction was a false positive. Implicitly, all action-sequence shortcuts not predicted, are labelled negative. If the predicted action-sequence shortcut was a false positive, then one of the negatively labelled action-sequence shortcuts (the one representing what a user actually did) is false negative.

Figure 5.4 shows how action-sequence shortcut prediction can be evaluated as a binary classification task.

Note that the number of false positives equals the number of false nega-
Figure 5.4: Action-sequence shortcut prediction viewed as a binary classification task. Figure (a) shows the case for a correct prediction, (b) shows the case for an incorrect prediction.
tives. Furthermore, the number of true positives, false negatives, true negatives, and false positives equals the number of action-sequence shortcut candidates.

The most popular evaluation measures used in prediction or classification learning are classifier accuracy, sensitivity, specificity and positive predictive value.

**Accuracy**

Classifier accuracy measures the proportion of correctly classified instances; that is

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + TP + TN}.
\]

**Sensitivity - (Recall)**

Classifier sensitivity is the fraction of actual positive examples that are correctly classified; that is:

\[
\text{Sensitivity} = \frac{TP}{TP + FN}.
\]

The higher the sensitivity, the more actual outgoing communications are correctly predicted. The sensitivity in Figure 5.4(a) is 1 and Figure 5.4(b) is 0.

**Specificity**

Classifier specificity is the fraction of actual negative examples that are correctly classified; that is:

\[
\text{Specificity} = \frac{TN}{TN + FP}.
\]

The higher the specificity, the less actual outgoing communications are incorrectly predicted. The specificity in Figure 5.4(a) is 1 and Figure 5.4(b) is \(\frac{4}{5}\).
Positive predictive value - Precision

The positive prediction measures the reliability of the classifier’s positive predictions. In other words, it answers the question “given a predicted action-sequence shortcut, how likely is it that this action-sequence shortcut will actually be used?” It is calculated as follows:

\[ PPV = \frac{Sensitivity \times Prevalence}{Sensitivity \times Prevalence + (1 - Specificity) \times (1 - Prevalence)} \]

Note, that sensitivity and specificity are independent from the population, that is they are independent of the proportion of positive and negative action-sequence shortcuts. However, the positive predictive value depends on the population.

In our case, only one action-sequence shortcut can be positive, therefore the Prevalence is inversely proportional to the number of action-sequence shortcuts.

\[ Prevalence = \frac{1}{\text{Number of action-sequence shortcut candidates}} \text{ in every trial.} \]

The positive prediction value in Figure 5.4(a) is 1 and Figure 5.4(b) is 0. Since prevalence is inversely proportional to the number of action-sequence shortcuts in each trial:

\[ Sensitivity \equiv PPV \equiv (1 - \text{error-rate}). \]

5.6.2 Stability Measure

In the previous section we evaluated each action-sequence shortcut prediction approach in terms of its positive predictive value (PPV) or the fraction of times the correct action-sequence shortcut is presented \((1 - \text{error-rate})\). Since there is only one actual positive action-sequence shortcut in each trial, sensitivity is the same as these two measures. We assumed that a correctly
5.6. EVALUATING ACTION-SEQUENCE INDUCTION

predicted action-sequence shortcut would represent some gain in user interface efficiency. Gain was measured as the number of key presses the action-sequence shortcut saved a user.

Efficiency and predictability are two key aspects when mobile interfaces are designed. We assumed these two aspects should be reflected in the approach we used to select action-sequence shortcuts. We have measured action-sequence shortcut approaches in terms of efficiency, and we now propose a method for determining the predictability of an action-sequence shortcut approach. We measured predictability as the percentage of the time the predicted action-sequence shortcut remained the same from one trial to the next.

5.6.3 Results

Using the same action-sequence prediction approaches introduced in Section 5.4 and the same data set introduced in Section 5.5.1, we measured how constant the predicted action-sequence shortcuts were. That is, we measured the number of times the action-sequence shortcut presented to a user changed. Table 5.5 shows that for each user, the percentage of the time the induced action-sequence shortcuts remained constant with each approach. We assumed that an approach which produced stable action-sequence shortcuts was desirable, since action-sequence shortcuts that remain stable for long periods provide a more predictable interface.

5.6.4 Discussion

Table 5.5 shows that the Most Frequent approach and the Contingency Based approach both produced action-sequence shortcuts that were considerably more constant and stable than any other approach. The Most Frequent approach produced action-sequence shortcuts that were stable 94 percent of the time, or more, for all users. Similarly, the Contingency Based approach produced action-sequence shortcuts that were stable 97 percent of the time, or more, for all users. These results indicate that both these approaches are well suited in terms of stability of shortcut prediction. However, the Most Frequent approach has a significant advantage over the Contingency Based
Table 5.5: Percentage of the time the induced action-sequence shortcuts remain constant from one iteration to the next.

<table>
<thead>
<tr>
<th>User</th>
<th>No Shortcut</th>
<th>Last Performed</th>
<th>Most Frequent</th>
<th>C4.5 Based</th>
<th>Naïve-Bayes Based</th>
<th>Contingency Based</th>
<th>Hybrid Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>27.42</td>
<td>99.43</td>
<td>53.36</td>
<td>35.84</td>
<td>99.32</td>
<td>81.23</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>40.40</td>
<td>96.97</td>
<td>94.95</td>
<td>41.41</td>
<td>100</td>
<td>75.76</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>20.61</td>
<td>97.83</td>
<td>59.08</td>
<td>24.93</td>
<td>99.19</td>
<td>80.76</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>30.57</td>
<td>98.09</td>
<td>73.57</td>
<td>42.04</td>
<td>98.73</td>
<td>85.35</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>13.98</td>
<td>98.86</td>
<td>19.47</td>
<td>22.61</td>
<td>99.36</td>
<td>85.45</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>32.14</td>
<td>97.32</td>
<td>55.36</td>
<td>40.18</td>
<td>99.11</td>
<td>73.21</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>26.47</td>
<td>98.04</td>
<td>47.55</td>
<td>32.84</td>
<td>97.06</td>
<td>71.57</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>20.88</td>
<td>97.19</td>
<td>40.96</td>
<td>26.91</td>
<td>97.99</td>
<td>71.89</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>20.35</td>
<td>98.25</td>
<td>30.88</td>
<td>23.16</td>
<td>98.25</td>
<td>70.88</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>16.72</td>
<td>99.50</td>
<td>25.08</td>
<td>22.07</td>
<td>98.33</td>
<td>81.94</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>17.61</td>
<td>94.32</td>
<td>46.02</td>
<td>23.30</td>
<td>98.86</td>
<td>67.05</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>22.14</td>
<td>97.67</td>
<td>38.93</td>
<td>27.97</td>
<td>97.67</td>
<td>80.19</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>22.13</td>
<td>97.54</td>
<td>71.31</td>
<td>40.98</td>
<td>100</td>
<td>86.07</td>
</tr>
</tbody>
</table>
5.6. EVALUATING ACTION-SEQUENCE INDUCTION

(a) User 2

(b) User 4

(c) User 5
Figure 5.5: Percentage of the time the induced action-sequence shortcut remains constant from one iteration to the next for users 2, 4, 5, 9, 13, and the average for all users.
5.6. EVALUATING ACTION-SEQUENCE INDUCTION

approach. As noted previously, the Contingency Based approach requires a number of examples before a statistically significant action-sequence shortcut can be found (if at all), therefore in the short term it may not predict any shortcut at all. The Most Frequent approach does not suffer from this problem and is able to present a shortcut at all times.

An interesting result shown in Table 5.5 is that the Hybrid Approach performed considerably better than the Na"ive-Bayes Based approach for all users. This can be seen in Figure 5.5(f). This result is not surprising given that the Hybrid Approach employs the Most Frequent approach if it cannot predict a shortcut with a high level of probability. The large difference between these two approaches, most notable in Figure 5.5(c), would indicate that the Hybrid Approach deferred to the Most Frequent approach frequently.

Another interesting feature in Table 5.5 is the relatively poor performance of the Last Performed approach and the learning based approaches, the C4.5 Based and the Na"ive-Bayes Based approaches. This is most noticeable for User 5, Figure 5.5(c), and for User 9, Figure 5.5(d). However, for users 2, 4, 13 (Figure 5.5(a), Figure 5.5(b), Figure 5.5(e) respectively), the C4.5 Based approach outperformed the Na"ive-Bayes Based approach. This suggests that the decision tree learner has an advantage over the probabilistic classifier in this setting.

When comparing Table 5.3 with Table 5.5, we see that the approaches that saved the most key presses, the Last Performed approach, the C4.5 Based approach, and the Na"ive-Bayes Based approach, did so by producing action-sequence shortcuts that changed often. In order to select the action-sequence shortcut prediction approach that is both efficient and predictable, a measure is needed that combines these two properties. In the next section, we state and discuss three possible ways of combining efficiency and predictability measures.

\footnote{The inability to predict a shortcut may not be a disadvantage if we take into account the distraction caused by presenting an unwanted shortcut to a user.}
5.7 Combining Efficiency and Predictability

In the previous section we presented the efficiency and predictability of the action-sequence shortcut prediction approaches, shown in Table 5.3 and Table 5.5 respectively. In this section we state and discuss three possible ways of combining action-sequence shortcut efficiency and predictability measures.

In the previous section, a measure of efficiency was taken to be the percentage of key presses saved. Furthermore, the measure of predictability was the percentage of times an action-sequence shortcut remained the same over a series of trials.

5.7.1 Average

The simplest method of combining the measures of efficiency and predictability is to take the simple average of the two measures, i.e.: \((p + v)/2\). Let \(p\) be the efficiency of each action-sequence shortcut prediction approach averaged over the 13 users (see Figure 5.3(e)). Let \(v\) be the predictability of each action-sequence shortcut prediction approach averaged over the 13 users (see Figure 5.5(f)). The average of these two measures is shown in Figure 5.6.

One possible variation of this measure would be to have a weighted average. This would enable the system designer to reflect the importance of saving key presses compared to reducing the number of times the action-sequence shortcut changes.

5.7.2 Harmonic Mean

Another approach would be to use a harmonic mean for combining the measures of efficiency and predictability, i.e.: \(2pv/(p + v)\). The harmonic mean prevents one measure from performing well at the expense of the other. Note, the harmonic mean is used by information retrieval researchers for combining precision and recall into the F1 measure [97]. Again, let \(p\) be the efficiency of each action-sequence shortcut prediction approach averaged over the 13 users (see Figure 5.3(e)). Let \(v\) be the predictability of each action-sequence shortcut prediction approach averaged over the 13 users (see Figure 5.5(f)). The harmonic mean of these two measures is shown in Figure 5.6.
5.7. COMBINING EFFICIENCY AND PREDICTABILITY

5.7.3 Threshold

A third approach would be to use a thresholding technique. If a threshold was applied to the stability measure $v$, then any action-sequence shortcut with stability above this threshold would simply be compared in terms of efficiency or the measure $p$. Similarly, if a threshold was applied to the efficiency measure $p$, then any action-sequence shortcut with efficiency above this threshold would simply be compared in terms of stability or the measure $v$. The thresholding technique provides a means of specifying a tolerance for a certain amount of either instability or inefficiency in an action-sequence shortcut.

5.7.4 Results

The real world data presented in Section 5.5.1 was used to evaluate the Average and Harmonic Mean techniques. For each action-sequence shortcut prediction approach, the data was averaged over the 13 users. Figure 5.3(e) shows the number of key presses saved for each action-sequence shortcut prediction method averaged over the 13 users. Figure 5.5(f) shows the stability of each action-sequence shortcut prediction method averaged over the 13 users. Figure 5.6 compares two ways of combining the information in Figure 5.3(e) and Figure 5.5(f).

5.7.5 Discussion

Figure 5.6 shows that the Average and Harmonic Mean techniques rate the different methods for predicting action-sequence shortcuts very differently. The largest difference can be seen in the No Shortcut method, with the Average technique rating it at 50 percent and the Harmonic Mean technique rating it at 0 percent.

The Average technique’s high rating of the No Shortcut method is somewhat misleading considering that the method does not predict a shortcut. In fact the No Shortcut method will be 100 percent consistent, but will never save the user any key presses. The No Shortcut method serves as an extreme example of how both the efficiency and stability measures need to be taken
CHAPTER 5. INDUCING ACTION SEQUENCES

Combining Key-Press Savings Measure and Consistency Measure

Figure 5.6: Average and Harmonic Mean approaches for combining the key press saving measure (efficiency) and the consistency measure (stability).

into account if meaningful results are to be inferred.

The Last Performed method performs equally well with both the Average and Harmonic Mean techniques. From this we can gather that the Last Performed method provides a good balance between action-sequence shortcut efficiency and stability.

If the Threshold approach was employed, and set to allow action-sequence shortcuts with stability of 95 percent or higher, then only the Most Frequent, No Shortcut, and Contingency approaches would be considered. Of these the Most Frequent would perform the best.

It would be interesting to investigate which evaluation approach is most in-line with user preferences. This would help inform how to rank the evaluation approaches, however, this is beyond the scope of this thesis.
5.8 Conclusion

This chapter introduced an approach for predicting action-sequence shortcuts on a mobile phone interface. The approach was a natural extension of the menu prediction approach presented in Chapter 3.

The action-sequence shortcut prediction approach was conceptually divided into two parts. The first part involved identifying relevant action-sequence shortcut candidates. To achieve this a regular expression was used to specify those action sequences of interest. The second part involved the use of a learning approach to predict which action-sequence shortcut to present to a user. We restricted our focus of attention to inducing action-sequence shortcuts, bi-grams, for outgoing communications on a mobile phone.

Different learning approaches for action-sequence shortcut prediction were evaluated on a data set acquired from the ContextPhone project [96]. Two measures were identified as useful in evaluating any action-sequence shortcut prediction approach. The first measure considered the efficiency of an approach in terms of predictive accuracy. The second measure considered the stability of the action-sequence shortcuts predicted over time. Of all the action-sequence prediction approaches evaluated, the Last Performed approach performed best in terms of predictive accuracy. However, the Most Common approach performed well in terms of both predictive accuracy and stability.
Chapter 6

Conclusion

6.1 Summary

The mobile phone user interface is significantly restricted by computational and physical size limitations. As a result, mobile phones are often considered to have poor user interfaces. This thesis has sought to address mobile phone user interface issues through the use of adaptive user interface elements. In applying adaptive user interfaces to mobile phones, several important considerations were explored, primarily involving the use of machine learning as a means of achieving adaptation. The specific contributions of this thesis are summarised below. Possible future research directions are then discussed.

6.1.1 Menu Item Prediction

Predicting a user’s next menu item selection was proposed as a non-intrusive way of achieving an adaptive user interface. Instead of modifying the menu, the menu item the user will select was predicted and highlighted, thus reducing the number of key presses required by the user. Menu item prediction represented a means of increasing efficiency while maintaining the default ordering and content of a menu.

Menu item prediction was presented from a machine learning viewpoint, with several unique characteristics. These were the ability to learn in an environment with a rapidly shifting target concept, using limited computational resources and limited training examples.
A learning system that used a restricted hypothesis space, together with a windowing technique, was presented as a means of learning in this unique environment. A manually specified and restricted hypothesis space allowed the learner to be biased heavily towards concepts that were likely to occur in the mobile phone environment. Furthermore, this bias allowed the learner to perform well with relatively few training examples. To address the problem of a rapidly shifting target concept, a windowing technique was used. This technique identified the hypothesis that minimised disagreement over a window of recent training examples. Particular focus was applied to an optimum windowing strategy in this environment. Two basic windowing strategies were presented. The first used a fixed size window. The second did not use a fixed size window, instead it considered all recent examples, and selected the hypothesis that had been consistent for the longest run into the past. This strategy was called the Most Recent Correct Hypothesis approach.

A model of concept drift was presented. It was parameterised by the rate of concept switching and by the probability of selecting the target concept from a class of concepts. Using this model of concept drift, a theoretical analysis of the learning-based approaches was performed. An abstract problem domain, called Being-the-Leader, was introduced to simplify analysis of the learning-based approaches. The difference between the hypothetical problem domain and the actual problem domain, called Follow-the-Leader, was determined. This allowed a comparison to be made between the learning-based approaches. The comparison showed the advantage of the Most Recent Correct Hypothesis approach when compared to fixed window learning-based approaches.

The learning-based approaches for menu item prediction were compared with several simple strategies. These strategies included, reordering menu items by their selection frequency, using the last menu item selected as the next menu item predicted, and using the most selected menu item as the next menu item predicted. Experimental results on data collected from two users indicated that the relatively simple strategy of using the most selected menu item performed well for one user but poorly for the other. The only approaches that performed consistently well for both users were the learning-based approaches that used a restricted hypothesis space and some form of
6.1. SUMMARY

Windowing technique. These results demonstrate the need for a predictive approach to incorporate learning if it is to be useful for a wide range of users. The consistent and good performance of the learning-based approaches was also seen in the scenario-driven data. It must be noted that the Most Recent Correct Hypothesis approach did not perform significantly better than the fixed window approaches when evaluated on actual user data. However, only a limited amount of training data was available.

Menu item prediction was extended to remove the restriction of a manually specified hypothesis space. Instead of a specified hypothesis space, an approach using the minimum description length (MDL) principle was employed to identify concept shifts online. The MDL principle was used to evaluate the code length of consecutive hypotheses. Changes in concept were identified by increases in the relative code lengths of hypotheses over time. Using the data collected from two users, the performance of the MDL approach was compared with the approach that used manually specified hypotheses. In general, the MDL approach performed badly for one of the users, and roughly the same or better for the other user. The results suggest that an approach which combines manually specified hypotheses and the MDL approach may perform better than either approach individually.

6.1.2 Action Sequence Prediction

To achieve greater savings in terms of key presses, an action sequence prediction approach was presented. The basis of action sequence prediction was to predict a user’s future actions and present this as a macro to the user. The process of predicting action sequences was conceptually divided into two parts, the first concerned with identifying relevant action sequences and the second with predicting which action sequences to present to the user as a shortcut. Identifying relevant action sequences was accomplished by the use of a regular expression that allowed action sequences to be specified in a general and compact way. Several learning-based approaches were considered for predicting which action sequence to present to the user.

As with menu item prediction, several action sequence prediction approaches were considered. An experimental evaluation of these approaches
was performed using data collected from 13 users over a period ranging from three to 12 months. The evaluation showed that the simple strategy of using the last performed action sequence as the next action sequence predicted could outperform decision tree and statistical learning approaches. In keeping with the assumption that users prefer predictable user interfaces as well as efficient interfaces, a novel evaluation measure was considered that measured the rate at which predicted action sequences changed.

When the action sequence prediction approaches were evaluated using the stability measure, the approach that produced the most stable predictions was the strategy of predicting the most frequently occurring action sequence. Given the stability of predicting the most frequently occurring action sequence, and the predictive accuracy of the learning-based approaches, a hybrid approach was presented. The hybrid approach predicted the most frequently occurring action sequence most of the time, deferring to a Naïve-Bayes approach only when the action sequence predicted by the Naïve-Bayes approach could be made with high probability.

Several methods for combining the predictive accuracy and stability measures were presented. The first method took the average of the two measures, and the second method took the harmonic mean of the two measures. Using the harmonic mean showed the most promise as it acted to balance the two measures. That is, it did not perform well in one measure at the expense of the other. The hybrid action sequence prediction approach clearly performed better than the other action sequence prediction approaches when evaluated by the harmonic mean.

### 6.1.3 Overview

Although predictive accuracy is often seen as an important property to optimise from a machine learning viewpoint, in the context of adaptive user interfaces we have assumed that a user desires both efficiency and predictability in an interface.

The results from the experimental evaluation of both menu item prediction and action sequence prediction have highlighted the fact that although machine learning techniques can improve predictive accuracy, they do not
necessarily produce predictions that are stable over time. In fact, some of the relatively simple prediction strategies outperformed learning techniques in terms of stability. Furthermore, given the computational cost required by these learning-based approaches, it might be that such simple strategies would be better suited to the adaptive user interface setting. However, it is also apparent from the experimental evaluation that the relatively simple strategies are not ideal when given a wide range of user behaviour. The learning-based approaches have the significant advantage of being able to adapt to individual users. An important conclusion from the experimental evaluations is that the characteristics that make simple strategies effective, combined with the adaptive advantage of the learning-based approaches, can achieve both high efficiency and predictability. The hybrid approach used for action sequence prediction is a prime example of how this can be accomplished.

6.1.4 Further Research

There are a number of directions in which this research could be extended, and we now remark on some of them.

One area of further research surrounds the shortcomings of the Most Recent Correct Hypothesis approach. A disadvantage of this approach is the need to minimise disagreement between the hypothesis space and the window of recent training examples. As the number of hypotheses increase, this approach becomes less feasible to implement. One possible solution mentioned by Helmbold and Long[42] is the use of a randomised algorithm that with high probability approximately minimises disagreement over the hypothesis space.

The model of concept drift presented in this thesis did not incorporate the notion of repeating concepts. An approach that identified repeating concepts and used such information to improve concept tracking could be considered in future work.

Another important area in which this research could be extended would be in the customisation of adaptive user interfaces for similar user groups. For instance, as seen in the data collected from the two users, one user pre-
dominantly used the mobile phone to send text messages (SMS), while the other user performed more varied actions. It may be that different prediction approaches for different user groups would perform better than the one overall approach. Applying a learning approach to identify similar groups of users, and then customising the prediction approach accordingly, could prove successful.

The evaluation of the menu and action sequence prediction approaches was based on a relatively small dataset. One possible extension would be to collect data from a wider variety of mobile phone applications and from a wider set of users. Furthermore, it would be beneficial to obtain feedback on whether users would use such features if they were incorporated into their mobile phones.

Gathering data on the cognitive effect of menu item prediction could also prove beneficial. We analysed menu item prediction in terms of discrete and absolute loss, but more complex loss functions that more accurately matched the type of loss psychologically experienced by a user could be considered. The performance and analysis of our menu item prediction approaches under differing loss functions would then need to be examined.

The use of information-theoretic approaches in identifying stable concepts, such as the minimum description length principle presented in this thesis, could be researched further. In particular, the benefit of seeding such approaches with manually specified hypotheses to improve their initial learning rate could be investigated.
Appendix A
A.1 Fitts’ law

Fitts’ law[37] is a model of human psychomotor behaviour. It is used to model the time it takes to point at something, based on the size and distance of the target object. Fitts’ law states that the time required to move from a starting position to a final target area, is a function of the distance to the target and the size of the target. It has been applied to the evaluation of input devices that allow a user to click on objects on a computer screen. Formally, Fitts’ law can be stated as follows:

\[ MT = a + b \log\left(\frac{2A}{W} + c\right) \]

MT is the movement time; a and b are empirically determined constants; A is the amplitude or distance of movement from start to target centre and W is the width of the target.

Fitts’ law states that movement time is a logarithmic function of distance when the target width is held constant and a logarithmic function of the target width when distance is held constant.

When applied to interface design, Fitts’ law can be interpreted as stating that more frequently used targets, such as buttons, should be made larger and placed closer to the users starting position. Finally, Fitts’ law only models untrained human movements in one direction.

A.2 Goals, Operators, Methods, and Selection Rules Model

GOMS is a task analysis model initially introduced by Card et al.[16]. It is used to represent the knowledge required to accomplish a task on a computer system. The knowledge required to complete a task is modelled in terms of Goals, Operators, Methods and Selection rules.

Goals
A goal is the task the user is intending to accomplish. Goals can be defined at various levels of abstraction. High-level goals could be the to send an SMS and a low-level goal could be to add a recipient to an SMS message. A higher-level goal can be defined in terms of a hierarchy of subgoals.

**Operators**

Operators are the actions performed to bring about a goal. Operators are atomic elements in the GOMS model. The model assumes each operator takes a fixed amount of time to execute, regardless of the context in which it is performed.

**Methods**

A method is a sequence of operators or sub-goals that accomplish a goal. A method specifies the procedural steps to accomplish a goal. For the one goal, many methods can exist. For instance, a user may delete SMS messages by individually deleting each message or may mark each message then delete all marked messages.

**Selection rules**

Selection rules specify which method should be used to accomplish a goal. A selection rule determines which method is appropriate in a given context and is usually specified by an “IF ... THEN ...” statement.

There are several variation to the generic GOMS model presented above. These are the Keystroke-Level Model (KLM), the Natural GOMS Language (NGOMSL), and the Cognitive-Perceptual-Motor GOMS (CPM-GOMS).

A GOMS model is used to analyse the usability of an interface. It can be used to predict a user’s performance in terms of interaction time in a new interface, or conversely, quantify the interaction time in an existing interface. Finally, a GOMS model does not account for user errors, user fatigue, or user differences.
Appendix B
Context-1.xml: An exert from the location and profile log for user 1.

```xml
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE events SYSTEM 'log.dtd'>
<events>
    <event>
        <datetime>20030221T171303</datetime>
        <location>
            <start />
            <location.value>
                <location.network>0</location.network>
                <location.lac>0</location.lac>
                <location.cellid>0</location.cellid>
            </location.value>
        </location>
    </event>
    <event>
        <datetime>20030221T171303</datetime>
        <profile>
            <start />
            <profile.value>
                <profile.id>Kokous</profile.id>
            </profile.value>
        </profile>
    </event>
</events>
```

Comm-1.xml: An exert from the communication log for user 1.

```xml
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE events SYSTEM 'log.dtd'>
<events>
    <event>
        <datetime>20030213T173147</datetime>
        <communication>
            <comm.sms />
            <comm.incoming />
            <comm.duration>0</comm.duration>
            <comm.number>18</comm.number>
            <comm.contact_name>11</comm.contact_name>
        </communication>
    </event>
    <event>
        <datetime>20030213T173147</datetime>
        <communication>
            <stop />
        </communication>
    </event>
</events>
```
Appendix C
Figure C.1: *Follow-the-Leader* variant of menu prediction. Discrete loss is shown versus fixed window size. Discrete loss is averaged over a 100 runs. The fixed window size varies from size 1 to size 1200.

Figure C.2: *Being-the-Leader* variant of menu prediction. Discrete loss is shown versus fixed window size. Discrete loss is averaged over a 100 runs. The fixed window size varies from size 1 to size 1200.
Bibliography


BIBLIOGRAPHY


