Machine Learning for User Interfaces
Alex Baumann
Good user interface design requires a strong commitment to the well established principles of good interface design, as well as some creativity to allow the design to stand out among others, or perfectly suit the need. Although much research has been conducted related to human-computer interaction, and user interface designs, we still do not know everything about good interface design, nor can we perfectly tailor an interface to suit all individuals who may use it. For this reason, machine learning and artificial intelligence can be used to monitor the actions of users, and their successes and failures, in order to dynamically determine how to modify the interface and/or gain future knowledge.

This learning and collecting of data could be used in several ways, including dynamically altering the layout, structure and appearance of the interface to best suit the actions of the user, automating repetitive tasks, and providing direct feedback to the programmer about how successful the interface is, allowing them to alter further releases of the interface. The first method of dynamically altering the layout, herein referred to as adaptive user interfaces, “is an interactive software system that improves its ability to interact with a used based on partial experience with that user” (Langley). This is often implemented very naively by some interfaces as warnings or suggestions within the interface that are prompted by certain actions the user takes. These systems do not often, however, learn from the subsequent choices of the user in order to determine the best choice to make in the future. The use of data collected while the user interacts with the system provides several advantages, including a steady source of training data and updates to the knowledge base after each interaction (Langley). Though the use of
learning on constantly collected data would require more computation as the knowledge base grows, the training sets can be reduced in size and complexity to allow rapid learning that occurs outside of the user’s knowledge of it. It is important that the learning must all take place without the user having to wait or otherwise conform to restrictions set by the learning. This is because the learning is intended to create a better interface for the user, and should in no way hinder their current interaction with the interface.

This form of learning can be used to tailor the interface to show different levels of complexity for different levels of user, as determined by the learning process. If the learning determines that the user is well skilled and knows what they are doing with the limited interface, the program can choose to show the user more advanced features. This learning process may also be used to determine what the user generally picks, or what it determines to be the best choice or set of choices, as based upon what the user has done in the past, and the knowledge in the knowledge base. This means that if the user always picks one or two choices out of a set of larger choices, these should always be preferred, and shown first and foremost to the user. Similarly, if the knowledge base contains information that one or more choices will be more suited for the current task, as determined by their success in the past, etc., then these choices should also be highlighted to the user as being the preferred choices.

The layout of the interface is another factor that could change as based upon the learning process. If, for example, a user tends to use a set of interface controls all the time and these controls appear in opposite sides of the interface, these controls could be grouped closer together to allow them to be accessed more easily. This, however, has
many drawbacks, since users generally get accustomed to an interface (or any other thing within their lives), and expect it to be the same when they try to do the same thing another time. Microsoft Windows uses this technique with its menus, which show a shortened list of menu items that include those choices that were accessed last. Though this can be very convenient due to the hiding of controls that are often unused, it can also be disorienting, due to the fact that menu items that one becomes accustomed to the location of, suddenly fall in a different place due to subtle changes in activity. This can perhaps be alleviated by only changing such interface layouts after a significant amount of usage, such as many choices of a particular menu item before it moves, rather than simply one.

One advantage that shifting menu items has over the moving of controls within an interface within a two-dimensional space, is that these menu items are presented as a list, and humans can more easily relate the position of an item relative to the rest of the list, even if the list changes, than if the item moves freely in two-dimensional space. In other words, even when these menu items move, they still remain in the same position in the list relative to the other items in the list, such that items that are at the end of the menu always appear at the end of the menu even if other items are shown, and do not move above other items suddenly. In order for interfaces to adapt their layout, they must do so in a way that does not disorient the user. The feedback from such changes to the interface can also be used in the learning process, perhaps to determine the best ways in which to move items within a layout that do not disorient the user.

Though it may be difficult to design an interface that allows movement of the controls, based upon the user’s actions, that is not disorienting, the best approach to this
situation would likely be to allow the placement of the controls to be learned by the system. This means that the placement of the controls are not only determined by the actions of the users, but also by the user’s response to the new location of the control. Since a user may not be happy with the location of the new control, it may have to move back to its original position, or one determined to be better by the user. Though this may still be disorienting to the user, this effect of changing placement of controls will likely diminish as the system learns both where the user wants controls, and how to and not to move controls within the interface.

Another aspect of computing that machine learning can help to alleviate are the repetition of tasks that are required to perform standard actions within an interface. Repetitive tasks are one aspect of life that many users would rather do without. For this reason, computers are often used, and are perfectly suited for the repetitive execution of well-defined tasks. These tasks, however, cannot always be well-defined by the programmer or user without significant effort (Witten, Manning & Maulsby). By learning the tasks that users do and how they do them, rather than having the user define a task for the computer to do, a learning system is able to model the user’s behavior and attempt to duplicate it. Clearly this learning process must be designed in a manner that takes into account the unpredictable nature of human actions, and can see the repetition within complex and often different tasks. The learning system must gather data from example actions the user performs and pull out what is relevant to the task at hand in order to avoid having to clarify the action with the user (Witten, Manning & Maulsby). These repetitive tasks that a user would want to have automated can be presented to the user in various ways. One way is by providing suggestions to the user about actions that
it appears they are trying to do and asking if they want to have that task performed for them. Another is to provide a list of available repetitive tasks that can be performed when needed.

The information collected by the background machine learning system can also be invaluable to the programmer of the interface. For instance, the programmer could use this information to determine how well the interface is working for the user, and can be used to test the accuracy of predictions about user actions. The system could be setup to help guide the user by suggesting what they would like to do based upon their actions, and the knowledge base’s recorded actions, and perform the preprocessing needed to achieve this suggestion (Motoda & Yoshida). The application could not only begin to learn what suggestions are viable for this particular user, but could also be sent to the programmer so that they could analyze the suggestions and user actions in order to alter the program structure in future releases. Though the programmer cannot provide accurate solutions for all users, they can alter their learning application to take advantage of what they can learn from the learned data. This means the programmer can alter the way in which suggestions are made and the method in which learned data is analyzed, to better suit the user-base in general. One example is this of error messages, in which the programmer is often the writer of the text within these, with very little of it generated by the code. Many error messages are confusing, unclear and do not help the user to determine the best course of action to take (Witten & Thimbleby). By observing the user’s actions in response to error messages, the programmer can analyze the effects of how the error messages are phrased and shown to the user. This can provide a framework for generating simple and well understood error messages.
It is unlikely that robust learning frameworks will become commonplace within standard end-user applications any time soon, but their necessity for the future of computing is quite apparent. Though users continue to make due with the current interface designs, many aspects of the process leave much to be desired. Through the user of machine learning, robust and dynamic interfaces can be designed that tailor their layout and functionality to the user’s preference without the need for extensive direct manipulation by the user. And although these systems may be difficult to use at first, and the early generations may not be ideal for all users, their implementation in future applications seems both necessary and inevitable.
@inproceedings{langley97machine,
    author = "Pat Langley",
    title = "Machine Learning for Adaptive User Interfaces",
    booktitle = "{KI} - Kunstliche Intelligenz",
    pages = "53-62",
    year = "1997",
    url = "citeseer.ist.psu.edu/langley97machine.html"
}

@article{author = "I.H. Witten, H.W. Thimbleby",
    title = "User Modelling as Machine Identification: New Design Methods for HCI",
    publication = "Advances in Human Computer Interaction",
    pages = "58-86",
    year = "1993",
    url = "http://www.ucl.ac.uk/harold/srf/advances.pdf"
}

@unpublished{author = "Hiroshi Motoda and Kenichi Yoshida",
    title = "Machine Learning Techniques to Make Computers Easier To Use",
    institution = "The Institute of Scientific and Industrial Research, Osaka University",
    year = "1998",
    url = "http://www.ar.sanken.osakau.ac.jp/~motoda/papers/aij98_motoda.pdf"
}

@article{author = "I.H. Witten, C.G. Nevill-Manning and D.L.Maulsby",
    title = "Interacting with Learning Agents: Implications for ML in HCI",
    publication = "ML'96",
    pages = "51-58",
    year = "1996",
}