Designing User Interactions with AI: servant, master or symbiosis

Notes of talk at AI Summit London, 22nd Sept. 2021.

https://alandix.com/academic/talks/AI-Summit-2021-UI-with-AI/

Alan Dix

Director of the
Computational Foundry,
Swansea University,
Swansea , Wales, UK

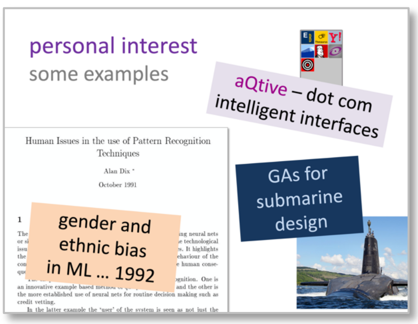
and

Professorial Fellow,
Cardiff Metropolitan
University,
Cardiff, Wales, UK

Abstract: All AI ultimately affects people, in some cases deeply buried, in others interacting directly with users whether physically, such as autonomous vehicles, or virtually, such as recommender systems. In these interactions AI may be a servant, such as Alexa operating on command; or AI may be the master, such as gig-work platforms telling workers what to do. However, potentially the most productive interactions are a symbiosis, human and AI complimenting one another. Designing human-in-the-loop systems changes the requirements of both AI algorithms and user interfaces. This talk will explore some of the design principles and examples in this exciting area.

Introduction

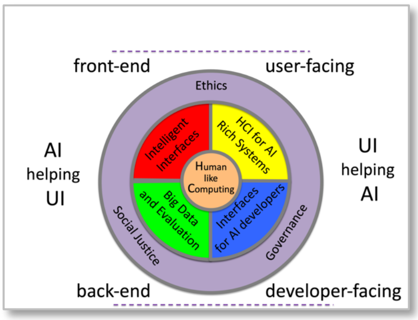
At one time artificial intelligence existed in clearly defined applications, such as expert systems or theorem provers. There were user interfaces for AI systems and AI technology might be used at a low-level, such as speech-to-text, more broadly, but did not fundamentally affect modes of interaction. This is no longer the case. The more low-level aspects of AI are becoming more sophisticated, such as facial recognition for identity verification and more clearly 'intelligent' features are becoming a central part of many interactions from movie recommendations to automatic photo tagging and predictive text.



My own central area of expertise has been in human–computer interaction since the field began in its current form in the mid-1980s, including writing one of the key textbooks in the area [DF04]. However, I have also been almost as long engaged with AI technologies both in their own right and as they relate to user interfaces and more broadly human interactions with technology. This has included using genetic algorithms for early submarine design, an intelligent internet start-up in the dot-com years [DB00], and, in 1992, possibly the first paper to highlight the potential danger for gender and ethnic bias in black-box machine learning algorithms [Dx92].

Addressing this topic is thus one deeply rooted in past experience, but also rapidly changing in both significance and scope.

AI and HCI connecting



There are a number of key ways in which AI and HCI can interact with one another. The diagram on the left summarises these.

Perhaps most obvious is if there is some form of user interaction where AI is used, such as book recommendations based on past reading behaviour; these are sometimes called '***intelligent interfaces***' (top left of diagram). Early work in this area includes Alan Cyper's Eager system [Cy91], which automated tasks based on a few examples and there has been an annual ACM conference, Intelligent User Interfaces (IUI) dedicated to the topic since the early 1990s.

However, there is also an increasing use of data-analysis and intelligent tools being used as part of the development and evaluation of user interfaces (bottom left of diagram), not least the extensive use of A–B testing [KL09] where decisions between two variants of a web-based interface are made by releasing both and gathering usage data.

Together these can be thought of as ***AI helping UI***, since the focus is on effective user experience and the AI is being used as a tool to deliver this.

In addition, we can look at systems where AI is the core element, but where effective UI can help in the use or development, that is ***UI helping AI*** (right hand side of diagram). Just as with AI helping UI there is a front-end and back-end version of this.

At the front end (top right of diagram) we have ***HCI for AI rich systems***, for example, how we design effective interactions for semi-autonomous cars or in smart cities. This raises issues, such as the need for handover when human intervention is needed and the way in which we interact with systems that are often hidden – the ghosts in the walls of a smart home!

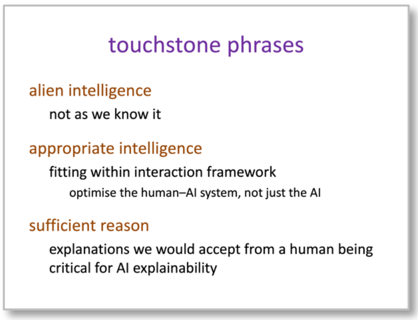
Below this we have ***interfaces for AI developers***, ways in which effective UI design can make it easier for developers to create AI systems and to understand their behaviour. This includes aspects of visualisation and also simply managing the large numbers of data files and unimaginably vast datasets.

In the centre of the picture is the concept of ***human–like computing***, the need to have AI which in some way comprehensible to humans or in some cases understands humans. This is increasingly being seen as essential for ***explainable AI***, both for developers and end-users … although differently for these different user groups.

Perhaps most important is that all of this is set within a ***social, political, and environmental context***. The press is full of stories where AI technology can be both used for good (such as developing new approaches to health), but also harm (such as racial bias in vision systems). Numerous institutes and programmes are being created to examine the ethical and legal implications of AI.

In these notes we will focus on the top half, user facing applications, and in particular design considerations and heuristics for the top-left.

Touchstone Phrases



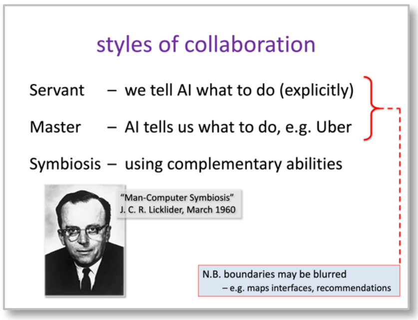
When thinking about intelligent interfaces, there are a few touchstone phrases that I have found useful over the years.

alien intelligence – Often AI is seen in terms of making systems that mimic human intelligence. However, even when AI is ‘intelligent’ in the sense that it does things that we might regard as intelligent if a person did them, the manner in which this is done is totally different. Often it can be more helpful to see AI as *alien* ... intelligent, just ‘not as we know it’. Crucially, while we may want to have some sort of broad understanding of how it is working this frees us to see human and AI as offering complementary aspects to an overall socio-technical system.

appropriate intelligence – It is easy to chase the impossible idea of a perfect AI. Sometimes, this is necessary, for example if designing systems that operate autonomously on a satellite. However, this often leads to fragile systems that fail catastrophically when the gaps in intelligence appear. When designing AI for user interactions, it is better to think of how imperfect AI fits within the wider interaction framework. That is we optimise the human–AI system, not just the AI, creating a system that *as a whole* is robust and effective.

sufficient reason – Explainable or comprehensible AI often seems impossible when we consider the billions or trillions of parameters in a deep neural network. However the explanations we would accept from a human being are rarely complete. If asked why you chose a particular option for lunch, you might discuss preferences as well as more factual or reasoned aspects such as the available menu options, or health conditions. Even a court ruling will include both matters of law and fact alongside (evidenced) judgements such as the veracity of witnesses. In each case the explanation is sufficient for the situation, we do not expect an explanation at the level of firings of neurons in the brain.

Styles of collaboration



A crucial question when designing a human–AI interaction, is who is going to be in control.

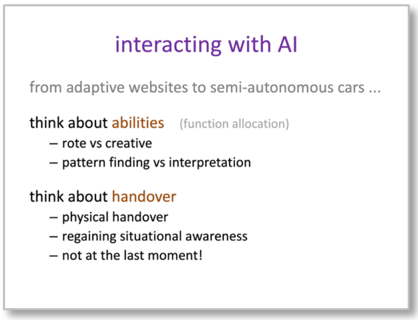
AI as ***servant*** – *we tell AI what to do (explicitly)*. When a user says to a voice controller, such as Alexa, "turn up the heating", there is a clear line of control. The voice controller uses extensive AI: to process and interpret the user's voice; to work out which room or rooms should be heated; to estimate by how much the heating should be raised based on past behaviour and preferences. However, this AI is all there in order to preform the task the user has given to the AI. Many of the examples of AI augmented interactions will fall into this category.

AI as ***master*** – *AI tells us what to do*. Contrast this with someone working for one of the gig-economy platforms such as Uber. They are presented with jobs chosen by complex adaptive algorithms, based on their location and possible past behaviour, and offered a payment rate related to current and predicted demand. In some platforms the worker can choose whether or not to take the task, but the line of control is usually still largely from the system. Of course, there are other humans within the system as a whole, notably customers hailing a ride, for whom the relationship is different, but for the workers the AI is effectively their master.

In reality, this servant–master distinction may be blurred. For example, when using a navigation system, you initially tell it where you want ago, but thereafter, more-or-less passively follow the navigation instructions turn-by-turn. Similarly, recommendation systems in online shopping sites in one sense leave the ultimate decision to you – in the end you can buy anything you want. However, by suggesting some things rather than others, they also gently nudge you towards certain products.

Perhaps the best alternative is ***synergistic interaction***, when humans and AI work together. In 1960 the visionary Joseph Licklider wrote "*Man-Computer Symbiosis*", looking forward to a day when computation could '*augment the human intellect*' in a similar way that mechanical tools augmented humans' physical abilities [Li60]. Licklider's original concept was perhaps closer to that of AI as servant "*Men* (sic) *will set the goals, formulate the hypotheses, determine the criteria, and perform the evaluations*", with computers performing the mundane and routine tasks. As AI becomes more powerful, this may become more of a synergetic interaction, using complementary abilities, more akin to human–human collaborations. We can see elements of this in more conversational interactions with voice assistants or with web chatbots, where there may be times when the AI-based system may take the initiative, seeking clarification or making suggestions, not passively doing what it has been told.

Synergistic Interaction with AI



There are many kinds of synergistic human–AI interactions. In a semi-autonomous car, it is very clear that AI is active when the car steers itself. Sometimes the presence of the AI is less clear, for example, consider an adaptive website, such as newspaper, where the content is customised for the user, but the user clicks on links or navigates pages.

When considering such synergies, the first thing to consider is the ***abilities*** of the human and computer. For example, rote tasks vs creative thinking in a design studio, or pattern finding vs interpretation in big-data analysis. We can then assign tasks to each (*function allocation*) that are appropriate, just as we would within a team of people.

Case study – cut your cloth

This is not new. In the garment industry large stamps, rather like giant pastry cutters, would cut all the pieces for a garment out of a pile of cloth, so that each layer had all of the pieces needed for a single garment. Skilled pattern arrangers would work out ways to arrange the pattern pieces in order to minimise wastage and then the metal stamps would be constructed to cut the cloth.

In the late 1970s Benetton invested in new numerically controlled cutters that worked on continuous cloth rather than stamping out pieces from piled cloth. It was now possible to consider laying out the pattern pieces for several copies of the same garment over larger areas of the cloth. In principle this could allow even more efficient use of the cloth, but the complexity of so many pattern pieces overwhelmed the human pattern arrangers.

A computer-aided system was developed that given an arrangement would `jiggle' it to move pieces so that they did not overlap, but were otherwise as close as possible, rather like shaking a jar of nuts. However, the human operators were still part of the process, they would look at the resulting layouts and spot more strategic changes, perhaps noticing that a gap looked about the right shape for a bodice piece, moving the piece there and then letting the automated system repeat the smaller scale re-arrangements.

This used the complementary abilities of human operator and automated CAD to perform a task neither could do on their own – synergy.

Sometimes this *function allocation* is clear and static, one aspect is far better for the human to do, another for the computer. However, there can be things in the middle, tasks that both the computer and user can do. If the whole task can be performed by a computer, this may lead to complete automation, but if it is part of a larger activity, then choice may be made. This could be based on human wellbeing, delegating boring tasks to the computer, or could be based on economics choosing whichever can perform the task most cheaply.

However, this balance can change over time. For example, in a semi-automated car the AI may be able to manage steering on the open road, but pass this back to the user when in a busy urban environment, and then perhaps be passed back to the AI for tight parking. Here the complexity and nature of the steering task has changed depending on the environment and context. In fighter aircraft there are many alerts and conditions that ideally require the pilot's attention. However, even quite crucial tasks may be a distraction in the middle of close combat. In such cases automated systems detect the pilot's activity and awareness and based on this may automatically suppress alerts or make autonomous decisions that would normally have been made by the pilot. Here the task has remained the same, but the pilot's abilities are changing due to other demands.

In both these cases who does what is changing over time, sometimes called *task migration* or *dynamic function allocation* depending on the community. When this happens, we need to think about ***handover***, how control is seamlessly passed back and forth between human and AI. In some cases, this is explicit at the initiative of the human, for example, when a pilot engages autopilot. The more difficult cases tend to be when the automated system takes the initiative in swopping control. Indeed, there have been several highly publicised accidents involving semi-autonomous cars when drivers have not been able to retake control rapidly enough.

The semi-autonomous car is a good example of this to hep us consider the main design elements for task migration, but similar situations can arise in control rooms or desktop systems.

First there is the choice of *when* control needs to be passed. In some ways this might be obvious, perhaps taking control when the driver appears to be sleepy (based on sensors or cameras), or passing it back when a road situation is encountered that the AI cannot deal with. However, the choice, while autonomously determined in itself, depends on the whole subsequent interaction.

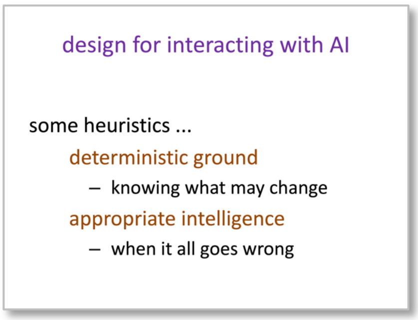
In the case of the steering wheel of a car, there is a *physical handover*. If, for example, the car were in the midst of steering a corner, the human would need to apply just the right amount of force on the steering wheel to avoid the car under- or over-steering. This could involve highly reactive haptic feedback, or, where possible, simply avoiding initiating handover when it is physically hard.

More problematic is regaining *situational awareness*, that is the general understanding of where one is and what is happening. If the car has been driving itself on a highway for an hour, the driver may be listening to the radio, or thinking of something entirely disconnected to the road situation. Extra aids may be needed to help the driver make sense of the situation, made particularly difficult as by definition the handover is initiated because something complicated is happening that the AI cannot deal with. Visual cues may help here, such as highlighting a potential hazard by projection on the windscreen, although again care needs to be taken as this could lead to *automation bias* [Cu4], focusing too much on one thing.

Perhaps crucially, these steps emphasise the importance of choice of time for handover. Rather than a last moment `*take over now*', various forms of more subtle indicators can be used when the system detects the potential for a difficult situation ahead, especially if it detects that the driver is not fully alert.

For information systems or desktop applications, the time scales are often less extreme, but similar principles apply. We are all constantly bombarded by notification alerts on our phones and laptop computers. If human intervention is needed in an otherwise automated process, then a well-designed system can choose a *pace* and *mechanism* for the notification that reflects the urgency of the task. For example, a reminder of action needed for a meeting the next day might be better managed using an email, or waiting for a lull in typing, rather than popping an alert.

Design for interacting with AI



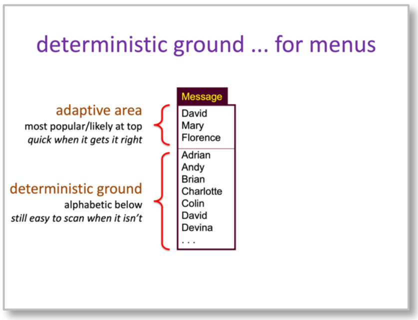
Two high-level heuristics for adaptive interaction are

- **deterministic ground**
  Does the user know knowing what may change? Ideally some elements are fixed to enable learning, whilst others intelligently adapt to improve efficiency.
- **appropriate intelligence**
  What happens when it all goes wrong? Are the intelligent elements embedded in a broader interactions that make the most of their good points, whilst shielding the user from the consequences of AI errors.

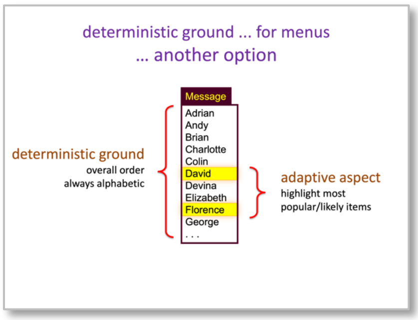
Deterministic ground

As an example of deterministic ground, let's look at long menus, which have been a target for adaptivity (more or less intelligent) since nearly their inception.

Given data on menu use, an obvious adaptation would be reorder items based on most recent or most frequent of use. Indeed, this is the approach used in default views of several cloud file services. This is useful if the desired item is towards the top of the list, as is likely to be the case. However, the times when you want to find something that you only access infrequently, you may have to simply scroll indefinitely scanning every item in an apparently arbitrary order – there is no deterministic ground. Not surprisingly, even where the default view of files is of this form, it is usually supplemented with hierarchical views.

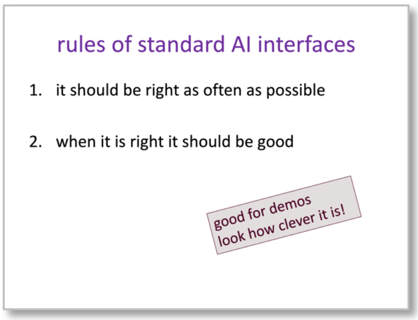


A more common approach is to reserve a small number of menu slots at the top for common/recent items decided by algorithm; and then have a standard order, often alphabetic, for the rest. For example, Microsoft Word uses this approach for its font menu. Note that this design has an *adaptive aspect* (most popular/likely at the top) which is quick when the AI gets it right; and also a *deterministic ground* (alphabetic below), which is still easy to scan when it doesn't.



This heuristic of having an adaptive aspect and also deterministic ground can be used to explore other options for adaptation. For example, for medium length menus (which can be scrolled through rapidly), we can always order the entire menu alphabetically (deterministic ground) and in addition highlight the most popular/likely items (adaptive aspect). The highlighting makes it easy to spot the most likely items, but it is still possible to choose anything with no loss of efficiency.

Appropriate Intelligence



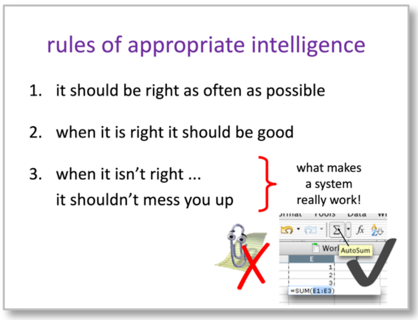
When designing an AI powered application, there are two obvious rules:

1. it should be right as often as possible
2. when it is right it should be good

This is exactly what you need for good for demos, you start to use the system and it comes up with clever suggestions, effective adaptations or accurate predictions. Of course during the demo, one avoids the minority of cases where the algorithm fails.

However, this does not lead to the best ultimate user experience when deployed. Those small number of troublesome errors are precisely what users remember and can cause a disproportionate level of difficulty or confusion.

For real systems for people to use, we instead need the rules of ***appropriate intelligence***:



1. it should be right as often as possible
   (or at least often)
2. when it is right it should be good
   (but doesn't have to be amazing)
3. when it isn't right ... it shouldn't mess you up

It is the last of these that makes a system really work!

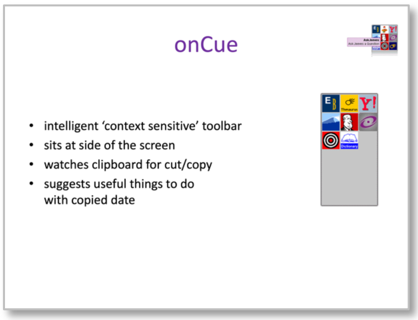
Older readers may remember 'Clippy' an early use of AI in Microsoft Word. While you typed, an algorithm was working in the background trying to work out what kind of thing you were doing. When it thought it knew, Clippy, a little animated paperclip, would appear perhaps suggesting a standard letter template when you wrote "*Dear …*". Some of the suggestions were potentially useful, but the act of making the suggestion interrupted your typing, both visually, but also taking the keyboard focus, so throwing away what you were in the middle of typing. Although the Clippy character was endearing, many users grew to hate it, and turned off the behaviour. The problem was that when the intelligent algorithm got things wrong it really messed up the interaction – rule 3!

Around the same time an `intelligent' feature was added in Excel, the Sum button. When you press this with a cell selected, Excel scans for things you might be wanting to add up. Its rules are quite simple – not complex AI – first scanning upwards to see if there are numbers and then if that fails to the left. There are a few extra features such as noticing sub-totals to avoid double counting them, but really very simple. The inferred selection is then highlighted and this is reflected in the cell formula (e.g. "`=Sum(B3:B8)`"). If the chosen range is correct, you just hit enter, if not you adjust the inferred one or select the required range. Crucially the latter is exactly what you would need to if you just type "`=Sum()`" in the formula area, and then select the range by hand. The only user-interaction cost of the Sum button getting things wrong is visually checking the range. It does not mess you up – *appropriate interaction*.

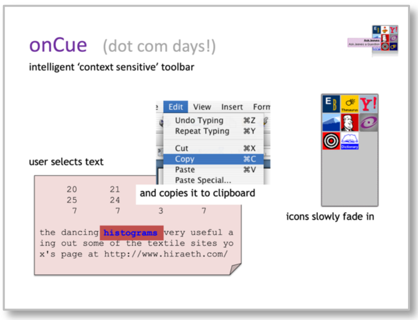
Not surprisingly, even though Clippy was far more cute and involved far more complex AI that Excel Sum, Clippy was dropped in later versions of Word, but the Sum button remains in Excel to this day.

Given there were skilled developers and designers on both teams when Clippy and Excel were developed, it is hard to tell whether the appropriate intelligence in the design of Sum was deliberate or accidental. However, it is possible to build appropriate intelligence principles into systems from the start.

Case study – onCue, an intelligent internet assistant



OnCue was an intelligent 'context sensitive' toolbar developed during the dot-com years (1998–2000) [DB00]. Much of the time it simply sat as a small icon or toolbar at the side of the screen, but it was constantly watching the clipboard and every time you cut or copied any text it would try to work out what kind of thing you had copied, and then suggest things you could do with the data in web and desktop applications. The engine was built using an agent-based framework that included simple pattern matching based on regular expressions (e.g. to recognise phone numbers), but could also trigger more complex matching, such as for tables of numbers. The suggestions were partly a short cut enabling single click access to services and partly informing the user about services they might not know about, especially at this point in the web's development.



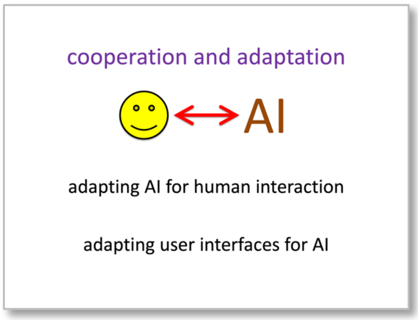
So onCue was useful when it was right, but often the user simply wanted to cut/copy the text to somewhere else, and not do anything beyond that. If onCue had, for example, popped a notification in the middle of the screen, it would have been very annoying.

Instead, the interactive behaviour of onCue was carefully designed to ensure appropriate intelligence.

1. It did not take keyboard focus, merely change the service icons in its toolbar.
2. The icons in the toolbar slowly faded in over a period of about a second.

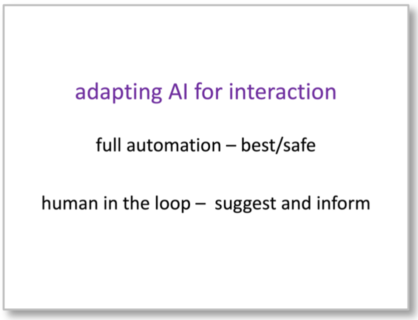
The first of these avoided an obvious interruption. You could simply ignore onCue and continue what you were doing. However, the second was also critical avoiding visual distraction – onCue was in the edge of the display, in peripheral vision, which is particularly reactive to sudden change. By making the change slowly, it was nearly impossible to see the change happening, even when you were expecting it, but whenever you looked at the onCue toolbar it was up-to-date.

Cooperation and Co-adaptation



When we regard humans and AI as working synergistically together, we may need to adapt our ideas of what constitutes the best AI and the best UX. This works both ways, we may need to adapt the way we develop AI to give better human interaction, but also we may need to adapt the way we design user interfaces in order to provide better opportunities for AI to learn and thus ultimately help.

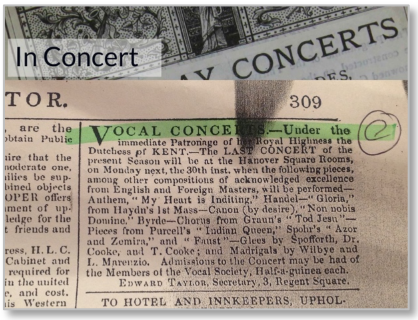
Adapting AI for interaction



When we make AI choices for ***full automation***, we are usually after the best, or most safe automated decisions. We want to *maximise accuracy* of predictions and find the *single best* possible solution.

However, if we assume we are designing for a ***human-in-the-loop*** system, where people will also take part in ultimate decisions or actions, we can change the targets of our algorithms to make them more able to *suggest alternatives* (not just a single best) and inform the user on how to use the information (measures of confidence, explainability).

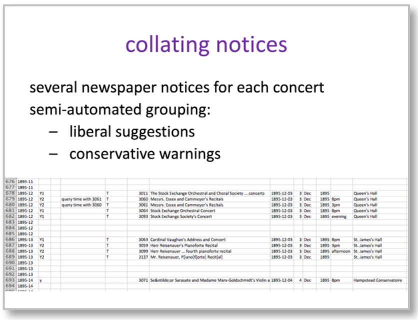
Case study – matching concert notices



To see how this can work out in practice, we'll look at an example from a project (InConcert) dealing with concert performances in London from 1750 to the early 20th century. One dataset was derived from notices and reviews in 19th Century newspapers and similar material.

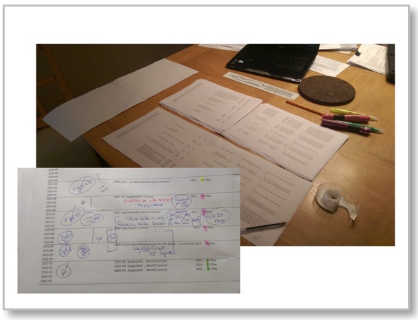
There could be several notices for a single concert in different newspapers, each of which might only contain partial information. So one of the first tasks was to match notices that referred to the same concert. As a fully manual effort this would have been extremely time consuming; however a fully automated solution was also not acceptable as this is an archival resource and the final matching must be approved by a musicological scholar.

So a semi-automated solution was developed. Some elements of this were identical to what would be expected for fully automated matching, in particular identifying key features



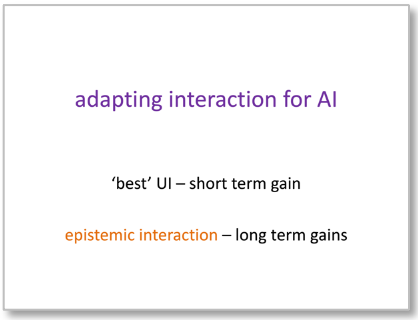
that would be combined to make a final decision. However, because this was to be presented as suggested groupings of notices to the scholar the way in which these factors were combined was adapted.

Rather than trying to create perfect group of notices for a single concert, instead a *liberal matching* algorithm was used, effectively taking a fairly low confidence bar for putting things in the same group. This meant that multiple references to the same concert were almost certain to be allocated to the same group (minimising the need for the scholar to search). It also meant that a single semi-automated group might contain notices about several concerts. A simple interaction mechanism was provided so that the scholar could simply agree with a grouping as representing a single concert, or mark items in the group as belonging to several manually assigned sub-groups.



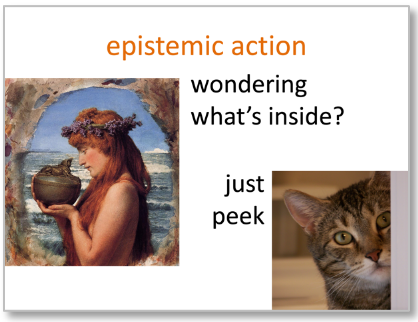
In addition, this liberal matching was supplemented by *conservative matching* (near exact matches) to add warning labels to lower confidence groups. This helped steer the scholar to the most problematic cases, but without preventing them from checking them all.

Adapting interaction for AI



Some of the information used to inform the AI in a interaction will come from fixed sources, perhaps large text or speech corpora used to train a model such as Open AI's GPT-3 [BM20]. However, if it is to adapt dynamically to the users interactions, this information must come from the current interaction whether in the user's explicit communications, "*I want to drive to Cardiff*", or by interpreting the user's actions, such as turning off the route into a service station suggesting the need for a break.

The user interface may naturally provide such incidental cues, but can also be adapted to give more information to the AI system. In theories of embodied cognition and ecological psychology human actions that furnish information are called *epistemic actions*. As a parallel, we can think of user interface designs that create opportunities for better information availability for the AI as *epistemic interactions*.

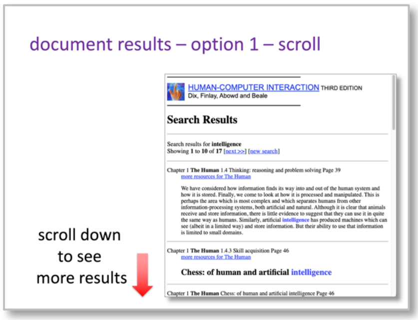


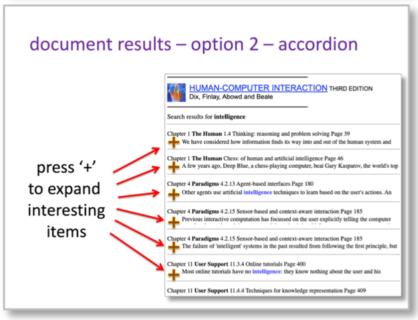
Mostly we want to design the very best user interface for the current activity, whether that is in terms of efficiency for highly functional tasks or broader user experience. Occasionally however we may introduce elements that may help the user in the longer term, for example to help them learn the system better, or help them to know about additional functionality of which they are unaware. Similarly, we might adapt the interaction to make it slightly less optimal for the immediate activity in order to provide better information for the AI to improve future interactions.

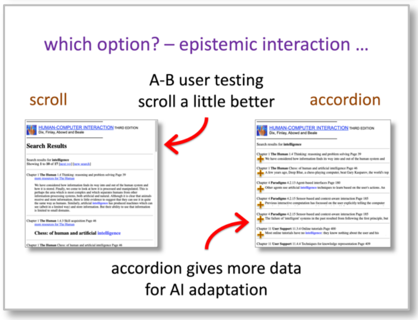
An obvious example of this is "*was this useful?*" questions in a help system. By responding positively or negatively the user helps the AI create a better understanding of the user and of its own information to aid future interactions for this or other users. This is an example of an *explicit communication* being used. In such cases there is an interruption of the user's task to provide the information, so ideally the prompt also has some more obvious benefit for the user, for example "*yes this was useful, please add it to my personal FAQ list*".

To avoid such explicit questions, we can also make small modifications to interaction that enable the AI to gather more *incidental information*, even if these modifications make the UI slightly less good at the moment of interaction.

Example – epistemic action for search results







As an example, imagine we are designing the search result web page that returns paragraphs from a book. We consider two options:

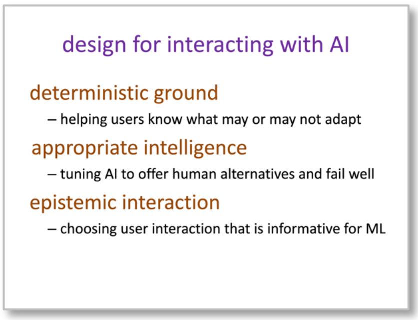
1. A long page of results that the user scrolls through to find interesting results.
2. An accordion-style interface with the short snippet of the result, where the user can open up interesting looking items.

Let's imagine we have run an extensive A–B user test comparing the two interface options and the infinite scroll interface comes out marginally better.

Normally that would be the end of the story, we select the scroll interface and go on to consider the next UI design choice.

However, it is far easier to obtain relevance data from the accordion interface. With the scrolling interface, we could monitor when the user delays on a certain page, suggesting it is interesting, but without eye tracking data, it is guesswork which of the results visible on the page is the one the user is reading. In contrast, the act of opening a result and how long the user leaves it open gives a strong indicator of the relevance of each search result, and thus helps the AI to tune future searches.

So long as the immediate UI difference was not too great, we might therefore opt for the slightly less good, but more informative *epistemic interaction* potential of the accordion interface.



Design for interacting with AI

In summary, we have discussed various issues when designing user interfaces that use AI. Some date back many years, others have become more evident recently as various AI techniques have become more common in user interfaces.

We've discussed three heuristics in more detail:

- **deterministic ground** – helping users know what may or may not adapt
- **appropriate intelligence** – tuning AI to offer human alternatives and fail well
- **epistemic interaction** – choosing user interaction that is informative for ML

However, this is a rapidly changing area with a growing number of conferences and workshops dedicated to the connections between AI and HCI. In addition, I am in the process of writing a short book on "*AI for HCI*" as part of the Routledge/CRC "*AI for Everything*" series … watch this space.

https://alandix.com/ai4hci/

References

[BM20] Brown, T., Mann, B., et al. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33:1877—1901.

[Cu4] Cummings, Mary (2004). Automation Bias in Intelligent Time Critical Decision Support Systems. AIAA 1st Intelligent Systems Technical Conference (PDF). doi:10.2514/6.2004-6313 (Reprinted in In Decision making in aviation (pp. 289-294). Routledge, 2017)

[Cy91] Allen Cypher. 1991. EAGER: programming repetitive tasks by example. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '91). Association for Computing Machinery, New York, NY, USA, 33–39. https://doi.org/10.1145/108844.108850

[DF04] Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). Human-computer interaction. Pearson Education.

[DB00] Dix, A., Beale, R. and Wood, A. (2000). Architectures to make Simple Visualisations using Simple Systems. Proceedings of Advanced Visual Interfaces - AVI2000, ACM Press, pp. 51-60.

[Dx92] A. Dix (1992). Human issues in the use of pattern recognition techniques. In Neural Networks and Pattern Recognition in Human Computer Interaction Eds. R. Beale and J. Finlay. Ellis Horwood. 429-451.

[KL09] Kohavi R, Longbotham R, Sommerfield D, Henne RM (2009) Controlled experiments on the web: survey and practical guide. Data Min Knowl Discov 18:140–181.

[Li60] Licklider, J. C. (1960). Man-computer symbiosis. IRE transactions on human factors in electronics, (1), 4-11.