Human AI Interaction Design Course

Group: Alpha

I. Objectives and Target Users

Our aim is to develop a high-fidelity prototype of a human-centric, interactive, recommender system app in the "events" domain. It will be named "DU Know Whats On?". The app will consider the human needs of the user and will aim to be usable, adaptable and provide an efficient and effective method to find upcoming events that they are interested in. Our target audience will be Durham Students between 18-25 years. The app will be free to download, making the app accessible for all Durham Students. For some students, they can struggle to engage with the community, know what local businesses have to offer and might not be involved in many societies and so they may not know what events are being run by different colleges or what deals local businesses are offering. The idea for the app stemmed from one of our development team trying to find a "Comedy Night" to go to in Durham yet could not. That same week, one was being held at a different college but had not been publicised on social media. It is our aim to connect students to events they might be interested in, centralising the way societies, club venues and local businesses offer events and provide personalised recommendations so that the user can discover a more diverse range they could be interested in more effectively. The AI defines success through collecting feedback from the user on whether they have found a recommendation "useful" or "not useful" and personalises the ratings overtime to better performance. Through having diverse recommendations, the AI does not take "failures" as being crucial due to the low cost of wrong predictions. Incorporating a "do not show events like this again" button will also help track failures from the system. Our other target audience will be university societies and local business owners, with an age range of 20+. By centralising the way businesses, societies and clubs publicise their events, we aim to increase the attendance and thus revenue for them, incentivising them to join our app. This was experienced with the same member of the development team who performed in a choir and experienced little attendance due to the lack of publicity for the event. We aim to bridge these demands together, benefiting all our users. Although "adding an event" will be through manual entry into a template, the app will automatically convert this into a page that can be viewed by users who may be interested. Another example of automation will be the chat-bot, s a virtual assistant to help the user navigate through the app if they are struggling. Finally, we ensure that the final choice in the events that the user is attending is down to user and we aim to help inform that decision through our recommender system.

In later stages such as implementation in the real-world, we aim to generate profit within our app through adding a 5% booking fee to each ticket we sell and collecting feedback data within the app that we can later sell to local businesses if they upgrade to a "premium" account. Lastly, we aim in future development to expand the target audience to several other universities, firstly collegiate like Oxford, where the app could be called "DU Know Wox on?", as well as to other universities where we shorten the name to "Whats on?". The app will be used daily or weekly, with local businesses providing offers probably on a monthly basis. For nightclubs this might differ to daily use as this is part of their business.

II. Prototype Development Lifecycle

Section 2.1: Acknowledging Feasibility section From the beginning, we decided that the recommender systems domain was a good place to start as we are not "reinventing the wheel" as there is lots of surrounding literature and example code on Github with similarly designed recommender systems apps. We then investigated the demand for the product by polling a focus group to decide between two ideas we had (an events recommender system or a food recommender system). The focus group was more supportive of the events recommender system which demonstrated that there was a real-world issue in the events domain that our app could resolve. The qualitative feedback that we received reinforced this: "it is difficult to find out what all the options are but easy when you know what you are looking for". Our app would solve this by recommending relevant events making it easier for the user to find appropriate events. We then looked into the cost of data acquisition by conferring with Dr Suncica Hadzidedic. Moreover, we had access to the NCC and so we were sure this app would computationally feasible for us. In 2020, an app named "Fygo" was developed that worked with local businesses to provide offers to students who used the app. This app was very popular and highlighted how "useful" apps could spread through the Durham student culture quickly, giving confidence that the app could be successful. As well as this, we did a case study on the Marriot Hotel, where in January, it had been offering students a discount but was relying on "word of mouth" to publicise this. It had been running for nearly two weeks before everyone had heard and then the venue became very popular. This highlighted that the app could benefit businesses by publicising offers more effectively. Moreover, as our team all regularly interacted with recommender systems (such as Netflix to watch movies and FIXR to book Durham events) we were confident we understood what components and practices in a recommender system make it valuable to a user. Furthermore, the cost of wrong predictions on the user were relatively low and so we would not be harming the user. Moreover, our team had similar mental models with other Durham students (as we are all Durham students) which makes it easier to align our product to our target user's models.

Section 2.2: Designing a paper prototype We then developed a paper prototype, drawing out different models and ideas of features and discussed them with the same selected focus group. Every member preferred the second paper prototype as they said it felt more "natural" to use, giving us an idea of the model to implement.

Fig. 1. Paper prototype meeting.

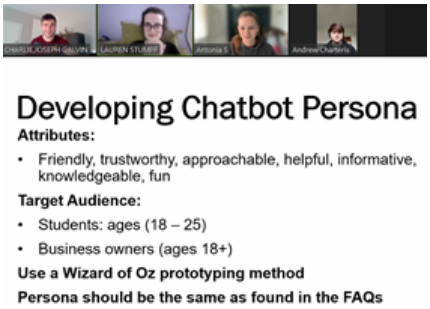

Fig. 2. Discussing chatbot persona.

Section 2.3: Developing Low-Fidelity Prototype We decided to implement or app using the Python library Kivy as there were projects on GitHub that could help in our design and we all knew how to use Python already, reducing development time. We each designed a different page in the app and put them together for a low-fidelity product, which we gave to the same users in the focus group and collected feedback. It became clear that the pages felt disjointed, but overall the users found the app relatively easy to use as it fit with their mental models. We discussed the persona for the chatbot, which we wanted to be friendly, helpful, approachable and easy to understand. After developing a script that matched these attributes, we ran a "Wizard of Oz" study where we gave the computer to the users and they believed they were conversing with an intelligent system when it was a member of our development team. The user feedback was encouraging and we recorded the behavioural attributes of the users whilst they were using the system. Both indicated that the persona was friendly and helpful to the user.

As well as this, we ran the recommender system with 3 selected users, 2 Durham students as well as a local business owner and ran a personal example by participants study. We fully informed the participants that this data was not going to be recorded or used within the data set for training prior to the study. Lastly, through collected feedback from the focus group on the low-fidelity prototype, we found that the users felt "overwhelmed" from the onboarding process we had in-place and that they would prefer more "tinkering" and have the ability to use the app straight away. We decided to reduce the amount of onboarding from this feedback, but still included some for the "older" business owners.

Section 2.4: Developed the final High-Fidelity prototype We asked the same users for their feedback from the final high-fidelity prototype. We did this by giving the same 3 selected users the final app for a week and asked them to fill in a feedback form everyday. We also recorded the diversity, novelty and mean squared error of the recommender system. Diversity and novelty are two metrics commonly used in recommender systems that helps give an indication of whether our recommender system is overfitting. We found that we had improved the persona of the chat-bot that redirects the user to a human if the chat-bot is finding it difficult to understand the user. We also implemented semantic analysis to ensure that the chat-bot understands when the user is feeling frustrated and redirects them to a human to avert any further frustrations. We also improved graphics with the UI making it more usable and adaptable with an advance in explainability, specifically talking about how explicit feedback is collected and used. As well as this, an improved level of "on-boarding" we noted, the users no longer felt overwhelmed.

III. Data Collection and Evaluation

For reference the recommender system that we implemented was Neural Collaborative Filtering [1]. Initially, we considered generating synthetic data to create our own data set to train a recommender system but felt that this would be difficult to balance bias, generate enough data-points and had little experience in doing so. However, it became apparent it was infeasible to gather enough data to train a deep learning model and so we chose to use the Kaggle Event Recommender System [2] data set instead. The Kaggle data set was chosen as it was the largest data set that was was relevant for our domain. Because the data set was so large we could afford to remove data to fine tune it to to minimize the effects of mismatch between the data and the aims of our product. To make this data set suitable for our purpose, we filtered users by their ages (keeping ages 18 - 25 to make it representative of our target audience). Moreover, to capture the local dynamics of Durham we chose to only take events in the region Manchester else we would be capturing more general trends. By attaining a high accuracy on data from this region it indicates that our model is able to capture the local dynamics and acts as a proof of concept that it would be able to capture the local dynamics of Durham if we had appropriate data to train it. We note that if we moved past the prototyping phrase we would create our own data set to capture the local dynamics of Durham.

The Kaggle data set contains 110 features describing the events. Most of these features were redundant to our purpose as they were counts of the number of times different words were used in the description of the event. To reduce the number of features we chose to focus on location, date, any tags associated with the event and the description. To encode the description into meaningful data that could be utilised by the AI we transformed the description into a 7 dimensional vector through the Bidirectional Encoder Representations from Transformers (BERT). The BERT model, a state-of-the-art model, produces a vector with high and telling semantic meaning. The location was already able to be exploited by the model (as it was in longitude and latitude form) and the tags were one-hot encoded. In the future we would create a data set with images for each event which could be displayed in the app. This would help the model as the image encodes meaningful data about the event. For the user, we chose to use their birth year as this respects user privacy while allowing the model to learn patterns in the data.

To prevent over-fitting we split our data set into separate training and testing subsets and ensure that the accuracy of the model when tested on the training and testing subsets are approximately equal. Moreover, we employed early stopping when it become clear the model was no longer improving in training. We found, from plotting the error, that around 13 epochs was best. Any further epochs weakens the model's ability to generalize as the model begins to over-fit to the training data. Our models initial accuracy scored relatively highly indicating that under-fitting was not an issue with our model. If we did encounter under-fitting we would have increased the complexity of the model (through adding more layers or making the layers wider or deeper) and trained for longer. Despite the steps we took to prevent our model from over-fitting, when we deployed this model in a prototyping stage we found that the model had a low novelty [3] - an indication of over-fitting. This was reflected in the user feedback with the low fidelity prototype in Section 2.3. All users reported a lack of diverse recommendations and consistent recommendation of one specific category of events. To mitigate this, we added dropout to one layer and reduced the complexity of the

model by reducing the number of layers. This helped increase the novelty and reduced the effects of over-fitting.

Our project had the potential to cause allocative harm [4] therefore it was important we were aware of any bias in the model and took steps to minimise this. We identified representational bias [5] in our data set (as the population had not been sampled randomly) and popularity bias in our model. Moreover, on further analysis it was clear that the data was scraped through social media networks and therefore it only accounts for people in similar social circles. However, unfortunately the data-set does not contain any information relating to socio-economic status and race and therefore we are unaware of the severity of this effect but hope when we can design our own data set, to reduce the effect of this. To mitigate the representation bias, once we had selected users from Manchester, we researched demographic statistics from this area and sampled a data set that was inline with these proportions.

Furthermore, our model contains the sensitive attribute gender. To make our model fair we first discussed which fairness definition was most appropriate. We theorized Disparate Treatment (fairness through unawareness [6]) as we argued that in large feature spaces (which our data-set had) sensitive attributes are generally redundant given the other features. Moreover, there was a low mutual information between gender and the other features indicating that the gender feature was not encoded elsewhere in model so we did not need to consider treating any other features differently. Therefore, we chose to train the model without the gender feature. Our model was vulnerable to popularity bias (whereby only popular events were recommended). To mitigate this, we chose to prioritise the the diversity and novelty of the recommendations and tuned the recommender threshold to maximize these metrics. As our recommender system predicts between 0 and 1, the recommender threshold refers to what threshold the recommended prediction should exceed in order to be recommended (we chose 0.62).

The complexity of the recommender system and purposely including a regularization term, as well as the discussed work on preventing over-fitting all contribute to preventing data leaks. These help prevent data-leaks as they ensure that model captures general trends and does not over fit to an individual therefore making it very difficult to extract information regarding an individual by interacting with our model.

Whilst we have not included it in the prototype, when deploying the app each account would be identified through a username and password that we necessitate to be over 8 characters long with a special symbol. We would keep sensitive information such as name and age under password protected files with a layer of authentication and do not request information that we do not need to help provide a good user experience (e.g. address). As well as this, we will encrypt the username that we use in the model, ensuring that if any "recommendation data" is leaked, this contain no personal information about our users. In the training phrase the the Kaggle data set has taken extensive steps to anonymise data (e.g. by removing precise location data) which we would continue with if we moved past the prototype phase.

IV. Mental Models, Explainabiltiy and Trust

When investigating the feasibility of developing the system (section 2.1), we considered how all members of the development team fit the description of the target audience, and so were better informed of the mental models are target users might have when using the app. Through using well known apps like Netflix, Amazon Prime, Snapchat, Facebook, Instagram among other examples, we were able to better align the mental model of the app with what users already had. This included trivial things such as placing the navigation bar at the bottom with the home button centred in this bar, as well as scrolling vertically downwards like Snapchat, Instagram and Netflix. Having a profile page as well as the settings in the right-hand corner like Facebook, as well as having the search bar at the top of the page as with all of these apps. We initially made sure that the physical layout of the app made sense to the user when consulting them with our paper prototype in section 2.2, where we held a focus group discussing which design the users preferred and reasons for it and it was unanimous that these aspects above all felt "natural" to them. Also, the steps to providing explicit feedback and purchasing a ticket from when the app is closed to being able to do it is only 3 clicks, from opening the app and being on the homepage, to clicking on a genre of events the user is interested, and finally choosing an event. This easy layout ensures that the user is not "mentally overwhelmed", increasing the app's usability. Although this is one more click than experienced in the recommender system app FIXR that many already use in Durham, which again has a very similar mental model regarding the physical layout of the app, navigation bar, vertical scrolling, etc., when we asked for feedback from the low-fidelity product in 2.2, the users said it felt very easy to use and that they were not overwhelmed at all.

With this in mind, the main aspects of the AI that should be explained is the chat-bot located in the top-right corner of the screen, as well as the FAQ's page which is not common in these apps but is helpful in the prototype for establishing explainability and trust. The calendar should also be explained as this again is uncommon but can help the user "see" what they have on and when, which is very beneficial for them as we have found from feedback.Lastly, adding an event should be explained to the user as although it is simple to do, it can give the user the confidence they need to add an event in the future. We include a message in both settings and the FAQ's that outlines as the user uses the product more and provides a greater amount of feedback, the performance will increase and the recommendations will be more personalised, managing their expectations. This will allow the users to connect feedback to adaptation and personalisation to establish a relationship, as the user will have more trust in the system when they see it perform better over time. Through including the "do not show me events like this again" button, this will also improve the relationship as it shows how effective the feedback is for the AI output when the algorithm responds by not recommending similar events immediately. We have also written a description of the app in the app store to help manage users expectations and not over-promise them. Through describing the app as a "DU Know Whats on is a Durham-specific events app prototype that adapts to your preferences to recommend you personalised events", this will mean that the user has more patience with the app and avert any frustrations, hopefully establishing a degree of trust through our honesty of expressing what the app can and cannot do in the marketing message. With the settings page, we also have a section talking about the "data we collect", increasing the explainability. As well as this, we have included a message from the chat-bot that explains that it's in its early stages of development. This is because when people interact with a chat-bot, they assume that it is "very intelligent" and they may have higher expectations of what it can do, which can lead to disappointment with it being

in its early stages of development. By communicating this to the user, we help manage their expectations. Lastly, some users may expect the AI product to find exactly what they are looking for, and can become disappointed when this has not happened. Through providing a confidence score for the recommendation, this can limit the users expectations, avoiding dissatisfaction. In the on-boarding process, we first explain the chat-bot, and how although it is in its early development, it can be used as a place to ask for help with navigating within the app. We then explain the FAQ section and that it can help the user understand better the inner-workings of the algorithm and how the users data is being collected, before explaining how the feedback they provide helps with the AI personalise their experience and highlight how they can "opt-in" to this by going to the settings page and moving the toggle button to allow us to collect explicit feedback. This message will also be included in the settings page. Lastly, we will comment on how the calendar can highlight what events the user has on a certain day, as this is a feature which is a bit different from other apps so is not in the mental model the user may have. The onboarding process helps the user get to know the app, but allows them to "tinker" with looking at events and providing explicit feedback on their own, as this was the feedback we received from the low-fidelity prototype in section 2.3. To offer explanations in the inner workings of the app, we have added a small description of how the algorithm works in the FAQ's section, simply explaining a "deep learning" model which takes in user's feedback on events to better personalise recommendations overtime. The language used is quite simple as to not overwhelm the user, as well as it is in the FAQ's so it is up to the user to find out more, giving them a choice. We discuss how we use the feedback in the on-boarding stage at the beginning as well as in the FAQ's and the settings page, where the user can opt-in using a toggle button, again to give them a choice. The recommendations provide intrinsic explainability in the algorithm offering local explanations of why an event has been recommended along with the confidence score of how "confident" the algorithm is that the user will find the recommendation "useful". This improves the app's interpretability and trust through greater transparency. This confidence score also allows the user to gauge when they want to trust the system and when they do not, improving the reliability and in itself calibrating their trust. This also improves the effectiveness of the product, helping the user make good decisions. We did not feel like the explanations needed to be contrastive as this did not fit with the mental model from the users. Through including the confidence score, we manage influence on the user's decisions as it is up to them to decide whether to trust a recommendation, but the algorithm quantifies how confident it is that they will find the recommendation "useful", which although can influence the user in making a decision, also gives them the information to decide whether to trust the system or not as well as provides a reason why it has been recommended. Providing the user with all this information manages the influence the algorithm can have on their decisions. Lastly, through testing how much on-boarding to do (section 2.3, 2.4) and keeping the users central to the development of the prototype, we have added the right amount of on-boarding for the users to understand how to use the model without them feeling overwhelmed. As discussed above, the app has high explainability in regards to data collection, follows common mental models that are target audience already uses and the app has had feedback that it is "usable and adaptable" when we gave the users the high-fidelity product in section 2.4. All of this leads us to believe that the app has been optimised for understanding.

V. Feedback and control

The app collects implicit feedback through the events that the user clicks on. When prototyping we also collected implicit feedback from the events that the user searched for however we could not find a way to convert this into meaningful data that our AI was able to learn from. The implicit feedback is opt-in where the users can opt-in in the settings page. This guarantees we have the users express permission to collect this data. We are clear with the user about what data is collected from them and how, regarding the actions they take when they use the app. This can be seen in the settings section under the terms of service. More detail on this can be seen in in the explainability section (Section 4). In the initial stage of prototyping after the low fidelity prototype (Section 2.2), we found the implicit feedback was very valuable and could be used to benefit the performance of the model (in terms of accuracy) in comparison when this was not included in the algorithm. It also did not have a disadvantage in "overwhelming the user" as it is collected the data without the user having to do anything, meaning it could only improve their experience. In order to translate the implicit data into meaningful data we update the model via transfer learning with a smaller learning rate. To do this, we wait until one thousand pieces of implicit feedback for the user has been collected. We then analyse these pieces for any anomalous data (which for example could be events that the user clicked on by accident). We do this by calculating the Z-score for each piece of data and drop the top and bottom 5 %. We then proceed to update the model and repeat the process by waiting for another new once thousand pieces of implicit feedback to be collected. This ensures we are harnessing the implicit feedback correctly and not updating the model on erroneous data.

We also collect explicit data. Specifically, we have useful and not useful buttons available to the user. We have chosen useful and not useful as opposed to like and dislike buttons as 'like' can be ambiguous and unspecific. Moreover, the ultimate goal of our recommender system is to provide useful feedback.We purposefully only use one word to be clear and concise (as too many mental obstacles may prevent the user to provide feedback) and discuss in depth what we mean by useful in the how your data will be used section. This explicit feedback it a lot easier to utilise in the model as it has been provided purposely and therefore represents the users true feelings. In the prototyping process, we empirically verified the value of this feedback and its benefit to the AI as it helped increase the model's novelty and diversity and the user satisfaction.

As feedback is so valuable with respect to improving the AI, it was important to make this value clear (as to motivate the user to provide it). To motivate the user to provide feedback we do this in two ways. We provide material reward in the form of free entry to events as when the user provides explicit feedback they are entered them into a raffle for a free ticket (to an event of their choosing). Whilst this is a loss of money initially, we argue that it is beneficial overtime as this contributes to a linking users to events which increasing profit. We also communicate the value of feedback (and so hope to motivate the user) with a popup that says thank you for your feedback it helps improve the recommendations. By not including a specific time frame, we also temper the users expectations as the changes in recommendations (as it is a deep learning model so it takes quite a lot of data for the model to notably change) happen slowly and gradually so its likely not immediately noticeable. This is also acknowledged in the FAQ section and the setting section to make it clear to the user. However, this also communicates the importance and impact of

the users feedback and so encourages the user to help improve the model further. Whilst making the value of feedback clear, we also make sure the user does not feel pressured into providing information they are not comfortable. We also have a "contact us" page in the settings, where users can provide more thorough feedback on how satisfied they are with the product.

VI. Errors and graceful failure

When building our prototype, we considered the possible errors a user could encounter and how to mitigate them.

One possible error a user could encounter is the system failing to complete an expected action, such as not returning the recommended search results. This could potentially be caused by server-side or back-end connection problems. To overcome this, an error message will request the user to reload the application or try again later. Alternatively, this could be caused by a user's search criteria being too specific. Our system aims to mitigate this through allowing a user to filter their search, preventing slip errors through imposing constraints on their search.

Another potential error could be the user making mistakes when navigating the system. Although the user won't initially be familiar with the mental model of the system, our use of onboarding aims to mitigate any mistakes the user may make. This is further supported by giving the user good defaults on the homepage. This will allow the user to use the application by choosing one of the default choices even though the user is not yet familiar with the system.

The recommender system in our application recommending irrelevant or unhelpful events could be a possible failure of our app. This would likely be caused by a lack of data or the data preferences supplied by user. We would aim to overcome this by gaining more information from the user on their preferences, by getting the user to fill in a form with their favourite events. This would allow us to update their rating on our system.

Although errors can occur through our application, when designing our system we tried to include functionality preventing the user getting stuck in the application. This was done through implementing the use of back buttons and a navigation panel for robustness of the application and to ensure that the user can always go back to the previous page.

VII. Design patterns

In our FAQ page, we show what our app can do, emphasising the benefits of the app. The focus groups helped formulate what information the users need to know about how the system works.

Allowing a user to give feedback was a design pattern which we incorporated into our system. Feedback included useful/not useful buttons, reporting an error or blocking a user. The useful/not useful buttons were an important feature because our application makes recommendations based on past behaviour and feedback buttons allow users to fine tune recommendations to their own needs. This will modify the recommendations on what they say, as well as what they do. Through enabling users to report an error or block users, we take responsibility and are accountable for errors. Through enabling the user to report an error, we will be able to update or change our system. This will ensure other users won't encounter this problem. Giving users the choice to block other users, provides the user with more freedom. This in turn builds comfort and supports personalisation of the system for the user. Overall, feedback allows us to improve the performance of our system.

Another important design pattern we introduced was the use of confidence levels, which enable us to convey a level of certainty for how good the predictions are that our system is making to the user. Users might not understand how our application is making predictions, but this will allow the user to know how much confidence we have in the prediction being made.

A vital design pattern for any application are the system controls, which allow users to understand what our application can do. The system controls of our application include a navigation panel, a settings page and a search page. System controls are interpretable which reinforces the user's mental model, since they have become accustomed to having controls in parts of an interface.

Our system controls also incorporated the idea of familiarity, which was an important aspect in the design of our User interface. We wanted to design our application so that our user felt comfortable with using it. This is reflected in the layout of our system. Individual pages were built with similar features in mind, such as the settings page containing icons for each section. The use of icons on the settings page are also present in the majority of applications. Familiarity improves user experience through building trust.

Transparency of our data was a key idea that we considered when designing our system. Through settings and FAQ pages, we inform the user of how we collect and use their data, as well as giving the user the choice to opt-in to data collection. The opt-in toggle for data collection makes it easy for users to adjust their settings. This increases the user's trust with the system and gives the user more control of how they use the system.

References

[1] Xiangnan He et al. "Neural collaborative filtering". In: *Proceedings of the 26th international conference on world wide web*. 2017, pp. 173–182.

[2] *Kaggle Events Dataset*. https://www.kaggle.com/c/event-recommendation-engine-challenge.

[3] Liang Zhang. "The Definition of Novelty in Recommendation System." In: *Journal of Engineering Science & Technology Review* 6.3 (2013).

[4] Solon Barocas, Moritz Hardt, and Arvind Narayanan. "Fairness in machine learning". In: *Nips tutorial* 1 (2017), p. 2.

[5] Ninareh Mehrabi et al. "A survey on bias and fairness in machine learning". In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–35.

[6] Cynthia Dwork et al. "Fairness through awareness". In: *Proceedings of the 3rd innovations in theoretical computer science conference*. 2012, pp. 214–226.