CAPRIO with Inclusive Pedestrian Path Recommendations

Brian T. Nixon*[‡], Zhiyong Chen*[‡], Sai Konduru[‡], Yuxin Liu[‡], Constantinos Costa[‡][§], Panos K. Chrysanthis[‡]
[‡]*Dept. of Computer Science*, *University of Pittsburgh*, *Pittsburgh, USA*
[§]*Rinnoco Ltd*, *3047 Limassol, Cyprus*
nixon.b@cs.pitt.edu, {zhc79, sck42, yul287}@pitt.edu, {costa.c, panos}@cs.pitt.edu

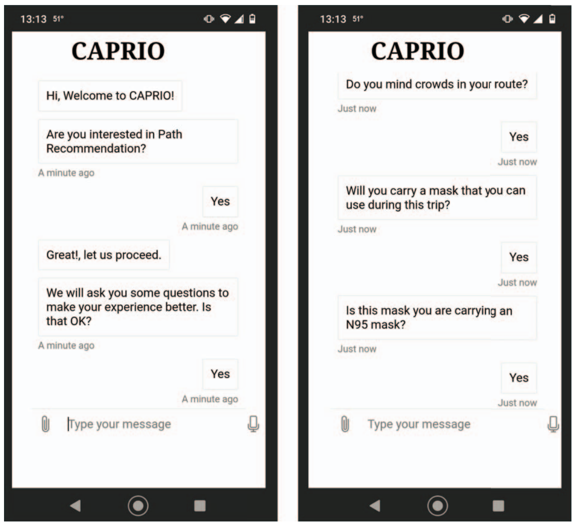
***Abstract*—Accessibility and usability have been key concerns in the design of computer interfaces through which users interact with applications and systems. Recently, chatbots have gained popularity with service providers for improvements in this area. In this paper, we present our experience in designing and implementing CAPRIO's inclusive chatbot-based interface for pedestrian path recommendations. Our CAPRIO system provides inclusive usability by extracting user preferences in a non-intrusive dialog and using them to build a more accurate model for the user's intent. It uses the Microsoft Bot Framework (MBF) and NLP modeling to support text and voice dialog.**

***Index Terms*—accessibility, diversity, inclusion, pedestrian path recommendations, HCI, chatbot, indoor, outdoor**

I. Introduction

Context-awareness in human-computer interactions is essential for the everyday activities of an individual to maximize the utility of the computer systems and applications which the individual interacts with. For example, a context-aware path recommendation system can provide recommendations that are robust to a wide range of constraints of great importance to many user groups. For instance, during inclement weather, such as the 2021 heatwave in the USA and the 2021 polar vertex in Canada, pedestrians are more vulnerable to heat and cold-related injuries and illnesses. In addition, moving crews need to avoid paths with many steps and narrow staircases. As the examples suggest, context is subjective and while these systems can provide utility for many users under several scenarios, benefiting users from all backgrounds requires holistic thinking about their design.

In this paper, we describe our experience in transforming CAPRIO [1], [2], our indoor-outdoor path recommendation system, to provide *inclusive utility*, i.e., providing equal service and value to all users. To achieve inclusive utility, we discovered that both the user interface and the path-finding algorithm in our case, must be designed to cooperatively support accessibility and usability, aiming to obtain the user preferences in a non-intrusive manner. While CAPRIO considers accessibility as a mobility constraint and enables users to personalize their recommended indoor-outdoor paths through the collection of user-specified preferences, its initial interface was not inclusive. As shown in Fig. 2 (left), CAPRIO provides a traditional interface that requires users to manually provide their preferences by adjusting sliders and buttons, which creates barriers for visually impaired or blind user groups. Furthermore, CAPRIO's current preference collection inhibits the usability of the system by requesting input parameters of excessive granularity rather than inferring them. For instance, the congestion tolerance constraint.



Hi, Welcome to CAPRIO!

Are you interested in Path Recommendation?

A minute ago

Yes

A minute ago

Great!, let us proceed.

We will ask you some questions to make your experience better. Is that OK?

A minute ago

Yes

A minute ago

Do you mind crowds in your route?

Just now

Yes

Just now

Will you carry a mask that you can use during this trip?

Just now

Yes

Just now

Is this mask you are carrying an N95 mask?

Just now

Yes

Just now

Fig. 1. Mobile interface of the CAPRIO chatbot

To improve the inclusiveness, we developed a chatbot interface for CAPRIO, that supports both text and voice chats, which assists in the removal of barriers faced by visually impaired and blind user groups. Furthermore, in terms of usability, the total time needed to collect a user's preferences can be reduced by presenting constraints as simple chat questions rather than confusing parameters. Lastly, unnecessary granularity can be removed by deducing preferences from the answers to a series of intelligent questions as shown in Fig. 1.

This demo paper focuses on our experience improving the inclusiveness of an existing path recommendation system, in our case CAPRIO, and not comparing path recommendation systems. The main contribution of this paper is twofold.

- We discuss how users' preferences can be used to express their capabilities and how these preferences are extracted in a non-intrusive text and voice dialog.
- We discuss our experience in developing a text-based and voice chatbot for CAPRIO's inclusive interface using Microsoft Bot framework (MBF).

*These authors contributed equally.

II. The CAPRIO System

CAPRIO is our path recommendation system within the University of Pittsburgh and the University of Cyprus campuses. Given a starting location and a destination location, it recommends a path through buildings that keeps the time outdoors and the overall travel time as low as possible and within user-specified limits. It also considers congestion delays as part of the travel time and accessibility constraints with respect to entrances, exits, and corridors.

The CAPRIO architecture consists of the *Data Layer*, *Processing Layer*, and *Application Layer*. The Data Layer is responsible for managing input data from various data sources such as local or distributed files, data streams from IoT devices, or other external APIs.

The Processing Layer is responsible for the core functionalities and services of the system such as processing data for path recommendations, performing localization, and providing congestion information. The Processing Layer uses ASTRO [3], an A*-based algorithm that can recommend a path with outdoor exposure, congestion tolerance, and accessibility constraints while keeping the travel time low for each request along with the corresponding source and final destination.

The Application Layer is equipped with an intuitive map-based interface layer that abstracts the complexity of the system and recommends routes as shown in Fig. 2.

III. Inclusive Dialog for Preference Gathering

Achieving *inclusivity-for-all* would require a solution that would take everyone's needs into consideration. In our investigation, we learned that arriving at such a solution requires identifying subgroups of users, authoring sensitive questions to obtain user needs, abilities, and disabilities in terms of preferences, and overcoming challenges faced during each of these steps [4].

A. Identify Subgroups of Users

Accessibility typically focuses on supporting users with disabilities and consequently inclusiveness is often associated with disabilities as well. For this reason, in order to design an inclusive system for all users, the initial step is to identify subgroups of users based on disability to handle as many scenarios as possible. The reasoning for this step is to potentially discover and generalize the types of barriers that users may face in accomplishing a specific task or goal, in our case while traveling and therefore provide a more robust path recommendation. While this step seeks to improve the inclusivity of a system by discovering scenarios and user groups whose preferences and requirements are not captured by the system, it turns out to be a very complex task. Identifying every disability and its proper spectrum is simply not possible given the near infinite number of conditions.

Rather than focusing on the conditions that affect the abilities of users, an alternate solution is to approach the issue bottom up rather than top down. Instead of identifying which subgroups users belong to, we focus more on the environmental barriers that users face while interacting and collaborating on a task. In the case of travelling, for instance, instead of highlighting the fact that someone uses a wheelchair as a result of being diagnosed with Cerebral Palsy, we concern ourselves with the fact that they are unable to climb stairs. By focusing on environmental barriers, users are identified through their abilities instead of the conditions that affect their abilities. This favors a more individualized approach that gathers information about each user to build a more accurate model of the type of paths or actions they wish to take.

B. Author Sensitive Questions to Obtain User Preferences

When the more individualized approach is adopted, the challenge is determining ways to ask users questions about their abilities in a sensitive and respectful manner. Inclusivity intends to benefit absolutely everyone and for this reason, in our case of context-aware path recommendation systems, the system should focus on aspects of environments that are commonly challenging for many people to overcome. For example, a wheelchair user cannot take the stairs and must use the elevator. A janitorial staff member who frequently moves heavy equipment will also opt to use the elevator. A restaurant caterer will need to travel with wheeled carts and once again uses the elevator. In each of these cases, climbing stairs is not the preferred way to navigate to different floors of a building where an elevator is an appropriate alternative irrespective of the circumstances surrounding the reasons for its use.

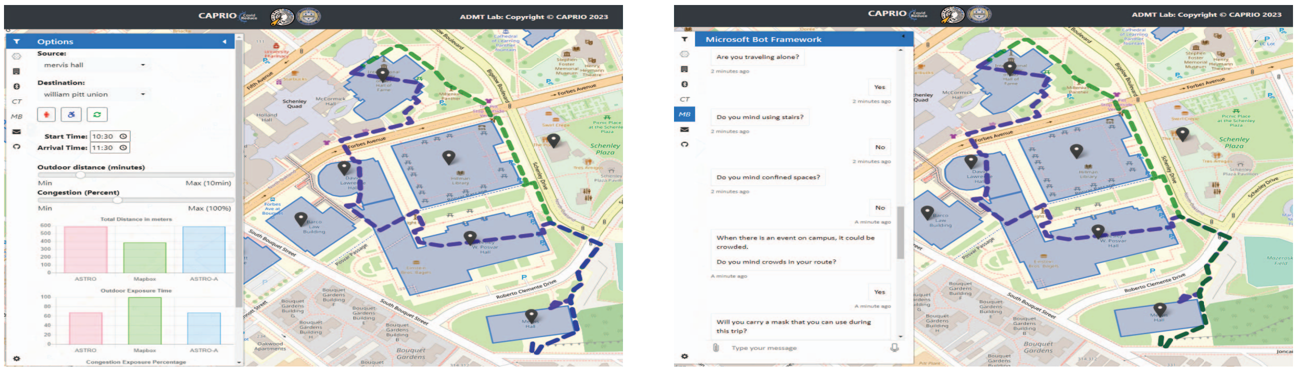
By framing questions in terms of environmental characteristics, there is no need to ask users about their capabilities and limitations because such information is irrelevant to identifying which factors may be used to identify the shortest and safest paths possible. Obtaining *preferences* (environmental elements) rather than asking individuals about their abilities (capacity to engage with elements) solves the issue of potentially asking intrusive questions that would discourage the use of the system and reduce its utility. For instance, asking, "do you have difficulty climbing stairs?" is far too invasive and personal. By contrast, asking, "do you mind using the stairs?" as shown in the dialogue above, we are simply interested in whether the user prefers to use stairs or not regardless of their reasons. Perhaps their reasons relate to their ability to overcome these barriers, but these reasons are not necessary for improving the path-finding algorithm in our case. A sample of an inclusive dialogue is shown in Fig. 1.

In short, inclusivity benefits *everyone regardless of ability* and questions within a chatbot dialogue should be asked based on *preference* instead of capability.

IV. CAPRIO's Chatbot Interface

The first challenge in developing an inclusive interface for CAPRIO was to select a suitable existing technology or develop our own from scratch. Towards this, we surveyed the existing frameworks and developed a taxonomy for determining which chatbot framework best meets our requirements.

We surveyed existing chatbot technology to determine the best option by considering cost, conversation limits, text and voice support, and development complexity. In order to be able



Source: mervis hall
Destination: william pitt union
Start Time: 10:30
Arrival Time: 11:30
Outdoor distance (minutes) Min Max (10min)
Congestion (Percent) Min Max (100%)

Are you traveling alone? 2 minutes ago
Yes 2 minutes ago
Do you mind using stairs? 2 minutes ago
No 2 minutes ago
Do you mind confined spaces? 2 minutes ago
No A minute ago
When there is an event on campus, it could be crowded.
Do you mind crowds in your route? A minute ago
Yes A minute ago
Will you carry a mask that you can use during this trip?

Fig. 2. (left) Manual interface for user preferences in CAPRIO, (right) Microsoft Bot Framework (MBF) chatbot interface for user preferences in CAPRIO

to maintain a free service and support future development, we restricted our pool to only free chatbots, which eliminated paid options (e.g., DialogFlow [5]). While free chatbot options exist, we considered their daily or monthly message limits and the requirements for improving the usability and accessibility of gathering preferences for CAPRIO.

A. Text Interface

For the text-based interface, we considered different chatbot types such as Rule-Based System, AI, and Live Chat. Although AI was initially preferred, the lack of training data led to the selection of a rule-based chatbot due to its flexibility in validating input. Accessibility requirements included support for multiple languages, multiple platforms, and voice functionality. Based on these requirements, AWS Lex and Microsoft Bot Framework (MBF) were selected as suitable chatbots due to their lack of conversation limits, language and platform support, as well as text and voice capabilities.

We have built interfaces using both the Amazon Lex and MBF solutions to compare them. The full architecture of the Amazon Lex solution requires multiple AWS modules along with the standalone client application to meet our requirements. Also, AWS Lex has a complicated architecture, which makes it harder to maintain. The MBF solution to the chatbot was user-friendly and was also able to meet the requirements of our use case. In MBF, developing a chatbot was straightforward and intuitive. Furthermore, with MBF's support for voice communication using Amazon Alexa or Web Chat, Microsoft's chatbot is more inclusive and usable in a larger range of scenarios. For these reasons, we found MBF to be preferable to AWS Lex.

B. Voice Interface

After developing the text-based communication chatbot shown in Fig. 1 and Fig. 2 (right), we focused on adding voice capabilities, which would greatly improve the accessibility of CAPRIO. A voice interface would provide a more natural and convenient way for users with typing or reading difficulties to interact with CAPRIO, thus enhancing their user experience.

MBF allows the bot to communicate with other applications using web channels through the Speech Services SDK. This SDK provides a Direct Line Speech channel that enables the connection of the bot with Microsoft Speech Cognitive Services to facilitate Speech-to-Text and Text-to-Speech functionality. In our case, we use the default trained speech model, Microsoft Azure Speech Cognitive Services, to perform speech recognition and synthesis functionality. While the process of integrating voice using MBF might seem straightforward, the integration required a human-in-the-loop approach to fine-tune the language in a way that maximizes usability.

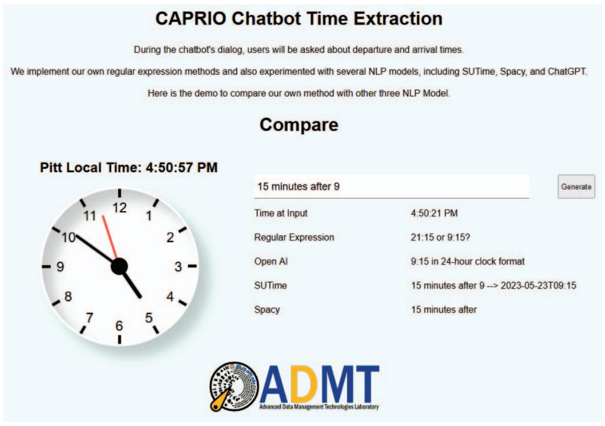
C. Related Problems

The addition of voice capabilities enhances the user experience and brings the chatbot closer to natural language communication; however, this upgrade has introduced challenges such as intent recognition and named entity recognition for determining arrival and departure times.

Intent Recognition: In a conversation with a chatbot, intent recognition is a crucial step in ensuring that the chatbot understands the user's request correctly. For example, in our case shown in Fig. 1, the chatbot asks the user if they will carry a mask that they can use during their trip. If the user clicks on the "Yes" button, it confirms that they will carry a mask with them. By providing users with a clear and simple button to confirm their intent in our original text dialog, the chatbot can avoid misunderstandings or confusion. However, if the chatbot interacts with the user using voice, there is no button for validating the user's intent.

To address the problem of intent recognition, we utilize natural language understanding models in Azure Cognitive Services to predict the user's overall intention. We synthesized a labeled training dataset for intent recognition that consisted of user responses, such as "OK", "okidoki", "sure", "yup", etc., and a matching intent of YES or NO. The NLP model was trained using our labeled training dataset, evaluated, and deployed on Azure. The chatbot subsequently makes external requests to the model to classify user intent.

Time Extraction: During the chatbot's dialog, users will be asked about departure and arrival times. Our original approach was to suggest a fixed input format "hour: minute", then validate the user's input and extract the time. But in a voice environment we can not specify the format of user input.



CAPRIO Chatbot Time Extraction

During the chatbot's dialog, users will be asked about departure and arrival times.

We implement our own regular expression methods and also experimented with several NLP models, including SUTime, Spacy, and ChatGPT.

Here is the demo to compare our own method with other three NLP Model.

Compare

Pitt Local Time: 4:50:57 PM

15 minutes after 9

Time at Input	4:50:21 PM
Regular Expression	21:15 or 9:15?
Open AI	9:15 in 24-hour clock format
SUTime	15 minutes after 9 --> 2023-05-23T09:15
Spacy	15 minutes after

Fig. 3. Comparing Time Extraction Interface

We have proposed two potential solutions: *regular expressions* and *NLP models*. In the English language, there are four different ways to indicate a particular time. The first involves expressing time in "minutes (To/Past) hour" format, while the second uses the "hours:minutes" format. The third simply uses the "hours" format, while the fourth is conveyed through specific keywords such as "noon" representing 12 PM. Based on the four expressions mentioned above, we have implemented regular expression code to extract time. However, time expression in English may have ambiguous cases, such as distinguishing between AM and PM. To resolve this ambiguity when extracting arrival time, we use intent recognition to determine whether it is AM or PM. On the other hand, when extracting departure time, we automatically resolve the ambiguity based on the common sense that the departure time is earlier than the arrival time.

Before developing our own solution, we explored existing systems that utilized voice chatbots to help drive their applications. One such example, Jarvis [6], utilized a *Conversational Unit Interface* to convert user's speech to query IoT devices. However, our problem requires a custom solution as these existing voice chatbots, such as Jarvis, do not translate to the problem of extracting preferences for path recommendations.

In addition to exploring existing voice chatbots, we also experimented with several NLP models, including SUTime [7], Spacy [8], and ChatGPT [9]. Among them, ChatGPT has the ability to recognize relative time. However, our study found that neither ChatGPT nor the other two NLP models were able to handle time ambiguity, such as distinguishing between 12-hour and 24-hour clocks as shown in Fig. 3. Our results show that the regular expression solution is deterministic, providing an error message when no time information can be extracted, whereas all three NLP models may produce incorrect time information. We also observed that the regular expression method's response time is comparable to the other solutions.

In order to debug and assess the completeness of our regular expression solution, we conducted a preliminary user study within our department which included diverse participants in terms of their speaking language background.

V. Demonstration

During the demonstration, attendees will have the opportunity to experience the key benefits of our inclusive chatbot interface, including accessibility through voice input and usability by utilizing the chatbot to generate path recommendations. In addition, the attendees will be shown the shortcomings of existing voice chatbots that motivated our solutions for time extraction and intent recognition through a special UI (Fig. 3).

A. Demo Artifact

We have extended our prototype CAPRIO system, which consists of an interactive map using OpenStreetMap along with several graph techniques. The back-end was developed using the Play Framework 2.7 and MongoDB 4.4 and the CAPRIO web interface were extended with the chatbot endpoints and sidebar as shown in Fig. 2 (right).

Particularly, CAPRIO provides two ways to input the user's preferences for a path recommendation. The first way is a query sidebar consisting of multiple UI elements such as buttons, dropdowns, and sliders, which requires manual input and adjustments from the user as shown in Fig. 2 (left). The second way is a sidebar with our proposed chatbot allowing users to express their preferences either by text or voice as shown in Fig. 2 (right). Demonstrating both ways during the conference will allow the attendees to understand the benefits of expressing the user preferences in a conversational way using our chatbot instead of manual input.

B. Demo Plan

Datasets: To generate paths within the University of Pittsburgh and the University of Cyprus, we will pre-load a diverse set of real datasets obtained from both universities.

Scenarios: The chatbot is publicly available on the CAPRIO website and conference attendees will be able to interact with the chatbot's web-interface through laptops, tablets, and smartphones and gain first-hand experience of our intent recognition and time extraction solutions. If the device supports voice input, attendees will also be able to experience the voice function.

References

[1] C. Costa, X. Ge, and P. K. Chrysanthis, "Caprio: Graph-based integration of indoor and outdoor data for path discovery," *Proc. VLDB Endow.*, vol. 12, no. 12, p. 1878–1881, 2019.

[2] C. Costa, B. T. Nixon, S. Bhattacharjee, B. Graybill, D. Zeinalipour-Yazti, W. Schneider, and P. K. Chrysanthis, "A context, location and preference-aware system for safe pedestrian mobility," in *IEEE MDM*, 2021, pp. 217–224.

[3] C. Anastasiou, C. Costa, P. K. Chrysanthis, C. Shahabi, and D. Zeinalipour-Yazti, "Astro: Reducing covid-19 exposure through contact prediction and avoidance," *ACM Trans. Spatial Algorithms Syst.*, vol. 8, no. 2, 2022.

[4] L. W. Leiby, C. Costa, and P. K. Chrysanthis, "Thinking inclusively with caprio," in *IEEE MDM*, 2022, pp. 378–380.

[5] Dialogflow. [Online]. Available: https://cloud.google.com/dialogflow

[6] N. D. Huynh, M. R. Bouadjenek, A. Hassani, I. Razzak, K. Lee, C. Arora, and A. Zaslavsky, "Jarvis: A voice-based context-as-a-service mobile tool for a smart home environment," in *IEEE MDM*, 2022, pp. 318–321.

[7] SUTime. [Online]. Available: nlp.stanford.edu/software/sutime.html

[8] Spacy. [Online]. Available: https://spacy.io/

[9] ChatGPT. [Online]. Available: https://chat.openai.com/