College of Engineering

COMP 437 – Intelligent User Interfaces Project

Final Report

Navigation Assistant

Doruk Taneli 60066

Spring 2020

Table of Contents

I. Abstract

In this project I wish to present a navigation interface that improves on the current navigation systems in three aspects. First, this application will consider time spent in traffic to be the main variable to pick the best route to a destination, instead of the shortest travel time. Currently most of the navigation applications only consider the shortest travel time constraint. Second, I would like to improve the interaction with the map. I would like to incorporate the intelligent user interface features into my application, such as multi-modality with touch and sound, recognition, user state sensing, user adaptiveness, and turn-taking. Through voice assistance, the driver can prompt searches with less thought and less distraction. The user can tell how he is feeling or what he wants, and the system will find the type of place the user needs and display nearby places. Plus the system will carry out small nuances depending on the situation and say different things each time so interaction will feel more natural. Lastly, the system will adapt to the user's preferences and act on the preferences without waiting commands from the user.

II. System Design

I built an android application using Android Studio. I used git and github for version control and kept a detailed log on google docs. In the log, I explained what I did, why I made some choices and which sources I used for what parts of the project. Many of the subjects here and some others are explained in more detail in the log. A link to the log is provided in the appendix.

At the start of the project I was planning to use yandex maps as it was completely free. However, yandex maps api documentation was not complete [1]. So I switched to Google Maps, thinking the free credits will be enough for my use case.

Basic map features may seem trivial, however google maps api do not provide many of the functionalities out of the box [2]. I spent a considerable amount of time figuring out how to use the Directions API. I was able to transform the string output into a series of points and then draw the resulting route on the map, using the tutorial of Mitch Tabian as a guide [3].

The project uses 3 Google Maps APIs in total, Maps SDK for Android, Directions API, and Places API. Maps SDK for Android is being used for the bare maps and showing user location . Directions API is used for showing routes from the user location to the selected places, along with

getting duration in traffic and total duration for each route. Places API is used for the search box, and getting places related to voice commands.

I used the built in android voice recognition and text to speech engines for interaction with the user. I wanted to use DialogFlow for more natural interactions, however the new java client library for DialogFlow does not support android [4]. So I manually implemented the user scenarios using Java.

III. Analysis and Results

The resulting project has some basic maps features and some more intelligent user scenarios. Starting with the basic features: the user can pick a destination by clicking anywhere on map, or using the autocomplete search bar at top. The user can then see possible routes to the destination, and view duration in traffic and total duration information of every route. The best route is determined with the shortest duration in traffic, as the target users of this project were manual transmission users. The user can also save locations as home or work, then quickly navigate to them using voice commands.

Second, there are some more intelligent user scenarios. The user can search for relevant places by telling his state. For example, I'm bleeding for hospitals, I'm out of yogurt for markets, I want to grab a coffee for cafes, etc. The user can search for hospitals, pharmacies, markets, greengrocers, bakeries, cafes, gas stations, liquor stores and restaurants.

There is also user adaptiveness. People generally prefer to buy gas from a specific brand of gas station. So, on the first use, starting gas station search will return all gas stations. When the user picks a specific gas station brand over and over, starting gas station searches will show that brand of gas stations. The user will still be able to see all nearby gas stations. If the user searches for all gas stations, the preference is reset and regular searches will show all gas stations until a preference is observed again.

I added some nuances to some user scenarios. The application asks if it should call the emergency services when it says something that will prompt a search for hospitals, and dials the number when it gets a positive answer. It says get well soon when prompted to search for pharmacies. The user can combine searches too. When the user says: "I want wine and cheese", the application will show both markets and liquor stores, grouping them by marker colors.

As demonstrated in three demos, the project was successful in displaying the intelligent user interface properties of multi-modality, recognition, user state sensing, user adaptiveness, and turn-taking.

IV. Conclusion

The app successfully demonstrates intelligent user interface properties, and it will help me remember those properties for a long time as I spent a lot of time implementing them myself. I also learnt a lot about android programming as my previous experience with android programming was limited. Plus reading lots of Google maps api documentation got me familiar with Google documentation style which I think will help me as Google is leading in software in many different areas.

The place results of the application are not optimal. I used the Place Autocomplete api to get information about nearby business establishments because it was the easiest way to get place information in android and I already had place autocomplete setted up for the search bar [5]. However, the Place Autocomplete api prioritizes the first word of the establishment name rather than location, so the places the api returns are not the best for our use case. The results can be improved by using Nearby Search or Find Place Request of the Places API [6].

The voice interaction with the users can also be improved by using a third party natural language processing system or training a new one, rather than selecting one of the pre-written sentences in random.

V. References

1. “MapKit reference for Android”, *Yandex*, https://tech.yandex.com/maps/mapkit/doc/3.x/concepts/android/mapkit-directions/about-docpage/

2. “Maps SDK for Android”, *Google Maps Platform*, https://developers.google.com/maps/documentation/android-sdk/intro

3. *Tabian, Mitch.* “Google Maps and Google Directions API”, youtube, Sep 19, 2018, https://goo.gl/pfGNnw

4. “DialogFlow Java Client Library”, *Google Cloud*, https://cloud.google.com/dialogflow/docs/reference/libraries/java

5. “Place Autocomplete”, *Google Maps Platform*, https://developers.google.com/places/android-sdk/autocomplete

6. “Place Search”, *Google Maps Platform*, https://developers.google.com/places/web-service/search

VI. Appendix

The drive links can be opened with a Koc University account.

Mid-demo:

https://drive.google.com/file/d/17vE0Q1Z8pljXgvVjp2DaRfifZIMwqpIR/view?usp=sharing

End-of-classes demo:
https://drive.google.com/file/d/1T9Bxf1hFkYyCf4zRL5m1n9bFCyPIeMZT/view?usp=sharing

Final demo:

https://youtu.be/NitnowR-SEU

Github:

https://github.com/DorukTaneli/navigation-assistant

Log:
https://docs.google.com/document/d/1ZuOX8XxH7o1N6phX2Y5MNjdX_o2AJJONsi2XFA0z710/edit?usp=sharing