1 Introduction

We propose development of an Android application which utilizes location data to generate customized region-based alarms. The idea is that when a user arrives at work or home, they are given reminders pertinent to the location. In order to achieve this goal there are a number of smaller problems that need to be overcome:

1. Determine the user’s current position as accurately as possible. This is a problem that is still largely unsolved in many indoor environments. While there are many acceptable approximations, our group will still need to evaluate and select the most viable.

2. Determine a way to bound the geographic regions. Since we wish to identify when a user crosses from one area to another, we will need some way to mark the edge of regions. As most people will not be pleased at having to mark out regions on a map by hand, this problem needs to be solved as automatically as possible.

3. Determine a reliable means of waiting for events and alerting the user when necessary. The reason this is such a problem is that having a process lurking in the background is traditionally very draining on battery and care needs to be taken to minimize the impact of such a process.

Our application will interest people who require more customized reminders or to-do lists. We believe almost every mobile user comes under the scope of this project. Also, social applications such as “Find My Friends” [1] are as likely to find benefit as scheduling and management applications.
2 System Design

The underlying architecture making up the core system of the project consists of the analytic engine, database, web server, and device application. These, along with minor components are shown below in Figure 1.

![System Architecture](image)

Figure 1: System Architecture

More specifically, as our analytics tool, we use a Python machine learning library called “scikit-learn” [4]. SVM was chosen as the classification algorithm since the use of non-linear kernels allows regions to be overlapped. We will also be using one-vs-all strategy for multi-class classifications as there will be multiple zones per user.

The data for our application will be stored on an SQL database via a Microsoft SQL Server. Both the database and the analytic engine will reside on the web server which will allow for rapid communication between the two components.

In addition to hosting the server-side applications, the IIS7 web server will act as a communication hub between them and the remote application. By using PHP as an intermediary between database and device application, the server provides a communication interface which is entirely architecture independent.

The device application, as the last major component, is an Android application which draws upon both the Wifi and GPS location sensors to determine the location of a user. This data is then transmitted to the server for analysis.

3 Related Work

Currently, there are few implementations of location-based alarms on mobile devices. An iOS app “Find My Friends” [1] has the capability to generate notifications when a friend enters or leaves a certain location. Also, the iOS’s pre-installed reminder app is able to set location-based reminders. The same is true of an Android app called “Geobells” [2] While effective, current practice is to simply calculate the geographical distances between two coordinates. This approach
is highly inaccurate and begins to break down as locations become more and more closely spaced. Even when these apps are successful in generating reminders based on users’ locations, however, they cannot deal with geographical zones that consist of multiple coordinates. Instead of using radius-based method, we train a machine learning classifier based on users’ labeled data and make predictions about whether a user is in a pre-defined zone.

For each user, there may be multiple geographical zones, which makes the problem a multi-class classification problem [5] rather than a binary classification problem. The normal solution to this is to use “one-vs-all” strategy [3] to train the classifier. To further complicate matters, in many cases two or more zones may overlap, preventing the data points from being classified using linear classifiers. Instead, non-linear kernels such as polynomial kernels [6] and RBF [7] kernels will be required. In our system, we will use both the one-vs-all strategy and an RBF kernel for SVM.

4  Current Status and Plan

Currently, we’ve finished building the web server and database and have achieved communication between most major components. The design of the database is shown below:

![Database Design](image)

(a) ER diagram  
(b) Relational diagram

Figure 2: Database design

For Android development, we’ve finished the first iteration of the UI design for the Android app:
Figure 3: UI design

For analytics, we’re currently testing our algorithm on a smaller dataset that consists of 100 data points and 10 labeled zones and have managed to generate zones using the data points:

Figure 4: Geographical zones generated from labeled data points

The timeline for remaining work is:

- March 21 - March 27: Select and implement a means of gathering user data and providing user alerts.
- March 28 - April 3: Implement prototype 1, design and implement testing harness
- April 4 - April 10: Collaborative and field-dependent test design
- April 11 - April 17: Run tests and adjust for failing test cases. Implement prototype 2
- April 18 - April 24: Individual testing by team members for day to day use.
• April 25 - May 1: Correct shortcomings and implement usability changes.

• May 2 - May 8: Provide deployment link over web site. Compile documentation for final report.

References


