CS 403X Mobile and Ubiquitous Computing
Lecture 7: Final Projects + Smorgasbord of Stuff!!

Emmanuel Agu
smörgåsbord

/ˈsmɒrɡəsˌbɔːrd/

definition

noun

- a buffet offering a variety of hot and cold meats, salads, hors d’oeuvres, etc.
- a wide range of something; a variety.
  "the album is a smörgåsbord of different musical styles"
Final Project Overview & Proposal Guidelines
Final Project

- Most projects will probably build an app
- App solves some societal problem
- App should be **mobile** or/and **ubicomp**
  - **Mobile?** Probably location-dependent, maps, deliver time-sensitive information
  - **Ubicomp?** Uses at least 1 sensor (accelerometer, microphone, camera, etc)
- Don’t build app that has no mobile or ubicomp aspects
- If you have questions, talk to me
Typical Paper

- Introduction
- Related Work
- Approach/methodology
- Implementation
- **Project timeline**
- Evaluation/Results
- Discussion
- Conclusion
- Future Work

Note: No timeline in final paper
Proposal

• Submit (Written 2 pages max PDF file): due Apr 16!!
  – Introduction
    • List team members
    • State problem app will solve. Preferably has social benefit
    • Why is problem important?
    • E.g. Find statistics: How much time, money, resources is being wasted on this problem today? How many people problem affects
    • Potential gain: how will your solution save time, money, etc?
  – Related work
    • What other research has been done to solve this problem (academic + commercial apps)
    • How is your app/approach/work different?
Proposal

- Methodology/Design/Tools:
  - Brain storm!
  - Summary of what you intend to do
  - How you intend to do it? Build android app, use scenario, etc
  - App screen mock-ups:
  - Don’t promise too much,
    - Some features can be future work
Methodology

- Preliminary design from team
- Screen mock-ups + flow
- Use Android Studio Design view, lucidcharts.com, hand-drawn?
Proposal

- Implementation plan:
- Timeline
  - Break down tasks, mini-deadlines, allot time for each task
- Proposal due April 16!!
Separate Vision and Prototype

1. Big picture
   if funds/time not an issue
   (e.g. company of 200 employees over 6 years)

2. Which reasonable Subset of the big vision can you do in 2.5 weeks?
   Can make simplifying assumptions
Typical Paper

- Introduction
- Related Work
- Approach/methodology
- Implementation
- **Project timeline**
- Evaluation/Results
- Discussion
- Conclusion
- Future Work

Note: No timeline in final paper
Final Paper: Evaluation

- Depends on what your project is.
- **Basic question:** How well did your solution work?
  - User studies
  - Measure performance. E.g. energy consumption, bandwidth consumption, etc

- User Studies
  - Pre-Survey:
    - Establish problem exists, need for your app, gather/refine requirements
  - Post-Survey:
    - Get users to use/rate your app, ask about likes dislikes
Recruiting Subjects For User Studies

- 3Fs: Friends, Family and ??
- Classmates (Do a trade with another group)
- On campus: post flyers, set up table at campus center
Discussion, Conclusion, Future Work

- **Discussion:**
  - How was your app received? Rationalize your findings in user studies, Say why certain features worked, did not work, etc

- **Future work**
  - Talk about features that would extend prototype
  - Revisit big vision
Your Team
Some Team Tips

- You already have a team!
- Everyone (team members) doesn’t have to do everything equally
- Team members can work on project aspects they are good at
- Example: Who is good at:
  - Android UI design (Android Studio design view, XML file, widgets, nice look)
  - Android programming (database, sensors, maps, backend)
  - Experimental evaluation/user studies
  - Machine learning
  - Writing, making presentations
Some Team Tips

- Team should have an honest conversation
- Doing something different doesn’t mean chilling
- Consider team online management tools, gantt charts, etc
- Assign tasks, mini-deadlines (every few days)
- Integrate features every few days => new version
- **Mantra:** Always have a working prototype, improve
What other Android APIs may be useful for ubicomp?
Speaking to Android
Ref: Professional Android 4 Development, Meier, Ch 11, pg 437

- Speech recognition:
  - Accept inputs as speech (instead of typing) e.g. dragon dictate app?
  - Note: Google (remote) service Requires internet access

- Speech-to-text
  - Convert user’s speech to text. E.g. display voicemails in text
Gestures
Ref: 3 cool ways to control your phone

- Search your phone, contacts, etc by handwriting onto screen
- Speed dial by handwriting first letters of contact’s name
- Also multi-touch, pinching
Doing More with Locations: Geocoding

Ref: Professional Android 4 Development, Meier, Ch 13, pg 513

- Maps, GPS discussed so far use longitude/latitude to pinpoint geographic addresses
- Users more likely to think in terms of street addresses

- **Geocoder** converts between longitude/latitude and street address
  - **Forward geocoding:** Finds latitude and longitude of an address
  - **Reverse geocoding:** Finds street address for given longitude/latitude

- Can also set proximity alerts
  - Intent delivered to your app when you are within a pre-set distance from a given location
More on Audio, Video and Camera
Ref: Professional Android 4 Development, Meier, Ch 13, pg 513

- Android MediaPlayer previously used to play audio
- Media Player can also:
  - Play videos (e.g. MPEG 4)
  - Record audio and video
  - Preview video
  - Manipulate raw audio from microphone/audio hardware, PCM buffers
    - E.g. if you want to do audio signal processing, speaker recognition, etc
More on Audio, Video and Camera
Ref: Professional Android 4 Development, Meier, Ch 13, pg 513

- Can control Camera parameter settings
  - Flash mode, scene mode, white balance
- Camera can also do face detection and feature recognition
  - Detects face up to a max number of faces + accuracy
RenderScript

- High level language for GPGPU
- Use Phone’s GPU for computational tasks
- Very few lines of code = run GPU code
Wireless Communication
Ref: Professional Android 4 Development, Meier, Ch 16, pg 665

- Bluetooth
  - Discover nearby bluetooth devices
  - Control your smartphone’s (device’s) discoverability
  - Communicating over bluetooth

- WiFi
  - Scan for WiFi hotspots
  - Monitor WiFi connectivity, Signal Strength (RSSI)
  - Do peer-to-peer (mobile device to mobile device) data transfers
Wireless Communication
Ref: Professional Android 4 Development, Meier, Ch 16, pg 665

- NFC:
  - Contactless technology
  - Transfer small amounts of data over short distances
  - **Applications:** Share spotify playlists, Google wallet
  - **Google wallet?**
    - Store debit, credit card on phone
    - Pay by tapping terminal
    - Fly through checkout?
Telephony and SMS
Ref: Professional Android 4 Development, Meier, Ch 17, pg 701

- **Telephony:**
  - Initiate phone calls from within app
  - Access dialer, etc

- **SMS:**
  - Send/Receive SMS/MMS from app
  - Handle incoming SMS/MMS in app
Google Fit API

http://en.wikipedia.org/wiki/Google_Fit

- Google Fit API: Single cloud storage record for all user’s fitness apps (myfitnesspal), gadgets (fitbit), etc
- Complimentary Google Fit app supports fitness tracking, view progress
- You can program app to access, read, write Google Fit record
Google Fit API
http://en.wikipedia.org/wiki/Google_Fit

- Google Fit API also has API for step counting
- i.e. Low end phones without step counter can use Google Fit’s step counting API
  - Implemented as a Google service
- Also **DetectorActivity** API to detect smartphone user’s current activity
- Currently detects 6 states:
  - In vehicle
  - On Bicycle
  - On Foot
  - Still
  - Tilting
  - Unknown
Alternate Implementation Options
AppInventor (http://appinventor.mit.edu/)

- MIT project, previously Google
- Use lego blocks to build app, easy to learn
- **Pro:** Quick UI development
- **Con:** sensor access, use third party modules restricted
PhoneGap

- Develop Apps using HTML, CSS, javascript
- **Pro**: Access to most native APIs, sensors, UI
- **Con**: Need to know HTML, CSS javascript
Making Apps Intelligent
(Sensors Inference & Machine Learning)
My Goals in this Section

- If you already know machine learning => set off light bulb
- If you don’t know machine learning => General idea of it, how it’s used
Example: Activity Recognition

- Android can now recognize 6 activities (in vehicle, on bicycle, etc)
- How is it done? Machine learning classifiers
- Next explain activity recognitions. Use it to explain
  - Machine learning + concepts
  - Data collection (FUNF)
  - Feature extraction, explain features
  - Inference:
    - Hard-coded rules by inspection, trial & error
    - Machine learning (supervised learning)
Activity Recognition

- Want our app to detect when user is performing any of the following 6 activities
  - Walking,
  - Jogging,
  - Ascending stairs,
  - Descending stairs,
  - Sitting,
  - Standing

- Need to collect sample data from sensors while user performing activity (called training data)

- **Example:** Phone’s accelerometer data sensitive to movements
Example Accelerometer Data for Activities

(a) Walking

(b) Jogging

(e) Sitting

(f) Standing
Example Accelerometer Data for Activities

(c) Ascending Stairs

(d) Descending Stairs
Gathering Accelerometer Data

- Can write simple app that retrieves accelerometer data while user is doing each of 6 activities (1 at a time)

- Label each data with activity performed. E.g. label the following data as sitting
Funf (funf.org)

- Can also download, FUNF app to gather data
- Capable of collecting user data
  - Log sensor readings
  - Web URLs visited
  - Phone calls + duration
  - SMS messages sent, etc
- Check boxes to specify sensors to log, sampling rate, intervals
Methodology (Data Collection)

- Data collected from 29 subjects
- Users carry phone in front pant leg pocket
  - For all activities
  - Perform each of 6 activities
- Accelerometer data collected every 50ms
  - 20 samples/second

Figure 1: Axes of Motion Relative to User
Segment Data (Windows)

- Raw time-series data cannot be used with classification algorithms
- Data divided into segments (e.g. 10 seconds)
Compute Features

- Within segments, compute features
- **Features**: Derivatives that capture important characteristics, but still stable
- **Examples**: moving average, standard deviation, min-max values within segment, magnitude within segment
Methodology (Feature Generation)

- **Average**[3]: Average acceleration (for each axis)
- **Standard Deviation**[3]: Standard deviation (for each axis)
- **Average Absolute Difference**[3]: Average absolute difference between the value of each of the 200 readings within the ED and the mean value over those 200 values (for each axis)
- **Average Resultant Acceleration**[1]: Average of the square roots of the sum of the values of each axis squared √(x_i^2 + y_i^2 + z_i^2) over the ED
- **Time Between Peaks**[3]: Time in milliseconds between peaks in the sinusoidal waves associated with most activities (for each axis)
- **Binned Distribution**[30]: We determine the range of values for each axis (maximum – minimum), divide this range into 10 equal sized bins, and then record what fraction of the 200 values fell within each of the bins.
Machine Learning

- Pull features + activity labels into Weka (or other Machine learning Framework)

- Export classifiers as Java JAR file
- Run classifier in your app
- Given an accelerometer pattern while user is performing activity => Guess (infer) what activity
Accuracy of Classifiers

- Classifiers can achieve > 90% accuracy for most activities

<table>
<thead>
<tr>
<th></th>
<th>% of Records Correctly Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>J48</td>
</tr>
<tr>
<td>Walking</td>
<td>89.9</td>
</tr>
<tr>
<td>Jogging</td>
<td>96.5</td>
</tr>
<tr>
<td>Upstairs</td>
<td>59.3</td>
</tr>
<tr>
<td>Downstairs</td>
<td><strong>55.5</strong></td>
</tr>
<tr>
<td>Sitting</td>
<td><strong>95.7</strong></td>
</tr>
<tr>
<td>Standing</td>
<td><strong>93.3</strong></td>
</tr>
<tr>
<td>Overall</td>
<td>85.1</td>
</tr>
</tbody>
</table>
What if you don’t know Machine Learning

- Visually inspect accelerometer waveform, come up with rules by trial and error
- E.g. If (min-max range < threshold), activity = sitting
Inference across multiple sensors

- Note that features can be from multiple sensors
- E.g. accelerometer features, gyroscope features, web URL features, etc.
Finding Idea to Work on
Pick an Idea to Work on

- Examples of previous projects from grad class:
  - Hearing aid
  - WiFi vulnerability
  - Mobile tweeter mining (mobile computing, ubicomp stuff),
  - weather prediction along user’s path

- Projects from Andrew Campbell class
  https://docs.google.com/document/d/1hg44pm9PPnPnxBfNthAktUD9XoHBLmkMdq6BmJiWJ/pub

- What else is detected in ubicomp (5W’s, 1H), examples ideas, how to do it in Android
Coming up with a Project

1. Click on papers,
   i. What areas you like?
   ii. What are your strengths? Machine learning? Signal processing?

2. Find papers you like within area or search Google Scholar, ACM digital library or IEEE Xplore

3. Can each paper be extended?
   a. Look at future work
   b. Repeat experiments + other things they didn’t try. E.g.
      i. Re-implement a simple idea: E.g. Bewell
      ii. Implement PART(S) OF complex idea (e.g. place sense paper)
      iii. Propose new idea based on your prior knowledge/experience (GREAT!!! Maybe publishable?)
Other Random Project Ideas?
Some Project Ideas

- **Machine learning:**
  - Detect personality type from detecting/analyzing daily interactions.
  - E.g. number of friends seen per day, number of people talked to per day, activity levels/type, etc.

- **Signal/processing:**
  - Detect speaker, extract conversations, convert speech to text, record
  - Detect emotion/stress levels from speech
  - Detect sleep duration, quality detection from accelerometer, microphone (iSleep paper)
Some Project Ideas

- **Image/Video Analysis:**
  - Detect a person's emotion/mood from an image video of their face
  - Detect if a person/student watching a youtube video is engaged/not engaged

- **Mobile Twitter**
  - Search Twitter messages, analyze how much important mobile topics are being discussed (e.g. security, malware, health)
References

- Busy Coder’s guide to Android version 4.4
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014