CS 520
Final project description

Final projects will be completed in teams of 4 or 5 students. Each team is responsible for a single project.
You should select a team and a project by **Thursday, October 17, 2019, 9:00AM EDT**.
Your mid-point check-in will be on **Thursday, November 7, 2019, 9:00AM EST**.
The final project will be due **Tuesday, December 10, 2019, 11:55 PM EST**.
There are four options for a final project[1] (each team will do one):

1. MSR 2020 Mining Challenge
2. Replication study
3. Mechanical-Turk-based user study of bias in ad recommendation systems
4. EleNa: Elevation-based Navigation

MSR 2020 Mining Challenge

The Mining Software Repositories conference runs an annual challenge in which they provide a dataset and ask you to answer research questions about the dataset. Read the description of this year's dataset, research questions, and challenge here:
https://2020.msrconf.org/track/msr-2020-mining-challenge#Call-for-Papers

Replication study

A replication study takes an existing research paper, replicates its experiments on the same data, and then extends the experiments to expanding that data set on which the experiments are run. For this project, we highly recommend selecting a paper with publicly available dataset and code to execute the experiments. The project involves a write up describing the process of replicating the experiments, deviations in the achieved results from the original ones reported in the paper, and lessons learned from applying the experiments to new data.

Here is a list of several papers that are good candidates for replication:

1. Automatic generation of oracles for exceptional behaviors from Javadoc comments.

   Paper: https://dl.acm.org/citation.cfm?id=2931061

   Source code: https://github.com/albertogoffi/toradocu

2. SimFix: Automated program repair

   Paper: http://sei.pku.edu.cn/~xiongyf04/papers/ISSTA18a.pdf

   Source code: https://github.com/xgdsmileboy/SimFix

   Dataset: https://github.com/rjust/defects4j

[1] In unusual cases, it is possible to convince the professor to do a self-defined project.

3. EvoSuite: Automated test generation

   Paper: https://dl.acm.org/citation.cfm?id=2685612

   Source code: http://www.evosuite.org/ and https://github.com/EvoSuite/evosuite

   Dataset: https://github.com/rjust/defects4j

4. Are mutants a valid substitute for real faults in software testing?

   Paper: https://homes.cs.washington.edu/~mernst/pubs/mutation-effectiveness-fse2014.pdf

   Source code and dataset: https://github.com/rjust/defects4j

5. SOSRepair: Expressive Semantic Search for Real-World Program Repair?

   Paper: https://people.cs.umass.edu/~brun/pubs/pubs/Afzal20tse.pdf

   Source code: https://github.com/squaresLab/SOSRepair

   Dataset: https://github.com/squaresLab/SOSRepair-Replication-Package

Mechanical-Turk-based user study of bias in ad recommendation systems

Companies such as Google train machine learning models to suggest which ads are relevant to which web pages and to which search queries. The training data they use are generated by humans — they show humans examples of ads, web pages, and queries, and ask them to rank which how relevant the ads are to the pages and queries.

This project aims to answer the question of whether different demographic groups (e.g., people of different gender, people of different race, or people in different locations on Earth) demonstrate different preferences for ads, and whether machine learning models trained on data generated on one group makes accurate predictions for other groups.

If you are interested in doing this project, contact Yuriy via email and he will meet with you to discuss project details. This project will expose students to some research, user study design, and more aspects of fairness in software engineering.

EleNa: Elevation-based Navigation

Navigation systems optimize for the shortest or fastest route. However, they do not consider elevation gain. Let's say you are hiking or biking from one location to another. You may want to literally go the extra mile if that saves you a couple thousand feet in elevation gain. Likewise, you may want to maximize elevation gain if you are looking for an intense yet time-constrained workout.

The high-level goal of this project is to develop a software system that determines, given a start and an end location, a route that maximizes or minimizes elevation gain, while limiting the total distance between the two locations to x% of the shortest path.

Components:

Your software system will most likely have four main components:

1. Data model that represents the geodata.
2. A component that populates the data model, querying, e.g., OpenStreetMap.
3. The actual routing algorithm that performs the multi-objective optimization.
4. A component that outputs or renders the computed route.

While all components are necessary for a working prototype, you may choose to focus on some of them in greater detail. For example:

- If you focus on developing and experimenting with several routing algorithms, it is sufficient to have a simple interface for entering the start and end location and a simple output that represents the route.
- If you focus on a sophisticated UI with proper rendering of the computed route, it is sufficient to have a basic data model and routing algorithm.

Resources:

- The A$\star$ algorithm: https://en.wikipedia.org/wiki/A*_search_algorithm
- Dijkstra's algorithm: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- OpenStreetMap wiki: http://wiki.openstreetmap.org/wiki/Main_Page
- The following paper, in particular Section 2, provides a very accessible introduction and overview of metaheuristic search algorithms:
  https://pdfs.semanticscholar.org/9c83/752460cd1024985981d4acaa7bc85e15c0f7.pdf