Software Engineering Large Practical

s1235260 Francisco Vargas

December 18, 2014

1 Development Process

1.1 Original Specification

The Original Specification comprised a web app in which users could rank paths. Paths being streets edges walks across parks etc. Besides the original path ranking, non users/users should be able to obtain a random walk that will be biased taking into account the ranking of previous users. The idea behind this is to throw a dice which is biased on users weightings at each decision step. A decision step in this sense is simply moving from one edge to another aiming to reach a final destination. So if the user inputs a start point and an end point, he/she will get a random path which may probably be pleasant. This specification was met fully.

1.2 Challenges and Solutions

The biggest challenge was working with spherical polar coordinates (the earth can be approximated to a sphere) and ensuring the path stayed sort of on track. The movements here are probabilistic (pseudo random) and hence constraining such movements is very difficult. Additionally, understanding the cosine law for spherical polars and implementing it to give constraints on the queries was a major challenge in this project.

Another significant challenge was finding a dynamic map. This took about 2 weeks of research before the start of the project. I used Mapbox and not leaflet like I originally proposed. Nonetheless their API's are connected. In terms of the data, which provides addresses (geocoding and reverse geocoding), I used openstreetmaps. I started using google maps but encountered many traffic problems.

Working in the web development framework was remarkably challenging since it is something I am new to. Due to this lack of experience, I made several designs and choice errors which I will expand on later chapters of this report.

1.3 Source Control (git)

As required git was used for source control. More specifically, I used a private repository in GitHub since I enjoy the graphs they provide. I find visual the aid useful when dealing with branches.
The general approach was to add elements with their relevant commits as time progressed. Commits became shorter and less clauses were used within them. When confronting a major bug fix, I would branch away from master and merge after the bug had been dealt with.

1.4 Unit Testing

I carried out unit tests formally towards the end of the project to verify if the custom mathematical functions I used worked as expected. Something important to note is that I should have carried out unit tests earlier on in the development process and along with branches and bug-fixes. I did not manage to do so since I am new to this development process and work-frame which led me to a struggle in the goal of managing everything perfectly.
I would like to point that I made an attempt to replace view tests by clicking on buttons and querying the database using sqlite3.
There is an interesting test inside the unit test folder which I think is worth mentioning. The test *visual_dice_test.py* computes histograms of two different weight distributions. One distributed in a symmetric normal (but discrete) like manner and the other uniformly distributed. The dice is rolled from 100 to 9000 times and its histograms at different number of roles are computed and plotted as png files. By inspecting these images one can confirm that the biased dice does indeed work. A formal unit test may not be relevant here since its probabilistic behavior does not give us information that can be asserted.

1.5 Documentation and Comments

During the development process small informative comments and debug logs were made. Towards the end, a more formal and dense documentation was delivered. I used the python enhancement protocol 8 (PEP8) to stylize and format my code nicely as much as I could.

2 Application Dissection

2.1 Functionality

1. The Craft Path button allows registered users to draw an edge and submit a rank on how pleasant they found it to walk that edge.

2. The Random Walk button allows the any user to create a path for the specified start and end points in Edinburgh. This path is created based on user rankings. It considers the top 6 edges from a given start point and it roles a weighted dice based on the rank of those edges. The landing side of the dice indicates what edge to move to. The start edge is updated and the procedure is repeated until there are no closer edges to the end point. The walk takes 1 second to be drawn due to a delay programed on the JS side.

3. Weights can be updated by users registered or not after they generate a random walk by entering a number between 0 and 100. This new weight is then averaged with the old weight of every single edge in that walk and this becomes the new result per edge.

4. Registered Users have a count (rank) of how many paths they have crafted.

2.2 Distance Metric on Latitude and Longitude

I required a distance metric between coordinates and two possible choices emerged:

1. Assuming the earth is flat for a small area such as Edinburgh. The following mathematical fact lead me in supporting this assumption:

   $$\lim_{x\to 0}\{\sin(\theta)\} = \lim_{x\to 0}\{\sum_{0}^{\infty}(\frac{-1^k\theta^{2k+1}}{(2k+1)!})\}$$

   as the angle θ gets very small (goes to 0), the non linear terms in the taylor expansion become very small which can lead us to discarding them, yielding:

   $$\sin(\theta) \approx \theta$$

   having $\sin(\theta)$ as the line segment corresponding to the arc segment θ one can see that this approximation seems reasonable for very small values. Instead of working with arcseconds and estimating how good this approximation would be, I decided to work under the assumption that the earth is round.

2. Using the spherical law of cosines the following formula can be derived for the distance of two coordinates given by latitude and longitude:

   $$\text{distance}_{sf} = \cos^{-1}(\cos(90-\phi_s)\cos(90-\phi_f) + \sin(90-\phi_s)\sin(90-\phi_f)\cos(\lambda_f - \lambda s))$$

   Full derivation of this law can be found in:

   http://www.math.unl.edu/ shartke2/teaching/2011m896/SphericalLawOfCosines.pdf

   Now using the fact that $\cos(90-\theta) = \sin(\theta)$ and $\sin(90-\theta) = \cos(\theta)$ we have the following:

   $$\text{distance}_{sf} = \cos^{-1}(\sin(\phi_s)\sin(\phi_f) + \cos(\phi_s)\cos(\phi_f)\cos(\lambda_f - \lambda s))$$

   This is the formula I used as a distance metric in my app. Working in polar spherical coordinates was quite a challenge but I considered it would be a good approximation to distances on the earth.

2.3 Random Walk

This is the main functionality of the app. a user enters two places in Edinburgh preferably by clicking on the map get the correct address. There is no need to include city of Edinburgh, Scotland.

After the button is pressed the top 6 ranked edges within one Km range that get closer to the end point are queried and their ranks are passed as parameters to the dice (decision_at_node_N). This picks one of the 6 edges to go to and repeats the process till the query no longer bring results i.e there are no more paths in the database that bring you closer.

This functionality was one of the biggest challenges. It is implemented as a while loop in a subsection of the post in the main view. It would have been a good idea to wrap all of it in a function that residing in map_graph.py to increase readability and maybe allow further testing.

3 Mistakes and Future Improvements

3.1 Test Driven Development

During my test cycle I believe I should have been testing whilst implementing features. If I had started testing at an early stage I would have figured out how to test views in the flask framework and saved time with manual tests verifying redirections etc.
This has been my first big programming project and web development. It was all quite new hence I made the novice mistake of leaving testing for the last moment.

3.2 Git Usage

I am glad with some of my git usage. I decided to be brave enough and try out branches when I thought they would be of great use (solving issues etc.). However use of git sped up my developing process significantly . Yet I still believe I should have broken my commits into smaller commits for two main reasons:

1. Smaller commits may overall add more documentation to the development process since commit descriptions will be mapped to smaller units of code;

2. Smaller commit messages are more readable and straight to the point.

Again the reason why I did not do this was the fear of submitting code with bugs plus the fear of leaving things out when committing. Given how git works the latter should not have been a worry.

3.3 Documentation and Code Structure

The idea of self documenting code is not strong through every line in my source code. The reason for this is that a lot of big and repeated procedures can be split up and higher order processes that can be merged into one function to keep modularity.
One bad example is my post method in the main view. I wanted to make an interactive template for the map so I just sent conditions between javascript and the server via a variable through jinja 2. This made my post method massive requiring a vast amount of comments to be understandable. Ideally I should have taken the computation to a different route catering for this single task.
In my javascript mapcon.js I used global like variables to pass conditionals in between functions. I believe this is not the best way to go about it and with more time and research I would have devised a better and more understandable mechanism to achieve these tasks.

3.4 Time and Planning

This is the section where I would partially like to give myself a pat on the back. The proposed idea was significantly challenging. The more I coded it the harder it became. Nonetheless, as estimated, a full month of working just on the app would be enough to complete the two requirements I proposed plus a little bit more.
Mathematics and design are quite close to my heart so, even though design may not be graded, I spent a good time customizing the CSS and trying to implement nice dynamic features since it is something I enjoy. I believe this may have been a setback time wise since I was left with little time to test the code. That being said, I should have tested from the start of the project.
My time was split in 1 month of revision and learning from October to November and one Month

of implementation November to December. This structure was ideal although I would criticize the learning month. It proved to be artificial and not as helpful as I would have wished. At lot of the learning happened whilst implementing the project.

3.5 Programming paradigms

The overall style is slightly convoluted this is due to pythons multi paradigm nature. Elements of functional programming can be seen in several functions such a list comprehensions lambda expressions and maps. Nonetheless many things could have been further abstracted in to higher order functions. A lot of the data structures used on the algorithmic side are lists of lists; lists of tuples and simple lists whilst these are comfortable to work their readability is not great. A good alternative would have been to use data structures from collections such as NamedTuples or OrderedDicts. Alternatively creating classes and conforming to an OOP style could have also brought an enhancement to my code.

3.6 Database and Queries

My original plan was to create an interface called BaseQueries using the python module abc and its functionality ABCMeta. This would allow me to use any relational database and not just sqlite3. Additionally all the relevant queries would be contained as methods. In a sense this would be quite similar to an ORM so I could have just used sqliteAlchemy.