Project Evolution

The aim of this document is to detail the process of us creating our webapp, Tweet Guard. We aim to account for our journey over the last 8 weeks, by splitting up the time into 4 two week cycles, and showing the implementation decisions that we decided to make, as well as the challenges we faced and limitations we discovered along the way. We hope this document will go alongside our project brief in highlighting the changes in our mindsets that have taken place over the length of this module, and the path we have travelled along the way in creating, managing and finalising our project.

Week 1-2

Implementation Decisions

We decided on the Flask framework for building our lightweight web application.

We used bootstrap in order to create a full-featured front-end with a professional looking style.

Inside Flask, we used SQLAlchemy in order to create and manage the database, which was SQLite, but will be converted to PostgreSQL when the site is finally deployed.

The TweePy library proved to be extremely important in fetching twitter profiles, displaying them and pulling tweets from a user.

Challenges / Limitations

The fetching and scanning of an entire account with a very high amount of followers can be extremely slow from first impressions and this is something that must be looked at.

Maximum of 3000 tweets pulled per user at a time, however this should be ok as recent tweets are the most important.

Requirements Change

At the beginning of week two we had to make the decision to get rid of "scan for all threats" functionality as we came to the decision that it was too complex and not even overly useful, as most user's would rather scan for something specific. Will leave it in "could have" in case we decide it could become useful later on in the weeks.

Week 3-4

Implementation Decisions

We used OAuth functionality to allow users to log in to our application through Twitter, saving the need for us to build our own verification system, and making it easier for us to allow a user to unfollow, block or mute another user.

We discussed our problem from last week of scans taking too long for high amounts of followers and decided due to the nature of the API we are using and rate limits that we should strip the amount of accounts possible to 10 per user per scan, in order to give users a more quick and balanced experience and to stay within the limits of the TweePy API. We note that in a more financially backed project this rate limit would more than likely be quite easily dealt with and we could allow our customers to scan a lot more accounts at the same time.

We eventually decided on using logistic regression to build our racism detection model, and decided to solely focus on implementing a racism detector for the time being, in order to prioritise the accuracy of that detector instead of mediocrity in 3 or 4 different detectors. This will hopefully ensure that we do not leave ourselves with too much development work to do in such a short timeframe, particularly with being a team member down for some amount of time.

The TweePy library was also chosen to be used to unfollow, block, and mute followers of a user.

Challenges / Limitations

We were unfortunately left with being a team member down for a longer amount of time than was originally expected, and this certainly had an impact on our plans for these two weeks. A few requirements changes had to be made in order to facilitate this impact.

We found towards the end of week 4 that after our racism detector had reached a certain level of accuracy, it became exponentially harder for us to increase the level of accuracy without a very high level of understanding on topics of machine learning, Bert models and the latest research in logistic regression and natural language processing. This leaves us with a challenge for the weeks to come, and we are going to try and solve it by 1) expanding our researching upwards further into our current scan model and see how we could improve it via changing the dataset it learns off, adding a birth model and looking into other avenues that would give us the same results, or 2) if this doesn't meet us with good results, trying to research other features to our site, such as the fact checker we had talked about, in order to expand the website's functionality so as to give our users a multitude of options and features to choose from.

Requirements Change

Due to still being a man down, we had to strip down requirements in order to not leave us under an unnecessary amount of pressure going into the latter weeks of the project. We stripped down Evan's task of creating models for 3 to 4 different forms of hate/antisocial behaviour into focusing on one main factor - the one we found most relevant and damaging in today's society: racism. We then also decided that going forward we would not add any more features to our project until we got exactly what we had left to do done. We made a list of 10-15 jobs that needed to be done in the next cycle. We then plan to have a "Beta" released. When we are all happy with how this beta looks and it is tested appropriately, then we will look at adding more features to our site if and only if we feel it is needed and we have an appropriate amount of time to do a good job on it. The phrase of the week is being realistic with the pace and projected output of our project.

Week 5-6

Implementation Decisions

This week was all about fine tuning our application to be ready for a beta release. We started out with an initial list of jobs that needed to be done. We gave out these jobs sequentially by volunteering, allowing the members of the team who felt their skills lied in that particular area to take hold of that particular task. Each member was equally bestowed with work for the next two weeks, with meetings scheduled for towards the end of the cycle to discuss any current problems, collaborate, and identify and future issues.

We eventually decided on using a Bert model in order to improve our scanning algorithm.

Challenges / Limitations

We were not held back too much in this cycle in terms of development, mainly due to 1) having correctly and fairly partitioned the jobs / workload correctly, and 2) being a lot more familiar with the frameworks we were using to complete these tasks, and therefore being able to preemptively identify possible problems before they appeared.

With regards to the Bert model, we noticed after its implementation that it was most definitely a more accurate model. However, it began to use a lot more resources, and therefore took a lot more time to run than other models. Our deployed build is planned to work on around 8GB RAM, and we are unsure as to how effective the Bert model will be under the stress of 1) many people using our site, and 2) high amounts of followers being scanned.

Requirements Change

The only requirement change was limiting the amount of followers that could be scanned for the Beta release, so as to ensure our customers would be able to provide feedback effectively and on all areas of the website.

Week 7-8

Implementation Decisions

For these weeks, the implementation decisions were not entirely up to us. We received multiple reports full of customer feedback and peer reviews, and were given a large amount to cover and fix before our web app's final release.

We made a definitive decision on what bugs we would be capable of fixing (almost all of them), and portioned these out to everyone based on their strengths, again. We then went about working on these as well as perfecting our screencast, lessons learned, and all documentation for final submission.

We used the feedback from our peers in various ways, and we all felt it was really beneficial to have another set of eyes look at our work. We made a list of all positives and tried to reinforce this, and made a list of all negatives (outside of bugs, which I already addressed) and chose what is important enough to implement, and what would have to be left out this time around. Overall, people seemed pretty happy with the site, but here are a few things we decide to change:

1. We decided to give the users an option to unfollow, block and mute people from the database search page, so that a user did not have to be scanned to be blocked etc.
2. We gave some of the database pages styling updates to make them more visually appealing.
3. We improved the login facilities to address bugs some users were experiencing upon first arrival.
4. Our admin system had security issues, so we made changes to it.

Challenges / Limitations

We came to as good a deal as we felt we could make on the performance/accuracy tradeoff of our scan algorithm, which unfortunately will stay limited to a small number of followers, due to the lack of computer resources we have available for the algorithm.

Requirements Change

As this was the end of our project, there were no requirement changes that would be carried forward to other weeks.