Product Brief - Team Software Project 2

RED - changes made due to other comments e.g Highlighted words and sentences in bold to improve readability

GREEN - changes made by our own decision e.g Gantt chart

Overview

Within our project, we aim to create a threat-analysing scanner for Twitter. The scanner will be created as a web application that will allow users to scan accounts for potential threatening behaviour that is considered to be anti-social, and provide the user with a detailed report at the click of a button on what the algorithm has determined to be dangerous. The user then has the option to disregard the report, or review it and make changes based on what they consider to be the main threat.

Our web application will be named **Tweet Guard Threat Scanner,** and will aim to provide a **user-friendly, quick environment that gives an accurate and detailed reflection** of the threat or lack thereof that may lie within the user's Twitter followers.

Overall, the aim is to create a **simple, sleek and minimalistic web app** that offers users an easy way to **filter out any negativity,** antisocial behaviour or danger from their twitter feed with the click of a button. We aim to create an algorithm that is accurate, helpful and will grow stronger as more users use it.

Why Is It Needed?

In digital times, digital crimes are also rocketing. People need a way to keep themselves safe from online threats like sexual predators, fraud accounts, internet trolls, etc. Our web application will warn its users whenever a “threatening account” tries to interact with them. This will be perfect for people such as children, women and others who face the most threat online.

Twitter is an open platform where people can look at and talk about current affairs across the globe. With differing points of view, some people may have a perspective on a topic which might be controversial or offensive to other users. Some may also exploit the platform for malicious purposes.

Twitter has a system in place where if a user believes something violates the platform’s rules they can report it. This helps to indicate to the Twitter team that something is going

on and they can then decide whether or not to take corrective measures. E.g. They may enforce a user to delete their Tweet, or suspend their account permanently (if a more serious/repeat offence is committed).
Violations are grouped under three separate categories:

1. Safety
This includes violent threats, terrorism, child sexual exploitation, harrassment, discriminational hate (e.g racism), intentions/encouragement of self-harm or suicide, sensitive media, and illegal activity.

2. Privacy
Includes disclosing other people's private information, and non-consensual nudity.

3. Authenticity
Includes spam, civic integrity, impersonation, synthetic media, and copyright/trademark.

Although the reporting of these violations on Twitter helps to make the platform a safer place, there are a few problems that it does not address:

- It is **inefficient to depend on Twitter staff** to go through each user report and determine what action should be taken. This takes too long for threats to be identified and eliminated, meaning more users are vulnerable for a longer period.
- **Users will be completely oblivious to a harmful account** as they cannot see if it has been previously reported before.

Removed 2 sentences to be concise^

The web application will quickly solve these problems:

- **Reported information is stored in a database,** so threatening accounts can immediately be identified by users.
- **Users will now be able to scan for harmful accounts and see all details of previous reports.**
- Whenever a threatening appearing account tries to interact with a user, the user will be able to **quickly discover if other people have had the same problem,** or if our algorithms have detected said threatening behaviour.
- The web application will show a danger level so that **threats of all sizes are accounted for**. This allows users to make their own decisions on how restrictive they would like their account to be.

So What Does It Do, And How Does It Do It?

In our web app, the main function that we are choosing to implement is a “scan” function, which uses **sentiment analysis and natural language processing** to provide a full report on a user’s Twitter followers. We aim to implement at least one criteria on which a user can be determined as dangerous, with **racism** within the tweets of a user at the forefront as our principle area of development.
Removed sentences for brevity and to avoid repetition^

The scan function can be used in two different ways:

- The user can **search for a specific account** and receive a full report on the account.
- The user can enter their own Twitter handle and be given analysis of up to 50 of their most recent followers under a selected criteria on which they choose to search, documented as a **full report which highlights the dangerous** users, and, if possible, **will give an option to unfollow a specific user or mass unfollow all threatening accounts.**

When an account is scanned, a full report is either taken from our database, or, if the account is not already present in it, we will scan the account’s recent activity and return the report to the user, adding it to the database for future users’ scans in the process. The returned report will contain information such as an overall “danger level” of the account, the amount of reports/flags our system or other uses have made on an account in the last month, the common reason for reporting and even some of the tweets in question that may contain said threatening qualities or anti-social behaviour.

we first introduce a transfer learning approach for hate speech detection based on an existing pre-trained language model called BERT (Bidirectional Encoder Representations from Transformers) and evaluate the proposed model on two publicly available datasets that have been annotated for racism, sexism, hate or offensive content on Twitter.

Added citation from research paper on why BERT is suitable for hate detection mitigating racial bias

The user will then even be given a chance based on the mentioned tweets to agree or disagree if there was indeed antisocial behaviour involved and to block/mute/unfollow the user.

Another feature we will be adding is a standalone report function, where users can manually send in a report on an account they believe to be threatening. This will allow for our users to alert each other on accounts that are threatening and that may have flown under the radar of our algorithm.

We would like to add more features as time goes on, **and one main feature we would like to add in future is a fact checker**. With this feature, a user could enter a tweet URL and be returned an answer on whether what is stated is true or false. We feel it would be relevant in today's society, with reports of "fake news" being broadcasted on a daily basis. Again, it would be implemented with our natural language processing and machine learning techniques.

What Other Solutions Are Out There?

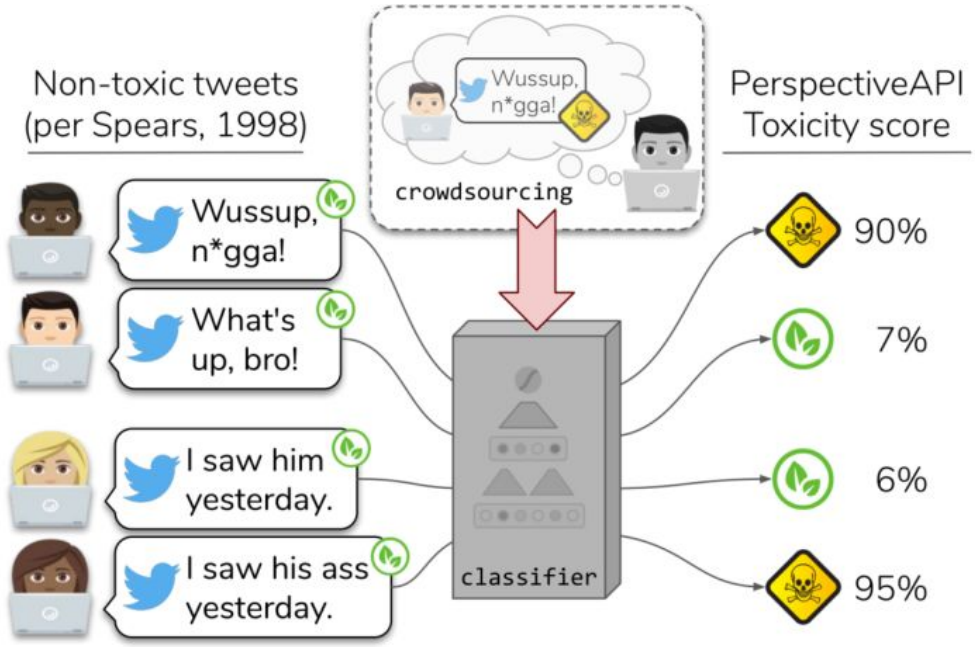
There are currently no solutions on the market that deal with this problem. Software security companies such as Proofpoint, FraudWatch International and Mandiant offer solutions to a problem that comes underneath the same category of social media threat protection. However they only protect branded social media accounts (different target market) and only solve a subset of the problems our product deals with.

Their security solutions monitor a wide range of social media sites for fake or fraudulent accounts impersonating a client's brand. They monitor and detect specific threats across these platforms, alerting the client when a potential impersonator is found which then must be confirmed by the client before the account is subject to removal.

Our product aims to give protection to all users of the Twitter platform specifically because it is where the problem we are addressing is most prominent.

In the past, there has been research done into similar topics that has produced varying results, with some reports of bias towards certain groups of people. We feel that due to the nature of our web app, with its main aim of producing a report for a user for them to then decide if what they have seen is untrustworthy or dangerous, this bias is obsolete. Only the outside bias of a user could influence the quality of our service.

Other solutions fail to mitigate racial bias in detection. In theory the racial bias stems from the training set used to train the TfidfVectorizer. Training sets label text as hate without out considering the context of the text, which proves how the test data that feeds these algorithms are biased from the start



Added paragraph on how other models fail to mitigate racial bias
Cited Image from referenced research paper

Project objectives & flexibility/constraints

	Target	**Tolerance**
Scope	Web Application Reporting System Database of reported threatening accounts Threat alert Danger Level Scan function (Search specific account for report) Sentiment analysis & machine learning	If time left over after necessary features are completed: Can extend functionality to scan all of your followers/ mutual accounts/ unknown accounts sending message requests Fact Checker to determine if a tweet provides true or false information
Time	7 weeks	None as we may finish early but then

		scope/functionality will be extended
Cost	No costs	--
Quality	Front end of application implemented in HTML, CSS, JavaScript, etc. Use a Python Web framework to speed up development Back end database implemented with SQLite Twitter API used for filtering keywords	Can use Bootstrap to implement front end Can use Flask framework instead of Django (maybe more lightweight for the purpose of the project) Database implementation is flexible Tweepy has what we need but pull limit of 3k followers
Risks	Team member becomes ill or cannot complete task for other reason	Spread the remainder of the task workload among other members
Benefits	There are benefits to making the design look nicer or extending functionality to add more useful features	If time is an issue it is more beneficial to keep to a simpler design and to focus on required features only

Our Team

Overview - Roles

We analysed our strengths and weaknesses and for the next 2 weeks have allocated roles which will be reviewed and re-evaluated as the project progresses, when people find out what they enjoy/ are struggling with.

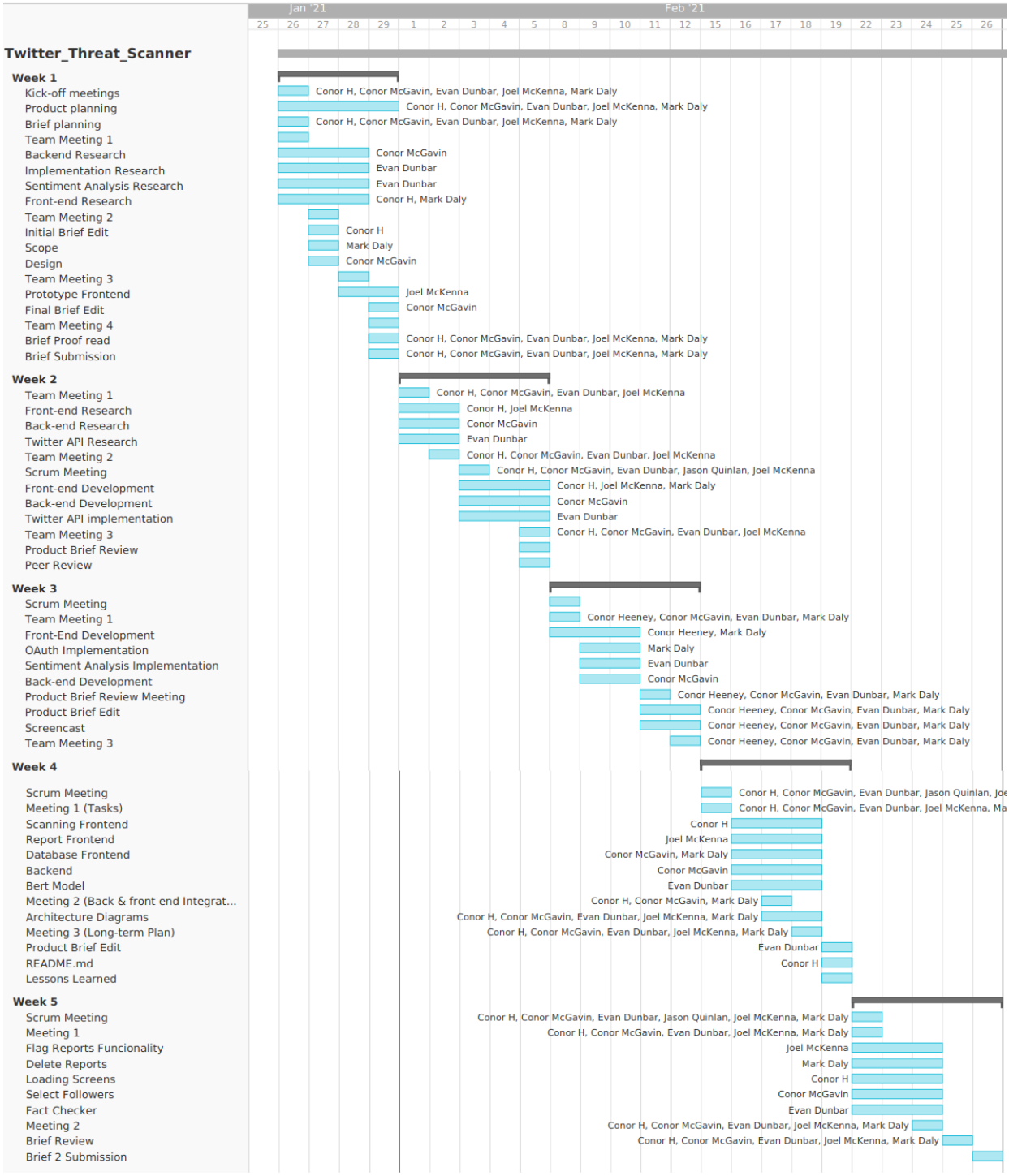
- Evan Dunbar - Bert Model Implementation
- Conor McGavin - Back-end, infrastructure using flask and SQL Alchemy
- Conor Heeney - Front-end design and development, Styling Scanning Pages
- Mark Daly - Front-end design and development, Login Pages and database
- Joel McKenna - Front-end design and development, Report user page

Our Approach

We will be taking an Agile approach to the project using Scrum techniques. The typical routine for a week involves our Scrum meeting with the product owner and the whole team. This is where we discuss reflections on the previous week and what needs to be done in the upcoming week. We then have a meeting with the development team, where we discuss the distribution of tasks among members. We then work on our own tasks, separately for the most part. About halfway through the week, we have another meeting to discuss progress and issues. Towards the end of the week, we create the Gantt chart for the week and update the product brief to reflect any changes to the specification.

We aim to understand each other's roles and work closely and collaboratively to produce an environment where one team member will be able to work independently of another, but know what is expected of them to produce from other team members. This ensures fairness and that there is never one person that feels they are doing everything. We aim to closely follow a plan and reassess on a weekly basis on where the project is, and where it needs to be.

Gantt Chart



Added Updated gantt

Conclusion

We hope this document puts into perspective both our vision of our product and our vision for good team communication and planning.

We are happy with our current product specifications and although we feel we have a lot to cover, we hope this product can continue to evolve, with more functionality and polish than is outlined in this document.

Thank you for taking the time to read this.

Added MoSCoW

MoSCoW

Must Have:
Scan Function - scan all followers and scan specific person for all threats
Report Function
Database of Reports
Login functionality w/ or w/o twitter integration

“Total reports by user” page
User lookup with scans and reports

Should Have:
Account summary / profile with calculated danger level and other statistics

Could Have:
Fact checker

Won’t Have (this time around):
N/A

Added Harvard Style referencing

References

CNRS UMR5157, T´el´ecom SudParis, Institut Polytechnique de Paris, Evry, France, 2020. *Hate Speech Detection and Racial Bias Mitigation in Social Media based on BERT model*.

Codementor.io. 2021. *10 Web App Ideas You Can Build As Side Projects | Codementor*. [online] Available at: <https://www.codementor.io/@mahil/10-web-app-ideas-you-can-build-as-side-projects-1bimcrdepx> [Accessed 21 February 2021].

Help.twitter.com. 2021. *The Twitter Rules*. [online] Available at: <https://help.twitter.com/en/rules-and-policies/twitter-rules> [Accessed 21 February 2021].

Sap, Maarten, et al. *The Risk of Racial Bias in Hate Speech Detection*. Association for Computational Linguistics, 2019.