A Framework for Automated Social Media Post Collection and Tailored Response Generation

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Ethan Chen

Spring, 2022.

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Rosanne Vrugtman, Department of Computer Science

A Framework for Automated Social Media Post Collection and Tailored Response Generation

CS 4991 Capstone Report, 2022

Ethan Chen
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
ejc5bgf@virginia.edu

Abstract
I interned for a startup credit card company that lacked the funding to staff sizable customer service and marketing teams. The startup needed a cost-efficient method to market its product and to conduct customer service online. To address the startup's needs, I created a framework capable of reading, storing, and crafting tailored responses to posts on various social media platforms.

I implemented the framework in the Python scripting language, and designed the code to be convenient to extend to other platforms. The framework implemented Twitter functionality, and implemented untested interaction with Google Play Store reviews. More work needed to be done to test program correctness, to integrate existing standards into the program, and to improve horizontal scalability.

1. Introduction
The credit card startup performed most of its business online through web and mobile applications. It did not have brick and mortar locations. As such, most customer complaints, feedback, and requests for support were posted online on social media. The company needed a low cost method of interacting with customers across various platforms.

The startup tasked me with creating a program to automate customer feedback collection and response on social media platforms. The program needed to be well documented, maintainable, and extendable. The project would improve customer service and feedback collection with minimal labor overhead.

2. Related Works
I implemented my program using the Python 3 scripting language [6]. My code utilized many built-in Python 3 modules. One third party Python module I used was Tweepy. The Tweepy library is a python wrapper. Tweepy allows developers to utilize Twitter's application programming interface (API) through python function calls [4]. The Twitter API can be used to scrape tweets and mentions off of Twitter, and to post tweets or replies onto the website [7]. I also used the Google Play Store's Reply to Reviews API to collate and respond to reviews on Google Play [5].

The two application interfaces enable the programmatic retrieval and creation of posts. My project builds upon Tweepy and the Reply to Reviews API by consolidating their functionalities under a single user interface, and by allowing users to schedule posts and define responses to feedback in advance.

3. Project Design
Several critical design decisions were made over the course of the project. In section 3 I provide an overview of the reasoning and implementation of my code base.

3.1 In-Depth Requirements
I was tasked with creating an automated solution for two platforms: Twitter and the Google Play Store. Twitter and Google Play are structured

differently. Twitter allows users to make posts on their own profile page and leave comments on others' posts, while Google Play restricts user interaction to app reviews.

However, managing separate frameworks for each social media platform would be labor intensive. It was my job to generalize the functionality of the two platforms and to create a single program capable of interacting with both Google Play and Twitter.

There were several specific requirements for the framework. First, the program had to periodically gather user feedback and save it in a database. The form of user feedback varies by social media platform. On Twitter, feedback is considered to be tweets that tag the company's handle. On Google Play, all reviews are considered customer feedback. Second, the framework had to be able to schedule posts. Finally, the program was to respond to customer feedback based on user-defined templates.

3.2 Program Architecture

The program was split into 8 modules. The *GooglePlayStoreInterface* and *TwitterInterface* are wrappers for the Google Play Store and Twitter APIs. Other modules call the *GooglePlayStoreInterface* and *TwitterInterface* modules to retrieve or post data to Twitter or the Play Store. The *DatabaseInterface* module interacts with the local database. *DatabaseInterface* handles database creation, storage, and retrieval. Any module that wishes to interact with the database instead interacts with *DatabaseInterface*. *LogHandler* is responsible for the creation and storage of program logs. Logs are used for debugging. The *TemplateHandler* handles the storage, retrieval, parsing, and matching of response template files. The *ReplyUpdateHandler* periodically calls the *TemplateHandler* to generate responses to reviews stored in the local database. The *DatabaseUpdateHandler* automatically retrieves user feedback from the *GooglePlayStoreInterface* and *TwitterInterface* for storage in the local database.

The design decision to split the program into independent modules with well-delineated responsibilities aided code organization and documentation.

3.3 Reply Templates

Program users can create template files to specify how the program responds to user feedback. Templates have whitelist and blacklist regular expressions. Templates also have a priority rating. If the whitelist regular expression is matched, the template is added to a pool of possible templates used to respond to feedback. If the blacklist regular expression is matched, the template will not be used to generate a response. The blacklist and whitelist features can be used to prevent the program from responding to feedback with slurs or profane language.

The priority rating is used to determine which reply template is chosen from the pool of applicable templates. The template with the highest priority is chosen. If there are multiple templates with the same priority score, a template is chosen at random. The inclusion of random choice was a deliberate design decision. Randomness can be used to introduce variety into generated responses.

The template text can include the $name$, $week_day$, $month_day$, and $year$ variables. The four variables are populated during reply generation, and are context sensitive. The $name$ variable is substituted with the name of the customer, $week_day$ turns into the current day of the week (in UTC), $month_day$ turns into the current month, and $year$ turns into the current year.

3.4 Post Templates

Program users can also create post templates to schedule repeating posts. The *Interval* field is used to specify whether the post is made on a daily, weekly, monthly, or yearly basis. The *Schedule* field is used to specify when the post is made. Post templates can also make use of the $week_day$, $month_day$, and $year$ variables.

3.5 Feedback Collection and Storage
DatabaseUpdateHandler regularly calls the *GooglePlayStoreInterface*, *TwitterInterface,* and *DatabaseInterface* modules to collect and store customer feedback. However, since Google Play Store reviews have different metadata compared to Twitter tweets, different database schemas were designed for each type of feedback. As a result the data for each platform are stored in a separate table on the local SQLite database.

4. Results
Both the Twitter and Google Play Store functionalities were implemented. The portion of the codebase for Twitter was fully tested. However, the Google Play Store application interface was untested when I left the company. The Reply to Reviews API requires an approved Play Store account and a published application. The startup's Play Store account was off the table, as it was a bad idea to pollute the official account with fake applications. I created my own Play Store account and developed a test Android application to publish to the platform. However, my app was not approved by the end of the summer.

I am unsure if my application was adopted by the startup. I developed the code, but was not responsible for deployment. There was still testing work to be done when I left the startup. I was not around long enough to discover what became of my project.

5. Conclusion
I created a framework capable of interacting with multiple social media platforms from a single, unified user interface. Only one tool needs to be modified and maintained to interact with any social media platform.

The framework is capable of accepting user-defined templates to procedurally schedule posts and respond to user feedback. The usage of templates helps developers avoid the modification of source code when implementing new responses and posts. The single user interface and the implementation of templates represent significant savings in the time and labor costs of social media feedback collection and response.

6. Future Work
The application needs to undergo more testing before deployment. The Google Play Store functionality of the application was untested. Multithreading was also implemented in a lackluster manner. I was not knowledgeable about test-driven development at the time of my internship, and multithreading testing was carried out by hand. My testing does not meet deployment requirements. Unit tests need to be written to ensure the thread safety of the code base.

Some aspects of the code need to be rewritten to better fit within existing standards. The template format I created was a simple text file. The program should be modified to accept templates in XML format. The database should not be stored locally but instead on a cloud service like AWS RDS. Additionally, the program should be modified to be runnable on a service like AWS Lambda for increased scalability.

7. UVA Evaluation
I had just finished my second year at the time of my internship and had yet to take the majority of my CS courses. As a result, there were some major flaws in both the design and implementation of my program. However, even if I were to tackle the project after four years of UVA CS, I still would not feel confident in my ability to design and implement a good framework. A large part of my anxiety stems from my lack of design pattern knowledge. Design patterns are as important as algorithms, and UVA should integrate design pattern education into the required curriculum.

I entered college planning on getting a computer science degree and entering industry. My career plan changed in my final year here. I became aware of research as a career path and began to involve myself in extracurricular research projects. I think the UVA CS curriculum could

do a better job of introducing and preparing students for research. The University of Rochester has undergraduate CS seminars that teach basic academic skills and introduce students to a specific research area [1, 2, 3]. Seminar courses are often centered upon reading research papers. I think that the UVA CS program should implement such a course. It would help draw students towards undergraduate research.

References

[1] Chen Ding. 2002. CSC 200 Undergraduate Problem Seminar, Spring 2002. University of Rochester. Retrieved April 27, 2022 from https://www.cs.rochester.edu/~cding/Teaching/200Spring2002/#Info

[2] Lane A. Hemaspaandra. 2015. CSC200/200H: Undergraduate Problem Seminar. Retrieved April 27, 2022 from https://cs.rochester.edu/courses/200/spring2015/150114-handout-KeyCourseInfoSyllabusVersion1pt4pt0.pdf

[3] Fatemeh Nargesian. 2022. CSC 200/200H - Spring 2022. Retrieved April 27, 2022 from https://fnargesian.com/assets/pdf/courses/csc200/CSC_200_Spring_2022.pdf

[4] Joshua Roesslein. 2022. Tweepy Documentation. Retrieved February 23, 2022 from https://docs.tweepy.org/en/stable/

[5] 2021. Reply to Reviews. Google Play Developer API. Retrieved February 23, 2022 from https://developers.google.com/android-publisher/reply-to-reviews

[6] 2022. The Python Standard Library. Retrieved February 23, 2022 from https://docs.python.org/3/library/

[7] Twitter API Documentation. Twitter Developer Platform. Retrieved February 23, 2022 from https://developer.twitter.com/en/docs/twitter-api