\subsection{Selecting a forum}

Selecting a forum proved to be a surprisingly challenging part.

I had initially decided to use GitHub commits as a dataset. I downloaded the dataset and played around with it to familiarize myself with it. But on reading more papers, I found that the GitHub dataset had been used quite a bit in sentiment analysis research. A lot of the ideas I had for the project had already been previously done.

I had to pivot towards a new topic and dataset. For this I looked around for forums that had more substantial sets of text in their comments than GitHub commits. I felt this would give a better idea of user sentiment since the length and nature of commit messages would limit how much the user was able to express his opinions.

Another consideration was the topic of the forum. My first instinct to mine for user comments was Reddit. I use Reddit a lot and was very familiar with how it worked. However, Reddit comments often tend to delve into unrelated jokes and subthreads which detract from the topic at hand. I wanted a forum that was more focused so that users could be evaluated in a more consistent way.

For this reason I chose HackerNews. HackerNews is a link-based social website that focuses on computer science and entrepreneurship topics. It is backed by YCombinator and thus tends to heavily prioritize new and upcoming topics in these fields. It has an active and passionate user base. This user base tends to be more focused and tend to stay on-topic quite a bit more than Reddit and other forums, making it an ideal candidate to mine for user comments.

\subsection{Obtaining the data}

Kaggle is a website that contains datasets from a wide variety of sources for a wide variety of purposes. I decided to use Kaggle to obtain the data I needed for this project. Kaggle had a HackerNews dataset ready. The approach was to create a Kaggle notebook with the dataset I needed that I could then manipulate within that virtual environment.

However, this proved to be a bit problematic as there was a bug because of which the data didn’t load along with the notebook. I had to look for alternative approaches to obtain the data. I found that HackerNews uploaded their data onto BigQuery. Kaggle allows you to access BigQuery using their API without having to create an account or obtain any authentication details.

The next step was to learn how exactly to query BigQuery for the data required. I found that this required writing SQL queries. I first wrote an SQL query to retrieve the top users on HackerNews. While this worked, calculating this using queries threatened to overwhelm the usage limit I had on the Kaggle notebook.

I then found that HackerNews had a page that listed the top users directly. I opted to forego the SQL query and directly use the list that HackerNews provided.

I then needed the list of comments for each of those users, along with at least the year in which they were posted. For this I wrote another SQL query. I ran this query in a loop through the list of top users and saved the results to a separate file for each user. Downloading the files was required since I didn’t want to risk going beyond Kaggle’s usage limits.

\subsection{Selecting a sentiment analysis tool}

The previous research done on sentiment analysis considerably helped with the search for a sentiment analysis tool. Since HackerNews comments primarily deal with computer science topics, a tool specifically geared towards the software engineering domain is more likely to give a good result. Thus, for this reason, I chose SentiStrength-SE.

SentiStrength-SE is based on the original SentiStrength tool. However, it is not actively updated, so a lot of SentiStrength’s newer features are not present in SentiStrength-SE. Key among them is the ability to use the tool from the command line. This breaks automation since I would have to select each file for analysis manually and also select the location to save the output to.

By default, the tool gives out a score between -5 and +5, indicating negative and positive sentiment respectively. There were other options as well, but changing them doesn’t affect the underlying way the sentiment is calculated and the default gives the widest range of values.

\subsection{Preprocessing}

The data in its raw form is not suitable to do any analysis on since it contains a lot of noise and unusable artifacts. This caused problems at various later stages. Thus, it requires some preprocessing before it can be used. For this, I elected to write a Python script to perform the preprocessing steps. Some of the steps taken are outlined below:

\begin{enumerate}

\item File encoding

The first concern was to make sure the file was opened using the right encoding scheme. Since these were comments made on the web, we cannot expect them to strictly be ASCII encoded. Thus this special handling was required.

\item Quotes

One of the first and biggest considerations was the use of quotations. Like most forums, HackerNews allowed users to use quoting notation to quote other users or people, either while replying to someone or simply to quote someone externally. However, since these words were usually not the words of the user themselves, it became important to remove these words from the corpus of text to be analyzed.

The key here was to find a way to demarcate which parts were words written by the user and which parts were just words that the user was quoting from someone else.

I observed that the quoted text were surrounded by the HTML <pre> tags, so the step involved carefully removing the words enclosed within these tags.

\item HackerNews-specific formatting

There were some tags that were HackerNews-specific and need to be removed. This usually looked like a ‘>’ followed by a ‘<p>’ tag.

\item XML tags

XML tags, particularly HTML tags, were used to enable richer text formatting for user comments. This was mostly used for bolding and italicizing text for emphasis. These tags provided no benefit to the sentiment analysis tool, so they needed to be removed as well.

\item URLs

URLs similarly do no provide a very meaningful input to the sentiment analysis tool and so can be safely removed.

\item Escaped Unicode characters

A lot of special and non-standard characters are ‘escaped’, a process by which they are represented using standard ASCII characters. To get an accurate sentiment, they have to be ‘unescaped’ to give the actual characters that were meant to be there.

\item Tabs (\t)

The way a user formatted their comment meant that sometimes a tab character (\t) would appear in the raw text version. However, the sentiment analysis tool used these tab characters to distinguish separate columns. Thus, I replaced all tab characters with spaces, since it ensured there was still whitespace between two characters while not affecting how the sentiment analysis tool read the comments.

\item Timestamp

The raw data contains the exact timestamp of when the comment was posted. While this is good information, for the purpose of this research, only the year is relevant. Extracting the year from the timestamp thus became another task to perform.

\end{enumerate}

\subsection{Processing}

Once the preprocessing script was run across all the files and the cleaned files were obtained, it was time to perform sentiment analysis on these files.

The steps to perform sentiment analysis were:

\begin{enumerate}

\end{enumerate}