Luke Manzitto and Casey Bates

CSCI 182 - Web and Data Mining

Programming Language Popularity on Twitter

For our project, we aimed to determine the popularity of different programming languages from user sentiment on Twitter. To do this, we scraped Tweets using the searchtweets-v2 Python wrapper for Twitter's search API to find which languages are mentioned and their frequency. We then took the content of Tweets mentioning these languages and performed a basic sentiment analysis to determine if the general consensus about each language is positive or negative. Using this analysis, we can draw conclusions about a language's relative popularity based on its mention frequency and average sentiment.

To scrape the data, we used a combination of the Twitter API itself and a Python wrapper for the API called searchtweets-v2. The Twitter API gives direct access to tweets, and allows us to search Tweets based on keywords such as names of programming languages. This API was built to support a project like this, as they specifically mention in their documentation that it can be used to perform sentiment analysis on historical data. The searchtweets-v2 library provides a Python wrapper for the "search recent tweets" endpoint of the Twitter API. This makes scraping tweets much easier, as API requests will not have to be formatted programmatically, and data will be returned in an easily interactable format.

Since we will be using these official tools to scrape data from Twitter instead of using a bot to crawl the website, we do not have to worry about following Twitter's robots.txt, as the API won't scrape those pages by default. Since used Twitter's own official API, as well as a Python wrapper for the same API, the ethics of scraping these Tweets is quite simple. We need not worry about violating the company’s privacy guidelines, as we will be using their own tools to conduct our research.

Once collected, the tweets are analysed to determine which languages are being discussed and what sentiment is being used in reference to each. Scraped data is stored using Python’s native dictionary data type, making it easy to store and retrieve language specific data. We used the natural language processing library nltk to remove stopwords and punctuation from each tweet. The matplotlib and WordCloud libraries were extremely useful in creating word clouds from specific posts to help identify sentiment, as well as charting sentiment values of each language. Finally, we used nltk’s pre-trained sentiment analyzer VADER in order to identify sentiment surrounding each language, as it delivers results quickly, and is accurate with short sentences like those often found in Tweets.

The twitter API made collecting tweets a simple process, but for our results to be accurate, we had to ensure that the tweets we were collecting and analyzing were relevant to our research. This meant that before anything else, we had to ensure that our query to Twitter’s API was complex enough to filter tweets. We decided to limit the scope of our search to only include tweets that were written in English for example. This was so we would have the ability to assess the data ourselves without entirely relying on our classifier, as neither of us are fluent in a second language. We also excluded retweets, to avoid potential duplicate or outdated tweets. For the queries themselves, we went with two different requests: “favorite programming language” and “worst programming language”. We chose these two as they represented polar opposite opinions. Initially, we had also included a “programming language” request, but we found that this ended up being much too general and would obfuscate our results. This was because many of the languages in our dataset were also the names of common English words. For example, the dataset references a language called “Hope”. We couldn’t account for whether the user intended to reference the programming language, or the emotional concept of hope. By querying for programming languages with descriptive adjectives, we found that the number of mistaken tweets was lowered drastically in our results.

After properly defining our query, our next step was to start cleaning our data. This started with the usual removal of stop words from the tweets, but rather than immediately moving to isolating what languages were mentioned, we decided to use the WordCloud library to show the most common terms associated with “favorite programming language” and demonstrate an unfiltered twitter perspective. We thought that creating these visuals of the most common words in returned tweets would provide an easy way to see what topics were being discussed the most, in addition to a possible way to identify the sentiment surrounding each language. Often our query would result in a variety of tweets that had nothing to do with the users’ favorite programming languages. It could be an advertisement to a bootcamp, a joke about a non-programming language, general thoughts on programming languages, etc.

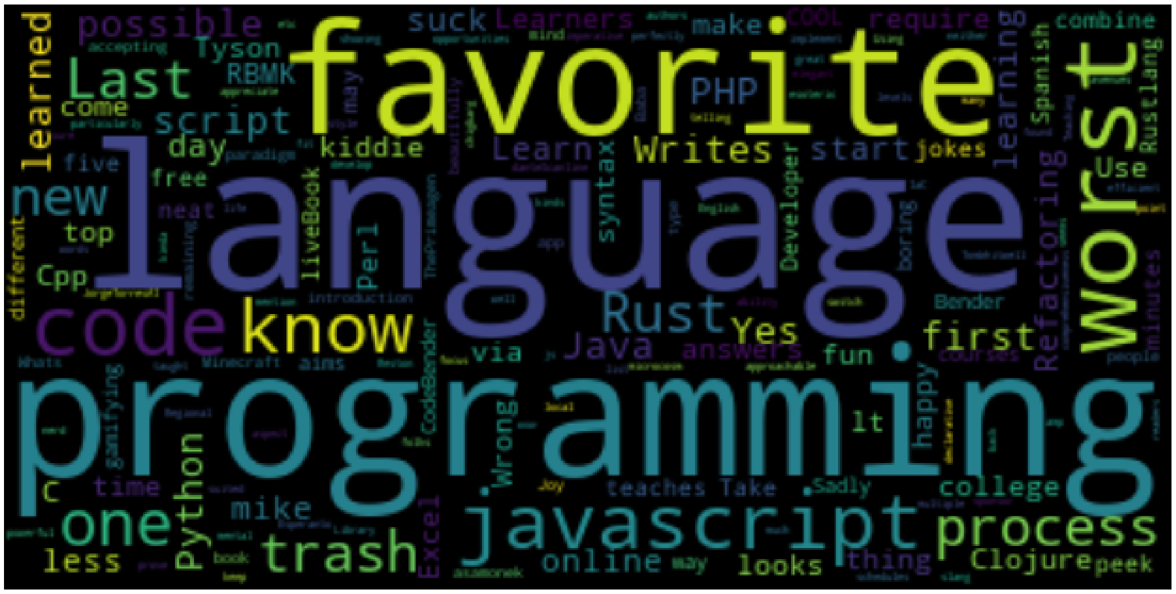


Figure 1: Word cloud based on all returned Tweets

After that, our next step was to use a dataset containing all programming languages to start creating word clouds of specific languages. This dataset came in the form of a JSON file from https://www.back4app.com/database/paul-datasets/list-of-all-programming-languages. We looked through each tweet, and if a language from our dataset was mentioned, the tweet would be appended to a list value in a language dictionary where each key is a unique programming language. In addition, the language name would be appended to a list of all of the languages mentioned so far in our query results. These lists would then be used to create new word clouds: one displaying which languages were mentioned most often (Figure 2), and others for each individual language, so we could demonstrate the most common words surrounding each language (Figures 3 and 4). Due to the scope of our research, we anticipated that a minority of the languages within that dataset would even be referenced at all. The dataset says it contains the names and information of 705 programming languages and our results only included 23 of them. That being said, most of those 705 languages are not even in moderate use, so our results will not be heavily skewed. We chose such a large dataset because it allowed us to include and recognize instances of less popular languages being referenced.

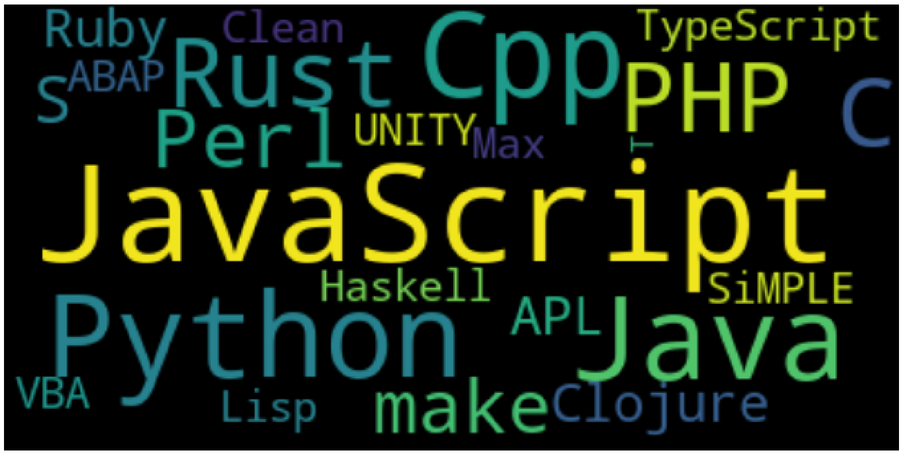


Figure 2: Word cloud of all mentioned languages

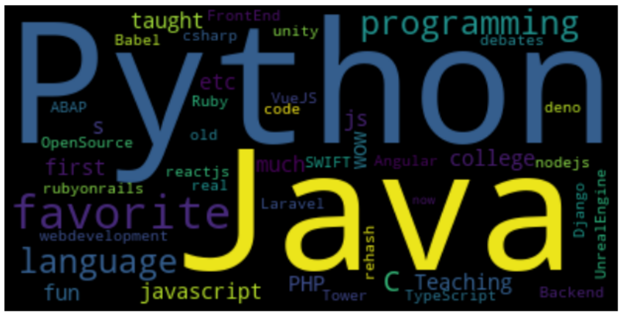


Figure 3: Word cloud of all tweets mentioning Python

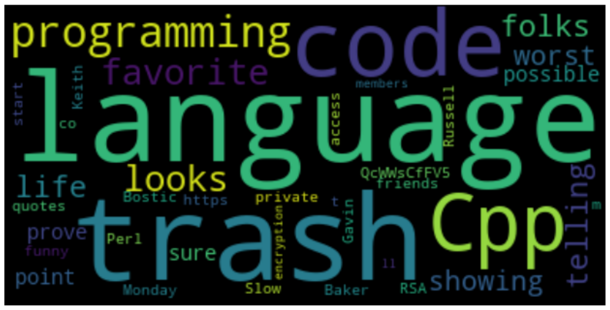


Figure 4: Word cloud of all tweets mentioning C++

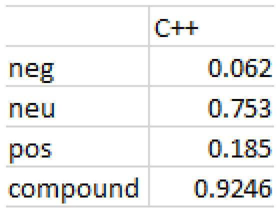
In Figure 2, we can see that Javascript is the most popular language by volume alone, with Python, C++, and Java following close behind. Looking at Figure 3, it is apparent why Python and Java have such similar popularity: they are often both discussed at the same time. In this word cloud, we can also see that the words “favorite”, “teaching”, and “fun” appear frequently when Python is mentioned, which indicates that this language will likely have a positive sentiment attached to it. On the other hand, Figure 4 presents a more mixed opinion about C++. Although its popularity is similar to that of Python, C++’s word cloud contains a more diverse set of words, with “trash” appearing extremely frequently, along with “favorite”, “worst”, and “slow”. These words likely indicate that this language will have a more neutral sentiment, or perhaps even a negative one.

After creating these word clouds, we used the **V**alence **A**ware **D**ictionary and s**E**ntiment **R**easoner (VADER) sentiment analyzer from nltk to mathematically determine the sentiment surrounding the programming languages in our dataset. VADER is perfect for this application, as it is pre-trained so it can deliver results quickly, and it excels at interpreting short sentences that might include slang or abbreviations, which is exactly what we expect to find in tweets. To use VADER, we created a new dictionary of each programming language, along with the text of all tweets that mentioned them. Inputting these to VADER, we were able to generate a “positive”, “neutral”, “negative”, and “compound” polarity scores for each programming language. The first three scores tell us how positive, negative, or neutral the sentiment surrounding each language is, and the compound score provides a normalized sentiment value on a scale from -1 to +1, where the value can be interpreted by the following table:

Sentiment	**Compound Score**
Positive	>= 0.05
Neutral	< 0.05 and > -0.05
Negative	<= -0.05

Figure 5: “Compound” Sentiment Polarity Meaning

Using these values, we can determine the overall sentiment of each language, and compare that to the visual inspection of each word cloud. For example, Figure 6 shows the positive, negative, neutral, and compound sentiment polarity scores of the C++ language.



	C++
neg	0.062
neu	0.753
pos	0.185
compound	0.9246

Figure 6: C++ Sentiment Polarity Scores

As predicted from the mixture of words in its word cloud, this language has an overwhelmingly neutral polarity score. However, its compound polarity score is surprisingly high, comfortably falling into the "positive" range described in Figure 5.

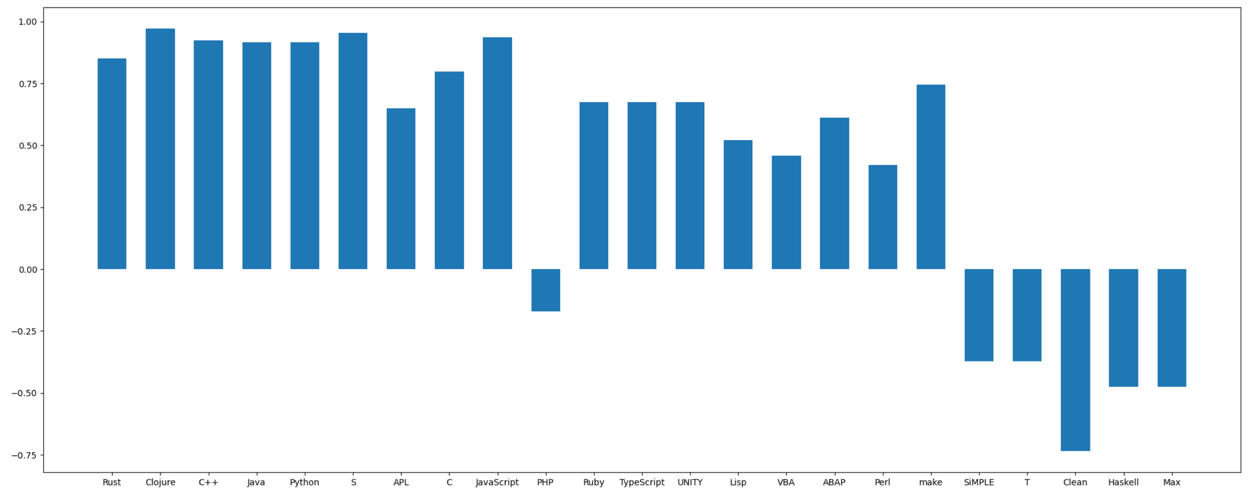


Figure 7: Comparison of Composite Polarity Scores

We then took those composite values and used them in a bar graph as a visual representation of the general sentiment around each language. To see the raw negative, neutral, positive, and compound values found for each language, please reference the polarity_scores.csv file.

While much of our data is interesting, there is a major concern that should be addressed when viewing these results. Most importantly, Twitter's API only allowed us to query recent tweets, so these results only include tweets created within the last week. At the time of this report, that will be the week of November 28, 2021 - December 5, 2021. This most likely accounts for why many of our composite scores were so extremely high, as depending on the week the general opinion of a language on Twitter can change drastically. While we wanted to work with a much larger set of tweets, Twitter's API unfortunately limited us to a much smaller than ideal data set. To gain access to Twitter's full-archive endpoint would have required Academic Research Access, which we would not have qualified for by Twitter's written guidelines.

While this means our results do not demonstrate the totality of Twitter users' opinions on programming languages across the site's history, they do reflect fresh opinions from its users. If we were to expand the scope of this project over time, we would like to incorporate a weekly refresh of this data and automatically save those results for comparison over time. Eventually, this would allow anyone to look back through each week on Twitter and the broad consensus surrounding many programming languages. This could benefit programmers looking to experiment with new languages, language designers looking for recent feedback and opinions, and to find trending languages.