PREDICTIVE POPULARITY: DATA ANALYSIS WITH PYTHON/REDDIT

OVERVIEW

Purpose: To accurately predict on a large scale the success of Reddit posts from the subreddit “AskReddit”, based on post metadata.

Method: A Python library utilizing the Python Reddit API Wrapper (PRAW) was written from scratch to collect posts and related information, and writing them to disk. An Amazon Web Services Ubuntu instance runs this code constantly.

The code is available at github.com/nickmahda/Project-Mango.

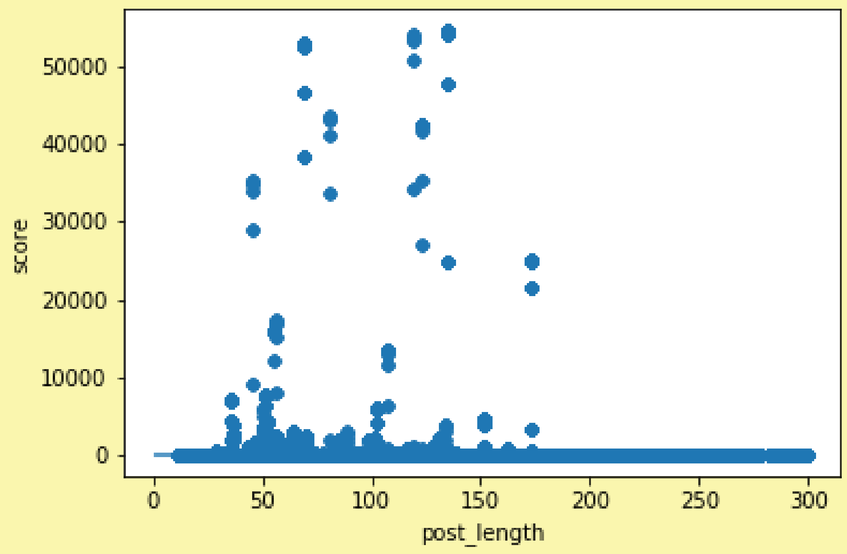
Analysis: The popularity of a post, quantified by the total score of the post (upvotes), was predicted using regression methods.

Total entries recorded	Total unique posts recorded	Total upvotes on all posts (combined)	Total comments on all posts (combined)
540805 entries	24277 posts	521809 upvotes	5458626 comments
Average upvotes/post	**Average comments**	**Average post title length**	**Time of day with most posts**
21.5 upvotes	22.5 comments	75.25 characters	4-6 AM UTC
Highest scoring post	**Time of day posted**	**Post with most comments**	**Time of day posted**
aus97z (55.7k)	11:54 PM UTC, 4:54 PM PDT	aujfms (28.925k)	9:36 AM UTC, 2:36 AM PDT

POST LENGTH ANALYSIS

All posts were analyzed based on the length of their titles, and were compared with their eventual score.

Ignoring the majority of posts that did not reach a substantial score, most posts that reached the front page were between 50 and 150 characters.



THE CODE

The code was run on a free 2-core AWS instance, which was constantly running and collecting data over the course of more than 3 months. The code was programmed in the Python language, version 3.6.5. Over time, the post collected new data at intervals varying from 5 minutes to 1 day, and used this data to find the speed at which posts’ scores grew. The following variables were collected:

- Score
- Post ID
- Date and time of post creation
- Number of comments
- Whether or not the post was interfered with (locked, edited, pinned, etc)
- The author of the post
- Attributes of the subreddit (# of subscribers, active users, etc)
- The current time

The program takes the state of all these attributes at various intervals, ranging from 5 min to 1 hr to 1 day, and places it as a unique entry in a database. Therefore, after 2-3 days, each post has about 20 different entries in the database, with the states of all of its attributes in each recorded instance.

After every post is updated, the program then writes the data to disk. After about a week, usually there are about 10,000 unique posts, each one having been updated 20 times for a total of ~200,000 entries in the database. This amount of data is crucial in order to make generalizations about the data without being limited by sample size.

For more information, see the repository at github.com/nickmahda/Project-Mango, ask me directly, or email me at nickmahda@gmail.com. The code will also be available for viewing at request.

THE CODE PT 2

```
                if (post.id in df['id'].values) or (post.id in df
                    logger.debug(f"{post.id} is a duplicate, cont
                    continue

                logger.debug(f"Picked up {post.id}")

                for _a in p_attr:
                    row_new[_a].append(getattr(post, _a))
                for _s in s_attr:
                    row_new[_s].append(getattr(s, _s))
                row_new['pickup_no'].append(0)
                row_new['post_pickup'].append(time.time())
                row_new['time_now'].append(time.time())

                logger.debug(f"Old {len(row['id'])} / new {len(row_ne

                df_new = pd.DataFrame(row_new, columns=attr)
                df_update = pd.DataFrame(row, columns=attr)

                modified = False if df.equals(df.append(df_new)) and

                df = pd.concat([df, df_new, df_update], ignore_index=
                if not config.DRY_RUN:
                    # df.drop(['pickup_no'], axis=1).to_csv(config.DA
                    df.to_csv(config.DATAFILE, index=False)

                logger.debug(len(df.index))

                del row, row_new
                del df_new, df_update

            except prawcore.exceptions.RequestException:  # You most
                retries += 1
```

It took about a month to finish the first iteration of the fetcher program, but the code stayed in development throughout the entire project. Some of the changes made over time were:

- changes to the way in which the program got data (efficiency)
- changes to the recorded variables
- additional functions to streamline the code

The code used over 10 other libraries, among them the Reddit API, Pandas (a data collection and organization library), NumPy (an advanced math library), and Statsmodels (a library to help with regression analysis).

For more information, see the repository at github.com/nickmahda/Project-Mango.

STATISTICAL ANALYSIS

Once the code was collected, the next necessary step was finding trends and patterns in the data. To aid with this, specialized math and data storage/collection libraries were used such as Numpy, Statsmodels, and Pandas (see right). These were used to create graphs and regression models for the data. Two such examples can be seen below.

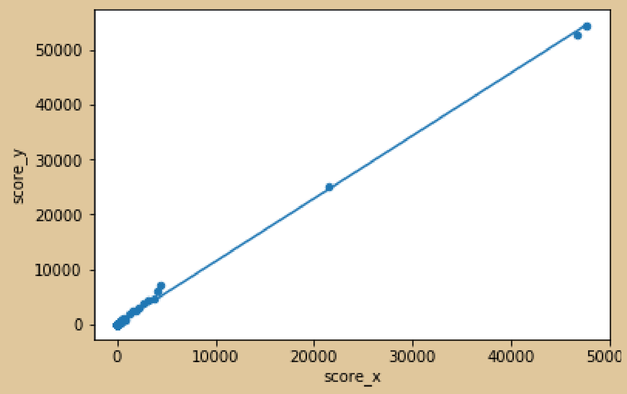


Fig. 1

The change in score of posts from 1 day after posting to 3 days after posting was predicted with an R^2 of 99%.

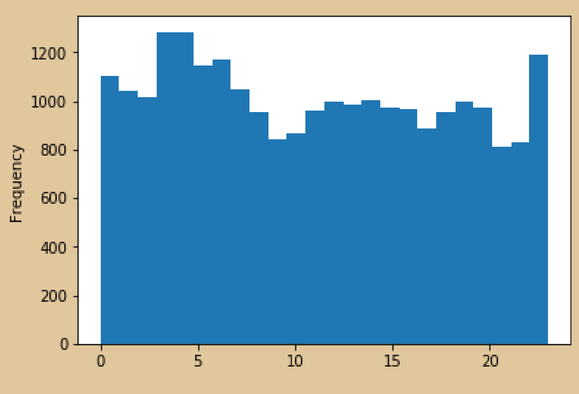


Fig. 2

The highest number of posts were found around 5:00 AM UTC, which is around 10:00 PM PST.

CONCLUSIONS AND FUTURE WORK

After analysis was finished, the following conclusions were drawn:

- Posts do not significantly change score after the first day, instead growing 1.2% per day with very little variance after they have hit the front page.

- Posts that do not receive more than 500 upvotes tend to stop growth completely after the first few hours.

- Most posts that reach high levels of popularity have between 50 and 150 characters in their titles.

Some future extensions of this work include image analysis on image-based posts to find trends in those posts, and doing more complicated analysis on the growth of the posts with more advanced math.

I would like to thank Reddit/the Reddit community for a functional and detailed API, Jacob Sisk for mentorship, and my family for their help, patience and support.