Spam Filter Study Project Report

Mihir Patel, Jacob Huber, and Ben Fintel

Abstract

Many websites utilize content provided by users. This kind of content can be constructive for the topic of the website, however, sometimes non-relevant content can be submitted as well. This non-relevant content can be referred to as “spam”. An example of spam could include content which contains attributes that are not allowed for the given website. This is the kind of spam that will be primarily researched in this paper. The research is based on a specific problem that we are running into. We are creating a website that allows users to post reviews of specific things. This website needs a method of filtering out content that violates our review posting guidelines. To do so, we can use a spam filtering package. We found that there are many different packages available to solve this task. To find the best package for our use case, we studied multiple different packages and recorded the results based on specific metrics. We hope others will be able to utilize our results to find the best Node.js based spam filter for their project.

Goals

We want to determine which of the packages that we have tested fit into what type of work that they say that they do. Which are the fastest, best at filtering, or the easiest to implement. Some packages may work in different types of uses depending on what specific filter it’s going for. However, we want a spam filter that works best for a website where users can leave reviews but what we find works best for us might not be what is needed for different developers. The spam filters will be running against the same tests that we have created and will be measured against speed, correctness, and usability. In the end we want to be able to state with confidence which filter packages are best for different types of everyday filtering different applications may need.

Result Measurement Method

Since the goal of this project is to sift out only the fastest and most efficient spam filters, we need to have a quantitative set of measurements to arrive at a conclusion. The categories we plan to test for each spam filter we use are: best percent spam caught for large text, best percent spam caught for small text, speed of spam caught for large text, speed of spam caught for small text. These results will take account of different test cases so we can cover a broad range of possible use cases. The way we are going to be measuring this is going to be done in a few ways. The first method will be the human observations, which will be what we can visually see and calculate ourselves from each run of the different spam filters. We can see how much each spam filter has missed, or maybe how much it caught that wasn’t necessary to filter out. The second method of measuring will be through computed calculations. We want to display the

data for the exact percentage of filtering that each filter caught for our test cases and how fast each filter ran. Using the combination of our human calculations and what the computer is able to calculate, we will be able to effectively measure which filter performs the best in each given situation. We also would like to note down the difficulty of implementing each spam filter so developers are aware of the effort required. Other readers will also be able to utilize our findings to select the best spam filter for their project.