Spam Scanner

Robbie Biesser
8/21/2020
CIS480

Introduction

The idea for Spam Scanner spawned from an interesting problem statement, "Spam costs U.S. organizations billions of dollars a year in spam prevention systems in addition to the consumption of system resources and decreased productivity." For the purpose of this project, spam is a term given to any unsolicited or unwanted message transferred electronically as email. It is widely known that spam has existed since the introduction of email as a form of communication and affects everyone with an email account. Spam Scanner is a client application where a user imports a mailbox of email messages and is presented with a score for each message representing the likelihood that the message is spam and is established in support of furthering research into identifying spam messages based on the content of a message. This report presents a single approach to this discovery and a summary of the findings from real messages.

User Experience

In order to provide enjoyable user experience, it was decided that the application should fit into the user's natural workflow and automate as much of the process as possible. To accomplish the task of calculating a score, the system requires, at minimum, a block of text to be the message and a listing of words that would be used to identify spam. If the user is forced to interact with a message that is likely spam, the identification method is considered a failure if it does indeed match the definition of spam after inspected by the user. There are many systems an email message passes through on its way from sender to receiver and it is understood that a message submitted for testing by Spam Scanner has reached the intendend recipient. A spam score is calculated for every message and can be investigated for any message the user selects. One objective of the user interface is to give visual clues about a message before the user interacts with the message and is accomplished through colored markers and an alert bar. In this way Spam Scanner functions much like a traditional email reader application with the added attention to identifying spam.

Development

Spam Scanner was produced over the course of a five week period with a series of weekly initiatives to develop what the application should become. It was soon realized that the task of developing a pattern matcher would be one small part towards exploring the available frameworks and libraries for creating a client application. There were a number of methods to approach

developing the user interface, but a traditional HTML web interface provides the simplest method for creating a consistent cross-platform codebase. React was chosen to create the user interface for its ease of development and available components. This was made possible because of an NPM package name Electron that allows for an application to be developed as a website and packaged as a standalone application. A decision needed to be made for the storing and retrieval of data and ongoing development has made client side databases for use in a web browser a possibility. RxDB is one such library that provides a wrapper for this functionality. These three formed the framework of the application and all that was left was working directly with the email messages. The mbox file format is a popular method for transferring received messages between email applications and made for an easy fit to provide the functionality Spam Scanner needed for importing large volumes of messages. Every popular email client allows for messages to be exported in the mbox format and this should be a comfortable user interaction. There are several other options to consider for importing messages and those should be considered for future development. The conversion from a plaintext email message in the from the mbox file to a JSON array of message objects is made possible through a module in the NodeMailer library known as Mailparser. To perform the function of pattern matching a simple match by regular expression could have been done, but research into the topic of natural language processing revealed several libraries are available that accomplish that same task. A library consequentially named Compromise takes a clever approach to inferring parts of speech and proved to be ideal for the purposes of the Spam Scanner application.

Testing

It was decided that providing at least thirty keywords or phrases would be sufficient for identifying spam on a testable level and would fit the scope of this project. Spam Scanner, however, has been developed to allow the user to manage the word list and accepts an unlimited number of values and range of weights. The weight is used to establish that certain words or phrases have a higher probability of identifying a spam message. A default list of suggested spam words and phrases is loaded when the application starts and the user is free to make changes as necessary. The initial concern was that traditional spam scanners are able to make use of a larger pool of emails and are able to posit the level of spam based extended criteria than just the content of the message itself. For instance, a human can determine that a particular email address has a history of sending spam emails and that email can be flagged regardless of the content of the messages. Another would be the ability to scan messages from multiple recipients to predict a correlation. Nonetheless, testing with Spam Scanner continued. The strategy of applying a weighted score to found messages proved to be successful in identifying a large number of messages as spam. The application was tested with messages from a personal spam folder and also a sample from the CLAIR collection of fraud email. It was expected that all of the messages should have been identified as spam since they were identified as spam from other systems. As messages were explored, more phrases were discovered and added to the list of spam words used by Spam Scanner.

An interesting result is that at no time during the testing of messages were the results able to identify every message as spam without also identifying legitimate messages. This is possibly due to the fact that the individuals authoring the messages changed their tactics to avoid the same words that were likely to be used by the application to identify spam. On one occasion it was observed that

an obvious phishing message was not treated as spam using the method described although a human would have been less likely to have been duped. There is a fine line between incorrectly matching legitimate messages and not matching known spam messages and Spam Scanner allows that line to be explored. A ten point scale was found reveal the most relevance where a threshold score could be set to allow legitimate messages a chance to contain supposed spam words but not mark the message depending on the context. The threshold is set so that if a message contains at least five instances of any of the spam words, it is marked as spam. If there is a word or phrase that should always be considered spam, it can be given a higher weight value. The point system described provides a sufficent method for identifying spam, but additional research could perfect the scoring system and the weights are adjusted from the user interface.

Future Development

Further testing was done to benchmark the performance of Spam Scanner at a larger scale and a mailbox with over 3,000 messages was imported with the operation taking a couple of minutes to complete. Considering the work that is being requested, that is a reasonable time to wait but also does not make for a quick user experience. Traditional spam utilities scan messages at a single point in time, when the message is sent or received, and is therefore a much quicker response time than processing an entire mailbox at once. It is not unrealistic to think that a mailbox may contain several thousand to tens of thousand messages. For a smaller amount of messages, within a couple of hundred, this solution remains snappy and responsive and the chosen architecture is sufficient. In continuing the fight against spam, research should be directed towards natural language processing and machine learning algorithms. Additionally, the interface should be expanded to function as a single solution mail reader. Spam Scanner explores some efficiencies of user interface design and the goal should always be to create an enjoyable user experience. There are several popular mail readers on the market, but Spam Scanner may find a niche in the custom language processing market.

Conclusion

In conclusion, Spam Scanner has proven that a client application dedicated to identifying spam could be beneficial and desired, but further research could lead to a more economic solution and that natural language processsing and machine learning are a worthwhile venture. A goal in the fight against spam should be to eliminate spam, not just identify it.

References

Radev, D. (2008), CLAIR collection of fraud email, ACL Data and Code Repository, ADCR2008T001, http://aclweb.org/aclwiki