Social Media with Automated Moderation

Client Company: *SnogJammers*

Digital Technology Supplier: *Internal SnogJammers Team*

Team 1 Members

Todd Lim

Fan Kuk Lam

Dan Ngo

Charles Mateer

Kakhaber Urigashvili

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

1.1 Introducing the Client Company: SnogJammers

SnogJammers is a social media company based in Los Angeles, California, USA. It provides an application that enables users to match and connect for dating purposes. It is a medium-sized company with 500 full-time employees: 150 of those employees are engineers. Over the last three years, SnogJammers' application has seen exponential growth in account creations; however, according to an exit survey of users who deleted their accounts in the previous year, 45% of those users cited spam content as the reason for deletion. SnogJammers defines spam as one of the following contents: (1) offensive/inappropriate; (2) low-quality; or (3) fraudulent. With the increasing sophistication of techniques and tools in the arsenal of malicious actors, simply relying on tech support personnel to review reported cases has become less sustainable to secure the platform from spam.

1.2 Introducing the Digital Technology Supplier: Internal SnogJammers Team

SnogJammers intends to create a cross-functional team of 10 employees, which will include data scientists/engineers, web developers, and development operations experts that will work specifically to develop a solution to mitigate spam content. This team will be working on various enhancements to the platform, such as user interface updates, provisioning of infrastructure, the creation of an abuse web service API, data collection/analysis, and training of a spam detection machine learning (ML) model.

As a part of the project, the team intends to integrate with third-party APIs for generic spam detection and Internet Protocol (IP) reputation analysis. A third-party spam filtering service can provide a baseline level of spam detection effectiveness. The external system would be integrated with the platform until the internal spam modeling methodology achieves the same or higher level of accuracy, but the IP reputation service would remain with the vendor.

2 BUSINESS REQUIREMENTS

2.1 Business Goals

- **Increase user engagement by creating a better user experience:** The objective would be measured by the increased user retention rate and longer app session intervals. Additionally, 45% of users reported that spam is the main reason for uninstalling the application. The goal is to reduce this number to less than 10% in a year.
- **Reduce the number of tech support personnel for reviewing spam related reports:** Support personnel spends 20% of their time on the tickets related to spam content. The goal is to reduce this metric to at worst: 5%. This would be accomplished by the automated system's ability to filter a large amount of spam content. Additionally, the manual review process would be streamlined, since all the content would be marked with a spam score.
- **Improve brand reputation:** The brand needs to increase its social proof by increasing its application store rating from 3.1 to 4 (out of 5 stars) in the Android Play store and from 2.9 to 4 (out of 5 stars) in the Apple Play store over a period of one year.
- **Reduce storage space and system strain from abusive content:** Based on recent analysis, at its highest, about 15% of the platform's backend storage was occupied by spam content. After content is confirmed to be spam, it is removed from the platform. A goal for this solution would be to reduce the average percentage of spam in our back-end systems to 5% by increasing the rate at which spam is flagged and removed.

2.2 As-Is and To-Be

The following sub-chapter shall include two sections: 2.2.1 and 2.2.2. The former, 2.2.1, will detail what SnogJammers' spam mitigation process currently looks like "As-Is." Meanwhile, the latter, 2.2.2, will discuss what the "To-Be" process would look like after the digital technology will have been implemented.

2.2.1 As-Is

Currently, spam content is managed in a largely manual fashion (with some semi-automated processes), whereby dedicated staff utilize a tech support system to view and manage a queue of reported abusive content. Users can click the “Report Spam” button in the application. The information is sent to the technical support ticketing system for review by currently available support staff. They are trained on and utilize a standard operating procedure (SOP) document to determine whether the flagged content is considered spam by the platform. Support staff can then mark the content to be removed from public viewing and, consequently, restrict certain account privileges shortly thereafter. The definition of spam is as follows:

- Offensive/Inappropriate content
    - Racism
    - Sexism
    - Nudity/pornography
    - Violence/bloody
- Low-quality content
    - Advertisement
    - Random messages that do not contribute to the conversation
- Fraudulent
    - Impersonation
    - Solicitation of financial info

Front-end validation is in place to automatically replace a known blacklist of words with random characters in order to reduce vulgar language displayed on the platform. However, this only targets text content and misses edge cases, in which words are misspelled. Moderators have to manually monitor posts with non-text content (images and videos) and flag them for review by the support staff mentioned above (to be compared against the SOP).

Both processes of review are time-consuming and error-prone. Increasingly, more novel ways of subverting the platform's review process are being implemented faster than the moderators and support staff can adapt to the new threats.

2.2.2 To-Be

SnogJammers wants to add further IT infrastructure to support an ML-based spam filtration system accessible to the front-end application via an additional web service, termed the Abuse Service API. The Abuse Service API will act as a single source of truth for determining whether the content is spam or not. Incoming messages from the application will be run through the API to determine its spam score in real-time before being posted to the platform. The spam score thresholds will determine if the content is: [1] immediately blocked (high confidence of spam); [2] sent to tech support for review (medium confidence of spam); or [3] posted to the platform (low confidence of spam). The primary target source of spam for this automated moderation will be for registered users participating in harassment, as well as botnets conducting unwarranted advertisement or fraud.

The Abuse Service API will be powered by two sources: an ML model developed by the internal data science team and by a third-party spam REST API. Creation of a highly accurate internal spam classifier will require a large amount of labeled data to train the model. Because of this, the Abuse Service API will initially utilize the above-mentioned third-party spam service and IP reputation analysis service on the back-end.

Once the internal model acquires enough training data to produce high-quality results, the Abuse Service API will eventually use the internal model as the mechanism for spam detection. The interface for the Abuse Service API would remain unchanged so that migration from third-party services to the internal model would appear seamless for other applications.

2.3 Requirements

2.3.1 Functional Requirements

[A] End Users: Improve the user experience for end users

[A.1] Button for reporting spam

[A.1.1] As an end user, I need to easily report spam with a button when I suspect that content falls into one of the three bucket definitions of spam.

[A.2] Channel for appealing account restrictions and blocked content

[A.2.1] As an end user, I need to file an appeal case via the notification email to appeal my spam case so that I can try to unlock my account messaging feature.

[B] Internal Engineering Team: Integrate third-party services for internal engineering

[B.1] Application programming interface (API) for internal applications

[B.1.1] As an engineer in the messaging team, I need to integrate with the abuse service API which I can send in message content and member ID. The abuse service needs to reply indicating if the content is spam or not so the messaging service can either allow or block the message request.

[B.1.2] As an engineer in the abuse team, I need to integrate with the third-party spam filter restful API so that we can forward the content for spam scoring.

[B.1.3] As an engineer in the abuse team, I need to integrate with our existing restriction service via restful API so that I can send a request to restrict a member from using the messaging feature.

[B.2] Metadata for restriction on end user accounts

[B.2.1] As an engineer in the abuse team, I need to be able to specify the origin of the restriction when I send the restriction request to the restriction service so that the support queue system will have the appropriate flag.

[B.2.2] As an engineer in the abuse team, I need to have a counter to count how many times a user sends spam messages so that I can issue an restriction to the user to disable the messaging feature.

[B.3] Database for training ML model

[B.3.1] As a data scientist in the abuse team, I need to create a database to collect content, IP reputation, member id, IP, geolocation from each spam request to train our in-house model.

[C] Technical Support Team: Improve the workflow for internal technical support

[C.1] Flagging feature on support queue items

[C.1.1] As a support team member, I need the support queue item to have a flag to show whether the system has restricted or issued a warning to a member automatically so that I can do a better investigation on the support case.

[C.2] Resource for referring ends users to rules and community guidelines

[C.2.1] As a support team member, I need to have a policy I can reference so that I can refer the user to the policy page without exposing our restriction reason.

2.3.2 Non-Functional Requirements

Scalability - The Abuse Service API needs to be able to handle up to 1,000 requests per second for the first year. The system needs to have the ability to scale above 1,000 requests per second.

Loose Coupling - The Abuse Service API will act as an abstraction layer for the underlying machine learning (ML) functionality. The system should have the ability to change the third-party spam service or ML methodology without changing its interface.

Law/Policy Compliance - All communications with third-party services need to be encrypted. Any PII such as phone numbers, emails and addressed have to be masked.

2.4 Business Benefit Justification

2.4.1 Return on Investment (ROI)

Expected costs:

- Paying for 3rd-party service. Fees for using 3rd-party spam detection service.

- Payroll (employees). An opportunity cost of using technical employees to work on spam automation project versus other company priorities.
- Provisioning new infrastructure (web servers and databases).

Expected benefits:

- Increased storage after reducing the percentage of spam in our back-end systems
- More ubiquitous, productive, and positive content, which is free of spam, will allow users to use the mobile application more efficiently
- More robust machine learning pipeline, which allows the company to sustain an effective spam filter without the need for paying for third-party services

Returns on investment:

- Increased number of satisfied users allows the company to sell more advertisement space for revenue

2.4.2 Measuring Success

SnogJammers will use its existing analytics, system monitoring tools, and user surveys to compare baseline metrics with post-implementation metrics to gauge its success. We aim to achieve the following improvements over the next year:

- Decrease the percentage of spam that makes it into the database from 70% to 30%
- Decrease the percentage of content that was misclassified by as spam from 55% to 25% (the misclassification is caused by limitations of the blacklisting methodology)
- Decrease the percentage of users deleting their accounts due to spam from 45% to 20%
- Increase the percentage of catching bad actors/spammers from 20% to 40%
- Improve social proof of the app by garnering higher rated reviews in the iOS App Store and Android Play Store
- Receive fewer negative comments related to spam on the platform's social commerce
- Increase total active accounts by 10%

3 TECHNICAL SPECIFICATION AND PROTOTYPE

3.1 Architectural Approach

To fulfill the stated requirements, SnogJammers plans to enhance its user-facing application. This entails building an in-house central component, which will interface with other services and databases. This will be called the 'Abuse Service,' and it will directly interface with the following services: 'Messaging Service (in-house),' 'CleanTalk API (third-party),' 'Vision API (third-party),' 'IP geolocation API (third-party)', and 'Internal Machine Learning Model (in-house),' as well as the Abuse Service Database (in-house). These services are further discussed later in Sections 3.1 and 3.2. They can also be seen at a high level in Figure 1 below.

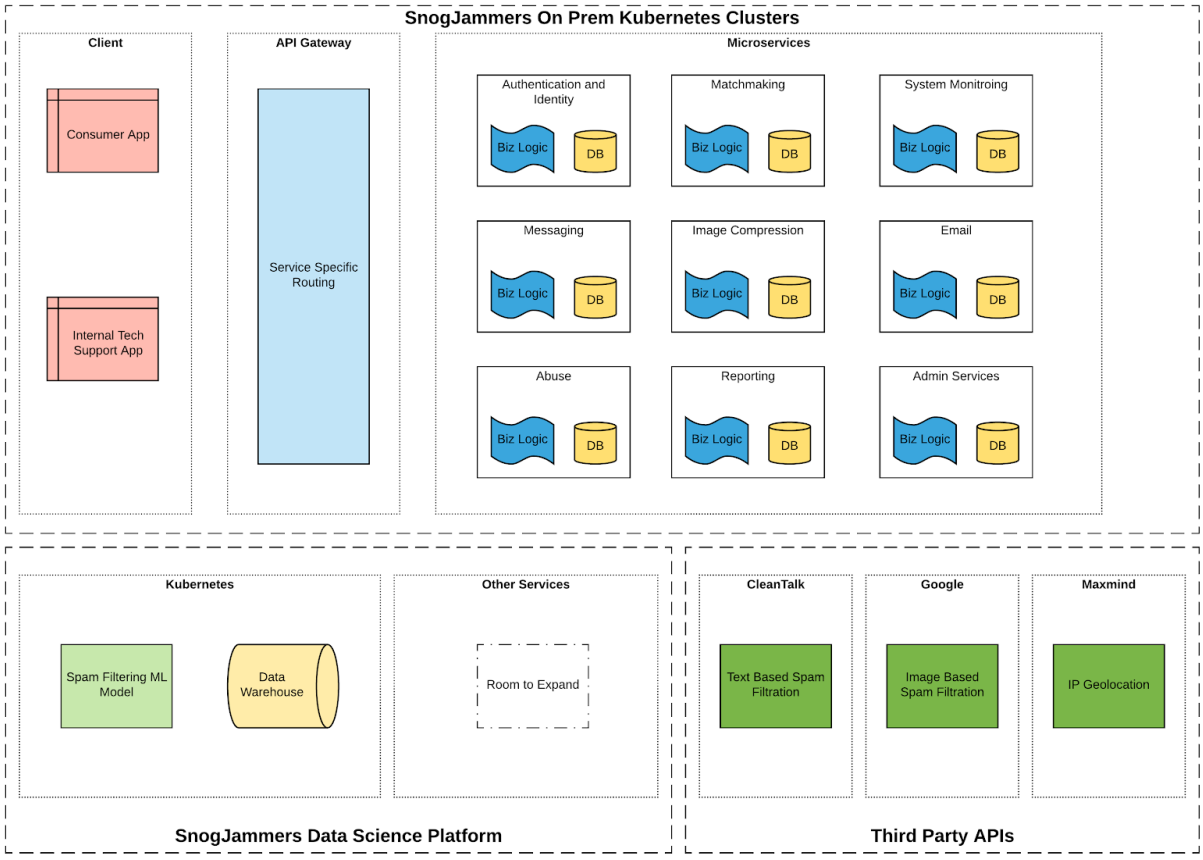


Figure 1. Business Context Architecture

To improve the user experience for end users, two functional requirements were identified: (1) button for reporting spam; and (2) channel for appealing account restrictions and blocked

content. On the mobile app, end users will have an easily visible ‘report spam’ button built into the user interface (UI) and an email icon that opens up the device’s default email application in order to communicate with technical support, respectively. Since the service will scan and feed users’ messages to an internal database, we will protect our users’ privacy by not associating the stored messages with any users and quickly preprocessing them before storing them into the database with techniques, such as stemming or removing stop words/names. This will be in one of our items in our Terms of Service (ToS). User-generated content will be retained for 30 days, and only data scientist(s) who works on the model and produce reports have access to it; no one outside of the team will have access, unless it is approved by our security team for a valid reason.

To integrate third-party services for internal engineering, three functional requirements were identified: (1) API for internal applications; (2) metadata for restriction on end user accounts; and (3) database for training a ML model. In this case, the Abuse Service API will listen, intercept, scan, and filter messages by interfacing directly with the Messaging Service, which is a microservice application responsible for accepting users’ chat messages and then saving it to a database; this is used to determine if a message is spam in real time. The Abuse Service API also interfaces with the CleanTalk API for real-time email and IP address spam checks, as well as the Google Vision API for image analysis, as it relates to spam. Data will get stored in an internal data warehouse that pulls from the respective microservice databases (for messaging and abuse) for use in machine learning workflows. These machine learning workflows will ultimately be used as an internal solution.

To improve the workflow for internal technical support, two functional requirements were identified: (1) flagging feature on support queue items; and (2) resource for referring end users to rules and community guidelines. These enhancements will be made to our existing in-house IT ticketing system by incorporating a feature that allows the Abuse Service API to pass flagged message data, which would indicate a system restriction or warning to a user before technical support even receives the item. The resource used by technical support for end users, in this

instance, would simply need to be available either in offline or online format, so the Abuse Service API would not be involved for this requirement.

Additionally, three non-functional requirements were identified: (1) scalability; (2) loose coupling; and (3) law/policy compliance. We will target scalability and loose coupling by leveraging our current Kubernetes containerization architecture and microservices design pattern. In terms of law/policy compliance, we will adhere to the General Data Protection Regulation (GDPR) and general privacy standards by encrypting communications with third-party services, such as CleanTalk and Google Vision. Unless there is a legal issue tied with a user (e.g. crime investigation-related), they have the right to delete their data. If the users delete their account, the data will be automatically deleted within 15 days based on the member id.

The new solution is an enhancement to the current systems landscape, which consists of other microservices. Adding a new microservice that exposes an API endpoint for integration would allow the internal engineering team to use the messaging service as a pilot.

3.2 Software Solution

The central component of the proposed solution will be a REST API, called the 'Abuse Service.' The Abuse Service will be written in the (Java) Spring Framework, which has traditionally been used by the SnogJammers engineering team to build APIs and the team is very proficient with the framework. The Abuse Service will expose certain REST endpoints to the Messaging Service to indicate whether content is spam or not.

On the back-end, the Abuse Service will use third-party spam services and an internally trained model to determine whether text content is spam or not. Initially, 100% of traffic will use the third-party spam service as a mechanism for classification. As we acquire labeled data and train our model internally, increasing amounts of traffic will be classified by the internal model. The percentage of traffic classified by the internal model should be configurable, as it is expected to increase over time. The Abuse Service will provide a stable interface, so consuming applications

do not need to be concerned with what methodology is used behind the scenes. For the third-party spam service, the Abuse Service will use CleanTalk (cleantalk.org). CleanTalk provides a REST API for both text content spam analysis and IP reputation in one API - this is the main reason for selecting this vendor.

For image recognition, the Abuse Service will use Google Vision API to detect if images are inappropriate. The Google Vision API has a SAFE_SEARCH_DETECTION type that detects inappropriate and abusive images. The Google Vision API uses the same models as Google Search uses for its own SafeSearch feature. The main competitor of the Google Vision API is Amazon Rekognition. The reason for choosing Google Vision over Amazon Rekognition is that Google Vision has a specific feature for Nudity/Violence Detection.

Architecture Viewpoint

Figure 2 below shows the relationships amongst the different parts of the software architecture.

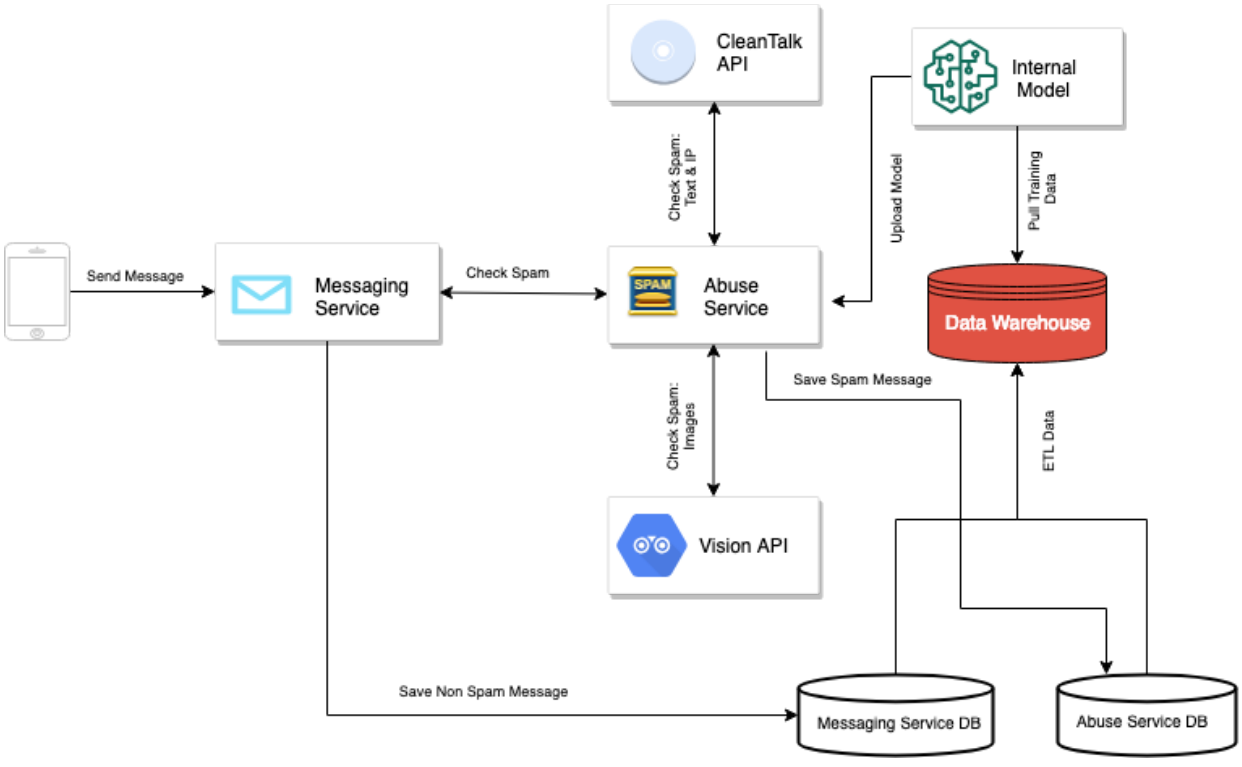


Figure 2. Software Solution Architecture

A message comes to the Messaging Service from the SnogJammers mobile app. The Messaging Service calls the Abuse Service and passes the user message some content. The Abuse Service invokes the third-party CleanTalk API or the internal statistical model to determine if the content is spam. The Abuse Service returns a response to the Messaging Service to indicate whether the content is spam. If the message has image content, the Abuse Service will also communicate to the Google Vision API to determine abusive/inappropriate content. The Abuse Service will then calculate an overall score and return a response to the Messaging Service. Based on the response from the Abuse Service, the Messaging Service will save a message to the "messages" or "spam_violations" table. In the background, the internal model will become trained based on data from the "messages" or "spam_violations" tables. The updated model is periodically uploaded to the Abuse Service by an automated process managed by the data science team. As mentioned previously, eventually all requests for text classification would be handled by the internal model instead of the CleanTalk API. The architecture diagram would be almost identical, since the Abuse Service will still use CleanTalk for IP reputation.

Deployment model

The Kubernetes cluster is currently used for hosting the majority of the companies' applications. SnogJammer went with the decision to use Docker and Kubernetes instead of using something like AWS for the following reasons:

- Avoidance of vendor lock in
- Cost savings via ability to run many applications on a host without conflicts
- Ability to autoscale faster using Docker containers. Auto-scaling can be done in seconds using Docker and Kubernetes versus minutes using EC2 instances
- Ability to run on-premise, cloud, or hybrid environment

The Abuse Service API will be hosted on our internal Kubernetes cluster. Docker images and Kubernetes deployment configurations need to be created for the service. Hosting on a Kubernetes cluster will allow new services to take advantage of zero-downtime deployments, auto-scaling, and high availability. The data science team will use the existing Kubernetes

infrastructure to run TensorFlow models for discovering insights, as well as host Jupyter Hub for scaling computational usage.

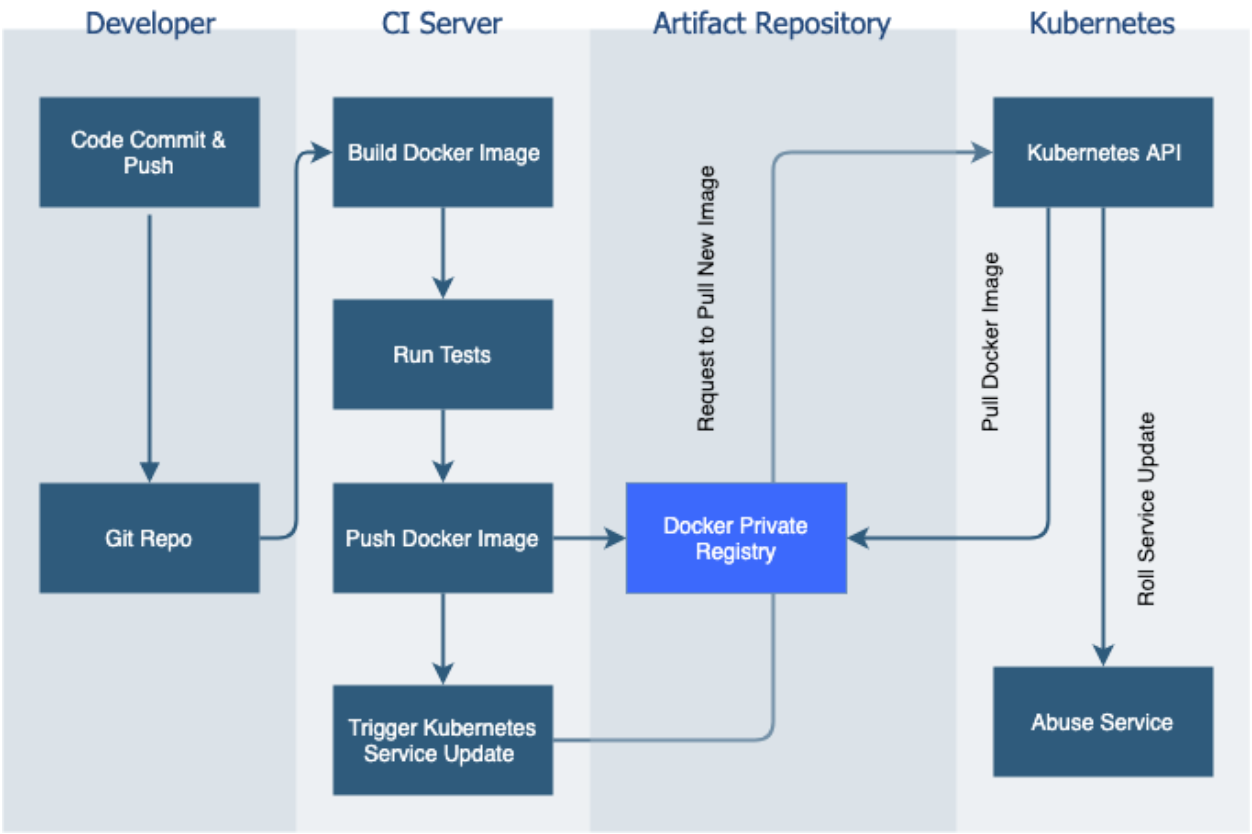


Figure 3. Continuous Integration Continuous Delivery

System Metrics

- The Abuse Service should be able to handle up to 200 requests per second.
- The Abuse Service response time needs to be within 200 milliseconds to 99% of requests.
- The Abuse Service availability needs to be 99.99 %, where the daily average downtime would not exceed 8.6 seconds.

Since the Abuse Service API will be hosted on Kubernetes, most of the SLA requirements would be taken care of in core Kubernetes features. The API should be configured with auto-scaling features. The minimum number of instances should be 5, and the number of instances should scale up to 20, if average CPU for active instances is over 90% for the last 2 minutes. The cooldown period should be 3 minutes of average CPU for anything less than 90%.

3.3 Integration with Other Applications and Data Sources

In order to seamlessly integrate the proposed functionality into our system, we intend to leverage design patterns from our existing microservices architecture. The primary changes to the system will include: updating our front-end application; adding a new REST endpoint to our API Gateway; pushing updates to our messaging microservice; adding a new abuse filtering microservice; integrating with third-party APIs; and expanding on our data science platform to use in creating spam identification models. Lastly, we intend to update our policy documentation for handling spam content under the new system.

Front-End Application Updates

Our mobile application will need to be updated to add user interface (UI) elements to allow users to refute a spam/abusive content claim against them (in the event that our Abuse Service catches a "grey case," in which the model has medium confidence that a piece of content is spam. This UI enhancement will be minimal and will utilize our existing continuous integration/continuous deployment (CI/CD) pipelines. Unit tests and quality gates will need to be set up for the new feature in our test suite and added to our continuous Jenkins build pipeline. Since our development pipeline and front-end development team is well-established, we do not foresee any major impediment to implementing this enhancement.

We will also need to add a new dashboard UI for our internal tech support web application. The proposed solution will be to use existing UI templates and frameworks to create a new tab on our tech support team's web portal application that will ingest new information coming from the updated Messaging Service and new Abuse Service to depict statistics on blocked spam content. Additionally, it will provide a queue for users to refute spam claims and for tech support to manually review spam/abuse "grey cases," in which the Abuse Service was unable to identify with high confidence. Similarly, to our mobile application, we will leverage existing Development Operations (DevOps) pipelines and internal build servers to accomplish this task.

See mockups in Section 3.5 for examples of the intended UI enhancements outlined here.

Additional REST Endpoint on API Gateway

We intend to add more REST endpoints to our system's API gateway to accommodate HTTP GET/POST/PUT/DELETE requests to our new Abuse Service microservice. The endpoint will be implemented using SnogJammer's templated API design patterns and standard operating procedures (SOPs), including the appropriate level of error handling, test coverage, and documentation. Expected request and response types are outlined in Section 3.2.

See sections 3.1 and 3.5 for architecture diagrams.

Updates to Messaging Microservice

The existing Messaging Service is the core backbone service for the SnogJammers application and handles the bulk of the messaging communication (e.g. public posts, direct messages, tying messages to user profiles) between SnogJammer's users. The Messaging Service will need to be updated to leverage the new Abuse Service API as middleware. Logic will need to be added to be able to craft and ingest requests/responses to the Abuse Service API, as outlined in Section 3.2, as well as to be able to either flag a message as spam (for ingest into the tech support queue for review) or remove a message entirely from the system before it makes it into the Messaging DB.

Existing DevOps pipelines will be leveraged for this update to our core messaging functionality. Extensive unit and integration testing will be required to ensure that the Abuse Service middleware is properly functioning and not slowing down the Messaging Service beyond acceptable limits. Development will occur on our existing development environment and iteratively released (following agile sprints) onto a testing environment for quality assurance. Final changes (once the entire system integration has been performed as outlined in this section) will be pushed to a user acceptance testing (UAT) environment for final integration, regression, and stress testing before finally being pushed to production.

New Abuse Service Microservice

Section 3.2 outlines the core functions of the Abuse Service. This service will follow SnogJammer's development reference guides and SOPs in order to integrate with current DevOps pipelines and development/test environments.

Integration with Third Party APIs

Third-party libraries for spam recognition (for text, images, and IP/geolocation analysis) will be leveraged by the Abuse Service, as described in Section 3.2. Endpoints in the Abuse Service will utilize RESTful principles to accept responses and send requests to these third-party APIs in order to provide an interoperable interface so that ultimately those vendor APIs (except for the IP Reputation Service) can be replaced by SnogJammer's internal spam/abuse detection model.

Data Science Platform

Based on our need to have a flexible and cost-effective environment for our internal spam/abuse ML modeling platform, we have opted to leverage our on-premise Kubernetes data science platform using the Tensor Flow framework. Our data science team will be updating their existing Extract-Transform-Load (ETL) pipeline to include the additional data elements outlined in Section 3.4 into their data warehouse.

Policy Changes

As part of these updates, our policy team is working on updates to SnogJammer's Terms of Service (ToS) to define spam and abusive content to end users. Additionally, internal documentation is being updated for use by tech support to help them determine "grey cases" of spam and abusive content that make it past our new Abuse Service. This documentation will be made available on the new tech support dashboard page.

3.4 Data Design and Management

Enhancements will need to be made to the data science team's data warehouse to include data collected from the Abuse Service.

The following data will be collected from users and stored in this data warehouse: IP address; member id; user behavior (e.g. time stamps, posts per hour, words per post, average word length, words per sentence, consonant to vowel ratio); text content (e.g. words text, average word length, words per sentence, consonant to vowel ratio); geolocation; and image content.

Presented on mobile app will be:

- user content that is not is restricted by system flag in the support system
- screen for restricting the usage of messaging feature

Flow in transactions:

The HTTP request object will contain: IP address, cookie, HTTP header information, member id, and the targeted content.

The data will be populated mainly from the data warehouse. Our data scientists will be validating the data offline to make sure the format is correct and the text content is obfuscated correctly (being sure to anonymize any personal information). We can set up a dashboard to monitor our system performance for analytics purpose (false positives and true positives). All data will be complied to General Data Protection Regulation (GDPR) standards. We'll monitor user behavior for suspicious activity (posts per hour, words per post, average word length, words per sentence, consonant to vowel ratio), all of which will be searchable within the database. We'll also monitor how each of our ML algorithms (that our models use) perform on the same data by examining how well each of the following perform when it comes to detecting spam: neural networks, naive Bayes, and logistic regression. We'll chart each models' sensitivity, specificity, positive predictive value, and negative predictive value over time.

Data ETL

Our data science team will set up an ETL process to pull various data points from the Abuse Service and the Messaging Service databases. This data will be loaded into an existing data warehouse for further analysis. Loaded from the Abuse Service API database to the Data Warehouse will be: Member ID, Request IP, IP Reputation, Message Content, and Confidence score. Loaded from the Messaging Service API database to the Data Warehouse will be: Member ID, Request IP, and Message Content. That data will be in a tabular structure for searchability when it comes to SQL queries (access content from a particular user, etc)

Internal Model

The data science team will train its own model to ultimately use in the Abuse Service and test out the performance of the model after we integrate with the third-party spam filter vendor. The algorithm we choose for the internal model will be chosen based on testing the performance of each of the following: random forests, naive Bayes, logistic regression, and neural networks on our internal data during model training. Performance will be evaluated, and the algorithm that performs best will be chosen for the internal model. Initially, we will use the vendor's spam filter result for all the Abuse Service traffic. After we have collected data and trained the model for two months, we will start to integrate the internal model with production traffic. Initially, we will have simple rules, such as:

- If the internal model has 90%+ confidence a piece of content is spam, we block the message.
- If the internal model has 50% to 89% confidence that a piece of content is spam, we send it to the support queue for review.
- Otherwise, the content will be considered a clean message.

The initial rules will be static and implemented in the Java application. After we integrate the internal model to 100% production traffic, we will start to collect more data, such as member restriction history and message sending rate. From there, we will need to integrate with a business rules engine so that our data scientists and engineers can update the rules without waiting for application deployment. The effort for integrating and implementing the rules engine

and collect more data will be in a release 2 effort, whereby we intend to gradually begin deprecating the third-party spam/abuse filtering in favor of our own.

Data Flow

The following diagram in Figure 4 below shows how data will move between systems under this new project. Here, we can see that a large portion of the existing SnogJammer's microservice infrastructure will be unaffected by this change.

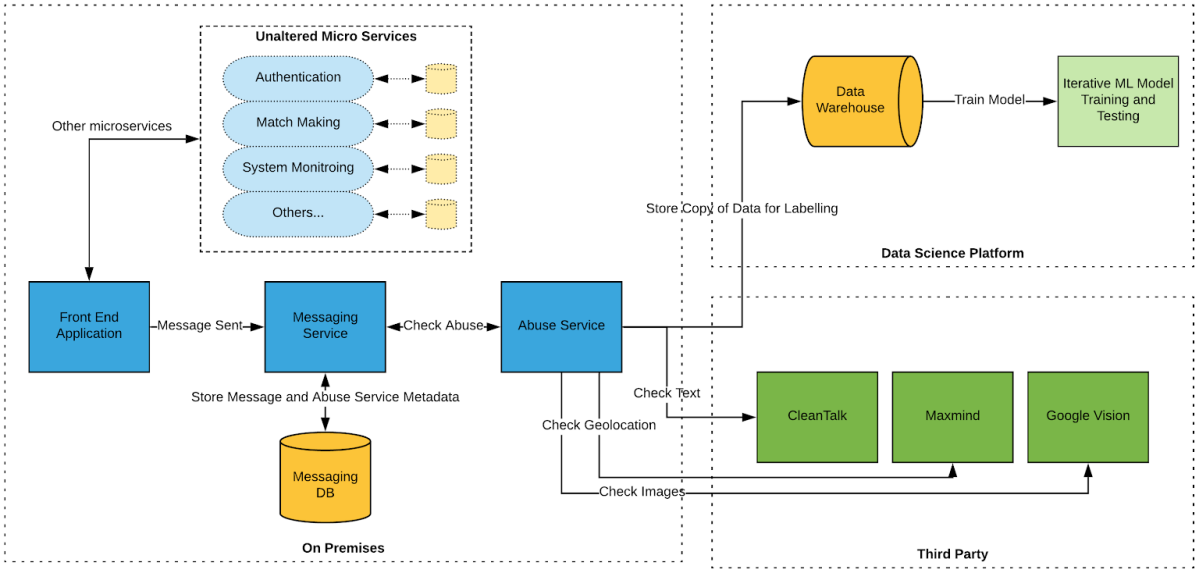


Figure 4. Data Flow Between Systems

3.5 Solution Demonstration

Process Flow Diagram

The following diagram in Figure 5 below shows the intended series of events for the new spam/abuse filtering system, starting with a registered end user sending a message to another user, who is using the front-end mobile application.

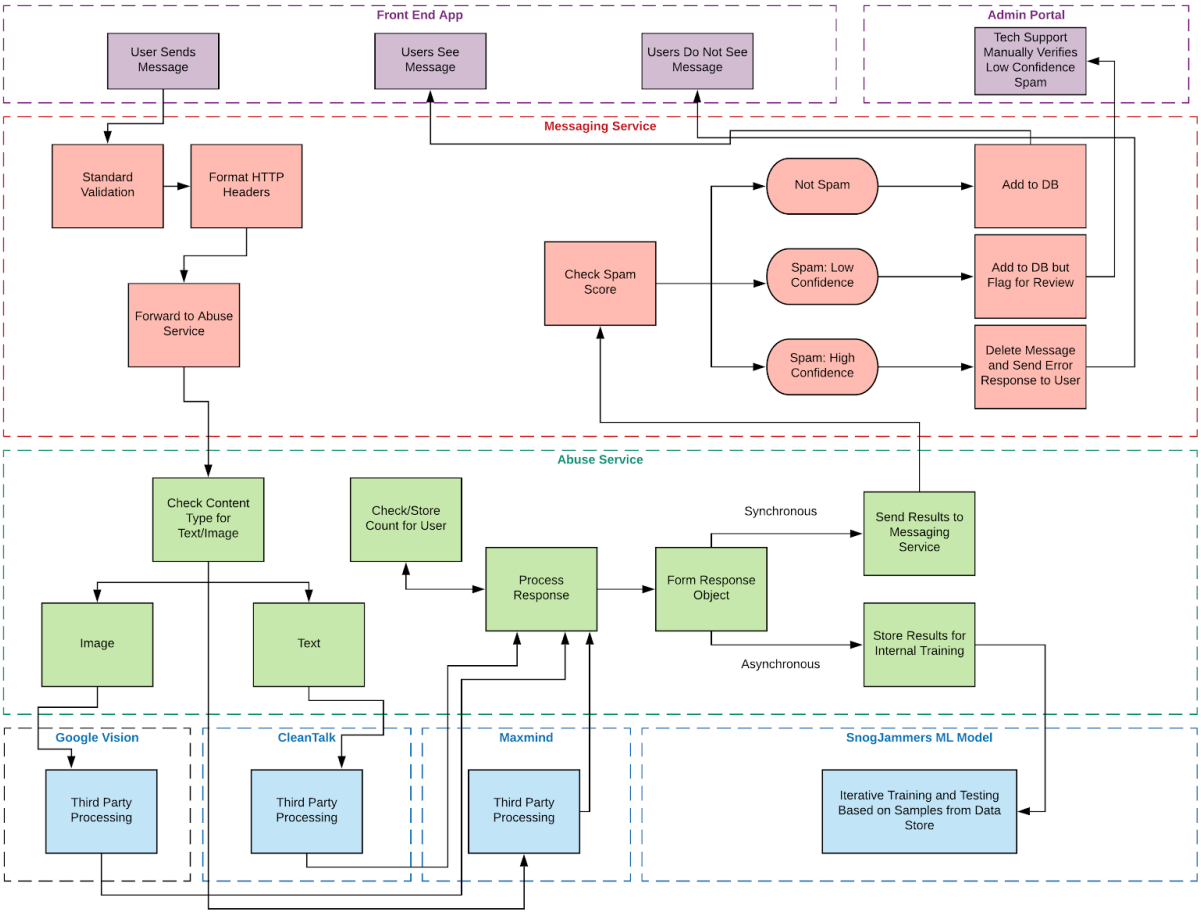


Figure 5. Process Flow

The message is passed on to the Messaging Service, which will need to be modified to forward a well-formed (see Section 3.2) request to the new Abuse Service. The Abuse Service would then parse out text and image content to be forwarded to the CleanTalk and Google Vision APIs, respectively. This step is temporary and would ultimately be replaced by SnogJammers' own ML-based text and image filtering software. The returned responses are processed and synchronously passed back to the Messaging Service to determine whether the content should be removed, reviewed, or posted to the SnogJammers platform. Simultaneously, an asynchronous process (configured to not interfere with operational data processing) copies the Abuse Service response to SnogJammers' ML cloud environment for model training and testing.

End User Experience

This series of mockups shows the general user experience (UX) that SnogJammers intends to implement on the front-end mobile application. Here, the user SpamBoy5000 sends a spam message (and/or an image that breaks the Terms of Service (ToS)), as shown in Figure 6 below.

Figure 6. Mockup: Breaking ToS

The spam filtration process described above is invoked, and the message is not sent. In addition, the user is met with an alert message that states this fact and provides a means for the user to refute the claim, if they believe their message was blocked in error. This would occur when the Abuse Service returns a "grey case," which would indicate uncertainty on whether the message was actually spam/abusive. The message would be stored in our system's database (DB) and flagged for review by the tech support staff.

If a message was confirmed as spam/abusive by the Abuse Service, then the message would be completely blocked, also meaning that it would not be stored in our operational DB. The message would appear to have been passed on to the target user and then the spam user, as shown in Figure 7 below. However, no content would have actually been sent; and neither would it persist, since it would be stored temporarily in user device storage. The message would, however, be temporarily stored in our data science team's test/train database for further model refinement.

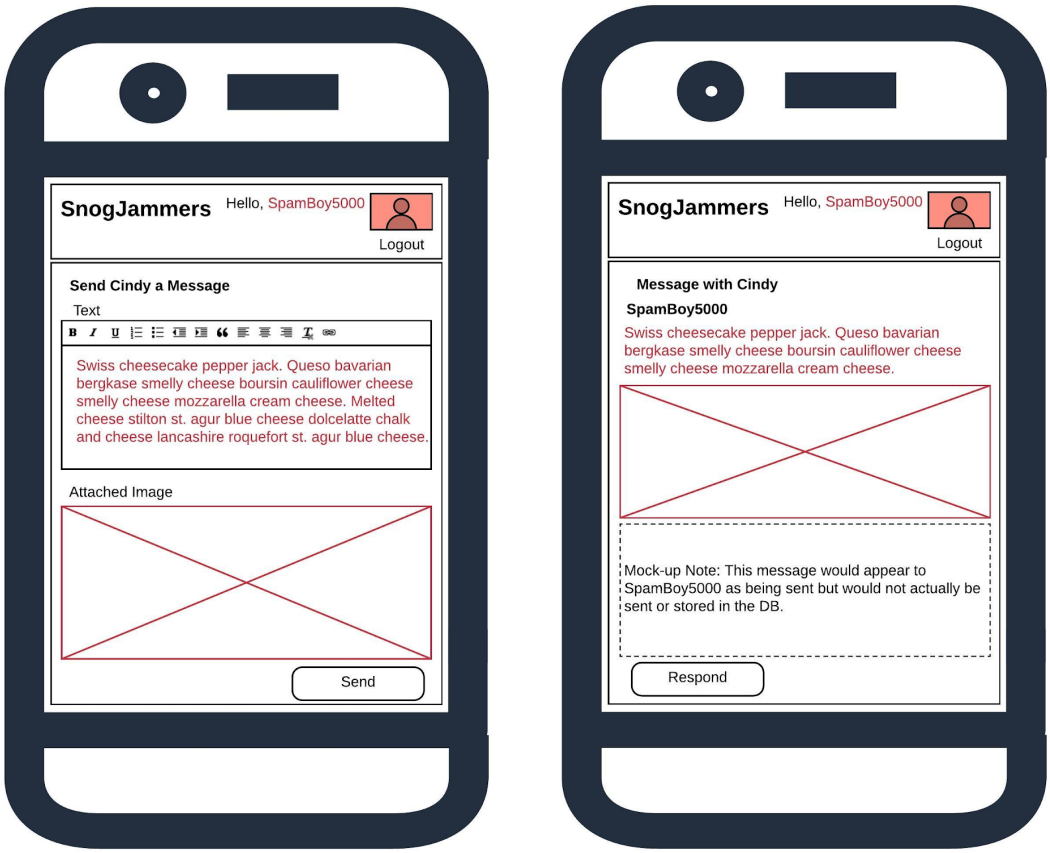


Figure 7. Mockup: Message Blocked After Confirmation of Spam by the Abuse Service

The alternative can be seen in the following mockup in Figure 8 below. In this case, a user, Jeff, sends valid content, and it is, in fact, posted to the intended recipient.

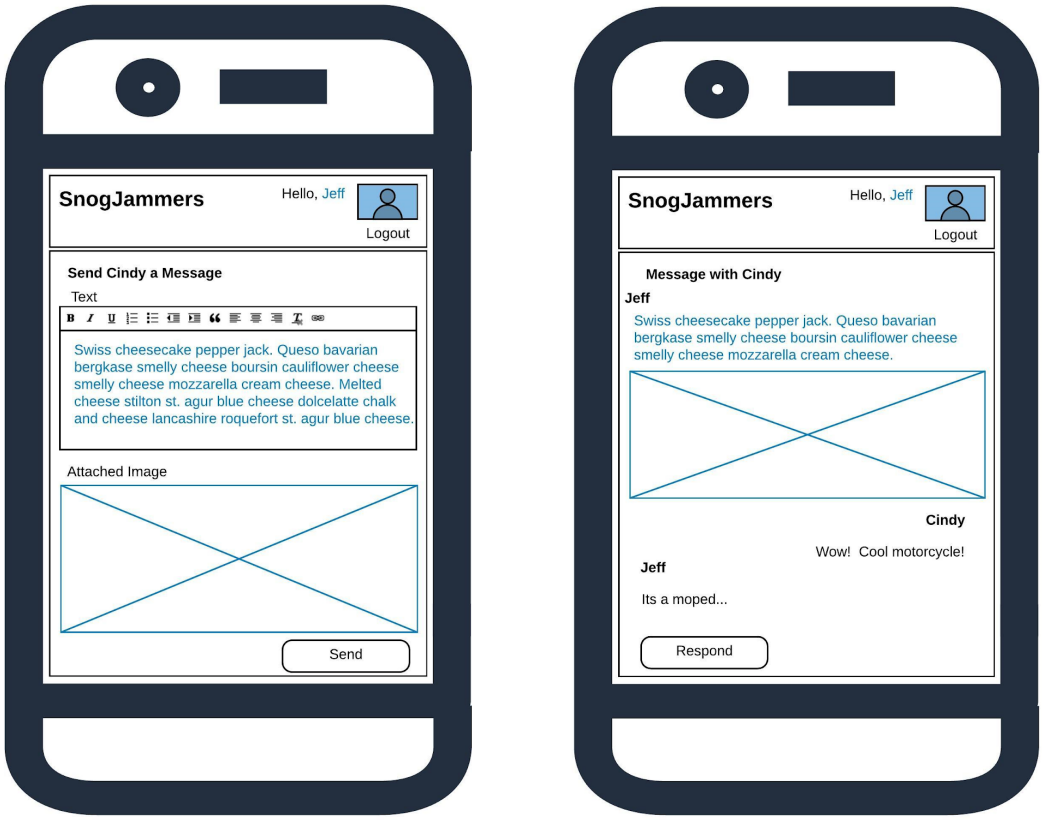


Figure 8. Mockup: Message Allowed After Confirmation of Valid Content by the Abuse Service

Tech Support Dashboard Mockup

The following mockup in Figure 9 below shows the administrative view for internal SnogJammer's tech support staff. There will be a new admin panel (for abuse) added that will intelligently present tech support staff with only ambiguous spam content that the Abuse Service was not able to mark as spam with high confidence.

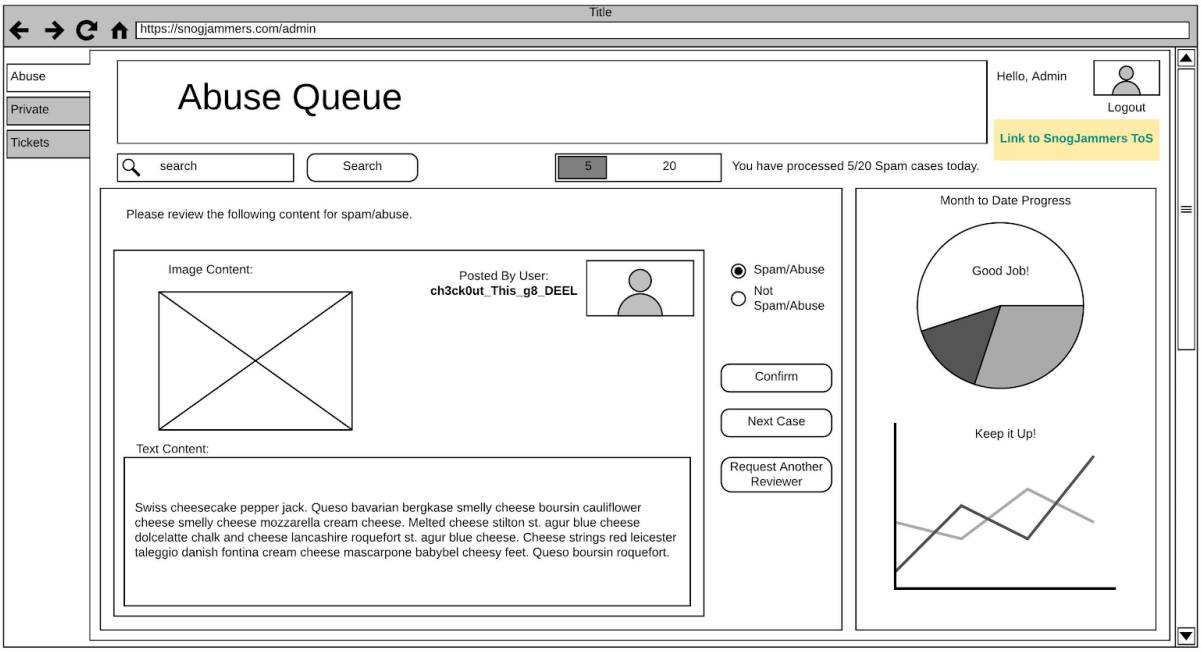


Figure 9. Mockup: Technical Support Administrative View

4 IMPLEMENTATION PLAN

4.1 Solution Delivery Roadmap

This section provides an overview of our planned solution delivery. We have split up the project into three major releases with the duration of each equivalent to a fiscal quarter. First, Release 0 focuses on rudimentary spam detection. Next, Release 1 focuses on preparing the machine learning model to be production-ready utilizing third-party support. Finally, Release 2 focuses on ensuring that the internal SnogJammers automated spam moderation system is production-ready and able to supersede the third-party spam detection engine. See Table 3 for more specific details for each role in change management sans Data Science because it is just responsible for preparing the internal model to be test-ready and switching the third-party spam filter to the internal model (i.e. monitoring and evaluating performance of the in-house model after switch).

Table 1. Solution Delivery Overview

Date	**June 3 - Aug 30 (Q1)**	**Sep 2 - Nov 29 (Q2)**	**Dec 2 - Feb 28 (Q3)**
Release #	Release 0 (W 1-13)	Release 1 (W 14-26)	Release 2 (W 27-39)
Goals	PHASE I (W 1-6) Software Engineering, Project Management ● Add a button to the Messaging page to help users report spam messages more easily ● Add a new option on the support case filing page for appealing spam reports ● Spin up a microservice for the newly-created Abuse platform	PHASE 1 (W 14-17) Software Engineering, Project Management, Data Science, Change Management ● Work with the Messaging team to integrate the newly-created Abuse Service into the Messaging microservice ● Ramp up the traffic of our internal ML model to the Abuse Service, and start to block spam messages	PHASE 1 (W 27-39) Software Engineering, Project Management, Data Science, Change Management, Tech Support ● Run the in-house model in parallel to the third-party spam filter ● Evaluate the performance of the in-house model ● Gradually shift traffic to the in-house model ● Implement code cleanup by removing code used for calling the third-party spam filter

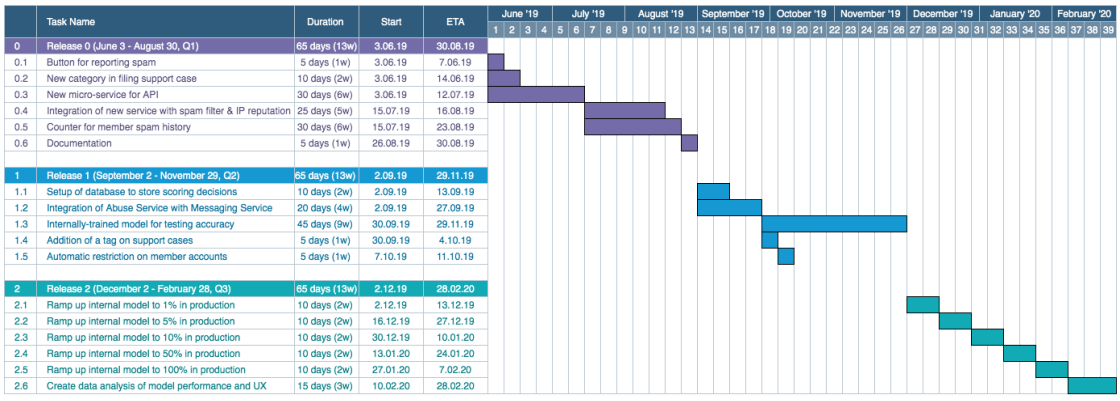
	PHASE II (W 7-12) Software Engineering, Data Science ● Integrate with vendors for spam filtering, IP reputation score, and fetching of geolocation ● Design and implement database schema for storing scoring decisions; it will be used for performance evaluation and in-house model training; this will include a counter for member spam history	PHASE II (W 18-26) Software Engineering, Project Management, Data Science, Change Management, Tech Support ● Add tags to specify whether users are restricted by the system automatically or by tech support personnel manually ● Get the service to automatically block obvious spam messages, as well as send messages in the 'grey area' to the review queue ● Update Terms of Service ● Provide training for tech support to handle different cases ● Begin to collect data and pick features for training the internal model ● Evaluate the performance of the third-party spam filter service ● Start to pick features and train model for Release 2	
	PHASE III (W 13) Tech Support ● Write and improve internal wiki pages		

In Release 0, our first phase consists of three deliverables: new user interface (UI) components and logic for reporting spam in the app, a new category for filing support cases, and a new microservice API; this lasts 6 weeks. The second phase of Release 0 consists of two deliverables: integration of a new service with a spam filter and IP reputation assessment, as well as a counter for member spam history; this lasts another 6 weeks. Finally, the third phase of Release 0 is documentation, which lasts 1 week.

In Release 1, our first phase consists of two deliverables: database setup to store scoring decisions and integration of the Abuse Service with the Messaging Service; this lasts 4 weeks. The second and final phase of Release 1 consists of three more deliverables: an internally-trained model for testing performance, addition of a tag on support cases, and automatic restriction on member accounts; this lasts 9 weeks.

Then, in Release 2, we intend to ramp up the internal model's production capacity in the following iterations: 1%, 5%, 10%, 50%, and finally, 100% over periods of two weeks each. The last three weeks focuses on creating data analysis of model performance and user experience.

In order to build our own ML model and system that would replace third-party spam-filtering, there will be additional labor, design, data subscription, and planning required. While we do replace the external vendor's spam filter, we do not intend to replace the IP reputation source; we will continue to use a vendor's service for that. In terms of labor, we will need our internal engineering team to clean data, pick features, and evaluate models based on performance and accuracy. We will also have engineering stand ups and sprint planning meetings to groom these additional features. Additional task level information can found in Table 1. Solution Delivery Overview and Figure 10. Solution Delivery Roadmap.



	Task Name	Duration	Start	ETA
0	Release 0 (June 3 - August 30, Q1)	65 days (13w)	3.06.19	30.08.19
0.1	Button for reporting spam	5 days (1w)	3.06.19	7.06.19
0.2	New category in filing support case	10 days (2w)	3.06.19	14.06.19
0.3	New micro-service for API	30 days (6w)	3.06.19	12.07.19
0.4	Integration of new service with spam filter & IP reputation	25 days (5w)	15.07.19	16.08.19
0.5	Counter for member spam history	30 days (6w)	15.07.19	23.08.19
0.6	Documentation	5 days (1w)	26.08.19	30.08.19
1	Release 1 (September 2 - November 29, Q2)	65 days (13w)	2.09.19	29.11.19
1.1	Setup of database to store scoring decisions	10 days (2w)	2.09.19	13.09.19
1.2	Integration of Abuse Service with Messaging Service	20 days (4w)	2.09.19	27.09.19
1.3	Internally-trained model for testing accuracy	45 days (9w)	30.09.19	29.11.19
1.4	Addition of a tag on support cases	5 days (1w)	30.09.19	4.10.19
1.5	Automatic restriction on member accounts	5 days (1w)	7.10.19	11.10.19
2	Release 2 (December 2 - February 28, Q3)	65 days (13w)	2.12.19	28.02.20
2.1	Ramp up internal model to 1% in production	10 days (2w)	2.12.19	13.12.19
2.2	Ramp up internal model to 5% in production	10 days (2w)	16.12.19	27.12.19
2.3	Ramp up internal model to 10% in production	10 days (2w)	30.12.19	10.01.20
2.4	Ramp up internal model to 50% in production	10 days (2w)	13.01.20	24.01.20
2.5	Ramp up internal model to 100% in production	10 days (2w)	27.01.20	7.02.20
2.6	Create data analysis of model performance and UX	15 days (3w)	10.02.20	28.02.20

Figure 10. Solution Delivery Roadmap

Our Abuse Service microservice is a new addition to our platform. We have the following migration steps for ensuring its implementation within the existing system:

1. Work with the Messaging team, who owns the Messaging Service, to integrate the new Abuse Service via REST API calls to get the spam decision on each message.
2. Setup a dashboard for monitoring service health and number of messages classified daily as spam, as well as the number of support cases in order to appeal account restriction(s).
3. Ramp up the integration of the Abuse Service on operational traffic in a tiered approach.
4. Monitor the metrics.

Since the Abuse Service is a new service, there is no old system to switch from. The only switch we have is in Release 2, when we switch from the third-party vendor to our internal model. We will be implementing a parallel run for both spam classifiers to compare their performance. We will be gradually shifting traffic from the vendor to the internal model.

For project risks, we have a lack of reference and task spillover. For dependencies, we have accuracy of vendors and ML model with data. For constraints(s), we have computational load for scoring. Finally, for our management plan, we want to follow spam trends, monitor the false positive rate of our vendor, and monitor the machine health.

4.2 Operationalization

Table 2. Non-functional Components that Support the System

Component	Purpose
Kubernetes Cluster	Hosting and application configurations
Github Enterprise	Code repository
Docker private registry	Docker images repository
Jenkins	Building and deployment
Jira	New feature requests and incident reports
Internal Wiki	General information about the project
Elastic Stack	Performance monitoring and alerting

Tech support ticketing system	Manual review of spam content by tech support team
Terms of Service pages	Provide our users with the details of new spam policies

Table 3. Change Management

Roles	**Process**	**Description**
Software Engineering	Release to non-production	The software engineering will automate deployment to stage environment using Jenkins.
Change Management and Software Engineering	Release to production	The Change Management team will use Jenkins to release code into production after approving the request. The updates will be communicated using a Jira release ticket.
Project Management	New features and enhancements	Request for new features and enhancements would be entered into Jira. Those requests would be prioritized by project management.
Tech Support	Incident reports	Incident reports for the Abuse Service functionality can be entered into Jira by any employee of the company including technical support personnel.

Table 4. Infrastructure Operations

Process	**Description**
Hosting	The Abuse Service API will be hosted on the Kubernetes cluster similar to other microservices in the company. Zero downtime deployments will be handled by standard Kubernetes API functionality.
Code Repository	The software engineering team will use Github for storing code and collaborating on development tasks.
Build Artifact Repository	Build artifacts (docker images) created by Jenkins builds would be stored in the Docker private registry. This process will be automated.
Application Performance Metrics	The following information is already captured for every service running on the Kubernetes cluster: CPU usage, Memory, Network I/O, HTTP request/response

	information. All this information is forwarded to the Elasticsearch cluster (document store database). The information is displayed using dashboards in Kibana (visualization software - part of the Elastic Stack).
Application Alerts	Automated alerts will be set up using existing Elasticsearch Watcher functionality (alerting/notification software - part of the Elastic Stack). Any user of Elastic Stack will be able to set up alerts based on the data logged onto Elasticsearch about the Abuse Service.

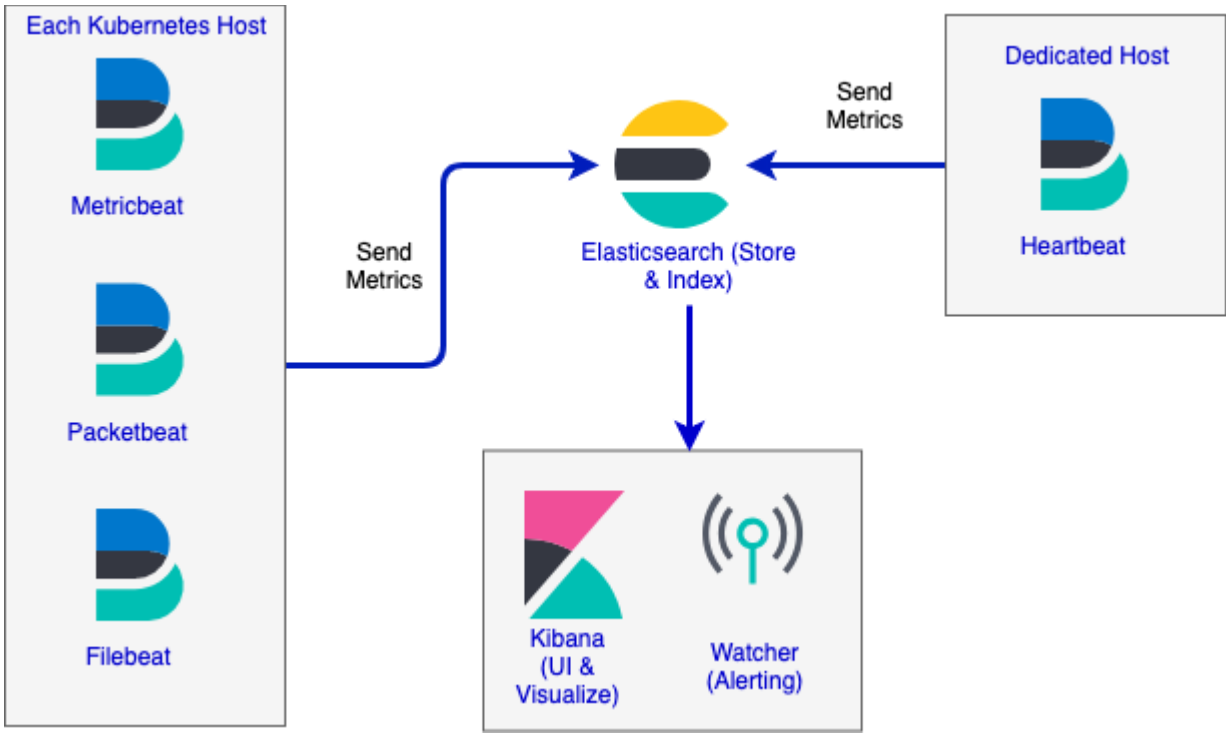


Figure 11. Current System Used for Operational Metrics and Application Alerts

Service Level Agreement (SLA)

- Abuse service has 99.99% uptime
- Abuse Service - response time within 200 ms for 99% of requests

4.3 User Enablement

This section outlines how SnogJammers intends to socialize the platform changes with both internal (largely tech support staff) and external (registered users of the platform) users. This will

include how SnogJammers intends to handle user access, roll out, and training activities, as related to the release phases outlined in Section 4.1.

Internal Engineering User Access

In terms of back-end system access for all phases of development, service access should always be protected by access control lists (ACL). Individuals who want to access the system resources should be granted the associated permissions/privileges with an Abuse Service team member's approval for actions, such as: SSHing into the production server for debugging purposes, commiting new changes, deploying new versions, and viewing data collected by the service and system monitoring.

User Acceptance, Communication, and Training

Before release, information about the new system changes will be emailed out to all currently registered users, regardless of updating to the newest version of the app, to ensure that users are aware of the ToS updates. This communication will include information on user rights and privacy, as well as provide a way for users to request additional information about the changes. Particularly, this messaging will outline that the new system will store anonymized segments of messages for no more than 30 days for use in spam detection modeling.

Our legal, human resources, and tech support divisions will work collaboratively to develop training programs and collateral to help support the new tech support workflows, as well as offer a one-day training and Q/A session for tech support staff. Engineering staff spearheading the project will present system functionality at company-wide knowledge transfer meetings. Lastly, an email alias will be set up for issues related to the project and system updates to be sent to for review.

In order to notify internal staff of the system changes, we will send out a company-wide notification email describing the changes at a high level, as well as what those changes will mean to tech support staff, in terms of enhancing their workflows. Additionally, we intend to work

with our policy team to update our terms of service (ToS) which will be released with the app's content update. This will do the following for users:

- Inform them of the changes to the system
- Educate on ways to report/refute spam or abuse in the new system
- Identify constitutes spam or abusive content
- Inform them about information security and privacy (including GDPR requirements)
- Show ways for users to request more details

User Rollout

During Release 0, much of the proposed software and system changes for this project should occur in the background with little to no interruption on daily system operations. Mobile application clients, or end users, will utilize their respective platform-update mechanism for major version updates.

Upon downloading and installing the latest update (which will be marked as a mandatory content update), users will have the ability to report or refute spam cases. Other than those UI additions, very little of the user experience will be impacted, as much of the spam filtration logic will be implemented and integrated on the back-end. After extensive unit testing and integration testing (in a simulated test environment), the Abuse Service will be rolled out in test intervals against varying levels of operational traffic in order to assess its impact on the end-user experience.

We intend to follow the roll-out schedule below for piloting the introduction of the Abuse Service (powered by third-parties at this point) into our operational system:

- 1% of the traffic for 1 day for testing to see if there is any major issue
- 5% of the traffic for 2 days
- 10% of the traffic for 2 days
- 50% of the traffic for 1 week
- 100% of the traffic, once the above tests have been completed and met target benchmarks

In parallel to the above, we will run a replica instance of the Abuse Service in our data science platform utilizing our own internal model. The goal here will be to fine-tune and compare the performance to the active third-party system, as it is being exposed to operational traffic for ultimate use in Release 2.

Tech support will begin to benefit from their enhanced spam filtration dashboard and queuing system immediately. Support will be able to test the functionality of the system by using the review queue starting on Day 1 of the first release.

As of Release 2, as mentioned in Section 4.1, we intend to begin transitioning to the internal model in the Abuse Service and phasing out the third-party spam detection engine. We will accomplish this in a series of two week iterations, starting with 1% of operational traffic utilizing our internal model (with the rest using the third-party system), then 5%, 10%, 50% and finally 100%. Internal engineering staff will be monitoring this transition and it will occur seamlessly for our internal tech support users and external customers.

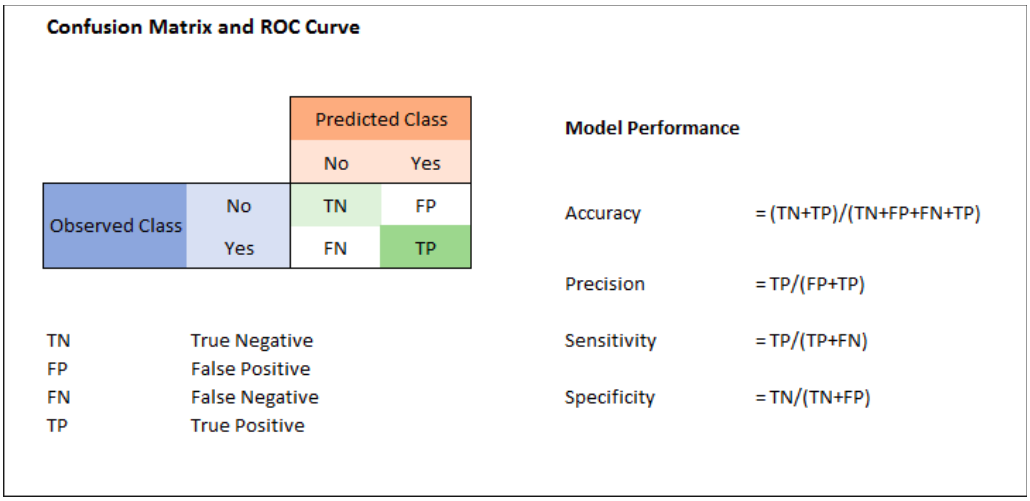
4.4 Success Metrics

The data science team will be primarily responsible for measuring and evaluating the fruits of everyone's labor. They will compile quarterly reports, as well as update data dashboards so that the company can track progress over time. Frequently, when organizations ask their new customers, "How did you hear about us?," a common answer is 'word-of-mouth.' Information cascades can be very powerful. Knowing that, it's important to ensure positive information is spread. The top row outlines the long-term improvement to the platform's reputation.

Table 5. Success Metrics

Impacts to company's top or bottom line	• Improve social proof of the app by garnering higher rated reviews in the iOS App Store and Android Play Store • Receive fewer negative comments related to spam on the platform's social commerce

	• A more attractive interface and improved user experience has attracted more "eyeballs" and opened up more advertising dollars.
System adoption: user subscriptions, transaction volumes, etc.	• An increase in total active accounts by 10% • A decrease in the percentage of users deleting their accounts due to spam from 45% to 20% • Data science will track and update the volume of users on the platform over time.
System operation parameters: scope, availability, etc.	• Abuse Service is able to extend to other abuse areas by implementing more endpoints to be integrated by different internal microservices • Abuse Service is able to change its internal implementation without asking consuming client to change



Additionally, negative predictive value = (TN) / (FN + TN). Positive predictive value and precision are the same.

Figure 12. Confusion Matrix for Model Predictions

True negatives are the messages that are NOT spam that are *correctly* labeled "NOT spam". The frequency of true negatives should be *high*, relative to the amount of messages being moderated. False positives are the messages that are NOT spam that are *incorrectly* labeled "spam". The frequency of false positives should be *low*, relative to the amount of messages being moderated. False negatives are the messages that are spam that are *incorrectly* labeled "NOT spam". The frequency of false negatives should be *low*, relative to the amount of messages being moderated.

True positives are the messages that are spam that are *correctly* labeled "spam". The frequency of the true positives should be *high*, relative to the amount of messages being moderated. Ideally, the project will improve in all of the measures of model performance shown in the confusion matrix; the measures deemed the most significant are outlined in "Project Implementation Metrics" section below.

Project Implementation Metrics	● A decrease in the percentage of spam making it into the database from 70% to 30% ● An increase in precision (positive predictive value) from 80% to 90%. Precision answers the question "out of content the model predicts to be spam, how frequently was the model correct?" When content is predicted to be spam, the prediction should be correct (precision is high). Otherwise, a low precision means clean content is being incorrectly detected as spam, and thus will not pass the filter. ● Identify all spam content correctly at 99% (sensitivity). Sensitivity (true positive rate) answers the question "out of all content that is actually spam, how much of the content was correctly identified?

Bibliography

About suspended accounts. (n.d.). Retrieved April 24, 2019, from https://help.twitter.com/en/managing-your-account/suspended-twitter-accounts

S. Abu-Nimeh, D. Nappa, X. Wang, S. Nair, "A comparison of machine learning techniques for phishing detection", *Proc. Anti-Phishing Work Groups 2nd Annu. Ecrime Res. Summit*, pp. 60-69, 2007. Google Scholar

A. A. Akinyelu, A. O. Adewumi, "Classification of phishing email using random forest machine learning technique", *J. Appl. Math., vol. 2014*, pp. 1-6, Apr. 2014. Google Scholar

Alba, D. (2017, June 03). Pinterest Unveils Its New Spam-Fighting Tool. Retrieved March 14, 2019, from https://www.wired.com/2015/02/pinterest-stingray/

I. Alberts, D. Forest, "Email pragmatics and automatic classification: A study in the organizational context", *J. Amer. Soc. Inf. Sci. Technol., vol. 63*, pp. 904-922, May 2012. Google Scholar

A. Almomani, B. B. Gupta, S. Atawneh, A. Meulenberg, E. Almomani, "A survey of phishing email filtering techniques", *IEEE Commun. Surveys Tuts., vol. 15*, pp. 2070-2090, 4th Quart. 2013. Google Scholar

Aski, A., & Sourati, N. (2016, July 1). Proposed efficient algorithm to filter spam using machine learning techniques. Science Direct. Retrieved from https://www.sciencedirect.com/science/article/pii/S2405882316300412

T. Ayodele, S. Zhou, R. Khusainov, "Email classification: Solution with back propagation technique", *Proc. Int. Conf. Int. Technol. Secured Trans. (ICITST)*, pp. 1-6, 2009. Google Scholar

T. Ayodele, R. Khusainov, D. Ndzi, "Email classification and summarization: A machine learning approach", *Proc. IET Conf. Wireless Mobile Sensor Netw. (CCWMSN)*, pp. 805-808, 2007. Google Scholar

A. Bacchelli, T. Dal Sasso, M. D'Ambros, M. Lanza, "Content classification of development Emails", *Proc. 34th Int. Conf. Softw. Eng.*, pp. 375-385, 2012. Google Scholar

M. Balakumar, V. Vaidehi, Ontology Based Classification and Categorization of Email, New York, NY, USA:*IEEE Press*, 2008. Google Scholar

M. T. Banday, S. A. Sheikh, "Multilingual e-mail classification using Bayesian filtering and language translation", *Proc. Int. Conf. Contemp. Comput. Informat.*, pp. 696-701, 2015. Google Scholar

M. T. Banday, S. A. Sheikh, "Folder classification of urdu and hindi language e-mail messages", *Proc. 3rd Int. Conf. Comput. Knowl. Eng. (Iccke)*, pp. 59-63, 2013. Google Scholar

S. Baskaran, "Content based email classification system by applying conceptual maps", *Proc. Int. Conf. Intell. Agent Multi-Agent Syst. (IAMA)*, pp. 1-2, 2009. Google Scholar

Bell, K. (2018, July 24). Twitter wants to kill spam for good. Retrieved March 14, 2019, from https://mashable.com/article/twitter-api-restrictions-spam-abuse/#JKKfFnzedmqd

V. H. Bhat, V. R. Malkani, P. D. Shenoy, K. R. Venugopal, L. M. Patnaik, "Classification of email using BeaKS: Behavior and keyword stemming", *Proc. IEEE Region 10th Conf.*, pp. 1139-1143, Nov. 2011. View Article Full Text: PDF (715KB) Google Scholar

E. Blanzieri, A. Bryl, "A survey of learning-based techniques of email spam filtering", *Artif. Intell. Rev., vol. 29*, pp. 63-92, Sep. 2008. Google Scholar

A. Borg, N. Lavesson, "E-mail classification using social network information", *Proc. 7th Int. Conf. Availability Rel. Secur. (Ares)*, pp. 168-173, 2012. Google Scholar

J. D. Brutlag, C. Meek, "Challenges of the email domain for text classification", *Proc. ICML*, pp. 103-110, 2000. Google Scholar

J. M. Carmona-Cejudo, M. Baena-García, J. D. Campo-Avila, R. Morales-Bueno, "Feature extraction for multi-label learning in the domain of email classification", *Proc. IEEE Symp. Comput. Intell. Data Mining Symp. Ser. Comput. Intell. (SSCI CIDM)*, pp. 30-36, Sep. 2011. Google Scholar

J. M. Carmona-Cejudo, M. Baena-Garcia, J. del Campo-Avila, R. Morales-Bueno, A. Bifet, "GNUsmail: Open framework for on-line email classification", *Proc. 19th Eur. Conf. Artif. Intell.*, pp. 1141-1142, 2010. Google Scholar

M. D. del Castillo, A. Iglesias, J. I. Serrano, H. Yin, P. Tino, E. Corchado, W. Byrne, X. Yao, "Detecting phishing e-mails by heterogeneous classification" in Intelligent Data Engineering and Automated Learning–Ideal, Berlin, Germany:*Springer-Verlag*, vol. 4881, pp. 296-305, 2007. Google Scholar

S. Chakravarthy, A. Venkatachalam, A. Telang, "A graph-based approach for multi-folder email classification", *Proc. 10th IEEE Int. Conf. Data Mining (ICDM)*, pp. 78-87, Sep. 2010. Google Scholar

M. Chang, C. K. Poon, "Using phrases as features in email classification", *J. Syst. Softw., vol. 82*, pp. 1036-1045, Sep. 2009. Google Scholar

N. Chatterjee, S. Kaushik, S. Rastogi, V. Dua, "Automatic email classification using user preference ontology", *Proc. Int. Conf. Knowl. Eng. Ontol. Develop. (KEOD)*, pp. 165-170, Sep. 2010. Google Scholar

Chilling, M. (2018, February 11). Comparison of machine learning methods in email spam detection. Retrieved from https://www.matchilling.com/comparison-of-machine-learning-methods-in-email-spam-detection/

Christina, Karpagavalli, & Suganya. (2010, 12). Retrieved from A Study on Email Spam Filtering Techniques: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.4041&rep=rep1&type=pdf

W. W. Cohen, "Learning rules that classify e-mail", *Proc. AAAI Spring Symp. Mach. Learn. Inf. Access*, pp. 25, 1996. Google Scholar

Confusion Matrix. Everything About Data Science. 24 March 2016. https://scaryscientist.blogspot.com/2016/03/confusion-matrix.html?view=classic

A. Y. Daeef, R. B. Ahmad, Y. Yacob, N. Yaakob, K. N. F. K. Azir, "Multi stage phishing email classification", *J. Theor. Appl. Inf. Technol., vol. 83*, pp. 206-214, Sep. 2016. Google Scholar

Dasgupta, A. (2018, 10 17). *NAIVE BAYES: An Implementation Of Email Spam Filtering*. Retrieved from https://news.siliconindia.com/technology/NAIVE-BAYES-An-Implementation-Of-Email-Spam-Filtering-nid-205725-cid-2.html

R. Dazeley, J. L. Yearwood, B. H. Kang, A. V. Kelarev, B. H. Kang, D. Richards, "Consensus clustering and supervised classification for profiling phishing emails in Internet commerce security" in Knowledge Management and Acquisition for Smart Systems and Services, Berlin, Germany:*Springer-Verlag, vol. 6232*, pp. 235-246, 2010. Google Scholar

Dormehl, L. (2019, 05 01). *What is an artificial neural network? Here's everything you need to know*. Retrieved from https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/

N. O. F. Elssied, O. Ibrahim, W. Abu-Ulbeh, "An improved of spam e-mail classification mechanism using K-means clustering", *J. Theor. Appl. Inf. Technol., vol. 60*, pp. 568-580, Apr. 2014. Google Scholar

Farmer, C. (2019, Jan 22). Domain Reputation Or IP Reputation: Gmail Care About More? Retrieved from mailgun: https://www.mailgun.com/blog/domain-ip-reputation-gmail-care-more-about

Ferreira, H. (2018, 03 29). *Basics of Machine Learning and a simple implementation of the Naive Bayes algorithm*. Retrieved from https://medium.com/hugo-ferreiras-blog/basics-of-machine-learning-and-a-simple-implementation-of-the-naive-bayes-algorithm-80c1e67a2e8a

Fumo, D. (2017, June 15). Types of Machine Learning Algorithms You Should Know. Towards Data Science. Retrieved from https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

W. N. Gansterer, D. Polz, M. Boughanem, C. Berrut, J. Mothe, C. SouleDupuy, "E-mail classification for phishing defense" in Advances in Information Retrieval, Berlin, Germany:*Springer-Verlag, vol. 5478*, pp. 449-460, 2009. Google Scholar

Grigonis, H. (2018, January 19). Here's What Social Media Giants Are Doing To Fight Extremist Content. Retrieved March 14, 2019, from https://www.digitaltrends.com/social-media/senate-hearing-terrorism-and-social-media-extremist-content-january-2018/

J. C. Gomez, M. F. Moens, "PCA document reconstruction for email classification", *Comput. Statist. Data Anal., vol. 56*, pp. 741-751, Sep. 2012. Google Scholar

J. C. Gomez, E. Boiy, M.-F. Moens, "Highly discriminative statistical features for email classification", *Knowl. Inf. Syst., vol. 31*, pp. 23-53, Apr. 2012. Google Scholar

T. S. Guzella, W. M. Caminhas, "A review of machine learning approaches to spam filtering", *Expert Syst. Appl., vol. 36*, pp. 10206-10222, Oct. 2009.Google Scholar

I. R. A. Hamid, J. Abawajy, T. H. Kim, "Using feature selection and classification scheme for automating phishing email detection", *Stud. Informat. Control, vol. 22*, pp. 61-70, Mar. 2013. Google Scholar

T. Ichimura, A. Hara, Y. Kurosawa, T. Ichimura, A. Hara, Y. Kurosawa, "A classification method for spam e-mail by self-organizing map and automatically defined groups", *Proc. IEEE Int. Conf. Syst. Man Cybern., vol. 1*, pp. 310-315, Oct. 2007. Google Scholar

Inzaugarat, E. (2018, October 30). Understanding Neural Networks: What, How and Why. Towards Data Science. Retrieved from https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31

R. Islam, Y. Xiang, Email Classification Using Data Reduction Method, New York, NY, USA:*IEEE Press*, 2010. Google Scholar

M. R. Islam, W. L. Zhou, M. Y. Guo, X. Yang, "An innovative analyser for multi-classifier e-mail classification based on grey list analysis", *J. Netw. Comput. Appl., vol. 32*, pp. 357-366, Mar. 2009. Google Scholar

R. Islam, W. L. Zhou, M. U. Chowdhury, Email Categorization Using (2+1)-Tier Classification Algorithms, Los Alamitos, CA, USA:*IEEE Computer Soc*, 2008. Google Scholar

M. R. Islam, J. Singh, A. Chonka, W. Zhou, Multi-Classifier Classification of Spam Email on an Ubiquitous Multi-Core Architecture, Los Alamitos, CA, USA:*IEEE Computer Soc*, 2008. Google Scholar

M. R. Islam, W. L. Zhou, A. Hua, S. L. Chang, "Minimizing the limitations of GL analyser of fusion based email classification" in Algorithms and Architectures for Parallel Processing Proceedings, Berlin, Germany:*Springer-Verlag, vol. 5574*, pp. 761-774, 2009. Google Scholar

M. R. Islam, J. Abawajy, M. Warren, Multi-Tier Phishing Email Classification with an Impact of Classifier Rescheduling, New York, NY, USA:*IEEE, 2009*. Google Scholar

A. G. K. Janecek, W. N. Gansterer, "E-mail classification based on NMF", *Proc. 9th SIAM Int. Conf. Data Mining (SDM)*, pp. 1345-1354, 2009. Google Scholar

Javed, A. (2018, August 25). Unfolding Naïve Bayes from Scratch. Towards Data Science. Retrieved from https://towardsdatascience.com/unfolding-na%C3%AFve-bayes-from-scratch-2e86dcae4b01

D. M. Jones, Learning to Improve E-mail Classification With Numero Interactive, London, U.K.:*Springer-Verlag*, 2010. Google Scholar

D. M. Jones, "Learning to improve e-mail classification with numáro interactive", *Proc. 29th SGAI Int. Conf. Innov. Techn. Appl. Artif. Intell.*, pp. 377-390, 2010. Google Scholar

C. Jou, A. Selamat, N. T. Nguyen, H. Haron, "Spam e-mail classification based on the IFWB algorithm", *Proc. Intell. Inf. Database Syst., vol. 7802*, pp. 314-324, 2013. Google Scholar

M. Khonji, A. Jones, Y. Iraqi, "An empirical evaluation for feature selection methods in phishing email classification", *Comput. Syst. Sci. Eng., vol. 28*, pp. 37-51, Apr. 2013. Google Scholar

A. Krzywicki, W. Wobcke, A. Nicholson, X. Li, "Incremental e-mail classification and rule suggestion using simple term statistics" in Advances in Artificial Intelligence, Berlin, Germany:*Springer-Verlag, vol. 5866*, pp. 250-259, 2009. Google Scholar

Lam, F. (n.d.). Francolam/capstone. Retrieved April 24, 2019, from https://github.com/francolam/capstone/blob/master/src/AntiSpam.java

C. C. Lai, C. H. Wu, "Particle swarm optimization-aided feature selection for spam email classification", *Proc. 2nd Int. Conf. Innov. Comput. Inf. Control (ICICIC)*, pp. 165, 2007. Google Scholar

Laura. (2014, Sep). IP reputation. Retrieved from Word to the wise: https://wordtothewise.com/2014/09/ip-reputation/

Liberman, N. (2017, January 26). Decision Trees and Random Forests. Towards Data Science. Retrieved from https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991

W. Li, W. Meng, Z. Tan, Y. Xiang, "Towards designing an email classification system using multi-view based semi-supervised learning", *Proc. 13th IEEE Int. Conf. Trust Secur. Privacy Comput. Commun. (TrustCom)*, pp. 174-181, Sep. 2015. Google Scholar

W. Li, W. Meng, "An empirical study on email classification using supervised machine learning in real environments", *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 7438-7443, Jun. 2015. Google Scholar

M. Li, Y. Park, R. Ma, H. Y. Huang, "Business email classification using incremental subspace learning", *Proc. 21st Int. Conf. Pattern Recognit. (ICPR)*, pp. 625-628, 2012. Google Scholar

Lowd, D. (2018, June 13). Can Facebook Use AI to Fight Online Abuse? Retrieved March 14, 2019, from https://www.scientificamerican.com/article/can-facebook-use-ai-to-fight-online-abuse/

T. Ma, H. Xu, "The research on email classification based on q-Gaussian kernel SVM", *J. Theor. Appl. Inf. Technol., vol. 48*, pp. 1292-1299, Sep. 2013. Google Scholar

Marcin Pawlowski: Spam-fighting Infrastructure. (2015, May 14). Retrieved April 24, 2019, from https://www.youtube.com/watch?v=IkrLfE9TCf4

M. N. Marsono, M. W. El-Kharashi, F. Gebali, "Targeting spam control on middleboxes: Spam detection based on layer-3 e-mail content classification", *Comput. Netw., vol. 53*, pp. 835-848, Apr. 2009. Google Scholar

M. N. Marsono, M. W. El-Khaxashi, F. Gebali, S. Ganti, Distributed Layer-3 E-mail Classification for SPAM Control, New York, NY, USA:*IEEE Press*, 2006. Google Scholar

Maxmind GeoLocation Documentation: https://dev.maxmind.com/geoip/

Melendez, S. (2015, May 21). How Facebook Turned The Social Graph Into A Hacker Alarm System. Retrieved March 14, 2019, from https://www.fastcompany.com/3045006/facebook-threatexchange-fighting-malware-spam-hackers

J. R. Mendez, M. Reboiro-Jato, F. Diaz, E. Diaz, F. Fdez-Riverola, "Grindstone4Spam: An optimization toolkit for boosting e-mail classification", *J. Syst. Softw., vol. 85*, pp. 2909-2920, Dec. 2012. Google Scholar

J. R. Mendez, D. Glez-Pena, F. Fdez-Riverola, F. Diaz, J. M. Corchado, "Managing irrelevant knowledge in CBR models for unsolicited e-mail classification", *Expert Syst. Appl., vol. 36*, pp. 1601-1614, Mar. 2009. Google Scholar

Y. Meng, W. Li, L. F. Kwok, "Enhancing email classification using data reduction and disagreement-based semi-supervised learning", *Proc. 1st IEEE Int. Conf. Commun. (ICC)*, pp. 622-627, Apr. 2014. Google Scholar

Metz, C. (2015, 07 09). *Google says its ai catches 99.9 percent of gmail spam*. Retrieved from https://www.wired.com/2015/07/google-says-ai-catches-99-9-percent-gmail-spam/

S. Misina, B. Ruesch, "Incremental learning for e-mail classification" in Computational Intelligence Theory and Application, Berlin, Germany:*Springer-Verlag*, pp. 545-553, 2006. Google Scholar

T. S. Moh, N. Lee, S. Dua, S. Sahni, D. P. Goyal, "Reducing classification times for email spam using incremental multiple instance classifiers", *Proc. Inf. Intell. Syst. Technol. Manage., vol. 141*, pp. 189-197, 2011. Google Scholar

M. Mohamad, A. Selamat, "An evaluation on the efficiency of hybrid feature selection in spam email classification", *Proc. 2nd Int. Conf. Comput. Commun. Control Technol.*, pp. 227-231, 2015. Google Scholar

Moon, M. (2018, February 21). Twitter's new rules prohibit bulk tweeting to fight spam. Retrieved March 14, 2019, from https://www.engadget.com/2018/02/21/twitters-rules-prohibit-bulk-tweeting/

monapi. (2018). IP Address Anomaly API. Retrieved from monapi: https://www.monapi.io/

monapi. (2018). pricing. Retrieved from monapi: https://www.monapi.io/pricing/

G. Mujtaba, L. Shuib, R.G. Raj, N. Majeed, M.A. Al-Garadi. "Email Classification Research Trends: Review and Open Issues", May. 2017. Google Scholar

Nguyen, D., Alam, F., Ofli, F., & Muhammad, I. (2017, May 1). Automatic Image Filtering on Social Networks Using Deep Learning and Perceptual Hashing During Crises. Research Gate. Retrieved from https://www.researchgate.net/publication/315786191_Automatic_Image_Filtering_on_Social_Networks_Using_Deep_Learning_and_Perceptual_Hashing_During_Crises

N. A. Novino, K. A. Sohn, T. S. Chung, "A graph model based author attribution technique for single-class e-mail classification", *Proc. 14th IEEE/ACIS Int. Conf. Comput. Inf. Sci. (ICIS)*, pp. 191-196, Sep. 2015. Google Scholar

M. A. Oveis-Gharan, K. Raahemifar, "Multiple classifications for detecting spam email by novel consultation algorithm", *Proc. IEEE 27th Can. Conf. Elect. Comput. Eng.*, pp. 1-5, 2014. Google Scholar

S. Park, D. U. An, "Automatic e-mail classification using dynamic category hierarchy and semantic features", *IETE Techn. Rev., vol. 27*, pp. 478-492, Apr. 2010. Google Scholar

N. Perez-Diaz, D. Ruano-Ordas, J. R. Mendez, J. F. Galvez, F. Fdez-Riverola, "Rough sets for spam filtering: Selecting appropriate decision rules for boundary e-mail classification", *Appl. Soft Comput., vol. 12*, pp. 3671-3682, Nov. 2012. Google Scholar

Process-Software. (2019). *Common Spam Filtering Techniques*. Retrieved from http://www.process.com/products/pmas/whitepapers/explanation_filter_techniques.html

J. Pujara, H. Daumé, L. Getoor, "Using classifier cascades for scalable e-mail classification", *Proc. 8th Annu. Collaboration Electron. Messaging Anti–Abuse Spam Conf. (CEAS)*, pp. 55-63, Sep. 2011. Google Scholar

Punjabi, V. (2015, 11 15). *SMS SPAM-HAM Detection using Artificial Neural Networks*. Retrieved from https://www.youtube.com/watch?reload=9&v=iZPIKIlYfKU

I. Qabajeh, F. Thabtah, "An experimental study for assessing email classification attributes using feature selection methods", *Proc. 3rd Int. Conf. Adv. Comput. Sci. Appl. Technol. (ACSAT)*, pp. 125-132, 2014. Google Scholar

J. J. Qing, R. L. Mao, R. F. Bie, X. Z. Gao, D. S. Huang, K. H. Jo, H. H. Lee, V. Bevilacqua, H. J. Kang, "An AIS-based e-mail classification method" in Emerging Intelligent Computing Technology and Applications: With Aspects of Artificial Intelligence, Berlin, Germany:*Springer-Verlag, vol. 5755*, pp. 492-499, 2009. Google Scholar

D. K. Renuka, P. Visalakshi, "Latent semantic indexing based SVM model for email spam classification", *J. Sci. Ind. Res., vol. 73*, pp. 437-442, Jul. 2014. Google Scholar

Report Violations. (n.d.). Retrieved April 24, 2019, from https://help.twitter.com/en/rules-and-policies/twitter-report-violation#specific-violations

S. A. Saab, N. Mitri, M. Awad, "Ham or spam? A comparative study for some content-based classification algorithms for email filtering", *Proc. (MELECON)*, pp. 439-443, 2014. Google Scholar

A. A. Al Sallab, M. A. Rashwan, "E-mail classification using deep networks", *J. Theor. Appl. Inf. Technol., vol. 37*, pp. 241-251, Sep. 2012. Google Scholar

Sayan Dasgupta & Ted Hwa: Finding Spam in Short Text Fields. (2015, May 14). Retrieved April 24, 2019, from https://www.youtube.com/watch?v=fcdnaEACICc

E. Sayed, M. El.-Alfy, Discovering Classification Rules for Email Spam Filtering With an ant Colony Optimization Algorithm, New York, NY, USA:*IEEE Press*, 2009. Google Scholar

M. R. Schmid, F. Iqbal, B. C. M. Fung, "E-mail authorship attribution using customized associative classification", *Digit. Investigat., vol. 14*, pp. S116-S126, Aug. 2015. Google Scholar

Schmulen, M. (2015, May 20). What is Social Spam? (And How to Avoid Creating It). Retrieved March 14, 2019, from https://blogs.constantcontact.com/social-spam-infographic/

T. F. Shi, S. Sambath, E. Zhu, "Research on the application of e-mail classification based on support vector machine" in Frontiers in Computer Education, Berlin, Germany:*Springer-Verlag, vol. 133*, pp. 987-994, 2012. Google Scholar

L. Shi, Q. Wang, X. Ma, M. Weng, H. Qiao, "Spam email classification using decision tree ensemble", *J. Comput. Inf. Syst., vol. 8*, pp. 949-956, Sep. 2012. Google Scholar

Sift Science. (n.d.). Stopping Content Abuse Before It Happens. Sift Science. Retrieved from https://pages.siftscience.com/rs/526-PCC-974/images/ebook-stopping-content-abuse.pdf

S. Smadi, N. Aslam, L. Zhang, R. Alasem, M. A. Hossain, "Detection of phishing emails using data mining algorithms", *Proc. 9th Int. Conf. Softw. Knowl. Inf. Manage. Appl. (Skima)*, pp. 1-8, 2015. View Article Full Text: PDF (1041KB) Google Scholar

M. H. Song, H. Liu, Y. Yang, S. Shen, Z. Zhong, L. Zheng, P. Feng, "E-mail classification based learning algorithm using support vector machine" in Materials Mechanical Engineering and Manufacture, Stäfa, Switzerland:*Trans Tech Publications Ltd, vol. 268*, pp. 1844-1848, 2013. Google Scholar

Spam | Facebook Help Center. (n.d.). Retrieved April 24, 2019, from https://www.facebook.com/help/287137088110949

Sprengers, M. (n.d.). *The Effects of Different Bayesian Poison Methods on the Quality of the Bayesian Spam Filter 'SpamBayes'*. Retrieved from The Effects of Different Bayesian Poison Methods on the Quality of the Bayesian Spam Filter 'SpamBayes'

J. Stefanowski, M. Zienkowicz, F. Esposito, Z. W. Ras, D. Malerba, G. Semeraro, "Classification of polish email messages: Experiments with various data representations" in Foundations of Intelligent Systems Proceedings, Berlin, Germany:*Springer-Verlag, vol. 4203*, pp. 723-728, 2006. Google Scholar

Tewari, A., & Jangale, S. (2016, November 1). Spam Filtering Methods and Machine Learning Algorithm - A Survey. Retrieved from https://pdfs.semanticscholar.org/0cfe/f70cc930a0d597c78b396c498420ddbf900f.pdf

Tolentino, J. (2018, June 14). 5 Types of Social Spam (and How to Prevent Them). Retrieved March 14, 2019, from https://thenextweb.com/future-of-communications/2015/04/06/5-types-of-social-spam-and-how-to-prevent-them/

Twitter Terms of Service. (n.d.). Retrieved April 24, 2019, from https://twitter.com/en/tos#update

Y. W. Wang, Y. N. Liu, L. Z. Feng, X. D. Zhu, "Novel feature selection method based on harmony search for email classification", *Knowl.-Based Syst., vol. 73*, pp. 311-323, Jan. 2015. Google Scholar

Z. J. Wang, Y. Liu, Z. J. Wang, D. L. Liu, X. B. Zhu, K. L. Xu, D. M. Fang, "E-mail filtration and classification based on variable weights of the Bayesian algorithm" in Applied Science Materials Science and Information Technologies in Industry, Zürich, Switzerland:*Trans Tech Publications Ltd, vol. 513*, pp. 2111-2114, 2014. Google Scholar

M. F. Wang, M. F. Tsai, S. L. Jheng, C. H. Tang, "Social feature-based enterprise email classification without examining email contents", *J. Netw. Comput. Appl., vol. 35*, pp. 770-777, Apr. 2012. Google Scholar

M. R. I. Wanlei, W. L. Zhou, Email Categorization Using Multi-Stage Classification Technique, Los Alamitos, CA, USA:*IEEE Computer Soc*, 2007. Google Scholar

Webroot, Inc. (2018). BrightCloud IP Reputation Service. Retrieved from BrightCloud: https://www-cdn.webroot.com/5715/2511/0532/BC-IP-Reputation-DS_us.pdf

T. L. Wong, K. O. Chow, F. Wong, Incorporating Keyword-Based Filtering to Document Classification for Email Spamming, New York, NY, USA:IEEE Press, 2007. Google Scholar

K. Xu, C. Wen, Q. Yuan, X. He, J. Tie, "A mapreduce based parallel SVM for email classification", *J. Netw., vol. 9*, pp. 1640-1647, Sep. 2014. Google Scholar

Yadav, A. (2018, October 20). SUPPORT VECTOR MACHINES(SVM). Towards Data Science. Retrieved from https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589

K. Yelupula, S. Ramaswamy, "Social network analysis for email classification", *Proc. 46th Annu. Southeast Regional Conf. (ACM-SE)*, pp. 469-474, 2008. Google Scholar

Yenala, H., Jhanwar, A., & Chinnakotla, M. (2017, December 27). Deep learning for detecting inappropriate content in text. Springer Link. Retrieved from https://link.springer.com/article/10.1007/s41060-017-0088-4

Y. F. Yi, C. H. Li, W. Song, Email Classification Using Semantic Feature Space, Los Alamitos, CA, USA:*IEEE Computer Soc*, 2008. Google Scholar

S. Youn, "SPONGY (SPam ONtoloGY): Email classification using two-level dynamic ontology", *Sci. World J., vol. 2014*, pp. 1-11, Sep. 2014. Google Scholar

S. Youn, D. McLeod, A Comparative Study for Email Classification, Dordrecht, The Netherlands:*Springer*, 2007. Google Scholar

S. Youn, D. McLeod, "Spam email classification using an adaptive ontology", *J. Softw., vol. 2*, pp. 43-55, Sep. 2007. Google Scholar

B. Yu, D. H. Zhu, "Combining neural networks and semantic feature space for email classification", *Knowl.-Based Syst., vol. 22*, pp. 376-381, Sep. 2009. Google Scholar

M. Zareapoor, P. Shamsolmoali, M. A. Alam, J. K. Mandal, S. C. Satapathy, M. K. Sanyal, P. P. Sarkar, A. Mukhopadhyay, "Highly discriminative features for phishing email classification by SVD" in Information Systems Design and Intelligent Applications, Berlin, Germany:*Springer-Verlag, vol. 339*, pp. 649-656, 2015. Google Scholar

C. Zeng, J. Gu, Z. Lu, "A new approach to email classification using concept vector space model", *Proc. 2nd Int. Conf. Future Generat. Commun. Netw. Symp. (FGCN)*, pp. 162-166, 2008. Google Scholar

W. Zhao, Y. Zhu, "An email classification scheme based on decision-theoretic rough set theory and analysis of email security", *Proc. IEEE Region 10th Conf. (TENCON)*, pp. 1-6, 2007. Google Scholar

Zheng, X., Zeng, Z., Chen, Z., Yu, Y., & Rong, C. (2015, February 24). Detecting spammers on social networks. Retrieved March 14, 2019, from https://www.sciencedirect.com/science/article/pii/S0925231215002106

Zhou, Z., & Sun, L. (n.d.). *Network-based spam filter on Twitter.*

Z. Q. Zhu, An Email Classification Model Based on Rough Set and Support Vector Machine, Los Alamitos, CA, *USA:IEEE Computer Soc*, 2008. Google Scholar

Appendix

Example of a request to the Abuse Service API to check if text content is spam:

```
curl -X POST
"https://{{abuse-service-domain-name}}/antispam/scoreText"
-H "Accept: application/json"
-H "Content-Type: application/json"
-v
-d '{
"content":"some message content",
"memberId":"e03fc8a015f2",
"userPublicIP":"159.127.104.69"
}'
```

- content - text content of the message
- memberId - user id of platform user
- userPublicIP - IPv4 public IP of incoming request

Example response from the Abuse Service API:

```
{
  "isSpam": true
}
```

- isSpam - indicator for whether the message should be blocked due to spam

Example of a request to the Abuse Service API to check if image content is spam:

```
 curl -X POST
"https://{{abuse-service-domain-name}}/antispam/scoreImage"
-H "Accept: application/json"
-H "Content-Type: application/json"
-v
-d '{
"mediaUrl":"https://someimage.png",
```

```
"memberId":"e03fc8a015f2",
"userPublicIP":"159.127.104.69"
}'
```

- mediaUrl - url of an image to check for spam
- memberId - user id of platform user
- userPublicIP - IPv4 public IP of incoming request

Example response from the Abuse Service API:

```
{
  "isSpam": true
}
```

- isSpam - indicator for whether the message should be blocked due to spam

Example of a request to the Google Vision API:

```
{
        "image": {
              "content": "base64ImageString"
         },
        "features": [
          {
            "type": "SAFE_SEARCH_DETECTION"
          },
         // More feature detection types....
        ]
}
```

Example of response from the Google Vision API:

```
"safeSearchAnnotation" : {
      "spoof" : "VERY_UNLIKELY",
      "medical" : "UNLIKELY",
```

```
      "adult" : "VERY_UNLIKELY",
      "violence" : "VERY_UNLIKELY"
   }
```

Loaded from the Abuse Service API database to the Data Warehouse

- Member ID
- Request IP
- IP Reputation
    - Example of IP reputation object:

```
{
"121.229.228.80": { "appears": 0, "network_type": "unknown",
"frequency": "3", "updated": "2019-03-27 18:43:52", "spam_rate": "1",
"frequency_time_10m": 0, "frequency_time_1h": "3",
"frequency_time_24h": "3" }
}
```

- Message Content
- Confidence - a numerical spam confidence score from 1 to 100% and the sources contributing to the score (CleanTalk API, internal model, Vision API).
    - Example of Confidence object:

```
{
"score": 0.89, "sources": [
{"name": "clean-talk","reasons":["SPAM_WORDS"]},
{"name":
"business-rules-engine","reasons":["MESSAGES_IN_LAST_MINUTE"]},
{"name": "vision","reasons":["VIOLENCE"]}]
}
```

Loaded from the Messaging Service API database to the Data Warehouse

- Member ID, Request IP, Message Content