SPAM Blocker for Blogs – ANTI SPAM

Minor Project

IBM Career Education

Disclaimer

This Software Requirements Specification document is a guideline. The document details all the high level requirements. The document also describes the broad scope of the project. While developing the solution if the developer has a valid point to add more details being within the scope specified then it can be accommodated after consultation with IBM designated Mentor.

INTRODUCTION

The purpose of this document is to define scope and requirements of a SPAM Blocker for Blogs (ANTI SPAM) for a corporate house, which wanted their CXOs to have control over comments posted by readers on their respective blogs. Since, the blogs are hosted on Internet, it was extremely critical to prevent unacceptable content in comments.

The proposed system will provide an effective way to determine the quality of comment in the blog article, whenever a reader adds a comment. If the comment entered has unacceptable words, the system will not publish the comment instantly; instead it will mark the comment for moderation by the authorized user.

This document is the primary input to the development team to architect a solution for this project.

System Users

The entire intranet user base of the corporate house will primarily use the SPAM Blocker for Blogs, ANTI SPAM.

Assumptions

1. The stop words dictionary will be uploaded using a CSV file. Such a file will be a single column CSV file containing the list of stop words, i.e. the words that must not be present in the comment.
2. Whenever a new stop words dictionary is uploaded, the previously stored dictionary is cleared and a new dictionary is uploaded in place of the old one. All the necessary computations are carried out again for the newly uploaded stop words dictionary.
3. There is no need to create new blogging software or to integrate with an existing blogging engine for testing. For testing purpose, a simple test bed will be created, where users will be able to post comments. These comments will be tested for the proposed functionality as outlined in this document.

REQUIREMENTS

ANTI SPAM will determine the validity of comment entered on client side (i.e. without needing to make a round trip to the server) on the basis of the following rules:

1. Comment is not empty.
2. The words used in comment should not be present in the dictionary.

Basic System Operation

The system will leverage “bloom filter”, for performing a lookup in to the stop words

dictionary. Bloom Filter is an extremely memory efficient probabilistic data structure for a very quick negative look up. In other words, it can determine that the word in question is “not” in the dictionary of stop words.

The administrator uploads the dictionary in the system. Upon successful upload, ANTI SPAM builds the “bit vector” for the words in the dictionary. This bit vector travels to client side in form of a JSON, on every comment-posting page. The client side Java Script performs the negative look up using this bit vector.

All validations outlined earlier are performed on “post comment” button. If a comment clears the validations, it is immediately published. If the comment contains the stop words, it is automatically routed to a moderation queue. This queue is only visible to the administrator, who can either delete comment or publish the comment “as is”.

The system presents a moderation queue view to the administrator for viewing the entries along with the words identified as unacceptable by the filter.

About Bloom Filter

Bloom Filter is a probabilistic data structure. It is in form of a “bit vector” or an array of bits with all bits initialized to “0”. To add an element, it uses 2 (or more) hash functions that map the element to 2 (or more) of the array positions with a uniform random distribution. These mapped bit positions are set to “1”.

To perform a look up for an element, find the hash of that element using the same set of hash functions used during addition to get 2 (or more) array positions. If any of the bits at these positions are 0, then the element is not in the set.

Therefore, selection of hash functions will be critical, as the same set will be implemented in Java for server side operation to build the bit vector; and in Java Script to perform the negative look up on that bit vector.

The memory efficiency results from the bit vector usage. It will not require several bytes to store each word. For example in form of a bit vector, 10000 words can fit in to an array of about 1.5KB.

DEVELOPMENT ENVIRONMENT

ANIT SPAM will be developed as a web application using Java/JSP and DB2 database. Eclipse will be used as the IDE for the same. You may consider using a JavaScript framework like Prototype. Please refer to textbooks on data structures in your library for Bloom Filters & Hash Functions. You may also refer to http://www.serve.net/buz/hash.adt/java.001.html URL for concepts on hashing and to http://en.wikipedia.org/wiki/Bloom_filter URL for concepts on bloom filter.