Post Filter: An automated system that filters posts from a user’s wall in Online Social Networks

1Nishmitha S, 2Basavaraju M

Studying M. Tech, C.S.E1, Head of the Dept. of C.S.E2
Atria Institute of Technology, Bengaluru, under Visvesvaraya Technological University
Email: 1nishmitha810@gmail.com, 2basavaraju@atria.edu

Abstract—Social networking sites that facilitate communication of information between users allow users to post messages as an important function. Unnecessary posts could spam a user’s wall, which is the page where posts are displayed, thus disabling the user from viewing relevant messages. The aim of this paper is to propose and experimentally evaluate an automated system, called PostFilter, which exploits text classification under data mining, using the neural network learning model to filter out unwanted messages from Online Social Networking (OSN) user walls. Further, additional support is provided by allowing the users to specify filtering policies on the messages to be displayed on their walls. Performance is measured in terms of accuracy of classification of the incoming posts.

Index Terms—Artificial-Intelligence applications, Data mining, Information filtering, Short-text classification

I. INTRODUCTION

Today, there is a continued rise of social networking on the Web. Social networking accounts for 1 of every 6 minutes spent online and as MySpace declines, LinkedIn, Twitter and Tumblr have grown at impressive rates [1]. Social media are becoming increasingly important to recruiters and jobseekers alike. In Online Social Networks (OSNs), there is the possibility of posting or commenting unwanted messages on particular public/private areas, called in general “walls”. Unnecessary posts could spam a user’s wall, thus disabling the user from viewing relevant messages. Information filtering can therefore be used to give users the ability to automatically control the messages written on their own walls, by filtering out unwanted messages.

This paper is primarily based on the work carried out by Marco Vanetti, Elisabetta Binaghi, Elena Ferrari, Barbara Carminati and Moreno Carullo towards proposing a mechanism to filter online posts [2] and explains the effort at practically implementing what is proposed by them, along with certain modifications.

OSNs today do not provide much support to prevent unwanted messages on user walls. For example, Facebook allows users to state who is allowed to insert messages in their walls (i.e., friends, friends of friends, or defined groups of friends). Existing Filters as browser extensions and add-ons are: Spoiler Shield app to filter feeds from both Facebook and twitter, for iPhone and Open Tweet Filter which is a filter for twitter on Chrome. These filters only take keywords and filter out messages that contain the specific words. Applications that are trying to solve this issue through machine learning techniques are either in the beta phase or do not perform efficiently due to poor learning curves used to analyze the messages.

Most of the work related to text filtering by Machine Learning has been applied for long-form text. Wall messages are constituted by short text for which traditional classification methods have serious limitations since short texts do not provide sufficient word occurrences. Thus, a suitable text representation method is proposed in this paper along with a neural-network based classification algorithm [3] to classify each message as neutral or non-neutral, based on its content.

Besides classification facilities, Filtering Rules (FRs) can support a variety of different filtering criteria that can be combined and customized according to the user needs. More precisely, FRs exploit user profiles, user relationships as well as the output of the classification process to state the filtering criteria to be enforced. In addition, the system provides the support for user-defined BlackLists (BLs), that is, lists of users that are temporarily prevented to post any kind of messages on a user wall.

This system is intended to be a software application, that is, an add-on, for any social networking service that allows users to post messages. The social networking application is itself developed first with minimalistic features, the most important being posting short-text
messages. This is to emulate the behaviour of existing OSNs like Facebook and Twitter. Also, a database of users and relationships between the users is maintained. Thus, PostFilter acts as an intelligent software on top of existing OSNs, providing users with a view of clean (non-vulgar and non-offensive) or relevant posts.

The scope of the project with respect to developing this 'PostFilter' system is limited to the social network built here consisting of pseudo-users and to show its correctness and usefulness so that it can be used with existing on-line social networks on the World Wide Web like Facebook, Twitter or LinkedIn. The actual integration is not performed in the scope of the current project. Experiments are conducted to show the effectiveness of the developed filtering techniques, numerically assessing the performances of the short text classification stage.

II. PRODUCT PERSPECTIVE

The architecture of the system in support of OSN services is a three-tier structure (Fig. 2.1), each of the layers and the modules in those layers explained below [2]:

The first layer:

It is the bottom-most layer and aims to provide the basic OSN functionalities i.e. profile and relationship management by maintaining a database as well.

The second layer:

It provides the support for external Social Network Applications (SNAs). It contains two core components:

a. Content-based message filtering (CBMF)

It exploits the message categorization provided by the STC module to enforce the Filtering Rules specified by the user. Blacklists can also be used to enhance the filtering process.

b. Short-text classifier (STC)

It aims to classify messages according to a set of categories.

The third layer:

The Graphical User Interface (GUI) required by the supported SNAs. Users interact with the GUI to set up and manage their FRs/BLs. The GUI provides users with a wall where only messages that are authorized according to their FRs/BLs are published.

The path followed by a message, from its being posted on the GUI by a user to the possible final display on another user’s wall (GUI) can be summarized as follows:

1) The user tries to post a message by specifying a sender among his/her contacts, which is intercepted by the PostFilter GUI.

2) An ML-based text classifier extracts metadata from the content of the message and assigns a class to the message – neutral or non-neutral.

3) The CBMF module uses metadata provided by the classifier, together with data extracted from the social graph and users’ profiles, to enforce the filtering and BL rules.

4) Depending on the result of the previous step, the message will be published or filtered by the PostFilter GUI.

III. SHORT TEXT CLASSIFIER

A. Text Representation

Texts cannot be directly interpreted by a classifier or by a classifier-building algorithm. It is first mapped to a feature vector which is then fed into the classifier. Appropriate text representation is necessary for accurate classification. Two types of features are considered: Bag of Words (BoW) and Document properties (DP) to form the feature vectors [2].

The vector is represented in the Vector Space Model (VSM) [4] where each text document \(d_j\) has a vector \(w_{ij}\), \(w_{2j}\), ..., \(w_{Tj}\) for \(T\) terms with \(w_{ij}\) containing values in the range [0, 1] and being weight that represents how much term \(t_k\) contributes to the semantics of document \(j\). Here, terms are nothing but words in the short-text messages and document refers to any short-text message.

Semantics of a document can be measured by Document Properties described by:

- correct words: list of words that are known and acceptable for the domain language.
- bad words: list of "dirty words" in the domain language.
- capital words: More capital letters in any statement made by someone is considered to be loud, in general and hence, may help in deciding if a message is unacceptable. Here, we calculate the percentage of words within the message that have more than half of the characters in capital case.
• punctuation characters: Sometimes, punctuation characters like ",", ",@", ",*", when used multiple times continuously, indicate the intended usage of bad words. This property is calculated as the percentage of the punctuation characters over the total number of characters in the message.

• exclamation marks: it is calculated as the percentage of exclamation marks over the total number of punctuation characters in the message.

• question marks: it is calculated as the percentage of question marks over the total number of punctuation characters in the message.

The above will form different bags of words and the frequency of occurrence of each word is used as a feature for training the classifier.

tf-idf (term frequency – inverse document frequency) is a concept used to assign weights to terms forming the vector. However, it cannot be used directly since it will not contribute to the semantics of a sentence in any way. Therefore, probability of correct words and other features forming the document properties are considered to form the weights.

Therefore, the vector contains six entries for probability of correct words, bad words, punctuation letters, exclamation marks, and question marks, respectively, in any text message. Any stop word (for e.g., ‘is’, ‘the’, ‘and’, ‘of’, ‘to’, etc.) in the short-text message is not considered while forming the vector as they are not much helpful in identifying if the message is normal or not.

B. Text classification

We consider the Multilayer Perceptron (MLP) artificial neural network architecture for classification as against the RBFN model used in [2].

An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function [5] which was developed to model the frequency of firing of biological neurons in the brain. MLP utilizes a supervised learning technique called back propagation for training the network. Learning occurs by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result.

The training set should be sufficiently complete and consist of consistent pre-classified data. We form two separate lists of normal and abnormal messages, respectively. Abnormal messages comprise of hate messages and messages that contain sexual, vulgar and offensive words. Normal messages or neutral messages comprise of messages that are acceptable in day-to-day conversations in the English speaking world. The data set is formed by downloading personal tweets and also searching online for messages containing “bad words”.

The list of “bad words” and “filter words” themselves are formed by searching on the Web.

The training set is fed as inputs (after processing them to form feature vectors) to the network along with the two output classes they belong to: neutral and non-neutral, during the learning phase. MLP can distinguish data that are not linearly separable. After training, the newly input short-text messages (from users of the social network application) are fed to the classification algorithm as input elements. Since the algorithm is trained to identify particular vectors under certain classes, it maps the newly input vectors into previously classified vectors and outputs their corresponding classes.

Backpropagation is a deep learning algorithm. Observed data is generated by the interactions of many different factors on different levels [6]. Deep learning adds the assumption that these factors are organized into multiple levels, corresponding to different levels of abstraction or composition. Varying numbers of layers and layer sizes can be used to provide different amounts of abstraction. Different concepts are learned from other concepts, with the more abstract, higher level concepts being learned from the lower level ones. Deep learning helps to disentangle these abstractions and pick out which features are useful for learning. Thus, classification can be fine-tuned by adjusting the number of hidden layers in the network during the learning phase which sets aside MLP different from RBF (Radial Basis Function) networks.

IV. FILTERING RULES AND BLACKLISTS

The Message Filtering module contains two sub-modules whose functions are:

C. Blacklist Management

This sub-module enables authentic users to BlackList other users, thus preventing certain users from posting messages into their wall. It checks if the entered Username by the user exists in the database and if already not enlisted in this particular user’s Black List, then adds the Username.

D. Filtering policies management

This sub-module allows users to state constraints on

• Message creators and

• Content of the message.

Given the social network scenario, creators may be identified by exploiting information on their social graph. A creator specification creatorSpec implicitly denotes a set of OSN users, and is a set of relationship constraints of the form,

\[(m, rt, minDepth, maxTrust)\]  \(-----------------------(1)\)

that denotes all the OSN users participating, where

ISSN (Online): 2347-2820, Volume -2, Issue-3, 2014
m is a user in a relationship of type rt, having a depth greater than or equal to minDepth, and a trust value less than or equal to maxTrust.

A filtering rule FR is a tuple (author, creatorSpec, contentSpec, action), --------- (2)

• author is the user who specifies the rule;
• creatorSpec is a creator specification, specified according to Definition 1;
• contentSpec is a Boolean expression defined on content constraints of the form (C, ml), where C is a class of the first or second level and ml is the minimum membership level threshold required for class C to make the constraint satisfied;
• action ∈ {block, notify} denotes the action to be performed by the system on the messages matching contentSpec and created by users identified by creatorSpec.

To facilitate the processing of a filtering rule, especially a creator specification, we model a social network as a directed graph, where each node corresponds to a network user and edges denote relationships between two different users. In particular, each edge is labelled by the type of the established relationship (e.g., friend of, colleague of, relative of) between the users. Therefore, there exists a direct relationship of a given type rt between two users, if there is an edge connecting them having the label rt. Moreover, two users are in an indirect relationship of a given type rt if there is a path of more than one edge connecting them, such that all the edges in the path have label rt.

The corresponding trust value, which represents how much a given user considers trustworthy with respect to that specific kind of relationship the user with whom he/she is establishing the relationship, is calculated as a ratio of how many messages are blocked against the number of messages that are not filtered from the other user, purely based on classification of the messages. Without loss of generality, we suppose that trust levels are rational numbers in the range [0; 1].

When this module receives a message from a user, it checks with the Blacklist manager and filtering rules manager if the sender is Blacklisted or fits the condition specified in a filtering rule. If it does, then it does not forward the message to the ShortTextClassifier. If the sender of the message is not Blacklisted by the receiver or if does not fit any filtering rule specified by the receiver, then the message is sent to the ShortTextClassifier to be classified so that it can decide whether to filter the message or not, based on the class of the message. Also, further calculation of the trust level is performed to aid in the checking of filtering rules the next time a message is posted.

V. CLASSIFICATION ACCURACY

The accuracy of classification is an important performance requirement. When filtering messages, it must be ensured that neutral messages are not filtered. This may be possible if the prediction of a class for a short-text message is inaccurate. Inaccurate classification may also cause abnormal messages not to be filtered and instead, displayed on the recipient’s wall. Hence, it is necessary to compute the classifier accuracy. The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. In the pattern recognition literature, this is also referred to as the overall recognition rate of the classifier, that is, it reflects how well the classifier recognizes tuples of the various classes [7].

We can also speak of the error rate or misclassification rate of a classifier, M, which is simply 1 - Acc(M), where Acc(M) is the accuracy of M.

The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes. Given m classes, a confusion matrix is a table of at least size m by m. An entry, \(CM_{ij}\), in the first m rows and m columns indicates the number of tuples of class i that were labelled by the classifier as class j. For a classifier to have good accuracy, ideally most of the tuples would be represented along the diagonal of the confusion matrix, from entry \(CM_{11}\) to entry \(CM_{m,m}\), with the rest of the entries being close to zero. A confusion matrix for two classes is shown in Table 5.1. In the table, True positives refer to the positive tuples that were correctly labelled by the classifier, while true negatives are the negative tuples that were correctly labelled by the classifier. False positives (also called type I error) are the negative tuples that were incorrectly labelled. Similarly, false negatives (also called type II error) are the positive tuples that were incorrectly labelled. Given these terms, we use them to compute the classification accuracy along with a few other evaluation metrics called precision and recall.

Table I: 2-by-2 contingency table or confusion matrix

<table>
<thead>
<tr>
<th>Test outcome (TO)</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
</tr>
<tr>
<td>True positive</td>
<td>True positive (tp)</td>
</tr>
<tr>
<td>False negative</td>
<td>False negative (fn)</td>
</tr>
</tbody>
</table>
relevant instances that are retrieved [8]. Using the terms used in a contingency table, we compute classifier accuracy, precision and recall as follows:

\[
\text{Acc} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}
\]

(3)

\[
\text{Precision} = \frac{\text{tp}}{\text{tp} + \text{fp}}
\]

(4)

\[
\text{Recall} = \frac{\text{tp}}{\text{tp} + \text{fn}}
\]

(5)

In our context, for OSN filters,

- \( \text{tp} \): number of neutral messages that have not been filtered
- \( \text{fp} \): number of non-neutral messages that have not been filtered
- \( \text{fn} \): number of neutral messages that have been filtered
- \( \text{tn} \): number of non-neutral messages that have been filtered

Additionally, we can also compute f-measure or f-score which is the harmonic mean of precision and recall, computed using (2) and (3):

\[
\text{f-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

(6)

VI. EXPERIMENTAL RESULTS

The test data set is a combination of messages from the training set and other short-text messages downloaded from online social networks. As mentioned earlier, the training set consists of two separate lists of normal and abnormal messages, respectively and is formed by downloading personal tweets and also searching online for example messages.

Five separate test sets are formed ranging from 10 – 500 messages, each tagged with their respective class label: 0 for neutral and 1 for non-neutral. After classification, \( \text{tp}, \ \text{fp}, \ \text{fn} \) and \( \text{tn} \) are measured for each test set of short-text messages and the accuracy, precision and recall are computed. Figure 6.1 shows the performance graph measuring accuracy of classification for the test sets used. Figure 6.2 shows the performance graph for the same test sets in terms of error rate. Figure 6.3, 6.4 and 6.5 are the performance graphs that measure the precision, recall and f-measure, respectively, for these 5 test-sets.

VII. APPLICATION

As mentioned earlier, the social networking application is developed with minimalistic features, the most important being posting short-text messages. PostFilter acts as an intelligent software, as an add-on to the application, providing users with a view of clean (non-vulgar and non-offensive) or relevant posts. Neutral messages can be seen on the user’s wall as soon as a user logs in or by clicking on the ‘Home’
Fig. 6.5: Performance graph depicting f-measure button. Non-neutral messages can be seen after clicking on the ‘Spam’ button. Similarly, provision for ‘Blacklisting’ and specifying ‘Filter policies’ are made by keeping specific buttons on users’ walls. Figure 7.1 shows a user’s wall or home page with one message. Figure 7.2 shows a list of non-neutral messages along with the ones that are blocked due to blacklisted users or due to filter rules.

VIII. CONCLUSIONS

With the increasing usage of online social networks, there is an increasing demand for removal of undesired posts, a basic functionality provided by all social network applications, which spam a user’s wall. There is a need for an automated filtering system that filters unwanted messages with good accuracy. In this paper, we saw a content-based message-filtering system that uses the machine learning technique of artificial neural networks for short-text classification. We have seen the experimental results of classification based on the MLP classification algorithm. The results suggest a good accuracy of classification that helps decide which messages should be filtered, correctly, and which ones should not be filtered. Nevertheless, the classification is not 100% accurate. Hence, there is further scope of study and analysis for achieving a better accuracy, like finding out the optimal combination of factors that form the input vector. Also, other machine learning methods can also be used to find out if they can perform better classification.

Further, the blacklisting and filtering rules policies implemented here are minimalistic. They can be further expanded, for instance, blacklisting can be done based on certain conditions similar to filtering rules. In the same context, data mining techniques can be exploited to suggest the best privacy preferences to suggest to the OSN users, inferred on the basis of the available social network data, as future work.

REFERENCES


