Fraud Detection with Data Mining

Mohammed Fawsan*, Sampath Deegalla* and Roshanth Gardiarachchi†
*Department of Computer Engineering, Faculty of Engineering,
University of Peradeniya, Sri Lanka
†Ideamart at Dialog Axiata PLC,
Sri Lanka

***Abstract*—Ideamart (Dialog Axiata PLC) is the most popular telco based Software Development Platform in Sri Lanka which provides API access to the developers to create telco based applications on top of Dialog Axiata services. Since Ideamart has no control over the content which is being delivered to the end users, there is a huge risk of developers scamming the end users.**

In this research study to eliminate such problems, the data which was sent to the end users were collected and analyzed to detect the spam messages which fails to meet certain criteria expected by the Ideamart. In order to do that a data mining approach is used with a self learning system which is capable of improving itself with new data. The proposed new system is coupled with the existing Ideamart system via a web service.

***Index Terms*—spam, data mining, Ideamart, SMS, Fraud Detection**

I. Introduction

Ideamart [1] is a software development platform (SDP) offered by Dialog Axiata PLC [2] to developers and content providers to use Dialog network based features via shared APIs and monetize their ideas. The SDP allows developers to use network based features on Idea Pro via Application Programing Interface (API) to create telco based applications, The AllApps Appstore [3] presents the applications that are published for the dialog subscribers.

The concept of Ideamart is very new. The idea of allowing developers to use telco APIs to create applications was innovative and got the attention of Groupe Speciale Mobile Association (GSMA) Awards as well. But with great power comes great responsibility. The more Ideamart became popular the more threats Ideamart started to receive. One of the main threat was, a subset of developer community becoming fraudulent.

The current architecture of Ideamart Software Development Platform does not have any mechanism to catch the frauds, because there are many possible ways to commit fraud in the platform. Since the Telecommunications Regulatory Commission of Sri Lanka (TRCSL) is very strict on what to share and what not to share on telecommunication services, Ideamart faces a threat to its existence, as some of the developers/content providers commit fraud on certain applications.

subsection I-A discuss the types of frauds , scams and section II is about previous works related to text classification and spam filtering, algorithm used in this project and how the system is going to be coupled with the existing system are discussed in section III.

A. Types of Fraud

The rapid growth of Ideamart community is helping Dialog customers as well as the developers. Developers get the chance to monetize their idea while the customers get a better service. At the beginning of Ideamart, the number of SMS based applications published on the platform was low, so the quality of the application was monitored by the individuals of Ideamart team. But with time the situation started to change when more developers got exposed to the Ideamart platform. While some of the developers create quality applications, some started to make money with less effort. Thats where the scamming and spamming have started. Some nasty developers started to create applications and cheat people, These frauds were difficult to catch because they used different kind of methods to scam the end users. The main methods used by the frauds to scam the end users can be narrowed and classified as follows.

- Applications which sends no contents or less than one message per day
- The number of messages provided to the end user by any application reduce with time
- Applications which deliver uncensored content to the end users
- Applications which deliver irrelevant content or contents out of the context it was intend to provide
- Application which deliver Viral Spam Messages
- Applications which sends spam messages mixed with genuine content

II. Literature Survey

Though our goal is to address all the issues mentioned in subsection I-A, one of the biggest challenges was to come up with an efficient spam filter for short messages.

The spam filtering problem can be viewed as a text classification problem. Whether it can be SMS or email messages both fall in to the same classification method. Only difference is unlike emails, SMS messages are limited in size and it only contains text while emails may or may not contain graphics, word documents or executable files.

If we model the spam filtering problem to the classical text classification problem, we can divide it to two parts. First part is called as training and the second part is called as testing. In the training process the text classifier is prompted in a set of labeled messages and then training model is used to classify the class of unlabeled message. Other approaches further abstract the messages to a bag of words derived from the message and use it to classify unlabeled messages [4].

The problem of spam classification for SMS has been considered by Healy et al [5], who compare K-Nearest-Neighbor, Support-Vector-Machine and Naive Bayes classifiers on short messages. Messages were represented as bag-of-words improved by some statistical features. They conclude that Support-Vector-Machine and Naive Bayes substantially beat K-Nearest-Neighbor, disagreeing to their previous results for full email messages.

Even though the concept of Ideamart is new, the topic of fraud detection through spam filtering can be boiled down to the classical text classification problem. Hence all of the techniques discussed in the previous methods can be applied to this project, most of the implementations available out there are built for filtering spam messages in emails.

The problem with those implementations are, email messages are usually big in size, they contain different types of documents, graphics, external link etc. which has to be processed as well, but in the case of short message services, the scope is well defined and narrowed. i.e. The size of any message will not exceed 130 characters, and there will not be any attachments attached to the messages as well. Hence the proposed methods for email messages require large hardware resources, we can make use of the same idea, but implement the methods so that it can be run in a regular computer.

III. Methodology

The Data Model implemented in this project is the Multinomial Naive Bayes Classification. Which is a Probabilistic Data Model. A general Naive Bayesian spam filtering can be abstracted into the model presented in Figure 1. It consists of four major components, each responsible for three different processes: message tokenization, probability estimation and Naive Bayesian classification.

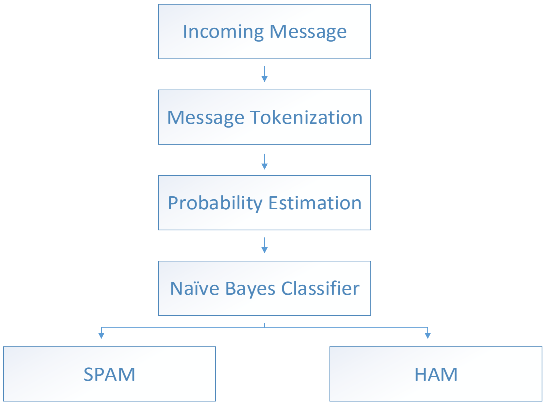

Fig. 1. Architecture of the Data Model

To create the spam filter for the short messages I have used the Multinomial Naive Bayes Classification.The Naive Bayes Classifier (NBC) is a widely used text classification method with a simple probabilistic approach based on Bayes Theorem. The NBC technique is widely applied in spam detection by estimating the conditional probability of a particular word for a given class and its frequency of term w presents in class c.

Since our goal is to classify a message to a ham/spam class and messages are just some bunch of words, we could easily apply NBC to our problem. In a Multinomial Naive Bayes approach, we take a training data set and classify them manually so that they will belong to class spam or to class ham which will give us two separate lists.

The first set of data contains all the words that have appeared in our spam messages along with its frequency. The second set of data contains all the words that have appeared in our ham messages along with its frequency. Now we can use these to find the probability of a particular word appearing in a spam message or in a ham message. And then, we can use Equation 1 to find the probability that a new message and decide whether it is ham or spam.

$$P(w|c) = \frac{count(w,c)+1}{count(c)+|V|} \tag{1}$$

- $P(w|c)$ - probability of a particular word given a class as the relative frequency of term w in text associated to class c
- $count(w,c)$ - frequency of the word w in classified class c
- $count(c)$ - total frequency of words presented in the classified class c
- $|V|$ - frequency of the vocabulary. that is total number of unique words presented in our vocabulary (without the word w)

After creating the spam filter, I have used the domain specific knowledge to create another system to improve the accuracy. Main reason for that is there were certain cases where the spam filter performed really poor. (For example messages with less than given words). In order to over come the issue of false positive I had to make use of the domain knowledge.

Then the spam filter along with the rule based domain knowledge is wrapped with a RESTful API to create the proposed system. Figure 2 shows the architecture of the complete system after integrating our proposed system.

In Figure 2 the *Web Application* is used by the Ideamart administration to monitor the selected applications behavior. The developers who are using the existing Ideamart API will have no direct connection to our proposed system.

They will keep using the existing API, but the Ideamart Administration can route a partial or full traffic through our system to query the state of messages being sent. Upon receiving the results from our system the Ideamart API can then decide whether to allow or block the message.

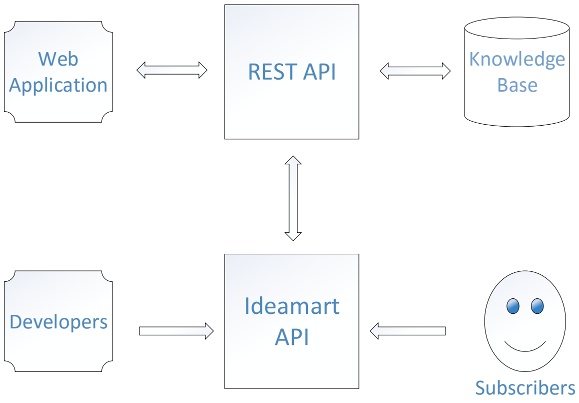

Fig. 2. Architecture of the Proposed System

IV. Results & Discussion

So far I have created the spam filter with the REST API. The Knowledge Base created by the spam filter was used on pure spam and ham message to test the accuracy of the program. The results after running the *Three Fold Cross Validation* test is tabulated in Table I.

TABLE I
Three fold cross validation on known dataset

Round	*TrainingData*	*Spam %*	*Ham %*
One	100% *spam*	99	1
	100% *ham*	5	95
Two	100% *spam*	98	2
	100% *ham*	5	95
Three	100% *spam*	97	3
	100% *ham*	5	95

When the same cross validation test is done on unknown data set, the results were Table II.

TABLE II
Three fold cross validation on unknown dataset

Round	*Accuracy %*	*ResultType*	*Spam %*	*Ham %*
One	92	*Test*	22	78
		Classified	14	86
Two	92	*Test*	19	81
		Classified	13	87
Three	92	*Test*	22	78
		Classified	14	86

From the above results we can see that the spam filter has the accuracy of 92% on unknown data set.

V. Conclusion

In this project, a spam filter is created using the Multinomial Naive Bayes algorithm. The program which does the spam filtering can also retrain itself by processing the detected spam messages. We can see that the Naive Bayes classifier works pretty well but there are other popular classifiers as well. Linear Classifiers [6], Stacking classifiers [7] are few of them. Implementing those classifiers and comparing them with the current result would give a better insight on which one is the best for SMS spam filtering. Those methods will be tried in the future improvements of this project.

Also a secure REST API is created to query the spam filter. The web application which will be giving real time analytics on the applications being monitored will be created in the latter part of this project.

Acknowledgment

I would like to thank Mr. Charith De Silva from Dialog Axiata PLC. for his vauable ideas and comments that greatly improved the project.

References

[1] Dialog. (2015) Ideamart-free app builder. [Online]. Available: https://www.dialog.lk/mobile-miscellaneous-idea-mart/
[2] ——. Dialog home page. [Online]. Available: http://www.dialog.lk
[3] ——. App store home page. [Online]. Available: https://allapps.lk/appstore/
[4] S. F, *Machine learning in automated text categorization.* ACM Computing Surveys 34, 2002, vol. 1, pp. 1–47.
[5] M. Healy, S. Delany, and A. Zamolotskikh, "An assessment of case-based reasoning for short text message classification," in *16th Irish Conference on Artificial Intelligence and Cognitive Science, (AICS-05)*, 2005, pp. 257–266, n. Creaney, Ed.
[6] G. Sakkis and I. Androutsopoulos. (2015) Linear classifiers. [Online]. Available: http://core.ac.uk/download/pdf/11309832.pdf
[7] W. F. Inc. (2015) Stacking classifiers. [Online]. Available: http://cgi.di.uoa.gr/~takis/emnlp01.pdf