Turkish Online Journal of Qualitative Inquiry (TOJQI)

Volume 12, Issue 3, July 2021:1951- 1959

Research Article

User-specific Models for Filtering Distracting Messages on Mobile Instant Messaging

N.M.K. Akilan [1], Akash Ghosal [2], Saad Yunus Sait [3]*

Abstract

The Internet has contributed to a huge transition in lifestyle for most people worldwide. Also, Mobile Instant Messaging (MIM) has been one of the prevalent modes of peer-to-peer communication in the last decade (e.g. WhatsApp, Facebook Messenger). WhatsApp, by comparison, has been shown to lead to poor academic performance related to WhatsApp addiction. It follows that somehow the harms should be controlled to use MIM to our benefit. A machine learning model which shall be able to classify messages as distracting or non-distracting could help by keeping the messaging systems professional and thus enhancing productivity. Online learning has been used in this paper to gradually adapt a model. In this work, multinomial Naive Bayes, linear SVM, perceptron and passive-aggressive models, have been attempted in conjugation with stochastic gradient descent to obtain online models. All of them provide F1-scores better than 0.86, with Multinomial Naive Bayes providing the best F1-score of 0.89. This work paves the way for user-specific models which can be learnt on the fly using user-feedback for users of social networking and instant messaging apps.

Keywords: *Online learning, incremental, stochastic gradient descent, mobile instant messaging.*

Introduction

For a long time, there has been a lot of emphasis on emails, and thus a large number of spam classifiers have come into existence. But, the present generation is more focused on mobile instant messaging applications rather than emails, and that is why a similar system was envisioned to be developed for these applications which would enable users to classify messages into different categories (distracting and non-distracting, for instance). As distracting messages are highly specific, and vary between users, different models need to be learned for different users.

[1] Student, SRM Institute of Science and Technology, Kattankulathur, Computer Science Engineering, as8595@srmist.edu.in

[2] Student, SRM Institute of Science and Technology, Kattankulathur, Computer Science Engineering, aa2745@srmist.edu.in

[3*] Research Associate Professor, Dr., SRM Institute of Science and Technology, Kattankulathur, Computer Science Engineering, saady@srmist.edu.in

Further, the models need to be updated to accommodate change in user behaviour over time. This leads naturally to online models, which can be updated gradually using user feedback.

MIM is used all over the world for the several benefits in it, especially for co-operative activities and knowledge sharing, but at the expense of loss in productivity of the individual; the latter should be curtailed if MIM is to be used for the benefit of users.

There has been very limited research when it comes to filtering distractions on mobile instant messaging systems. The closest research with regard to classifying conversations is (Avrahami, 2006) which predicts the social relationship (i.e. personal or business) between two instant messaging (IM) users based on characteristics of conversations like messaging rate and duration and were able to achieve accuracies of up to 80%. (Elnahrawy, 2002), (Ozyurt, 2010) are related works that classify the conversations not based on the type of conversation, but based on the subject/topic of the conversation. (Elnahrawy, 2002) and (Ozyurt, 2010) have both used SVM, Naive Bayes, and kNN models for classification. Other than that, several studies have been conducted on spam filtering. While most have considered training the model on the entire dataset, a limited number of researchers have also taken incremental models into account.

In a previous work (Sait et al), the authors have generated a dataset for distracting and normal messages, and developed classification models assuming that distracting and normal messages are the same for all. In this work, the authors develop online models for the purpose. Online models can be updated using a few (new) examples that become available from time to time. This is particularly helpful when dealing with data that comes in a stream, such as in social networking and instant messaging apps. Based on user feedback on the incorrect classification of a model, the new example may be used to update the model. This also enables user-specific models to be built for the purpose of filtering distracting messages.

With the objective of user-specific online models, incremental versions of Naive Bayes, linear SVM, perceptron along with the Passive-Aggressive classifier, an online model have been used to classify messages as distracting or normal. Incremental models are obtained by adjusting the models using stochastic gradient descent. All models perform well with an F1-score better than 0.86. Multinomial Naive provides the best F1-score of 0.89.

The next section shall deal with the literature review which aims to find the current knowledge in the field of text classification and online/incremental learning. The following section shall take the readers through the experimental setup which includes the various steps carried out during the research, including dataset preparation, preprocessing, feature extraction, training, validation, and testing. The final sections in the paper deal with the discussion of the results, and the conclusions that could be drawn from the research.

Literature Survey

Text classification has always been a vast topic of interest under the domains of machine learning and natural language processing. Different researchers have published surveys and papers pertaining to text classification.

(Gaurav, 2019) used various machine learning algorithms such as Decision Trees, Naive Bayes, and Random Forest and observed that Random Forest performs the best among all of these. (Bosaeed, 2020) used Support Vector Machine, and Multinomial Naive Bayes for spam classification.

Other than the very commonly known machine learning algorithms such as SVM, Random Forest, Decision Trees, Naive Bayes, etc. there has been use of lesser commonly used algorithms too. (Singh & Bhushan & Vij, 2019) used the Fuzzy c-means algorithm, and (Dendeturk, 2020) used the Artificial Bee Colony algorithm for the same.

The commonality among all the papers discussed above is that they used batch mode of training. Batch mode of training has been used time and again for solving various machine learning problems but online training is much more adaptable. In online learning, the system learns over time and is thus much more suitable for user models in instant messaging systems.
In 2001, Domeniconi and Gunopulos of the University of California highlighted the problem of large memory requirements in batch learning and considered incremental SVM as a solution to that. They showed that incremental learning performs at par with batch learning and in some cases is able to outperform it. They showed that incremental learning algorithms perform better with more and more training.

Since then incremental learning has been used for different purposes in different papers and surveys. (Ma, 2008) used incremental learning for Chinese Text Classification using quick clustering. (Tseng, 2009) designed mailNET which used incremental SVM for spam detection on emails and social media networks. (Shan, 2020) used incremental learning with LSTM for text classification. They categorized news, amazon reviews, tweets on Twitter into several classes. (Hegde, 2017) performed aspect based feature extraction and sentiment analysis of Review Datasets. They used incremental machine learning algorithms and compared them against each other. After using TF-IDF for feature extraction, incremental Naive Bayes gave 67.5% accuracy, incremental Linear SVM gave an accuracy of 70.5% and iterative Decision Tree gave an accuracy of 78.5%. Various other applications of incremental learning including drug discovery were shown by (Laskov, 2006). (Silva, 2017) used MDLText which is a classifier that supports online learning. They were successful in developing an incremental model for classifying small text messages.

In this paper, incremental linear SVM shall be used along with Stochastic Gradient Descent as an optimizer to classify the instant messages on mobile devices. This paper aims to make the model more user-specific by creating separate models for each user, where the user can give feedback on their choice of distracting and normal messages, thereby customizing the model.

Methodology

Experimental Setup

Fig. 1 illustrates the proposed architecture diagram of the system that may be used to filter distracting content for MIM.

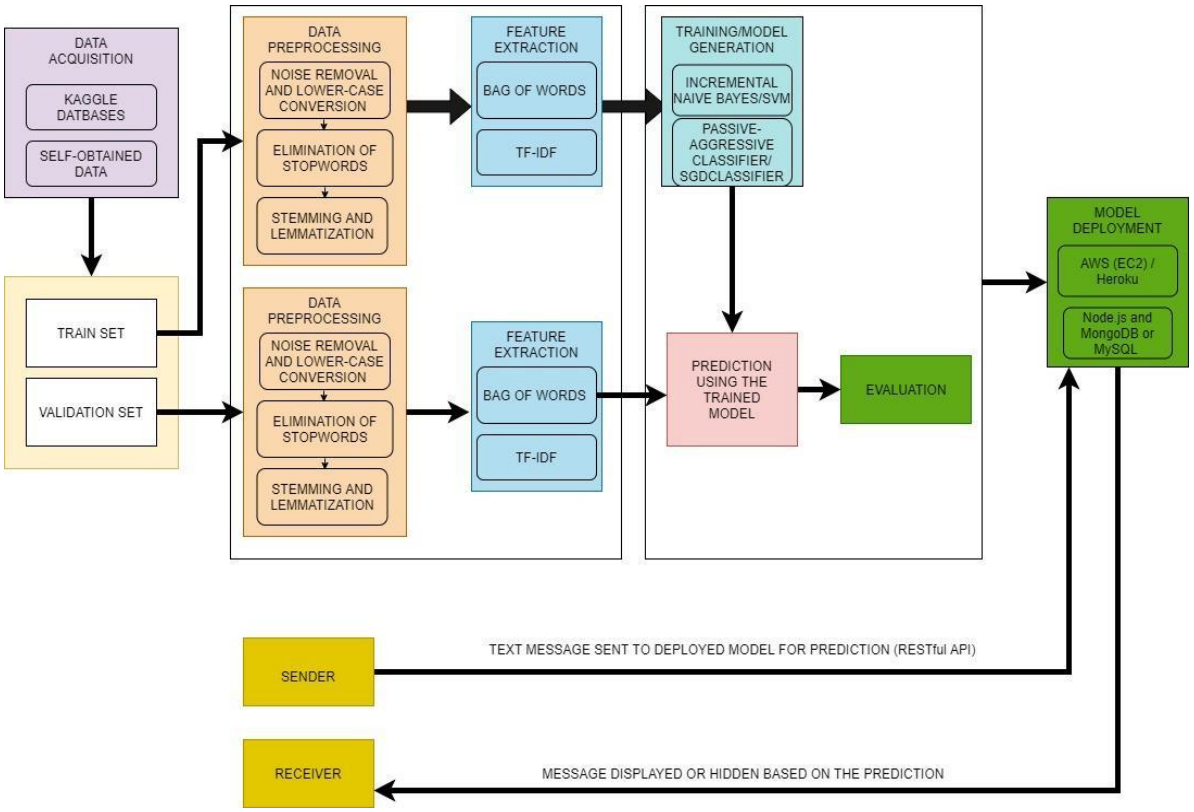

Figure 1. Proposed Architectural Diagram of the Experimental Setup

Dataset Preparation

The first step in carrying out the implementation is dataset preparation. The dataset can be prepared manually by collecting text messages from various sources and classifying them manually as distracting and non-distracting. Sometimes, more than one person could manually classify the messages and the majority vote would be considered for further processes. But this way of preparing a dataset has an issue of class imbalance. This happens when one class has more samples than the other class. For instance, if there are more samples of distracting messages than non-distracting messages or vice versa, that can be termed as the class imbalance problem. This can be solved by oversampling, undersampling, or SMOTE. Oversampling is done by duplicating the samples of the minority class. Undersampling is the opposite of oversampling. Undersampling is achieved by selecting a subset of samples of the majority class. SMOTE is a synthetic method of oversampling which involves interpolation between examples to get new examples. After the dataset is prepared, the data undergoes preprocessing.

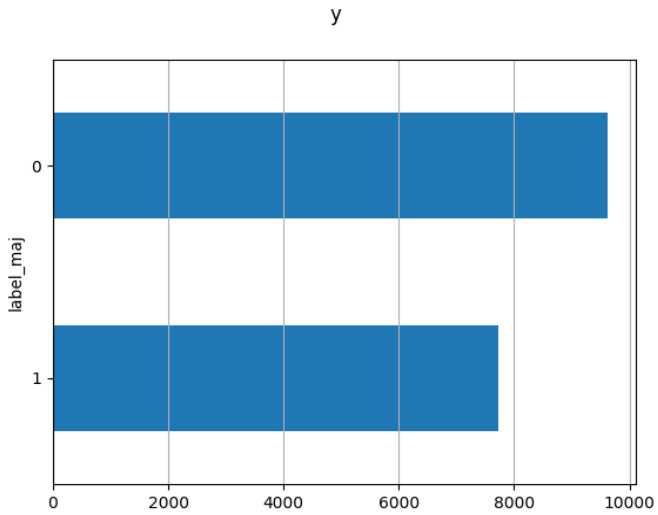

Figure 2. Comparison of the Number of Samples in the Two Classes (Non-Distracting [0] and Distracting [1])

The dataset for this work was collected as part of a previous work of the authors (Sait et al)). In this dataset, the two classes (non-distracting [0] and distracting [1]) consist of about 9629 and 7731 samples respectively.

Preprocessing

The first step towards preprocessing is translation or transliteration-translation of the text messages if they are not in English. Google's Cloud Translation API or Amazon Translate or equivalent on any other cloud platform may be used for this purpose. Conversion of any text message into English is necessary in order to bring regularity to the model. Moreover, there are many more libraries available for the English language than for most other languages.

The next step is to cleanse the data. Any unwanted symbol is first removed. That might include symbols such as -, :, ;, etc., because they do not affect our classification significantly. The next thing performed is to convert the data into lowercase as it was felt that ase does not matter when classifying distracting and non-distracting messages unlike in the case of sentimental analysis where case plays an important role. Then, removal of stop words was performed using Natural Language Toolkit (NLTK). Stop words refer to very commonly used, routine words that do not have any significant impact on text classification; these can be safely removed from the text. Lemmatization is performed next. It is used to find out the root of the words accurately using the lexicon of the language.

Feature Extraction

Feature extraction or word embedding can be done using Bag-of-Words (BoW), TF-IDF, Word2Vec, or by using deep learning techniques. This paper uses Bag of Words and TF-IDF.

Bag-of-Words is nothing but a representation of text in terms of occurrences of words, not considering the grammar. TF-IDF (term frequency - inverse document frequency) is a statistical measure showing the significance of a word in a document.

A bag-of-words (BoW) model is a method of extracting text features for use in machine learning applications involving text. The method is straightforward and versatile, it can be applied to a wide range of text features. BoW is a vector-based embedding with values representing the counts of words in the text. It's called a "bag" of words because all detail about the document's word order or composition is discarded. The model takes into account only the occurrences of the recognized words and not their location or context. The assumption is that documents with similar word counts are similar.

Sometimes, n-gram approaches are applied for feature extraction in which the vectors are created by considering n consecutive words rather than just a single word.

The use of word count has the disadvantage of causing frequently-used words to influence the document, but it doesn't provide as much "informational content" for the model like unique (technical) terms for that domain. One approach is to adjust the frequency of words depending on how often they appear in all texts, penalising words like "the" that occur very often. This is accomplished by using term frequency - inverse document frequency (TF-IDF). While the term frequency (TF) denotes the frequency of each term, inverse document frequency (IDF) reflects the rarity of the word in the document collection, giving greater weight to such words in the document collection. TF-IDF is the product of TF and IDF. In this work, TF-IDF features have been used to learn models.

Online Learning

Vectors that were produced as output during feature extraction are fed to classifier models which support online learning.

Online Learning can be defined as, "given a stream of data $s_1,s_2,s_3,s_4,...,s_t$, a sequence of models is generated ($h_1,h_2,h_3,h_4,...,h_t$). In this case, $s_i = (x_i, y_i) \in R^n \times \{1, 2,..., C\}$ and $h_i: R^n \times \{1, 2,...., C\}$ is a model function depending on h_{i-1} and the recent examples $s_1,s_2,s_3,...,s_p$ with 'p' being strictly limited".

Online learning algorithms are a subset of incremental learning algorithms that allow for endless/lifelong learning on a computer with limited resources.

The following issues confront incremental learning algorithms:
(1). The model must evolve progressively, i.e. it is built on the assumption that no full retraining is needed.
(2). Information that has already been learned has the risk of being forgotten.
(3). There is a restriction on the number of training instances p that can be stored.
Since there isn't an approach that works optimally in every case, an algorithm must be selected based on the task's preconditions. So far, a number of interesting incremental learning algorithms have been written, each with its own pros and cons.

This paper uses a linear SVM classifier coupled with SGD (Stochastic Gradient Descent) optimization. As with most other machine learning problems, an 80-20 split is done (the model is trained on 80% of the dataset and tested on the remaining 20%). linear SVM is being used because it works well with TF-IDF in practice. Other classifiers such as the Passive-Aggressive classifier have been used as well. The difference between online training and batch training is that in the former one can learn from one example at a time whereas batch training has to be done on the entire batch at once. The mathematical pseudo-code/algorithm for SGD is as follows:

$$\text{for i in range (m):}$$

$$\theta_j = \theta_j - \alpha * \nabla_{\theta_j} J(\theta; x^i; y^i)$$

where 'm' corresponds to the number of training data samples, 'α' is the learning rate, J(θ) is the cost function, (x^i, y^i) is a training data point, and 'θ_j' is a model weight or parameter. There can be multiple parameters which can be adjusted to minimize the loss. Each model parameter is adjusted by calculating the gradient or partial derivative of the cost function with respect to that parameter using one data point at a time. The learning rate is used to determine the pace of training or the step size.

Thus, this linear SVM model uses stochastic gradient descent (SGD) where the gradient of the loss is evaluated one sample at a time, and the model is revised along the way with a decreasing intensity schedule (aka learning rate). The partial fit approach in SGD allows for mini-batch (online/out-of-core) learning. The data should have a zero mean and unit variance for the optimal outcomes by using the default learning rate schedule.

The regularizer functions as a penalizer for the loss function. It uses either the squared Euclidean L2 norm or the absolute Euclidean L1 norm or a combination of the two to diminish model parameters to zero (Elastic Net). The regularizer helps in penalizing the weights and avoiding overfitting. For instance, L1 tries to do feature selection by attributing zero weights to insignificant input features and non-zero weights to significant features. To implement SGD for linear SVM, the following python code is used:

class sklearn.linear_model.SGDClassifier(loss='hinge')

The loss function is specified through the parameter "loss". By default, the value "hinge" is taken which fits a linear support vector machine. Some other possible options are "log", "perceptron", or a regression loss: "squared_loss", "huber", etc. The "log" loss models a logistic regression while the perceptron algorithm uses "Perceptron" as its linear loss.

Partial fit is an incremental fit performed on a single sample or a mini-batch (the whole dataset is not used to compute the gradient). To implement partial fit, the following python code is used:

partial_fit (X, y, classes=None, ...)

Here X is a subset of training data, y is a subset of target data.

Validation and Testing

K-fold Cross-Validation is used to validate models. In this, the training dataset is divided into K number of segments (in most cases, the number of segments is 5 or 10). Now, with every iteration, we keep one segment for validation and the remaining segments for training the model. Each iteration gives an accuracy (which are a1, a2,..., ak). The mean of a1, a2, …, ak is taken as the accuracy. The advantage of using this method of validation is that we can tune the parameters such as learning rate based on the accuracy of each fold. After a considerably good accuracy is achieved, the remaining 20% of the dataset is used for testing.

Results

Table 1 compares the results obtained from all models for both batch learning as well as online learning; 1926 examples were in the normal class and 1546 examples in the distracting class. In the case of batch learning an f1-score of over 0.89 was obtained for all the models, with best performance (f1-score) of 0.91 for linear SVM. On the other hand, for online learning an F1-score of over 0.86 was obtained for all models, with best performance of 0.89 for multinomial naive bayes.

Table 1

Accuracy, Precision, Recall and f1-score using batch and online learning using different classifier models

	Batch Learning		**Online Learning**	
Classifier Model	**Accuracy**	**f1-score (weighted average)**	**Accuracy**	**f1-score (weighted average)**
Multinomial Naive-Bayes	0.89	0.89	0.89	0.89
Perceptron	0.90	0.90	0.87	0.87
Passive-Aggressive	0.90	0.90	0.86	0.86
Linear SVM	0.91	0.91	0.86	0.86

The final outcome is a model which can classify incoming text messages into two classes (distracting and non-distracting). It works well with Hindi text messages and audio messages (Google's Speech-to-Text API is used to transcribe the audio message) as well. A model may be learnt for every user gradually as the user provides feedback on the filtering mechanism. Thus, for every user u: $u \in U$, where U is the set of all users, there exists a model m. The users can mark a non-distracting message as distracting or vice-versa and the model associated with that user gets updated accordingly.

Conclusion

In this work, it has been shown that online learning models can be used to classify messages into normal and distracting classes. A number of models have been experimented with, the best performance of which was an F1-score of 0.89 for multinomial Naive Bayes model. Online learning has been accomplished by using SGD to adjust models parameters. This work indicates that online models with good performance may be learnt for filtering distracting messages. This enables the learning of user-specific models; a user can tip-off an incorrect classification, and the feedback can be used to update online models. By making models user-specific, good classification performance can be achieved. Further, online models offer the benefit of adjusting to the drift

in the data by continuous update. The learning of optimal user-specific models for the case of filtering distracting messages is a subject of future work.

References

1. Almeida, T.A., Silva, T.P., Santos, I., & Hidalgo, J.M.G. (2016). Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering. *Knowledge-Based Systems*, *108*, 25-32.
2. Avrahami, D., & Hudson, S.E. (2006). Communication characteristics of instant messaging: effects and predictions of interpersonal relationships. *In Proceedings of the 20th anniversary conference on Computer supported cooperative work*, 505-514.
3. Bosaeed, S., Katib, I., & Mehmood, R. (2020). A Fog-Augmented Machine Learning based SMS Spam Detection and Classification System. *In Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, 325-330.
4. Dedeturk, B.K., & Akay, B. (2020). Spam filtering using a logistic regression model trained by an artificial bee colony algorithm. *Applied Soft Computing*, *91*, 106229.
5. Elnahrawy, E. (2002). Log-based chat room monitoring using text categorization: A comparative study. *In The International Conference on Information and Knowledge Sharing, US Virgin Islands.*
6. Gaurav, D., Tiwari, S.M., Goyal, A., Gandhi, N., & Abraham, A. (2020). Machine intelligence-based algorithms for spam filtering on document labeling. *Soft Computing*, *24*(13), 9625-9638.
7. Hegde, R., & Seema, S. (2017). Aspect based feature extraction and sentiment classification of review data sets using Incremental machine learning algorithm. *In Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, 122-125.
8. Laskov, P., Gehl, C., Krüger, S., Müller, K.R., Bennett, K.P., & Parrado-Hernández, E. (2006). Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research*, *7*(9), 1909–1936.
9. Ma, H., Fan, X., & Chen, J. (2008). An incremental Chinese text classification algorithm based on quick clustering. *In International Symposiums on Information Processing*, 308-312.
10. Özyurt, Ö., & Köse, C. (2010). Chat mining: Automatically determination of chat conversations' topic in Turkish text based chat mediums. *Expert Systems with Applications*, *37*(12), 8705-8710.
11. Sait, S.Y., Adak, R., Sharma, D., Prasad, A., Venkatesh, S.V., & Gaurav, A. Enhancing Mobile Instant Messaging Productivity by Filtering Distracting Messages. *Journal of Information Science*, *SAGE Journals.*
12. Shan, G., Xu, S., Yang, L., Jia, S., & Xiang, Y. (2020). Learn#: A Novel incremental learning method for text classification. *Expert Systems with Applications*, *147*, 113198.
13. Singh, A.K., Bhushan, S., & Vij, S. (2019). Filtering spam messages and mails using fuzzy C means algorithm. *In 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, 1-5.
14. Tseng, C.Y., & Chen, M.S. (2009). Incremental SVM model for spam detection on dynamic email social networks. *In International Conference on Computational Science and Engineering*, *4*, 128-135.