A Time-Sensitive Spam Filter Algorithm Dealing with Concept-drift

Jiaolong Liu

College of Computer Science and Technology, Zhejiang University

Hangzhou, Zhejiang, China

jl_liu@zju.edu.cn

Keywords: Data Stream; Classification; Concept-drift

Abstract. Spam, under a variety of shapes and forms, continues to inflict increased damage. Varying machine learning techniques have played an important role in spam filtering field in condition that ample training data is available to build a robust classifier. These methods include Decision Tree, Support Vector Machine (SVM), etc. However, spam filtering is a particularly challenging task as the data distribution and concept being learned changes over time. More seriously, data stream classification poses many challenges to the data mining community. In this paper, we proposed a time-sensitive spam filter algorithm dealing with concept-drift (TSSFA), which is appropriate for such dynamically changing contexts. We evaluate its performance on the TREC public corpus, and showed satisfactory result.

Introduction

In recent years, the increasing use of e-mail has led to the emergence and further escalation of problems caused by unsolicited bulk e-mail messages, commonly referred to as Spam or Junk. Evolving from a minor nuisance to a major concern, given the high circulating volume and offensive content of some of these messages, spam is beginning to diminish the reliability of e-mail.

Knowledge discovery on streaming data is a research topic of growing interest. Reference [2] shows the fundamental problem we need to solve is the following: given an infinite amount of continuous measurements, how do we model them in order to capture time-evolving trends and patterns in the stream, and make time-critical predictions?

Huge data volume and drifting concepts are very common in the Data Mining community. One of the goals of traditional Data Mining algorithms is to learn models from large databases with bounded-memory. It has been achieved by several classification methods. However, to be frank, these algorithms require multiple scans of the training data, which makes them inappropriate in the streaming environment. For that in such environment instances come at a higher rate than they can be repeatedly analysed.

A difficult problem with learning in many real-world domains is that the concept of interest may depend on some hidden context, not given explicitly in the form of predictive features. Typical examples include weather prediction rules that may vary radically with the season or the patterns of customer's buying preferences that may change, depending on the current day of the week, availability of alternatives, inflation rate, etc. Often the cause of change is hidden, not known in advance, making the learning task more complicated. Changes in the hidden context can induce changes in the target concept, which is generally known as concept-drift, as showed in [8]. An effective learner should be able to track such changes and to quickly adapt to them.

As we've known many methods before were proposed to deal with stream data, especially concept-drift. However these methods mainly concentrate upon whether there are concept-drifts or not, why they happen and how to deal with them. In other words, they focus only on the statistical index like precision and recall. But in fact, they cannot tell where and when exactly the concept-drift appears. This is what we want to do. In this work, we proposed a superior technique for both detecting concept-drifts and dealing with them. Our framework can also tell where and when exactly the concept-drift appears.

In this paper, we claim two main contributions in concept-drift detection and dealing in data stream environment. Firstly we proposed a time-sensitive spam filter algorithm dealing with concept-drift, namely, TSSFA. Given each word a timestamp, through TSSFA we get the corresponding timestamp, thus we can indeed know when concept-drift occurs. Secondly, we've done the evaluation and made a comparison between TSSFA and some state-of-art methods. It shows that our approach really works and has a good performance.

The rest of the paper is organized as follows: Section 2 reviews the main related work on spam filtering and concept-drift problem. Section 3 describes the algorithm's architecture in our approach and its workflow, followed in the next sections by the two components' description: feature extraction and the classification module. Section 5 presents the experimental work performed and the results obtained. The last part summarizes this article's conclusions and presents some further work that might be pursued.

Related Work

Filtering is a popular solution to the problem of spam. It can be defined as automatic classification of messages into spam and legitimate mail. As we've known that many researchers have done a lot of work on spam detection field. Such methods can be divided into two parts: rule base methods and statistics based ones. For rule base methods, there are Ripper, Decision Tree, Boosting, Rough Sets, etc. And for statistics base methods, there are kNN, Bayes, Rocchio, Support Vector Machine, Winnow, etc. However, when emails come at a high-speed and in a huge amount, namely in stream environment, these works tend to be insufficient. Especially when there are concept-drifts occur.

The most common concept drift technique is based on instance selection. It involves generalizing from a window that moves over recently arrived instances and uses the learnt concepts for prediction in the immediate future. Examples of window-based algorithms include the FLORA family of algorithms as in [8], FRANN and Time- Windowed Forgetting.

Incremental data mining methods are another option for mining data streams. These methods continuously revise and refine a model by incorporating new data as they arrive. Domingos et al introduce an efficient incremental decision tree algorithm called VFDT, as in [13]. They use Hoeffding bounds to guarantee that the output model of VFDT is asymptotically nearly identical to that of a batch learner. Many modifications of this work were made for streams. The first of these is due to Hulten, Spencer, and Domingos, who adapted the learning method to create the Concept-Adapting Very Fast Decision Tree (CVFDT) as in ([14,18]).

Ensemble methods (e.g. [1,3,7,18]) such as bagging, AdaBoost, random forest, random subspaces, etc. are popular used in the data mining community. One of the earliest ensemble based approaches for concept drift is Streaming Ensemble Algorithm (SEA) due to Street and Kim, as in ([18,19]). Reference [3] shows a particularly important ensemble method, namely Dynamic Weighted Majority (DWM). Now DMW has become a baseline algorithm for almost all experiments in this field.

There are some other ways of dealing concept-drift, such as OLINDDA (showed in [20]), which is a cluster-based approach. OLINDDA considers the problem of learning with a one-class classification. It uses the cluster radius to detect normal data and separates concept drifts and novel classes. The main shortcoming of OLINDDA is that decision is made based on the whole cluster.

TSSFA Framework

This section presents the TSSFA, a time-sensitive spam filter algorithm. It is divided into two main components: one that collects data and extracts the features of each email, and the other one classifies the instances and interprets the results. As showed in Fig.1, each component has some processes that have to be run in this particular order.

First, an email collection is done in order to collect data to be analysed. The next step is a mail analysis process: the message is analysed and several attributes are saved. An email message is

usually with HTML mark-up and graphical elements. Therefore before classification we have to do participle, stop-word removal, feature extraction, etc.

For the second part classification model the corpora are divided into training sets and testing sets. In the real process of spamming, the email corpus is dynamic so the size and the distribution of the data set fluctuate. Thus, we use slide windows techniques and divide them into different windows (or chucks). Next we apply classifiers on these windows, including traditional classification methods, as well as special methods for dealing with concept-drift. Timestamps of certain words can be judged through training phase, and then been used to the latter classification. The last part of the system is interpret the results and informs the user.

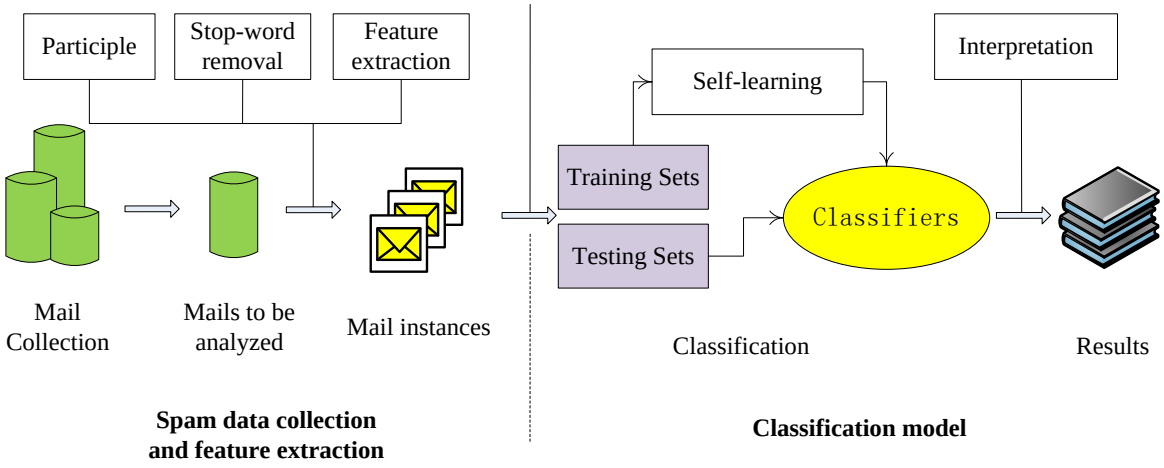

Fig. 1 Architecture of the TSSFA approach

Spam Data Collection and Feature Extraction

Experiment Setup. The information contained in an email is divided into header and body. Before a classifier can use the available information, appropriate pre-processing steps are required. These steps involved in the extraction of data from a message can be grouped into: (1) Tokenization; (2) Lemmatization; (3) Stop-word removal; (4) Representation.

Feature Extraction. For each message analysis method, its designer must choose a way of doing feature extraction. It is to decide which part of the messages is relevant for the analysis. The simplest way of doing feature extraction is the 'bag of words' model, which represents the message as an unstructured set of tokens. A few researchers may use SCPCD to extract the whole phrase, or use a higher-level statistical language model, such as bigram (bag of bigram), or trigram (bag of trigram). Those can keep the word order. Here in this paper, we just use bag of words for simplification.

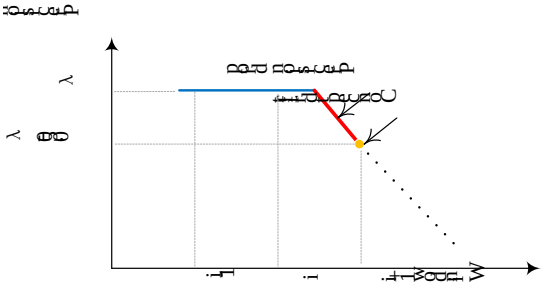

Fig.2 Concept-drift detection

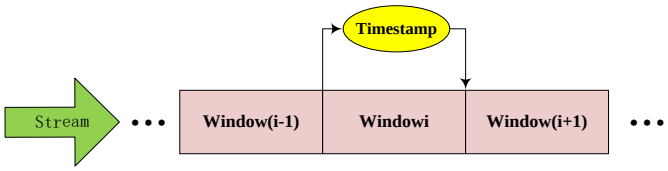

Fig.3 Seek timestamp

For each email *em*, we do pre-processing, as described in the previous section. After that we can get a words list. Next, we calculate word frequency for each word, and then using tf-idf we choose 25 highest of them. Thus to make a 25 dimensional vector $V_{em} = \{w_1, w_2, w_3 \ldots w_{25}\}$, in this way, the

email *em* can be represented by eigenvector V_{em}. For the case one email contains less than 25 words, we just make a completion with zero vectors.

Classification Model. In this part, we mainly do experiments in two aspects, namely find concept-drift and deal with it. First step, we give the criteria for detecting concept-drift, which is to find the time when concept-drift appears. Like most of the algorithms, we also use the index accuracy to identify it. Since SVM is a good classifier in text categorization, we use SVM as our base classifier. As the data streams continuously come, we use a certain classifier to classify the instances in each window separately. For each window, there is a classification precision (or accuracy). Usually the precision maintains at a relatively stable level, until concept-drift occurs. When concept-drift occurs, the classification precision will definitely drop. Suppose the mean precision of the classifier is Pre_m, the current precision of the classifier for the current window is Pre_{cur} and the error precision ε is:

$$\varepsilon = Pre_m - Pre_{cur}. \quad (1)$$

If $\varepsilon > 0.05$, which means there is a precision drop greater than 5%, we consider there to be concept-drift, otherwise the concept is stable. It is showed in Fig.2. Usually, in the window *i-1* and the other windows ahead, the classification precision maintains at a relatively stable level λ. But in window *i*, there is a classification precision drop, all the way to λ -0.05. Thus we can say that there occurs concept-drift.

As soon as there appears concept-drift, the algorithm starts to deal with it. In text categorization area, especially spam filtering area, the changing concept of a certain word may lead to totally different results. Our goal is to find which word triggers the drift.

For every email *em* in window *i*, we test on their eigenvector V_{em}. Some words may have different meanings in different time or occasion, and may affect the classification result. Therefore in order to judge whether the concept-drift is caused by a certain word w_i, we remove w_i from V_{em} first, and then we can get a new vector V_{em}' without w_i. So do the other emails that contain word w_i. Thus, with V_{em}' we make a reclassification on the original corpora.

If the classification precision rises again, it is sure that the word w_i lowers the average. Then we get the instantaneous time t_i of window *i*, and blind it to word w_i. We call it the timestamp of word w_i, and labeled as $ts(w_i)$. Within this way, each unstable word has its timestamp. In the next data streams, the classifiers will classify the data streams with a new meaning of these words. This procedure is briefly showed in Fig.3.

Experiment Work

This section presents our evaluation of TSSFA. The evaluations were offline evaluations using emails collected over an extended period of time.

Data Set. The evaluation was performed on the 2005 TREC Public Spam Corpus, which contains 92,189 email messages, with a chronological index labelling each as spam or ham (i.e. legitimate email). 52,790 messages are labelled spam while 39,399 are labelled ham. The corpus was created for the TREC Spam Evaluation Track using an iterative adjudication process.

From the corpus we choose a part of the messages (i.e. 10000 instances) for our experiments, from which the necessary amount was selected. Among them we manually add 10% noisy data, and some of them may lead to concept-drift.

Evaluation Metrics. For a binary classification problem, usually four situations would happen. If an instance is positive and is also predicted to be positive, we call it True positive (TP). If an instance is negative and is predicted to be positive, we call it False positive (FP). Correspondingly, if an instance is negative and is also predicted to be negative, we call it True negative (TN). And if an instance is positive and is predicted to be negative, we call it False negative (FN).

Previous work on spam filtering uses a variety of measures to report performance. The most common performance metrics is precision, here we use precision to evaluate the methods. It reflects each classifier's ability to determine the whole sample. The precision can be calculated as follows:

$$\text{Precision} = (TP + TN) / (TP + FN + FP + TN) \quad (2)$$

Comparison between TSSFA and other Classifiers. Traditional machine learning methods, such as Naïve Bayes method, Decision Trees, SVM, etc, appear to be excellent techniques for spam filtering. Therefore we've made a comparison between TSSFA and such traditional classifiers. We set the window size to be 100. For that we have 10,000 instances, so we get 100 different precisions. Because of paper space limited and in order to see it clearly, we choose 20 of them. The results are summarized and showed in Fig.4:

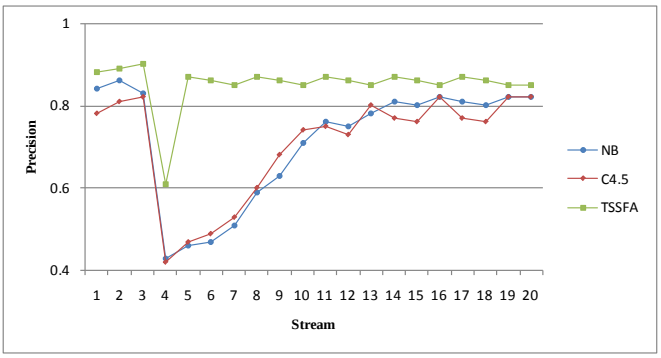

Fig.4 TSSFA VS traditional methods

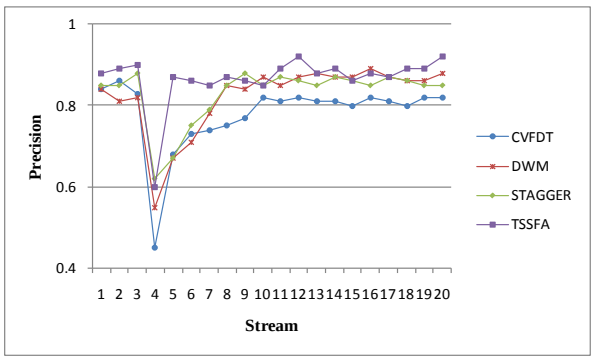

Fig.5 TSSFA VS special methods

As showed in the figure, originally (the first three chunks) the precision of these classifiers maintain at a relatively stable level. Their mean precisions are about 80%, more or less.

But in the 4th chunk, there appears unstable concept, mostly due to the changing meaning of some words. For this phenomenon, the classification precision drops suddenly in the current chunk. Facing this, the traditional methods do not do anything until the disappearance of those words, after a long time (sometimes even more than ten chunks as showed in the figure). However, when putting into this situation, TSSFA immediately discovers when and where concept-drifts happen, and deals with them as soon as possible. So the precision of TSSFA can rise again (in the next chunk, i.e. 5th chunk) to its original level quickly.

Besides traditional machine learning methods, there are also many other algorithms special for dealing with concept-drifts. These methods include CVFDT, SEA, STAGGER, etc. Here we've also made a comparison between TSSFA and these classifiers. The results are summarized and showed in Fig.5.

This group of experiments is similar to the former ones until the 5th chunk. Originally they maintain at a relatively stable level, and fall in the 4th one. All these methods can deal with concept-drifts, however they do not know when and where exactly concept-drifts happen, and they also need some time to adapt to that (maybe about three chunks as showed in the figure). That means, TSSFA is a faster one. In this aspect, our algorithm outperforms the existing methods.

Conclusion

In this paper we proposed a novel approach for spam filtering. Messages are classified with the TSSFA algorithm, which uses SVM as its base classifier. The TSSFA focuses not only on the statistical index precision, but also on some certain words, so that it can clearly know where and when exactly concept-drift appears.

We compared the algorithm with a great deal of other classifiers and showed great performances. Further development would be to add more foreign corpora to the classifier, like Wikipedia,

WorldNet, etc. In this way we can get the outer timestamp of a certain word more easily. Also adding more users' feedback would be another direction for further possible study.

References

[1] Wang H, Fan W, Yu P S, et al. Mining Concept-Drifting Data Streams Using Ensemble Classifiers[J]. KDD, 2003:226-235.

[2] Fan W, Wang H, Yu P S. System and method for mining time-changing data streams: US, US7565369 B2[P]. 2009.

[3] Kolter J Z, Maloof M. Dynamic weighted majority: a new ensemble method for tracking concept drift[C]// Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. IEEE, 2003:123-130.

[4] Delany S J, Cunningham P. An Analysis of Case-Base Editing in a Spam Filtering System.[J]. Lecture Notes in Computer Science, 2004, 3155:128-141.

[5] Delany S J, Cunningham P, Tsymbal A, Coyle L. A case-based technique for tracking concept drift in spam filtering. Knowledge-Based Systems，2005，18(4–5):187–195.

[6] Klinkenberg R. Learning drifting concepts: Example selection vs. example weighting.[J]. Intelligent Data Analysis, 2004, 8:281-300.

[7] Bifet A, Holmes G, Pfahringer B, et al. New Ensemble Methods For Evolving Data Streams[C]// In 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM. 2009:139--148.

[8] Widmer G, Kubat M. Learning in the presence of concept drift and hidden contexts[J]. Machine Learning, 1996, 23(1):69-101.

[9] Masud M M, Chen Q, Khan L, et al. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams[J]. Knowledge & Data Engineering IEEE Transactions on, 2013, 25(7):1484-1497.

[10] Lu N, Zhang G, Lu J. Concept drift detection via competence models [J]. Artificial Intelligence, 2014, 209(2):11–28.

[11] Gama J, Žliobaitė I, Bifet A, et al. A survey on concept drift adaptation [J]. Acm Computing Surveys, 2014, 46(4).

[12] Hofer V, Krempl G. Drift mining in data: A framework for addressing drift in classification [J]. Computational Statistics & Data Analysis, 2013, 57(1):377-391

[13] Hulten G, Spencer L, Domingos P. Mining time-changing data streams[J]. Proc.of Acm Sigkdd Intl Conf.on Knowledge Discovery & Data Mining, 2001:97--106.

[14] Domingos P, Hulten G. Mining High-Speed Data Streams[C]// Sixth Acm Sigkdd International Conference on Knowledge Discovery & Data Mining. 2002:71-80.

[15] Gonçalves P M, Santos S G T D C, Barros R S M, et al. A comparative study on concept drift detectors[J]. Expert Systems with Applications, 2014, 41(18):8144–8156.

[16] Gehrke J, Ganti V, Ramakrishnan R, et al. BOAT-optimistic decision tree construction[J]. Proc Acm Sigmod, 1999, 28(2):2469-2474.

[17] Akila V, Zayaraz G. A Brief Survey on Concept Drift[M]// Intelligent Computing, Communication and Devices. Springer India, 2015:293-302.

[18] Hoens T R, Polikar R, Chawla N V. Learning from streaming data with concept drift and imbalance: an overview[J]. Progress in Artificial Intelligence, 2012, 1(1):89-101.

[19] Kim Y. A streaming ensemble algorithm (SEA) for large-scale classification[C]// Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001:377-382.

[20] Spinosa E J, Gama J. OLINDDA: a cluster-based approach for detecting novelty and concept drift in data streams[J]. In Proceedings of the Acm Symposium on Applied Computing, 2007:448-452.