Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 225 (2023) 3001–3010

Procedia Computer Science

www.elsevier.com/locate/procedia

27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2023)

TCR-M: A Topic Change Recognition-based Method for Data Stream Learning

Kun Wang[a], Jie Lu[a,*], Anjin Liu[a], Guangquan Zhang[a]

[a]University of Technology Sydney, Australia Artificial Intelligence Institute (AAII), Broadway, Sydney 2007, Australia

Abstract

Data stream learning has received more and more attention in recent years, change tracking and real-time prediction of data streams under uncertainty have been highly focused. With the development of the information age, more and more different types of data streams have been generated, bringing challenges to the research in this field. Among them, text data streams, as one of the categories, also need to be mined and predicted in real-time. This paper addresses this problem by proposing a topic change recognition-based method (TCR-M) for data stream learning, thus helping support text data stream learning. We first propose a topic change recognition process that extracts the topics of the text data stream at each time point, tracks and determines the severity of the topic change, and locates the time points when significant changes occur. Next, an ensemble learning model is constructed and a separate base learner is simultaneously trained to correct the prediction results of the ensemble learning model, which is updated based on the topic change recognition results. To verify the effectiveness of the method, a number of text data streams are collected for evaluation, then outputting the topic change recognition results and prediction results. By comparing it with benchmark methods, the proposed method shows its efficiency. In future research, further improvements are needed for learning and application.

Keywords: Data stream; Topic change; Concept drift; Ensemble learning

1. Introduction

In recent years, data stream learning has received increasing attention, many research works aimed to deal with data stream for real-time prediction. Data streams are common in our life, with traditional data streams composed of structured data. However, with the development of the information age, data streams consisting of unstructured data such as text and images have gradually emerged. Many previous methods are designed based on traditional data streams to help the machine learning model to enhance its adaptability and effectiveness by identifying and handling

* Corresponding author
E-mail address: Jie.Lu@uts.edu.au

1877-0509
10.1016/j.procs.2023.10.293

the changes in the data stream in time. However, these methods can not be applied to text data stream learning directly and completely. In addition, the identification of data changes and the efficiency of learning tasks require further extension and improvement. Furthermore, there are not many relevant studies for text data stream learning currently, so it is necessary to think about how to identify and handle the text data stream changes in time to help support machine learning tasks.

Topic change identification is a common approach in some recent studies when dealing with text data streams. Previous related studies have identified topic changes by extracting topics and calculating the distance. Although this strategy can effectively detect the change that occurs, the severity of change also needs to be considered since not every change in topic requires adjusting the learning model. Therefore, this paper considers a method that can simply calculate the severity of topic change and find the time point of significant topic change, and based on this, help adjust the machine learning model to further support text data stream learning. The contributions of this paper are as follows.

- A topic change recognition process is proposed to help identify the change occurrence and severity in the text data stream. The topics are extracted, and the probability of each topic is calculated at each time point.
- A data stream learning process triggered by topic change recognition results is given. The learning model will self-update when a change is identified.
- An experiment on real-world text data streams is evaluated and discussed. The proposed method shows its efficiency by comparing it with benchmark methods.

The rest of this paper is organized as follows. The related work is summarized and discussed in Section 2. Section 3 gives the problem statement and introduces the proposed method. The experiment settings and results are described and discussed in Section 4, and the conclusion and future study are given in Section 5.

2. Related Work

This section summarizes the previous studies in data stream learning, and then gives related research works about topic change learning in dealing with the text data stream.

2.1. Data Stream Learning

Previous studies address certain works on data stream learning. Concept drift in the data stream will affect the learning efficiency that the machine learning model may not adapt to the change in time, leading to learning decay and performance degradation. Concept drift is known as the data distribution changes with time goes on [21, 9, 14, 17, 28], denoted as

$$P_t(y|x) \neq P_{t+1}(y|x) \tag{1}$$

where $P(y|x)$ expresses the data distribution. In order to reduce the learning decay caused by concept drift, it is necessary to identify and handle concept drift in time to help avoid performance degradation of the learning model. Nowadays, aiming to handle concept drift efficiently, many methods have been proposed and applied. The basic procedures of the previously proposed methods to deal with concept drift have three steps: detection, understanding, and adaptation. For drift detection, error rate is a commonly used method to identify concept drift. For example, methods like Drift Detection Method (DDM) [8], Early Drift Detection Method (EDDM) [1], and ADaptive WINdow (ADWIN) [2] are designed based on error rate. The DDM method can identify the warning level and drift level. Recently, there are some new methods for drift detection are proposed. For example, aiming to measure the changes in the data stream, a local drift degree (LDD) measurement method has been developed by research [19], which focuses on monitoring the regional density changes. Besides, research [18] develops a method to detect drift by the equal intensity k-means space partitioning.

For drift adaptation, many methods are designed to help the machine learning model adapt to concept drift in time. Ensemble learning methods are commonly used because of their excellent learning performance. Bagging [5] and Boosting [23, 24, 7] are two popular ensemble learning models. Many concept drift adaptation methods are designed mainly based on these two ensemble learning models. For example, the Learn++.NSE [6] is a classical method for concept drift learning, this method uses incremental learning to support drift adaptation. LeverageBagging [3] is another ensemble model for concept drift learning, it adds more randomization to the input, and output of the classifiers, thus helping the learning model adapt to the concept drift. Besides, ARF [10] method is designed based on the random forest model. This method is widely used in data stream learning due to its ideal performance. In addition, Boosting method is also applied and optimized for data stream learning. Online AdaC2 ensemble classifier [25] and Online Boosting [25] are designed based on the Boosting model for data stream learning.

However, most of these methods proposed in the past are designed based on traditional data stream data and are not well suited for textual data stream data. With the development of the information age, more and more text data stream data are generated, and this kind of data also changes uncertainly over time. Due to the difference in data types, it is clearly not possible to apply the previously proposed data stream learning methods directly. For this reason, it is very necessary to think about how to perform adaptive learning of text data streams.

2.2. Topic Change Learning

Text data stream learning has been gaining attention [11, 20, 13]. Related researchers aim to address real-time mining of knowledge information from text data, or give cluster analysis of text data. However, in an uncertain environment, text data streams may change uncertainly over time, which affects the adaptability of the machine learning model, leading to a degradation of learning performance. To address this problem, some related research works aim to handle changes in text data streams in time to help support model learning. Aiming to handle the text data stream well, some researchers focus on recognizing the change of topic for designing the learning method. For example, topic model inference methods have been applied to streaming document collections [26]. Besides, to deal with the short text data stream, research works [15, 16] propose topic drift detection and learning methods. The detection methods in these studies are based on the distance calculation results, the Cosine distance method is applied to measure the difference. Research [30] designs a three-layer method to detect concept drift in the text data stream.

From the perspective of the application, research works [27, 12] combines data stream learning and user interest drift recognition to support learning and prediction. Research [29] combines the methods of fuzzy learning and concept drift learning to deal with user interest drift. Text data stream mining has a wide range of real-life applications, and effective real-time learning and prediction can be useful in different fields. However, relatively few works have been done on text data streams. Although some studies have done some work on text topic change discovery, the severity of topic change needs to be considered and used to help model learning.

Therefore, it is necessary to consider methods to help support the text data stream learning task. This paper aims at the task of learning text data streams, starting from identifying the severity of the topic change, and thus designing methods that can further help improve the adaptability of machine learning models and reduce the impact of uncertainty changes.

3. Methodology

This section gives a detailed problem statement, and then develops related learning methods to deal with it. The learning methods include two processes: the topic change recognition process and the data stream learning process.

3.1. Problem Statement

Given a data stream $\{x_i, y_i\}_{i=1}^{n}$ at time t, where x_i is the text data instance, y_i is the label, and a machine learning model $F(x)$ is trained. Under the influence of uncertainty, the topic β may change at the next time step, leading to a prediction decay of the machine learning model. To help the model handle the topic change and maintain performance, it should consider how to recognize topic change and update the model to better deal with the data stream. So, the

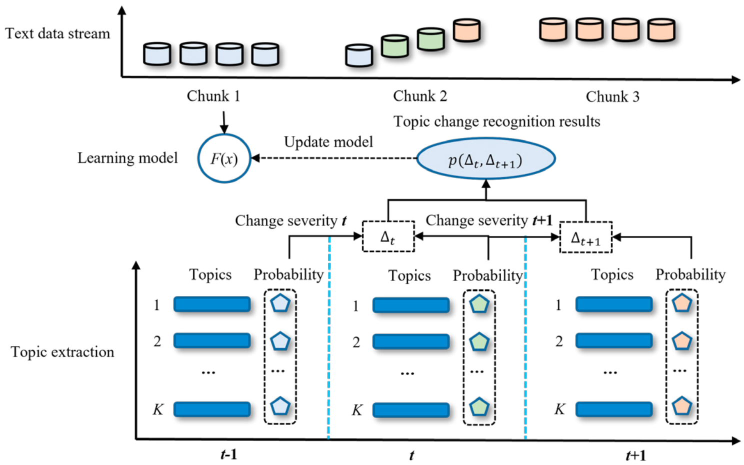


Fig. 1. A illustration of the learning process of the topic change recognition-based method for text data stream learning.

problem is how to find a method to reduce the loss caused by the changeable data, denote as

$$F(x) = \arg\min_{F(x)} L(y, F(x)) \tag{2}$$

where L is the loss function of the machine learning model. Aiming at this target, we propose a corresponding learning method in the next section.

3.2. Topic Change Recognition Process

As mentioned, the prediction performance of the learning model may be affected when the topic changes under uncertainty, so, it is needed to recognize the topic change in the data stream. First, we use the LDA model [4] to help extract topics from data instances. Here, assuming there are K topics extracted, and the probability of each topic P_{β_k} is calculated, where k is the topic index. Therefore, the probability of each topic can be expressed as

$$\{P_\beta\}_{k=1}^{K} = \{P_{\beta_1}, \cdots, P_{\beta_K}\} \tag{3}$$

Since the probability of each topic is different, and it may also change over time. Therefore, to track the topic change, we observe and record the change in the probability of topics. For each topic, the change of its probability in the timestamp can clearly reflect the topic change situation. Here, we find and locate the topic with the maximum probability, denoted as

$$P_{\beta_{max}} = \arg\max\{P_\beta\}_{k=1}^{K} \tag{4}$$

At the next time point, the probability of this topic may increase or decrease in different severity. We not only need to track the change but also need to focus on its change severity. Because some topic change with a relatively small severity may not affect the model learning performance obviously. But some topic changes with a significantly high severity may degrade the model performance. Therefore, we should then recognize the topic change severity. This process is also based on the probability of each topic. As we mentioned, there are K topics extracted, and the probability of each of them is calculated. So, we simply calculate the topic change severity of two consecutive time steps by

$$\Delta_t = |\{P_{t,\beta}\}_{k=1}^{K} - \{P_{t-1,\beta}\}_{k=1}^{K}|, \quad \Delta_{t+1} = |\{P_{t+1,\beta}\}_{k=1}^{K} - \{P_{t,\beta}\}_{k=1}^{K}| \tag{5}$$

Thus, the topic change severity Δ_t and Δ_{t+1} at time t and time $t+1$ can be got. Each of them is an array that contains K elements. And the Δ can be expressed as

$$\Delta_t = \{\Delta_{\beta_k,t}\}_{k=1}^{K}, \quad \Delta_{t+1} = \{\Delta_{\beta_k,t+1}\}_{k=1}^{K} \tag{6}$$

To identify whether the topic change is significant, we use the statistical test to calculate the significant level of the topic change severity of two consecutive time points. Here we use p as the p-value, and set 0.1 as the significant level, the p-value of topic change severity of two consecutive time point can be expressed as follow

$$p(\Delta_t, \Delta_{t+1}) < 0.1 \tag{7}$$

If $p(\Delta_t, \Delta_{t+1}) < 0.1$, it can be reflected that there is a significant difference between Δ_t and Δ_{t+1}, vice vera. Therefore, it should be paid more attention when topics occur significantly change, trying to update the learning process to avoid performance degradation.

3.3. Topic Change Recognition-based Data Stream Learning Process

Based on the results of the topic change recognition process, we try to update the machine learning model in time to avoid performance degradation. In previous research, incremental learning and retraining are two commonly used strategies for model updates. In this paper, we simply choose the retraining strategy. And, we use the ensemble learning model for learning and updating. Because the ensemble learning model is commonly used in the field of data stream learning due to its excellent performance. Here, we first learn a Bagging model [5] $F(x)$ with K base learners, and retrain it at each time point. The model can be denoted as

$$F_K(x) = sign \sum_{k=1}^{K} f_{\beta_k} \tag{8}$$

The number of base learners is set the same as the number of extracted topics. New learners also can be added to this Bagging model, but here we only simply keep a small number of base learners to reduce time-consuming. However, it should be noted that there are still errors in the prediction results. So, to further reduce the error, we separately train a base learner f to help correct the model results. This base learner is independent and updated based on the topic change recognition results. Unlike the ensemble learning model, which is retrained and updated at every moment, this learner is separately retrained only when the data changes. Finally, we compare the results of the ensemble model and separate base learner, then choose and keep a better result, this process can be denoted as

$$l_{min} = \min\left(l(y, f), l(y, F_K(x))\right) \tag{9}$$

Algorithm 1: Topic Change Recognition-based Method for Data Stream Learning

Input : Data chunks D of text stream data.
Output: Model prediction results.
Begin
Extract K topics from chunk 1; When $t = 1$, train a Bagging model F_K with K base learners on chunk 1;
Train a single decision tree model f with K base learners on chunk 1;
for $t = 2$ **to** T **do**
    Test models F_K and f, and then get better prediction results;
    Extract K topics from chunk t by Eq. (3);
    Get the maximum probability by Eq. (4);
    Calculate the topic change severity by Eq. (5);
    Calculate the p value by Eq. (7);
    if $p < 0.1$ **then**
        Retrain and update the model f on chunk t;
    end
    Retrain F_k on chunk t.
end
End

This can help reduce the error to a certain extent. The learning process is shown in Fig. 1. This method simply applies a retraining strategy to update the model and corrects the model's predictions based on the results identified as having changed. The method is also only a preliminary attempt to identify and predict the learning of changes in the text data stream, and further improvements are needed. The procedure of the proposed method is listed in Algorithm 1. In the next section, we conduct the experimental analysis to verify the effectiveness of the proposed method.

4. Experiment

This section outlines the description of experiment procedures, datasets information, and benchmark methods. And detailed discussion and analysis of the experiment results are also given.

4.1. Experiment Setting

In the experiment, we aim to test and evaluate the proposed method, trying to track and recognize the change of topics at each time point, then helping update the machine learning model. The detailed experiment procedures are given as follows.

Step 1: We extract topics of the text data stream at each time point, then calculate and track the change of the probability of each topic, locating the topic with maximum probability timely.

Step 2: We calculate the changing severity of the topic in the text data stream between two consecutive time points, then use the statistical test to find out the topic with significant change.

Step 3: We build a Bagging model and a separate base learner to deal with the text data stream, the model is updated based on the results of topic change recognition. In addition, we use this updated model for learning and outputting the prediction results.

To evaluate the prediction results, the accuracy and F1-score of the proposed method and benchmark methods are calculated and compared.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}, \quad F1 = \frac{2 \times \frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} \tag{10}$$

4.2. Datasets

The datasets chosen to be tested in the experiment is online user reviews collected from e-commerce websites. A small amount of review text data was collected over a continuous period of time, within the allowed range, on a selection of items that people frequently purchase on the Amazon website. The user scores corresponding to each data instance were also collected and used as labels for the data. The data were organized and pre-processed to eliminate some invalid reviews, and initially organized into a continuous stream of data to help the model conduct test experiments. It should be noted that the topics in each dataset are first extracted by the LDA method, which groups and summarizes the topics by calculating the probability. In this experiment, we initially extract 10 topics from datasets at each time point and only focus on tracking the probability changes of each of them. The detailed description of datasets is shown in Table 1.

Table 1. Datasets

Datasets	Time Period	Sample	Class
Monitor	2017.07-2021.12	3990	2
Phone shell	2020.10-2021.11	2124	2
Wet wipes	2014.06-2021.11	1571	2
T-shirt	2018.11-2021.12	4990	2
VC tablets	2005.10-2021.12	3349	2
Noodles	2014.10-2021.12	3420	2

4.3. Benchmark Methods

To evaluate the performance of the proposed method, we test it on datasets and compare its results with several benchmark methods. The benchmark methods are described as follows.

- **Baseline:** is the method that initially trains a Bagging model with a certain number of base learners, and we retrain it at each time point.
- **TCR-M:** A Bagging model is built and retrained at each time point, and a learner is separately learned to help modify the prediction results of the model.
- **TCR-M(retrain):** is the method which contains a Bagging model and a separate base learner, the base learner will be retrained based on the results of topic change recognition.
- **Learn++.NSE [6]:** is a classical ensemble learning method for data stream learning, and the incremental learning strategy is applied.
- **LeverageBagging[3]:** is a Bagging-based data stream learning method that adds randomization to the input and output of classifiers.

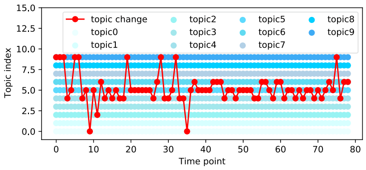
The experiment is implemented by Python 3.7, and the computational environment is: Red Hat Enterprise Linux Workstation release 7.9 (Maipo), Intel(R) Xeon(R) Gold 6238R CPU @ 2.20GHz. The Learn++.NSE [6] and LeverageBagging[3] were implemented by the ScikitMultiflow meta module [22], with default parameter settings. In this experiment, we have set the chunk size to 50, aligning it with the chunk sizes used in all benchmark methods. The datasets and the Python source code are available online[1].

4.4. Experiment Results Discussion

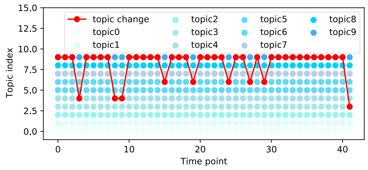
First, we extracted a certain number of topics for each dataset, in this experiment, we simply extracted 10 topics and calculated the probability of each topic at each moment. Since topics may change at each moment, we track topic

[1] https://github.com/kunkun111/TCR-M

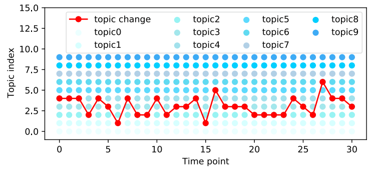
changes by locating the topic with the highest probability at each moment. Then, we locate and visualize the change of topics on the time stamp, as shown in Fig.2. It can be found that the topics have different probabilities at each time point. For example, the largest probability at one time point may be topic 1, while the largest probability at the next time point may be topic 10. This can help to track and observe the topic changes in time and lay the foundation for subsequent model learning.



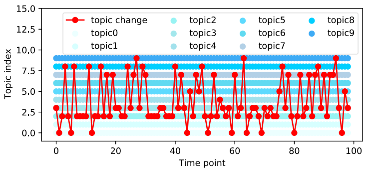
(a) Monitor



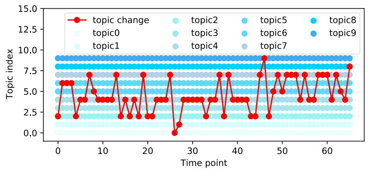
(b) Phone shell



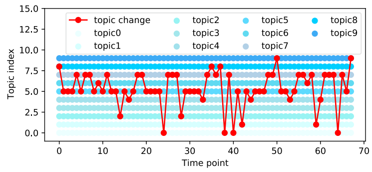
(c) Wet wipes



(d) T-shirt



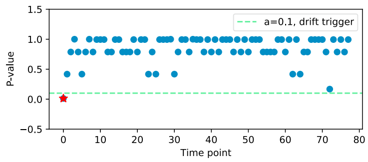
(e) VC tablets



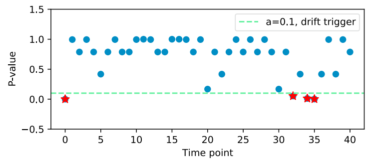
(f) Noodles

Fig. 2. A plot of the results of topic change tracking on datasets. We calculate the probability of the topics of each dataset. Here, we simply choose a topic with the maximum probability, tracking and observing the probability change at each time point.

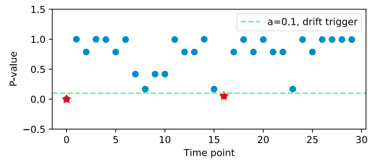
Secondly, we calculate the severity of topic change on two consecutive time steps and based on statistical tests, which helped to identify the significant topic change in time and marked the time point. Then, we adjust the model when a significant topic change occurred. By testing the topic change recognition process on each dataset, we identify and count the frequency of significant changes in topics timely, as shown in Table 2. It reflects that significant changes in topics were found on each dataset. To clearly show the change, we graphically marked the time points when a significant topic change is identified, as shown in Fig. 3. When changes occur, this timely process of change recognition can help the learning model to make real-time adjustments.



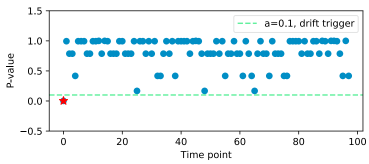
(a) Monitor



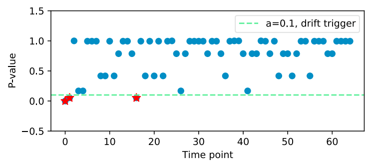
(b) Phone shell



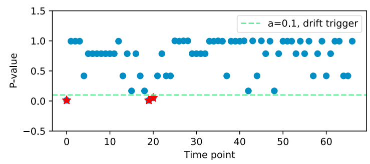
(c) Wet wipes



(d) T-shirt



(e) VC tablets



(f) Noodles

Fig. 3. A plot of the results of recognizing the topic with significant change severity. The red mark is the topic with significant change severity.

Besides, we construct an ensemble learning-based model and update it timely based on the topic change recognition results, then testing it on each dataset and compare the results with benchmark methods. The accuracy and F1 scores

Table 2. Topic Change Recognition results

Datasets	Monitor	Phone shell	Wet wipes	T-shirt	VC tablets	Noodles
Change frequency	1	4	2	1	3	3
Normal frequency	77	37	28	97	62	64

of each method are calculated and compared, and the results are shown in Tables 3 and 4. The results show that our proposed TCR-M method achieves relatively higher accuracy scores on most of the datasets, and the average accuracy scores are also relatively high, which reflects the performance of the proposed method to some extent.

Table 3. Accuracy Results Analysis (%)

Datasets	Baseline	TCR-M	TCR-M(retrain)	Learn++.NSE	LeverageBagging
Monitor	73.42	78.24	**79.72**	76.97	76.59
Phone shell	88.42	**91.70**	90.35	80.71	85.34
Wet wipes	78.56	83.23	**83.49**	72.71	69.29
T-shirt	72.77	75.50	**79.08**	78.68	76.31
VC tablets	86.81	88.78	**90.57**	84.63	87.20
Noodles	78.21	**84.09**	79.94	78.33	74.71
Average	79.69	83.59	**83.85**	78.67	78.24

Table 4. Macro F1-Score Results Analysis (%)

Datasets	Baseline	TCR-M	TCR-M(retrain)	Learn++.NSE	LeverageBagging
Monitor	52.84	56.02	58.13	**64.55**	58.04
Phone shell	53.42	51.59	54.34	48.74	**55.33**
Wet wipes	51.84	50.94	**59.37**	54.77	55.91
T-shirt	50.90	56.48	59.86	**64.91**	60.56
VC tablets	50.84	53.16	52.17	52.30	**54.83**
Noodles	49.78	46.76	49.87	51.54	**55.14**
Average	51.60	52.49	55.62	56.13	**56.63**

From the experimental results, it is found that the proposed TCR-M method has certain effectiveness in processing text stream data, but the method is only a preliminary attempt for text stream learning, and the stability of the model and performance in different application scenarios still need to be improved. In future research work, the current model learning strategy will also be improved and validated by multi-scenario testing.

5. Conclusion

In this paper, we propose a text data stream learning method based on topic change recognition. The method not only identifies and tracks topic changes in a time stamp, but also determines the severity of topic changes. It also updates the model based on the topic change recognition results, which further improves the prediction performance of the model and reduces the degradation of the model performance due to uncertain changes in the data. The effectiveness of the method is verified to some extent by testing and evaluating it on different text data streams. However, the method is currently only a simple attempt to learn the text data stream, and further improvements are still needed in the future to enhance its adaptability and robustness. And it should also be adjusted with more practical application scenarios so as to enhance its application in real-life scenarios.

Acknowledgements

This work was supported by Australian Research Council through the Discovery Project (No. DP200100700).

References

[1] Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R., 2006. Early drift detection method, in: Fourth International Workshop on Knowledge Discovery from Data Streams, Citeseer. pp. 77–86.
[2] Bifet, A., Gavalda, R., 2007. Learning from time-changing data with adaptive windowing, in: Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM. pp. 443–448.
[3] Bifet, A., Holmes, G., Pfahringer, B., 2010. Leveraging bagging for evolving data streams, in: Machine Learning and Knowledge Discovery in Databases, European Conference, Springer, Barcelona, Spain. pp. 135–150.
[4] Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022.
[5] Breiman, L., 1996. Bagging predictors. Machine Learning 24, 123–140.
[6] Elwell, R., Polikar, R., 2011. Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks 22, 1517–1531.
[7] Friedman, J.H., 2001. Greedy function approximation: A gradient boosting machine. Annals of Statistics , 1189–1232.
[8] Gama, J., Medas, P., Castillo, G., Rodrigues, P., 2004. Learning with drift detection, in: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, Springer. pp. 286–295.
[9] Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A., 2014. A survey on concept drift adaptation. ACM Computing Surveys 46, 44:1–44:37.
[10] Gomes, H.M., Bifet, A., Read, J., Barddal, J.P., Enembreck, F., Pfharinger, B., Holmes, G., Abdessalem, T., 2017. Adaptive random forests for evolving data stream classification. Machine Learning 106, 1469–1495.
[11] Gong, L., Zeng, J., Zhang, S., 2011. Text stream clustering algorithm based on adaptive feature selection. Expert Systems with Applications 38, 1393–1399.
[12] Jiang, Y., Yan, S., Qi, P., Sun, Y., 2020. Adapting to user interest drifts for recommendations in scratch, in: 16th International Wireless Communications and Mobile Computing Conference, Limassol, Cyprus, June 15-19, IEEE. pp. 1528–1534.
[13] Katakis, I., Tsoumakas, G., Vlahavas, I., 2006. Dynamic feature space and incremental feature selection for the classification of textual data streams. Knowledge Discovery from Data Streams , 107–116.
[14] Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., Woźniak, M., 2017. Ensemble learning for data stream analysis: A survey. Information Fusion 37, 132–156.
[15] Li, P., He, L., Hu, X., Zhang, Y., Li, L., Wu, X., 2016. Concept based short text stream classification with topic drifting detection, in: 2016 IEEE 16th International Conference on Data Mining, IEEE. pp. 1009–1014.
[16] Li, P., He, L., Wang, H., Hu, X., Zhang, Y., Li, L., Wu, X., 2018. Learning from short text streams with topic drifts. IEEE Transactions on Cybernetics 48, 2697–2711.
[17] Liu, A., Lu, J., Song, Y., Xuan, J., Zhang, G., 2022. Concept drift detection delay index. IEEE Transactions on Knowledge and Data Engineering doi:10.1109/TKDE.2022.3153349.
[18] Liu, A., Lu, J., Zhang, G., 2020. Concept drift detection via equal intensity k-means space partitioning. IEEE Transactions on Cybernetics 51, 3198–3211.
[19] Liu, A., Song, Y., Zhang, G., Lu, J., 2017. Regional concept drift detection and density synchronized drift adaptation, in: IJCAI International Joint Conference on Artificial Intelligence, pp. 1–7.
[20] Liu, Y.B., Cai, J.R., Yin, J., Fu, A.W.C., 2008. Clustering text data streams. Journal of Computer Science and Technology 23, 112–128.
[21] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G., 2019. Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering 31, 2346–2363.
[22] Montiel, J., Read, J., Bifet, A., Abdessalem, T., 2018. Scikit-multiflow: A multi-output streaming framework. Journal of Machine Learning Research 19, 1–5.
[23] Schapire, R.E., 2003. The boosting approach to machine learning: An overview. Nonlinear Estimation and Classification , 149–171.
[24] Schapire, R.E., Freund, Y., 2013. Boosting: Foundations and algorithms. Kybernetes 42, 164–166.
[25] Wang, B., Pineau, J., 2016. Online bagging and boosting for imbalanced data streams. IEEE Transactions on Knowledge and Data Engineering 28, 3353–3366.
[26] Yao, L., Mimno, D., McCallum, A., 2009. Efficient methods for topic model inference on streaming document collections, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 937–946.
[27] Yin, H., Zhou, X., Cui, B., Wang, H., Zheng, K., Nguyen, Q.V.H., 2016. Adapting to user interest drift for poi recommendation. IEEE Transactions on Knowledge and Data Engineering 28, 2566–2581.
[28] Yu, H., Zhang, Q., Liu, T., Lu, J., Wen, Y., Zhang, G., 2022. Meta-add: A meta-learning based pre-trained model for concept drift active detection. Information Sciences 608, 996–1009.
[29] Zhang, Q., Wu, D., Zhang, G., Lu, J., 2016. Fuzzy user-interest drift detection based recommender systems, in: 2016 IEEE International Conference on Fuzzy Systems, IEEE. pp. 1274–1281.
[30] Zhang, Y., Chu, G., Li, P., Hu, X., Wu, X., 2017. Three-layer concept drifting detection in text data streams. Neurocomputing 260, 393–403.