# **Final Graduate Directed Project Proposal**

### Information Systems Graduate Program

**School of Information Technology**

#### Student Name: Yangxuan Wu

UID#: 870318175

Email: ywu1214@ilstu.edu

Daytime Phone: +1 (309)-750-5376

Planned Graduation Semester: Fall Semester in 2020

IT 494 registration permit requested for \_\_\_\_\_\_\_\_Fall\_\_\_\_\_\_ for \_\_\_3\_\_\_ credit hours.

Semester/Year (1-4)

#### Project Title:

As members of the Project Committee for this student, we approve the attached project proposal.

CHAIR: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Name (typed) Signature Date

**OR**

CO-CHAIRS: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Name (typed) Signature Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Name (typed) Signature Date

MEMBER: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Name (typed) Signature Date

MEMBER: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Name (typed) Signature Date

## ***FOR SCHOOL USE ONLY***

|  |  |  |
| --- | --- | --- |
| *Date Received* | *School Approval by* | *Approval Date* |
|  |  |  |

**Project Description**

Project Title: Practical Anomaly Detection Through Machine Learning

Student Name: Yangxuan Wu

Estimated length of entire project description is 8-10 pages, single-spaced. Writing and references should follow the APA Guidelines.

1. **What will be produced by this project?**

Monitoring service performance and detecting exceptions is critical for Internet-based services,  such as search engines, online shopping, and social networking. Although dozens of Anomaly detectors like Anomaly Detection Toolkit (ADTK), which is a Python package for unsupervised/rule-based time series Anomaly Detection. Detection have been proposed over the years, it takes more than 10 days for the operator to select and tune the detector, and even eventually it will not work. This paper tries to solve this problem by practicing a new approach based on supervised machine learning. I will try to rebuild and use the Opprentice (Operator Apprentice) system, which allows the operator to manually annotate exceptions in the performance data with a convenient tool. Multiple detectors are used to extract the abnormal features of the performance data in parallel. Then a random forest classifier is trained by using features and tags to automatically select the appropriate combination of detection parameters and threshold.

In this article, I'll focus on time series performance exceptions (also known as volume-based exceptions) rather than other types of exceptions, such as using packet payload information for intrusion detection. The KPI(Key Performance Indicator) data that Opprentice is going to process is time series data in the format (timestamp, value). This data can be collected from SNMP, system slogs, network tracing, Web access logs, and other data sources.

1. **Who will use the results of the project?**

Direct users are anomaly detection practitioners. Any company that manufactures and operates network equipment can be a potential customer base. Typically, this product is sold to manufacturers and operators of network products. The users of the product are mostly the operation engineers and staff of the product quality inspection department in these companies, as well as other individuals or teams concerned about whether the overall quality and operation indicators of each batch of network products are anomaly.

1. **Describe the problems or difficulties currently experienced by the proposed user(s) which will be addressed by the proposed project.**

Tuning the parameters and thresholds for the detector is time-consuming because the optimal parameters and thresholds for a given detector are often highly dependent on the actual data in the service. Sometimes a combination of multiple detectors is even required to process complex parameters.Therefore, many time-consuming iterations are required between the exception detection practitioner and the operator to find the appropriate detectors and to tune their parameters and thresholds.

To address the definition challenge and detector challenge, using supervised machine learning techniques is a good choice to be able to capture complex parameters based on features and tags from the data (KPI data).First, operators can visually examine time series data and label the anomalies they identify.Operators can periodically (for example, weekly) mark cases as new data arrival because exceptions are often uncommon and the marking time is reasonable with the help of dedicated marking tools.Secondly, the severity of anomalies measured by different detectors can be used as a feature in machine learning, so each detector can be used as a feature extractor.Opprentice then learns from the tagged data and automatically obtains domain knowledge from operators.Specifically, multiple detectors are applied to KPI data in parallel to extract features.The feature and tag are then used to train the machine learning model: random forest, which can automatically select the appropriate combination of detection parameters and thresholds.The training objective is to maximize the operator's accuracy preferences.

The machine learn-based approach described above is feasible, but applying it to Opprentice presents many interesting and practical challenges.

First, marking exceptions requires a lot of manual work.In order to help operators label effectively, I need to find a special labeling tool with a simple and convenient interface.Secondly, I need to consider whether the training set contains enough exceptions and whether it will affect the performance of the machine learning algorithm. However, due to the low frequency of exceptions in practice, it is impossible for any training set to cover enough exceptions.For example, new exceptions may appear in the future. To solve this problem, it is a good choice to gradually retrain the classifier with newly tagged data. In this way, Opprentice is able to capture and learn new types of exceptions that do not occur in the initial training set. At the same time, another effect of infrequent anomalies is that the number of normal data in the training set is always greater than the anomaly. In learning this unbalanced data, the classifier tends to favor large (positive) classes over small (abnormal) classes. It results in a low detection rate that does not satisfy the operator's accuracy preference. I plan to address this problem by adjusting the machine learning classification threshold (cThld). Finally, in order to save manual work, I could neither choose the most appropriate detectors nor tune their internal parameters. I need to use multiple detectors with different parameters to extract the features. In this case, some features are either irrelevant to the exception or redundant to each other. Previous studies have shown that some learning algorithms are less accurate when dealing with these features. This problem can be solved by an integrated learning algorithm. Random Forests, which are relatively robust, are very effective for this problem.

1. **Describe the project results in more detail, including how they will be used.**

Opprentice Architecture:

1. First, the detector is selected as the feature extractor of the data. Based on feature and operator tags, the machine learning algorithm (random Forest) increments retraining the exception classifier with historical and up-to-date tagged data. The same set of detectors then extract the characteristics of the incoming data and use a classifier to detect/classify it. Here, the detector extracts only the features, rather than reporting the exceptions itself.

Labeling Tool:

1. To find Labeling Tool to help operators effectively flag anomalies in historical data.

Detectors

1. Detectors As Feature Extractors: First, when the detector receives an incoming data point, it internally generates a non-negative value, called severity, to measure how abnormal the point is. Most detectors are parameterized and have an internal set of parameters. The severity of a given data point depends on the detector and its internal parameters. The detector then further needs a threshold to translate severity into binary output, i.e. The threshold is referred to as the sThld.
2. Multiple fixed detectors are obtained by sampling the parameters of each parameterized detector. Detectors with specific sampling parameters are configured as (detectors).Thus, the configuration can act as a functional extractor: data point configuration (detector + sampling parameters)!Feature extraction, training, and classification (detection) in Opprentice are all designed for individual data points, not exception Windows.In this way, the machine learning algorithm can have enough training data, and the classifier can quickly detect individual abnormal data points.
3. EWMA(exponential weighted moving average) is a good choice for detector parameter sampling. It is a prediction-based detector with only one weight parameter.With the rise, forecasts rely more on recent data than historical ones

Machine Learning Algorithm and Confifiguring cThlds:

1. Due to the use of detectors without careful evaluation, there are redundant and irrelevant features. I can choose to use a decision tree to solve these problems. Decision tree is a popular learning algorithm because it is easy to understand and interpret. It is widely used to reveal the relationship between features and labels. At a higher level, it provides a tree model with various iF-then rules to classify data.
2. **Review existing software and literature relevant to the proposed project.**

Researchers have developed many detectors using different techniques [1-24]. In addition, the researchers attempted to address several challenges of using detectors in practice.

1. In order to automatically adjust the internal parameters of the detector, Krishnamurthy et al. [11] proposed a multi-channel grid search to find appropriate parameters from the data. Himura et al. [23] search parameters to maximize the rate of detected events. In contrast, in addition to the detector's internal parameters, we also consider the automatic selection detector and its thresholds.
2. in some jobs, ROC curves are used to evaluate the performance of different detectors without considering their thresholds [9,14].
3. MAD is used to improve the robustness of the detector against dirty data [3,15].
4. some solutions attempt to statically combine different detectors [8,21].
5. machine learning is also applied to anomaly detection [16,20,32], but it is the basic detector. Instead, use machine learning to combine different detectors.Another important challenge in anomaly detection is to obtain real values on the ground to evaluate the detector.

Three common solutions are :

1. real exceptions identified or confirmed by using domain operators [1,4,9,12,14,17];
2. generate composite exceptions by injecting real or predefined exceptions into background data [9,14,18,19];

(c) compare the anomalies reported by other detectors in pairs and take them as ground reality [1,8,10,17,18]. Solution (a) makes more sense because the basic goal of the project is to implement Opprentice's functionality to meet the needs of operators.

1. **Describe the benefits and advantages which the user could expect as a result of the project.**   
   I will follow theoretical guidance to try to build and use the Opprentice system, which allows operators to manually annotate exceptions in performance data using convenient tools. Multiple detectors can be used to extract the abnormal characteristics of performance data in parallel. Then the feature and label are used to train the random forest classifier to automatically select the appropriate combination of detection parameters and threshold. This avoids the need for operators to manually and iteratively tune detector parameters and thresholds, greatly optimizing the process of anomaly detection and improving efficiency.
2. **Describe the requirements, costs and commitments for the user as a result of this project.**   
   Operators want to be able to deploy multiple exception detectors in network services to detect data exceptions without having to manually and iteratively tune detector parameters and thresholds. Since this project is carried out based on theory and free data sources, there is no extra overhead and it can be implemented only by relying on common hardware and software.
3. **Provide a list of the major activities or steps you expect to undertake in completing this project.**
4. Collect KPI data from SNMP, system SLOg, network tracking, Web access logs, and other data sources;
5. Select the detector as the feature extractor of the data.The same set of detectors then extract the characteristics of the incoming data and use a classifier to detect/classify it.Here, the detector only extracts features, not reports the exception itself;
6. To find Labeling Tool to help operators effectively flag anomalies in historical data;
7. Detectors As Feature Extractors;
8. Multiple fixed detectors are obtained by sampling the parameters of each parameterized detector;
9. Learn the robust representation of multivariate time series by using key techniques such as random variable connection and plane normal flow;
10. Obtain the normal mode of multivariate time series;
11. Use these representations to reconstruct input data;
12. Use reconstruction probability and the data comparison to determine anomalies;
13. Explain the detected exceptions, according to the reconstruction probability of the uni-variate time series;
14. By collecting the anomalies found by the method and collecting the anomaly interpretation data, the quality parameters of network products and operation quality parameters can be optimized.

**What hardware, software, or other resources you will need to complete this project?** This project needs to use algorithms and programming methods in in Python. It also needs to find representative data sets as experimental parameters in SNMP, system slogs, network tracing, Web access logs, and other data sources. When designing the detection model, it is necessary to use LSTM(Long Short Term Memory)-based encoder-decoder for Multi-sensor anomaly detection, and apply it to multivariate time series prediction, and use the prediction error to determine the anomaly. To create a random model, it should rely on DAGMM which joints Deep Auto-encoder (AE) and Gaussian Mixture Model (GMM) simultaneously. However, the method studied in this project is still in the experimental stage and far from practical application. For example, in a real production application, users would simply enter the data set they want to analyze to get the desired exception detection and analysis results. But now you still need to manually enter and sort the data sets and manually select the algorithm to execute one by one to perform the corresponding steps in Python.

1. **Provide a bibliography of software and literature related to this project and proposal.**
2. Yingying Chen, Ratul Mahajan, Baskar Sridharan, and Zhi-Li Zhang. A provider-side view of web search response time. In Proceedings of the ACM SIGCOMM 2013 conference onSIGCOMM, pages 243–254. ACM, 2013.

[2] Ajay Anil Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, YinZhang, and Joanne Emmons. Detecting the performance impact of upgrades in large operational networks. In Proceedings of the ACM SIGCOMM 2010 Conference, SIGCOMM ’10, pages 303–314. ACM, 2010.

[3] Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. Rapid detection of maintenance induced changes in service performance. In Co-NEXT, page 13. ACM, 2011.

[4] David R. Choffnes, Fabián E. Bustamante, and Zihui Ge. Crowdsourcing service-level network event monitoring. In Proceedings of the ACM SIGCOMM 2010 Conference, SIGCOMM ’10, pages 387–398. ACM, 2010.

[5] Suk-Bok Lee, Dan Pei, M Hajiaghayi, Ioannis Pefkianakis, Songwu Lu, He Yan, Zihui Ge, Jennifer Yates, and Mario Kosseififi. Threshold compression for 3g scalable monitoring. In INFOCOM, 2012 Proceedings IEEE, pages 1350–1358. IEEE, 2012.

[6] He Yan, Ashley Flavel, Zihui Ge, Alexandre Gerber, Daniel Massey, Christos Papadopoulos, Hiren Shah, and Jennifer Yates. Argus: End-to-end service anomaly detection and localization from an isp’s point of view. In INFOCOM, 2012 Proceedings IEEE, pages 2756–2760. IEEE, 2012.

[7] Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. Rapid detection of maintenance induced changes in service performance. In Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies, CoNEXT ’11, pages 13:1–13:12, New York, NY, USA, 2011. ACM.

[8] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In Proceedings of the 6th International COnference, Co-NEXT ’10, pages 8:1–8:12. ACM, 2010.

[9] Augustin Soule, Kavé Salamatian, and Nina Taft. Combining fifiltering and statistical methods for anomaly detection. In Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, pages 31–31. USENIX Association, 2005.

[10] Yin Zhang, Zihui Ge, Albert Greenberg, and Matthew Roughan. Network anomography. In Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, IMC ’05, pages 30–30, Berkeley, CA, USA, 2005. USENIX Association.

[11] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In Proceedings of the 3rd ACM

SIGCOMM conference on Internet measurement, pages 234–247. ACM, 2003.

[12] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron. A signal analysis of network traffific anomalies. In Proceedings of the 2nd ACM SIGCOMM Workshop on Internet

measurment, pages 71–82. ACM, 2002.

[13] Jake D Brutlag. Aberrant behavior detection in time series for network monitoring. In LISA, pages 139–146, 2000.

[14] Fernando Silveira, Christophe Diot, Nina Taft, and Ramesh Govindan. Astute: Detecting a different class of traffific anomalies. In Proceedings of the ACM SIGCOMM 2010 Conference, SIGCOMM ’10, pages 267–278. ACM, 2010.

[15] Benjamin I.P. Rubinstein, Blaine Nelson, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J. D. Tygar. Antidote: Understanding and defending against poisoning of anomaly detectors. In Proceedings of the 9th ACM SIGCOMM Conference on Internet

Measurement Conference, IMC ’09, pages 1–14, New York, NY, USA, 2009. ACM.

[16] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. ACM Comput. Surv., 41(3):15:1–15:58, July 2009.

[17] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. Detection and identifification of network anomalies using sketch subspaces. In Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC ’06, pages 147–152, New York, NY, USA, 2006. ACM.

[18] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffific anomalies. In Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM ’04, pages 219–230. ACM, 2004.

[19] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffific feature distributions. In Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM ’05, pages 217–228. ACM, 2005.

[20] Makoto Yamada, Akisato Kimura, Futoshi Naya, and Hiroshi Sawada. Change-point detection with feature selection in high-dimensional time-series data. In Proceedings of the

Twenty-Third International Joint Conference on Artifificial Intelligence, IJCAI ’13, pages 1827–1833. AAAI Press, 2013.

[21] Shashank Shanbhag and Tilman Wolf. Accurate anomaly detection through parallelism. Network, IEEE, 23(1):22–28, 2009.

[22] Ayesha Binte Ashfaq, Mobin Javed, Syed Ali Khayam, and Hayder Radha. An information-theoretic combining method for multi-classififier anomaly detection systems. In

Communications (ICC), 2010 IEEE international conference on, pages 1–5. IEEE, 2010.

[23] Y. Himura, K. Fukuda, K. Cho, and H. Esaki. An automatic and dynamic parameter tuning of a statistics-based anomaly detection algorithm. In Communications, 2009. ICC ’09. IEEE International Conference on, pages 1–6, June 2009.

[24] Amazon cloudwatch alarm. http://docs.aws.amazon.com/AmazonCloudWatch/latest/

DeveloperGuide/ConsoleAlarms.html.

[25] He Yan, Lee Breslau, Zihui Ge, Dan Massey, Dan Pei, and Jennifer Yates. G-rca: A generic root cause analysis platform for service quality management in large ip networks. In Proceedings of the 6th International COnference, Co-NEXT ’10, pages 5:1–5:12, New York, NY, USA, 2010. ACM.