รายงานสรุปเนื้อหาและการนำไปใช้ประโยชน์จากการเข้าร่วมอบรม สัมมนา หรือประชุมวิชาการ

ข้าพเจ้า นาย กิติศักดิ์ โอสถานันต์กุล ตำแหน่ง อาจารย์ สังกัด คณะวิทยาศาสตร์ มหาวิทยาลัยแม่โจ้ ขอนำเสนอรายงานสรุปเนื้อหาและการนำไปใช้ประโยชน์จากการเข้าร่วมนำเสนอผลงานวิชาการระดับนานาชาติ "ECTI DAMT and NCON 2020" ระหว่างวันที่ 11-14 มีนาคม 2563 ณ. โรงแรมแอมบาสเดอร์ ซิตี้ จอมเทียน จังหวัดชลบุรี ตามหนังสือขออนุญาตเดินทางไปปฏิบัติงาน เลขที่ อว 69.5.11/25 ลงวันที่ 11 ธันวาคม 2562 ซึ่งการเข้าร่วมประชุมวิชาการดังกล่าว ข้าพเจ้าได้เลือกใช้งบประมาณการพัฒนาบุคลากรตามกรณีที่ 3 ดังนั้นจึงขอนำเสนอสรุปเนื้อหาและการนำไปใช้ประโยชน์ของการเข้าร่วมประชุมวิชาการดังต่อไปนี้

งานประชุมวิชาการระดับนานาชาติ International Conference no Digital Arts, Media and Technology (DAMT) ครั้งที่ 5 และ ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (NCON) ครั้งที่ 3 จัดขึ้นที่ พัทยา จังหวัดชลบุรี ประเทศไทย ระหว่างวันที่ 11-14 มีนาคม 2563 จัดขึ้นโดย Northern Section of the Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology Association of Thailand (ECTI ประเทศไทย) กลุ่มอุตสาหกรรมหลัก และสถาบันการศึกษาในภาคเหนือ รวมถึงมหาวิทยาลับบูรพาที่เป็นเจ้าภาพในปีนี้ โดยมีวัตถุประสงค์เพื่อสร้างโอกาสสำหรับนักวิจัย นักพัฒนาระบบ นักออกแบบระบบ วิศวกรและนักเทคโนโลยีได้แลกเปลี่ยนความคิดและหารือเกี่ยวกับการพัฒนาด้านเทคโนโลยีดิจิทัลและวิศวกรรมสำหรับการเสริมสร้างความเข้มแข็งของอุตสาหกรรม อีกทั้งงานประชุมวิชาการในครั้งนี้จัดขึ้นที่เมืองพัทยา ซึ่งเป็นเมืองที่มีเสน่ห์ มีทัศนียภาพที่สวยงามของธรรมชาติและสถานที่สำคัญที่น่าสนใจในภาคตะวันออกของประเทศไทย โดยโปรแกรมทางเทคนิคจะมุ่งเน้นการวิจัยและการแก้ไขปัญหาสำหรับการสนับสนุนเศรษฐกิจดิจิทัลเพื่อรักษาเสถียรภาพของการพัฒนาทางสังคมในระดับภูมิภาคและระดับโลกพร้อมกับการอนุรักษ์สิ่งแวดล้อม ทั้งนี้การประชุมวิชาการจะประกอบไปด้วยการนำเสนอผลงานด้วยปากเปล่าและการแสดงนิทรรศการในหัวข้อต่างๆ ที่น่าสนใจ ดังนี้

DAMT-2020

- Media Systems and Implementations
- Multi-signal Processing and Applications
- Digital Arts and Media

- Media and Medium Engineering
- Digital Economy for sustainable growth
- Geoinformatics
- Knowledge Management and Learning Organization

NCON-2020

- Devices, Circuits and Systems
- Computers
- Information Technology
- Communication Systems
- Digital Economy for sustainable growth
- Geoinformatics
- Knowledge Management and Learning Organization

โดยที่ข้าพเจ้าได้มีโอกาสนำเนื้อผลงานในหัวข้อ Simple Load Disaggregation Library based on NILMTK ซึ่งหัวข้อนี้เป็นการนำเสนอ library สำหรับการแยกปริมาณการใช้งานของเครื่องใช้ไฟฟ้าออกจากสมาร์ทมิเตอร์ ด้วยวิธีการใช้งานของ Machine Learning โดยที่นักพัฒนาสามารถนำ library ที่นำเสนอไปนั้น ไปต่อยอดงานวิจัยของผู้ที่สนใจต่อได้ ซึ่งตัว library นี้เองมีความกระทัดรัด สะดวกในการใช้งานและพร้อมที่จะถูกนำไปใช้ เมื่อเปรียบเทียบกับเครื่องมือที่ใช้ในการที่เผยแพร่อยู่ในปัจจุบัน

โดยเนื้อหาทั้งหมดของการเข้าร่วมประชุมวิชาการในครั้งนี้ ข้าพเจ้าสามารถนำมาใช้ประโยชน์ในงานประจำเช่น การสอน การวิจัย และงานบริการวิชาการได้

.....

นายกิติศักดิ์ โอสถานันต์กุล

ตำแหน่ง อาจารย์

19 มีนาคม 2563

ความคิดเห็นของผู้บังคับบัญชาชั้นต้น (ประธานอาจารย์ผู้รับผิดชอบหลักสูตรนวัตกรรมเทคโนโลยีดิจิทัล)

.....

.....

.....

.....

อาจารย์ กิตติกร หาญตระกูล

ประธานอาจารย์ผู้รับผิดชอบหลักสูตรวิทยาศาสตร์มหาบัณฑิต

สาขาวิชานวัตกรรมเทคโนโลยีดิจิทัล

19 มีนาคม 2563

Simple Load Disaggregation Library based on NILMTK

Kitisak Osathanunkul
Department of Digital Technology Innovation
Faculty of Science, Maejo University
Chiang Mai, Thailand
kit_o@mju.ac.th

Khukrit Osathanunkul*
Department of Information Technology
International College, Payap University
Chiang Mai, Thailand
osathank@gmail.com

Abstract— **Load disaggregation is a method to predict appliance power usage from a household power meter reading. A powerful open-source tool used in load disaggregation task is called Non-Intrusive Load Monitoring Toolkit (NILMTK). The toolkit provides many features such as built-in basic statistics, evaluation metrics and disaggregation algorithm comparison. However, it requires a steep learning curve. This can be difficult and troublesome for an unexperienced researcher to get started. Thus, this paper proposes a Simple Load Disaggregation (SLD) library to allow users to get started with the load disaggregation task quickly. The proposed library allows users to train and disaggregate an appliance load within a few lines of codes. The SLD modifies disaggregation algorithms from NILMTK. It removes all unnecessary elements and focusing only on making the load disaggregation task clean and simple.**

Keywords—non-intrusive load monitoring, nilm, load disaggregation, energy disaggregation.

I. INTRODUCTION

Load disaggregation is a technique of predicting an individual appliance electricity used from a household power meter reading. This technique is called Non-Intrusive Appliance Load Monitoring (NALM or NILM). The idea was firstly introduced in the mid-1980s by George Hart [1]. The technique uses a pattern recognition and event-based methods. Another technique [2] makes use of the combinatorial optimization technique. It learns the sum of appliance power consumption, then comparing this sum with the household meter reading. These two techniques were widely studied and it has been developed as disaggregation algorithm in an open-source community called Non-Intrusive Load Monitoring Tool-Kit (NILMTK) [3].

NILMTK is a python open source toolkit used in Non-Intrusive Load Monitoring (NILM). It is designed to allow researchers to work in this field with the same standard. So that benchmarks between different disaggregation algorithms can be compared using the toolkit. The features of NILMTK is not limited to load disaggregation, but it can also provide some data statistics, algorithm benchmark tools, evaluation metrics, and etc.

Due to many features of NILMTK, researchers need to study on the environment of NILMTK in order to get started. For using just load disaggregation function in NILMTK, users or researchers are required many steps to get started with the task. The whole package of NILMTK is needed to be installed. Then the users need to learn how to convert a dataset into NILMTK format and to learn how to deal with NILMTK basic commands and functions. Simply knowing a python language is not enough. A steep learning curve is definitely needed for unexperienced users.

In this paper, Simple Load Disaggregation is introduced to fill the gap. It is designed for users with little to no experience in this area. A load disaggregation function can be done by simply importing a library, training a model, then an appliance load can be predicted right away. It skips all unnecessary procedures required by NILMTK.

Section II discusses about related works. SLD is introduced in Section III. Section IV compares SLD with the original NILMTK in several aspects, and section V concludes the paper.

II. RELATED WORKS

A. NILMTK

Non-Intrusive Load Monitoring Toolkit or NILMTK [3] is a tool used in analysing load usage in a building. It is open source toolkit written in python. It is available on Github with a big community. Prior NILMTK published, it is almost impossible to find a way to compare literature findings and experiments in NILM. There is no standard in terms of experiment setup, data acquiring, data format, and etc. Therefore, NILMTK is introduced to tackle these issue. It is designed to be a standard tool for NILM tasks, and to be used among researchers. With NILMTK, researchers can have some guidelines on how obtained data and predicted data to be collected, stored, compared, evaluated and even represented in the similar manner or format. Thus, the results from disaggregation can be compared and discussed on the performance of an algorithm used.

NILMTK includes several features such as dataset conversion tools (to import dataset into NITLMTK environment), several disaggregation algorithm and evaluation tools.

B. Public Dataset

In NILM, many public datasets are available for researcher to work on. Datasets in NILM consists of both main power meter reading and an appliance power reading. The sampling rate of the dataset are typically ranges from 1 second to several second, but some datasets may also provide a higher sampling rate like 16kHz [4]. Popular NILM datasets are REDD [5], BLUED [6] and UKDALE [4].

1) REDD: A public data set for energy disaggregation [5] is published in 2011. It is the first dataset available for NILM research. During that time, this dataset becomes a standard dataset for researchers to benchmark their NILM algorithm. REDD dataset obtained data from 6 different houses in Massachusetts, US. REDD provides with both high and low frequency sampling rate. However, a lower frequency one is more useful as the data from the higher frequency one is often redundancy.

2) BLUED: Building-Level fUlly-labelled dataset for Electricity Disaggregation [6] is published by Kyle et al. from Electrical and Computer Engineering, Carnegie Mellon University, USA. The dataset provides both voltage and current data sampled at 12kHz with the duration of one full

week. The strong point of this dataset is that the appliance power data is labelled with a timestamp. That means ground truth can be used to confirm disaggregation results when evaluating algorithm.

3) UK-DALE: UK recording Domestic Appliance-Level Electricity [4] is the first public UK dataset. The dataset comes with a sampling rate of 16kHz for a whole house power meter reading and every 6 seconds for individual appliances. UK-DALE includes data from 5 houses with a length of 2 to 4 years. With a long data collection period allows researcher to investigate the nature of the data at the different time of the year. For example, during winter a heater is tend to be switched on more often or to consume more electricity than in the summer. This dataset also introduced a wireless device used for collecting its data.

C. Factorial Hidden Markov Model (FHMM)

Using Factorial Hidden Markov Model (FHMM) algorithm for load disaggregation has been introduced by Zoha et al. [7]. The idea of FHMM algorithm is to learn load patterns for each appliance. Zoha's experiments shows that with 5 multi-state appliance, FHMM can achieve the f-measure of 0.614. In addition, Kolter and Johnson [8] also uses FHMM in the load disaggregation in their experiments. Their results from REDD dataset shows the average accuracy of 47.7%. In addition to the typical FHMM algorithm, many researchers have proposed several variants to improve the performance of this algorithm. For example, Kim et al. are applying probabilistic models with FHMM in [9] and Parson et al. adopt HMM and modify Viterbi algorithm in [10].

D. Combinatorial Optimisation (CO)

CO is another disaggregation algorithm proposed by Hart et al [1]. CO reduces the difference between the sum of predicted appliance load and the household power meter reading data by finding the optimal combination of those different states. It finds a sum of an appliance power usage, and then uses this sum to compare with the one from a household power meter reading.

III. Simple Load Disaggregation

Originally, disaggregation algorithm modules in NILMTK are tied with its NILMTK environment. The module deals with power data stored in the Hierarchical Data Format (HDF5). This is well organized but it can be complicated to understand. In order to allow unexperienced users to work without a steep learning curve, a modification on the module is needed. Here, Simple Load Disaggregation (SLD) is introduced to fill this gap.

SLD presented in this paper is based on the original NILMTK version 0.2. The purpose of this library is to make a load disaggregation task simple and easy to use for an online or a near real-time application. It is also written in python and focus on using python dataframe rather than HDF5 like the original NILMTK. SLD is also published as a Github repository in [11].

There are two disaggregation algorithms provided in SLD; Factorial Hidden Markov Model (FHMM) and Combinatorial Optimisation (CO). Both algorithms are modified from NILMTK in similar manner where focusing on working with python dataframe and remove all unnecessary features apart from the load disaggregation function.

The following subsection explains the requirement of SLD, how to prepare data, how to create and train a model and then finally how to disaggregate a whole house power meter reading.

A. Software and Library Requirement

Disaggregation algorithms in SLD are a modification of the disaggregation algorithm of the NILMTK. So it inherits most of the major requirements from the original one. However, SLD requires just enough software and libraries to function a load disaggregation. The software and libraries required are python, numpy, pandas and hmmlearn.

B. Data Preparing and Preprocessing

Before getting started with the load disaggregation process, it is obvious that data is needed to be preprocessed in a certain format. In order to keep the task simple, SLD library required much less complicated data preprocessing to get started. Just like any machine learning preparation process, SLD library required two sets of data; training data and testing data.

1) Training data

Training data is the data that is used to train a model. A model can be more or less accurate depending on these training data feeding to the model. Here, training data must be included both main meter data (aggregated electricity meter reading) and an individual appliance meter data.

To prepare a dataframe for training a model, both main meter data and appliance meter data must be contained in the same dataframe. It must include at least three columns; timestamp, power and appliance meter reading. The example of the dataframe is shown below.

	timestamp	power	app1	app2	app3
0	1525689485	23.30	0	0	0
1	1525689490	318.4	295.1	0	0
2	1525689495	318.7	295.4	0	0
	...				

1.1) Timestamp

The first required column is the timestamp. It is used to identify the order of the event happened. Unlike the NIMLTK, timezone and daylight saving will not be considered. The first timestamp will be used as the reference point. It is recommended to keep the timestamp in a Unix Epoc format in which 10 decimal digit number. This column should be named as "timestamp".

1.2) Power

The second column is the training main meter data. It is typically a reading from a main of a household, or it can be a reading from a set of appliance depending on how the meter data can be acquired. The unit of the reading can be in Watt or kWatt. This column should be named as 'power'.

1.3) Appliance

This rest of the column is the reading for individual appliance. There must be at least one appliance meter data included in the dataframe as it will be used to train a model.

In python, data from a CSV file can be imported to a dataframe. It can be done by simply calling a method read_csv() from pandas.

2) Testing Data

Similar to the training data, testing data should contain timestamp and a house meter power. However, it does not require individual appliance power data in the testing data. So the format of the testing dataframe will be similar to the following dataframe.

```
   timestamp    power
0  1525689485   23.3
1  1525689490   318.4
2  1525689495   318.7
...
```

The power data in the testing dataframe here will be disaggregated into individual appliance power data later on. The results of the disaggregation will be depending on the model. If the model has been trained with 3 appliance data, the result data will also contain 3 disaggregated appliance data as well. The details of the model will be discussed in the later section.

C. Creating and Training a Model

In order to disaggregate main meter data, model is needed to be trained first. In SLD, models can be trained simply by calling train() function. The function requires two arguments; dataframe and a list of appliance. The dataframe is the one that is prepared from the previous data preprocessing step. On the other hand, a list of appliance must be stored as a python list. In the list, it should contain at least one name of appliance to be trained. A name of the appliance is stored as string type.

A list of appliance will be used to define how many appliances will be trained in the model. So that when disaggregating the main power, the number of appliance will be the same as the number of appliance in the list. For example, if three appliances like 'kettle', 'microwave' and 'heater' are focused, the python list should be as follow.

```
list_of_appliance = ['kettle','microwave','fan']
```

In addition, the name of the appliances in the list should exactly be the same name as the one in the dataframe. During the training process, each appliance will be used to train the model one by one until all appliances in the list will be included.

Given "df" is a dataframe containing data for a house, "list_of_appliance" is a list containing an appliance list and fhmm is an object class for FHMM algorithm. The example code of calling a training function can be as the following code.

```
fhmm.train(df, list_of_appliance)
```

After calling the train() function, the model is created. This model will be used to disaggregate an appliance power from the main power later on.

After a model is created (or the first training, the model can be further trained with another dataframe containing another house to learn more information. Doing so will allow the model to learn with different data from different houses.

The model can be stored for later use. This is because it is useful for further training, which helps to make the model more accurate. The saved model is small and fast to be loaded, it can be used for a near real time application. To save or store a model, it is recommended to use a python "pickle" library. With pickle, the model will be saved in a file with the "pkl" extension.

Here, SLD also provides a function to save and load a model. It is done by calling save() and load() function. For example, to save and to load a model to a file can be done by the following command.

```
fhmm.save("fhmm_trained_model")
fhmm.load("fhmm_trained_model")
```

To save, a model will be stored to a file called "fhmm_trained_model.pkl". On the other hand, once the model is reloaded, the same model will be ready to use as it is the same model.

D. Load Disaggregation

Load disaggregation is the main function of this library. It is used to extract appliance meter data from main meter data. A result of disaggregated appliance meter data will be found depending on a trained model and a selected algorithm. In order to disaggregate a power meter reading, it can be done by calling disaggregate() function. This function only requires one argument which is a testing dataframe. The example for calling this function is as follow.

```
prediction = fhmm.disaggregate(df)
```

This section has introduced SLD. This includes functions and data format to get load disaggregation task done. The next section will be discussed on what different between the SLD and the original NILMTK.

IV. Comparisons between SLD and NILMTK

This section compares SLD with the original NILMTK in different aspects. These are the software and library dependency, data preparing and preprocessing, disaggregation performance and feature and supports.

A. Software and Library Dependency

It is known that NILMTK requires lot of software and libraries for its feature-rich tool kit. There can be a troublesome when installing the NILMTK at the first time. A dependency confliction issue is the main cause of error during the installation process. This is because when the time goes by some libraries might have been upgraded to a newer version. Some library often requires a specific version of dependency library. This can cause the installation fail and some attention will be needed to resolve this issue. Installing NILMTK can be difficult from this kind of issue.

On the other hand, SLD only focuses on load disaggregation but it does not consider much about the details or environment of the data. So SLD required only few software and libraries just enough for disaggregation function to work. Its requirements are only python, numpy, pandas and hmmlearn. With this little requirements, SLD can be easily to install or import, and ready to get started easily.

Table I shows the list of the requirements of NILMTK and SLD. It is obvious that SLD requires less software and library dependency to get started with load disaggregation task.

TABLE I. SOFTWARE REQUIREMENTS OF NILMTK AND SIMPLE LOAD AGGREGATION (SLA) LIBRARY

Software and library	NILMTK	SLD
Python	>= 3.6	>= 3.6
Numpy	>= 1.13.3	>= 1.13.3
Pandas	>= 0.25.0	>= 0.25.0
Cython	>= 0.27.3	Not required
Bottleneck	>= 1.2.1	Not required
Numexpr	>= 2.6.4	Not required
Matplotlib	>= 3.1.0	Not required
Networkx	== 2.1	Not required
Spicy	>= 1.0.0	Not required
Scikit-learn	>= 0.21.2	Not required
Hmmlearn	Any	Any
Pytables	Any	Not required
Jupyter	Any	Not required
iPython	Any	Not required
iPykernel	Any	Not required
Nose	Any	Not required
Coverage	Any	Not required
Psycopg2	Any	Not required
Coveralls	Any	Not required

B. *Data Preparation and Preprocessing*

In NILMTK, this process can really be complicated and confusing for unexperienced researcher. It requires a lot of steps and some learning curves to get started. Data needs to be arranged into its specific format called as NILMTK-DF described in [9]. In the NILMTK-DF format, it is well organized using the Hierarchical Data format (HDF5). It provides lots of information such as metadata of a dataset, electricity meter reading, water meter reading, gas meter reading or even on-off switch data. The purpose of having this kind of format is to preserve as much information as possible. So that the other researcher can learn and understand on when, where, how data can be obtained.

NILMTK does provide some tools to convert some public dataset into NILMTK-DF format. For example, the RRED, BLUE and UKDALE datasets are the most famous dataset. Users can convert these datasets into NILMTK-DF format with a single command. However, converting a custom dataset in to NILMTK-DF format is not an easy task. A deep understanding of the NILMTK structure is mandatory. This procedure can be time consuming to convert a custom dataset into the right format. In contrast, SLD requires data preparation in a simple python dataframe. The format is clean and simple to understand. Users can simply prepared data in Microsoft Excel and import to dataframe easily. At this point, SLD allows users to prepare data in order to get started quickly.

C. *Disaggregation Performance*

When considering a disaggregation performance between NILMTK and SLD. Both of them does predict the exactly same results. This is because SLD modifies the disaggregation algorithm module from NILMTK. The modifications were to remove all unnecessary features and leave only the disaggregation function untouched. As a result, the SLD library can predict an appliance load with the same results.

D. *Feature and Supports*

NILMTK community has been published as Github repository since 2014. At the time of writing this paper it has been forked for almost 300 times with over 20 contributors. It contains lots of features such as dataset conversion, power meter selection (in case one house has several meters), basic statistics, evaluation metrics, disaggregation algorithm comparison and etc. As this is an open source community, NILMTK has a volunteer to support when users or researcher get stuck at some point.

On the other hand, SLD is pretty simple. It has only one feature. That is load disaggregation. SLD may have much less features when comparing to NILMTK, but it is also easy to understand and get going.

V. CONCLUSIONS

Simple Load Disaggregation library (SLD) is introduced. It predicts appliance electricity used from a house electricity meter reading. The main idea of SLD is to make load disaggregation job easy to use. Disaggregation algorithms from NILMTK is modified. SLD removes all unnecessary elements, and leave the core of disaggregation algorithm untouched. SLD is simple and easy to use, while it stills perform the same results as the original NILMTK.

REFERENCES

[1] G. W. Hart. Prototype nonintrusive appliance load monitor. Technical report, MIT Energy Laboratory and Electric Power Research Institute, Sept. 1985.

[2] G. W. Hart. Nonintrusive appliance load monitoring. Proceedings of the IEEE, 80(12):1870–1891, Dec. 1992. doi:10.1109/5.192069.

[3] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, Mani Srivastava. NILMTK: An Open Source Toolkit for Non-intrusive Load Monitoring. In: 5th International Conference on Future Energy Systems (ACM e-Energy), Cambridge, UK. 2014.

[4] Kelly J., Knottenbelt W. The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. Scientific Data. 2015;2 doi: 10.1038/sdata.2015.7.150007.

[5] Kolter Z. J., Johnson M. J. Redd: A public data set for energy disaggregation research. Proceedings of the In Workshop on Data Mining Applications in Sustainability (SIGKDD); 2007; San Diego, CA, USA. pp. 59–62.

[6] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges, "BLUED: A Fully Labeled Public Dataset for Event-Based Non-Intrusive Load Monitoring Research," in Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD), Beijing, China, 2012.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[7] Zoha A., Gluhak A., Nati M., Imran M. A. Low-power appliance monitoring using Factorial Hidden Markov Models. Proceedings of the 2013 IEEE 8th International Conference on Intelligent Sensors, Sensor Networks and Information Processing: Sensing the Future (ISSNIP '13); April 2013; pp. 527–532.

[8] Kolter Z. J., Johnson M. J. Redd: A public data set for energy disaggregation research. Proceedings of the In Workshop on Data Mining Applications in Sustainability (SIGKDD); 2007; San Diego, CA, USA. pp. 59–62.

[9] Kim H., Marwah M., Arlitt M., Lyon G., Han J. Unsupervised disaggregation of low frequency power measurements. SDM; 2011; pp. 747–758.

[10] Parson O., Ghosh S., Weal M., Rogers A. Non-intrusive load monitoring using prior models of general appliance types. Proceedings of the 26th AAAI Conference on Artificial Intelligence and the 24th Innovative Applications of Artificial Intelligence Conference; July 2012; pp. 356–362.

[11] Kitisak O, Simple Load Disaggregation Python Library, (2019), GitHub repository, https://github.com/amzkit/load-disaggregation

ECTI Association | ECTI NORTHERN | IEEE THAILAND SECTION | entral | Co-funded by the Erasmus+ Programme of the European Union

Certification of Appreciation present to

Kitisak Osathanunkul

Simple Load Aggregation

at The 5th International Conference on Digital Arts, Media and Technology (DAMT)
and 3rd ECTI Northern Section Conference
on Electrical, Electronics, Computer and Telecommunication Engineering (NCON)

11-14 March 2020 at Pattaya, Thailand

Asst.Prof. Pradorn Sureephong, Ph.D.
General Chair of DAMT and NCON 2020

แบบฟอร์มแจ้งความประสงค์การใช้งบประมาณสำหรับการพัฒนาบุคลากรคณะวิทยาศาสตร์

ประจำปีงบประมาณ พ.ศ. 2563

****

ข้าพเจ้า นาย กิตติศักดิ์ โชคภาพสกุล ตำแหน่ง อาจารย์ สังกัด สาขาวิชาบริหารเทคโนโลยีดิจิทัล

ได้ขออนุญาตเข้าร่วม นำเสนอผลงานในงานประชุมวิชาการ ECTI DAMT and NCON 2020

ตาม**หนังสือขออนุญาต** อว.๖๙.๕. 11 / 25 ลงวันที่ 11 ธันวาคม 62 โดยข้าพเจ้ามีความประสงค์จะขอใช้งบประมาณพัฒนาบุคลากรของคณะวิทยาศาสตร์เพื่อไปพัฒนาตนเอง ดังนี้

☐ กรณีที่ ๑ ใช้งบประมาณไม่เกิน ๖,๐๐๐ บาท สำหรับการเข้าร่วมอบรม สัมมนา หรือประชุมวิชาการทั่วไปที่เกี่ยวกับการพัฒนาวิชาชีพของตนเองฯ (ไม่ต้องรายงาน)

☐ กรณีที่ ๒ ใช้งบประมาณไม่เกิน ๘,๐๐๐ บาท สำหรับการเข้าร่วมอบรม ฝึกอบรม สัมมนา หรือประชุมวิชาการทั่วไปที่เกี่ยวกับการพัฒนาวิชาชีพของตนเอง ต้องส่งรายงานสรุปเนื้อหาและการนำไปใช้ประโยชน์ อย่างน้อย ๑ หน้ากระดาษ A๔ (เนื้อหาสรุปไม่น้อยกว่า ๒๕ บรรทัด)

☑ กรณีที่ ๓ สำหรับการเข้าร่วมนำเสนอผลงานวิชาการในรูปแบบโปสเตอร์ หรือปากเปล่า โดยต้องเป็นผู้เขียนชื่อแรก (First author) หรือต้องเป็นผู้เขียนหลัก (Corresponding author) ซึ่งได้รับการตอบรับเป็นที่เรียบร้อยแล้ว

- คนละไม่เกิน ๑๕,๐๐๐ บาท (สำหรับสายวิชาการ)
- คนละไม่เกิน ๑๐,๐๐๐ บาท (สำหรับสายสนับสนุนวิชาการ)

โดยต้องจัดส่งเอกสาร ดังนี้ สำเนาบทคัดย่อ หรือโปสเตอร์(ย่อขนาด A๔) หรือบทความฯ ฉบับเต็ม *และ*ต้องทำรายงานสรุปเนื้อหาและการนำไปใช้ประโยชน์ของการเข้าอบรม อย่างน้อย ๑ หน้ากระดาษ A๔ (เนื้อหาสรุปไม่น้อยกว่า ๒๕ บรรทัด)

☐ กรณีที่ ๔ สำหรับการเข้าร่วมอบรมเชิงปฏิบัติการเพื่อเพิ่มสมรรถนะในสายวิชาชีพที่เชี่ยวชาญตามตำแหน่งงานของตนเอง

- คนละไม่เกิน ๑๕,๐๐๐ บาท (สำหรับสายวิชาการ)
- คนละไม่เกิน ๑๐,๐๐๐ บาท (สำหรับสายสนับสนุนวิชาการ)

โดยต้องจัดส่งเอกสาร ดังนี้ สำเนาใบรับรองหรือหนังสือรับรองหรือใบประกาศนียบัตรหรือวุฒิบัตร จากการเข้าอบรมเชิงปฏิบัติการ *และ*รายงานสรุปเนื้อหาและการนำไปใช้ประโยชน์ อย่างน้อย ๑ หน้ากระดาษ A๔ (เนื้อหาสรุปไม่น้อยกว่า ๒๕ บรรทัด)

ในปีงบประมาณ พ.ศ. 2563 (๑ ต.ค. 62 - ๓๐ ก.ย. 63) ข้าพเจ้าได้ใช้งบพัฒนาบุคลากรฯ ไปแล้ว จำนวนทั้งสิ้น - ครั้ง ดังต่อไปนี้

-ครั้งที่ .... ในกรณีที่........... ใช้งบประมาณไปแล้วเป็นจำนวนเงินทั้งสิ้น.......................... บาท

-ครั้งที่ .... ในกรณีที่........... ใช้งบประมาณไปแล้วเป็นจำนวนเงินทั้งสิ้น.......................... บาท

(หากมีจำนวนครั้งเกินกว่านี้ ให้ทำรายละเอียดแนบท้ายเพิ่มเติม)

(ลงชื่อ) ผู้ขออนุญาต

(กิตติศักดิ์ โชคภาพสกุล)

11 / ธ.ค. / 62

(ลงชื่อ) ประธานหลักสูตร/เลขานุการคณะ/หัวหน้างาน

(.........................)

11 / ธ.ค. / 62

หมายเหตุ : ๑. งบประมาณที่ใช้สำหรับการพัฒนาบุคลากร หมายรวมถึงค่าใช้จ่ายทุกประเภทที่ใช้ในการเข้าร่วมการอบรม/สัมมนา/ประชุม เช่น ค่าลงทะเบียน ค่าใช้จ่ายในการเดินทาง และอื่น ๆ ที่เกี่ยวข้อง

๒. การใช้งบประมาณพัฒนาบุคลากรในที่คณะวิทยาศาสตร์จัดสรร ให้ถือปฏิบัติตามเงื่อนไขที่ได้กำหนดไว้ในแต่ละกรณี

๓. ให้แนบแบบฟอร์มแจ้งความประสงค์ฯ นี้มาพร้อมการส่งรายงานสรุปเนื้อหาและการนำไปใช้ประโยชน์ฯ ด้วย

เห็นชอบตามมติที่ประชุมคณะกรรมการประจำคณะฯ ครั้งที่1/2560

เริ่มใช้ตั้งแต่เดือน 1 กุมภาพันธ์ 2560