PAPER *Special Section on Intelligent Information and Communication Technology and its Applications to Creative Activity Support*

Software Development Effort Estimation from Unstructured Software Project Description by Sequence Models

Tachanun KANGWANTRAKOOL[†a)], ***Member***, **Kobkrit VIRIYAYUDHAKORN**[†b)], ***and*** **Thanaruk THEERAMUNKONG**[†c)], ***Nonmembers***

SUMMARY Most existing methods of effort estimations in software development are manual, labor-intensive and subjective, resulting in overestimation with bidding fail, and underestimation with money loss. This paper investigates effectiveness of sequence models on estimating development effort, in the form of man-months, from software project data. Four architectures; (1) Average word-vector with Multi-layer Perceptron (MLP), (2) Average word-vector with Support Vector Regression (SVR), (3) Gated Recurrent Unit (GRU) sequence model, and (4) Long short-term memory (LSTM) sequence model are compared in terms of man-months difference. The approach is evaluated using two datasets; ISEM (1,573 English software project descriptions) and ISBSG (9,100 software projects data), where the former is a raw text and the latter is a structured data table explained the characteristic of a software project. The LSTM sequence model achieves the lowest and the second lowest mean absolute errors, which are 0.705 and 14.077 man-months for ISEM and ISBSG datasets respectively. The MLP model achieves the lowest mean absolute errors which is 14.069 for ISBSG datasets.

key words: *software effort estimation, regression, deep learning, sequence model, recurrent neural network (RNN), gated recurrent units (GRU), long short-term memory network (LSTM)*

1. Introduction

Cost estimation is a heart of the software development process. It helps not only have a knowledge of the budget needed but also have an idea of the time needed to complete the project. The time estimation can be easily translated into budget once we know how much man-hour costs and all dependencies throughout the project. However, time estimation for a software project is an arduous tasks. It is a time-consuming task that requires professional software analysts.

Many methods have been published to estimate the effort on a software development project. Most of them are using the rule-based decision method, or referencing from previous data. L.H. Putnam firstly stated the empirical method for estimating computer software in 1978 [1]. A macromethodology that produces accurate estimates of manpower, costs, and times to reach critical milestones of software projects was proposed. It used four software state variables such as the state of technology, the applied effort, the development time, and the independent variable time.

Kemerer, and Chris F. proposed an empirical validation of four algorithmic models (SLIM, COCOMO, Function Points, and ESTIMACS) as general software-development cost estimators [2]. M. Shepperd and C. Schofield estimated the effort using the analogies database [3]. The feature of a software project is extracted as the query. The development effort is estimated by searching the most similar project from the database.

Later, the machine learning approaches which generally show the promising results [4] over the rule-based method in many regression tasks was proposed. A.L. Oliveira used the support vector regression (SVR) to estimate the effort of work [5]. The experiments were carried out using a dataset of software projects from NASA. They compared between support vector regression (SVR), radial basis functions neural networks (RBFNs) and linear regression for estimation of software project effort. The SVR significantly outperforms RBFNs and linear regression in this task.

In this paper, we proposed the software project estimators using deep learning techniques which are organized as follows. Section 2 describes the ISEM and ISBSG software projects datasets. Section 3 presents four proposed methods. Section 4 explains our experimental settings. Section 5 shows the experimental results. Section 6 concludes the paper.

2. Software Project Datasets

2.1 ISEM Dataset: Project Description Text

ISEM (Institute of Software Engineering improvement and quality Management) [6] is the first CMM/CMMI consulting team for software process in Thailand established in 1998. ISEM collected 1,573 pairs of a software project description written in English and its actual man-month while visiting 308 companies for CMMI appraisal throughout Asia. This dataset is not publicly available due to unable to disclose the clients' confidential information. The example of data is shown in Table 1.

2.2 ISBSG Dataset

ISBSG (International Software Benchmarking Standards

Manuscript received March 27, 2019.
Manuscript revised September 30, 2019.
Manuscript publicized January 14, 2020.

†The authors are with School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University, Thailand.
a) E-mail: tachanun@studentsiit.tu.ac.th
b) E-mail: kobkrit@gmail.com
c) E-mail: thanaruk@siit.tu.ac.th

DOI: 10.1587/transinf.2019IIP0014

Table 1 Examples of the ISEM software project description and the effort estimation (in man-month)

Product Description	Effort
Joint Office Automation System is based on the structural data management with the platform of www office automation system.	1.00
This system provides the operational support platform to local credit system data gathering, credit data sharing and credit application.	2.00
To realize centralized and unified management on the information such as order records, number, packages, preferential policies, etc. and background management authority settings	3.00
To provide our clients a set of complete advanced function and good user experience vehicle-mounted recreational and multimedia system.	7.00
To develop a universal interface module based on Modbus communication agreement, makes each isomerous station can be connected.	11.00
This is a system used to support Interaction between China Unicom Guangzhou branch and customers, and provide a presentation of analysis on interactive activity data to sales and management.	15.00
With the implementation of "Regulations of City Special Economic Zone on the Donation and Transplantation of Human Organ", It is developed to manage effectively the donor, transplant and other information.	18.00

Table 2 Examples of the ISBSG software project data and the effort estimation (in man-month)

Year	Industry Sector	Application Group	Application Type	Development Type	Prog. Language	...	Effort
1998	Service Industry	Business Application	Transaction/Production System	New Development	Oracle	...	10.51
2016	Communication	Business Application	CRM	Enhancement	Java	...	1.63
2000	Insurance	Business Application	Sales contact management	New Development	Java	...	103.18
2002	Wholesale & Retail	Business Application	Stock control & order processing	Enhancement	C	...	120.00
1994	Banking	Business Application	Transaction/Production System	New Development	EASYTRIEVE	...	10.65
2000	Communication	Real-Time Application	Real-time System	Enhancement	C	...	9.68

Table 3 Descriptive statistic of software project description datasets

Category	Statistics	ISEM Dataset	ISBSG Dataset
Data Points	Amount	1,573	9,100
Software Size Estimation (in Man-Months)	Min	1	0
	Mean	5.729	17.873
	Max	18	199.722
	Variance	5.700	700.575
	Skewness	1.031	3.217
Length of Software Project Description Text (in Words)	Min	4	7
	Mean	33.587	43.230
	Max	143	262
	Variance	307.208	925.927
	Skewness	1.076	1.957

Group) [7] is a not-for-profit organisation was founded in 1997 by a group of national software metrics associations. Their aim was to promote the use of IT industry data to improve software processes and products.

ISBSG's Development & Enhancement Repository contains software project data which is contributed by organisations around the world. It consists of 9,100 software projected developed in 32 different countries. Wide variety of industries and business types are represented in this dataset.

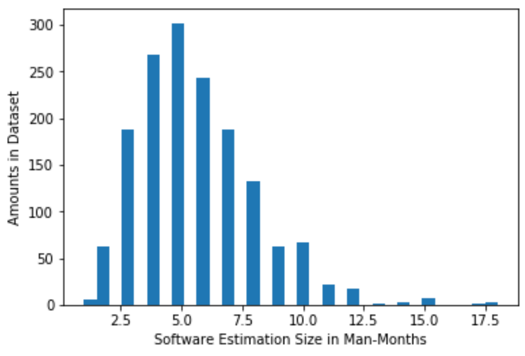
The data is arranged in Excel file format consisted of 253 columns shown in Table 2. They include:

- Project description details
- Size and size attributes
- Effort and effort attributes
- Defects
- Project schedule details
- Effort per phase
- Architecture
- Techniques
- Documentation used
- Platform details

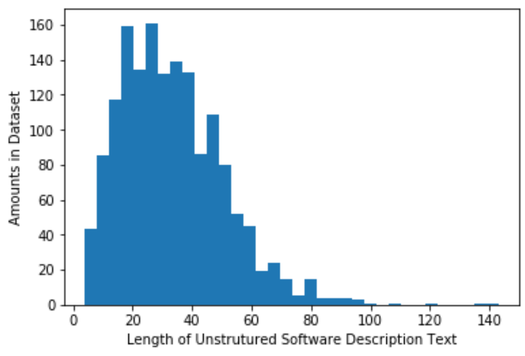
All numerical information that directly indicates to the effort estimates such as sizes, efforts, and their attributes are removed from the dataset. We selected the column "Effort: Summary Work Effort" as the prediction target, which is a total summarized of man-hours of that project. We convert the value into man-months by dividing 176 which is a number of working hours in a month assuming they are working 8 hours a day and 22 days a month.

2.3 Data Pre-Processing

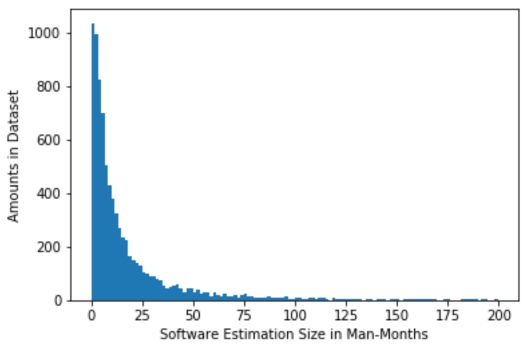
We concatenated all textual information in 253 columns of ISBSG dataset into a single raw text sentence for equally performance comparison between both datasets. We removed non-English characters and punctuation except the hyphen and then performed word tokenization using the space character. Then, we lemmatized every word using



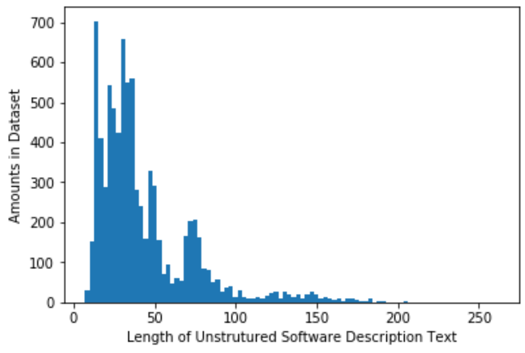
(a) Software Estimation Size (in Man-Months) of ISEM dataset



(b) Length of Software Project Description (in Words) of ISEM dataset



(c) Software Estimation Size (in Man-Months) of ISBSG dataset



(d) Length of Software Project Description (in Words) of ISBSG dataset

Fig. 1 Descriptive statistic of both ISEM and ISBSG datasets.

Wordnet lemmatization[†].

We converted a sentence into a list of word vectors by looking up the pretrained 50-dimensional GloVe [8] word embedding which is a dictionary mapping from vocabularies (including words and general name entities) to numerical vectors. These vectors contain fine-grained semantic and syntactic regularity of their representative words, which introduce their semantic information into our machine learning model. This pre-processing method is proved by various researches work [8]–[10] that significantly improved the performance across a range of natural language processing tasks.

After the pre-processing, statistic of software project estimation and software project description text in both datasets are computed in Table 3 and visualized in Fig. 1.

3. Proposed Methods

To perform ordinal regression tasks from text sequence, we split algorithms into two approaches as follows.

3.1 Average Word-Vector Approach

The 50-dimensional vector of all words in a sentence are arithmetic average. The average word-vector is fed into a ordinal regression algorithm as shown in Fig. 2. The regularization is applied using Adam optimization and dropout regularization techniques. Various regression algorithm is tested as follows.

3.1.1 Multi-Layer Perceptron (MLP)

Multi-Layer perceptron is considered as a universal function approximating [11]. It can be used to create mathematical models for either classification or regression analysis depended on the activation function.

Mean-squared error function is selected as the loss function, since it is a standard cost function in regression tasks. We feed an average word vector as an input into the network. In the training process, the network learns the estimation function from the average word vector for predicting the software effort estimation in man-months. The structure of the network is consisted of three layers listed as follows.

1. Input Layer: 50 Dimension average word-vector with relu activation function.
2. Hidden Layer: 20 Hidden nodes with relu activation function.
3. Output Layer: Single output nodes.

The equations of a multi-layer perceptron network are shown in Eq. (1) below.

[†]https://www.nltk.org/_modules/nltk/stem/wordnet.html

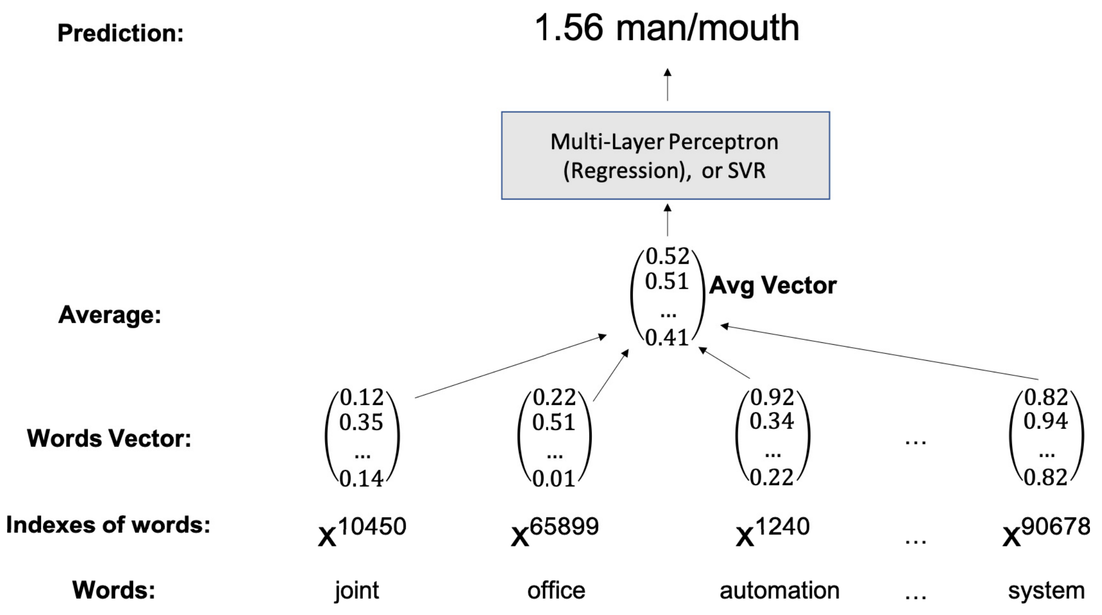


Fig. 2 Average word-vector approach

$$
\begin{aligned}
z_1^{(i)} &= W.avg^{(i)} + b_1 \\
a_1^{(i)} &= \text{relu}(z_1^{(i)}) \\
z_2^{(i)} &= W.a_1^{(i)} + b_2 \\
a_2^{(i)} &= \text{relu}(z_2^{(i)}) \\
\mathcal{L}^{(i)} &= \frac{1}{2n_y}\sum_{k=0}^{n_y-1}(Y-\hat{Y})^2
\end{aligned}
\qquad (1)
$$

where relu is a relu activation function and *avg* is an input average word-vector.

3.1.2 Support Vector Regression (SVR)

Support Vector Regression (SVR) has been proved in many research works that is significantly accurate more than other regressors [12], [13]. SVR is generalized from Support Vector Machine (SVM) in which returns a continuous-valued output for regression tasks, as opposed to an output from a finite set for classification tasks. SVM is designed to find a boundary that have the maximum margin separating the vector space while correctly classifying as many training sets as possible.

SVR is accomplished by introducing an ε-insensitive region around the function, called the ε-tube. SVR is formulated as an optimization problem that find the narrowest ε-tube centered around the surface, while minimizing the prediction error, that is, the distance between the predicted and the desired outputs as shown in Fig. 3.

In this study, the ε-insensitive loss function is used. Radial basis function kernel is selected for allowing non-linear boundary. The following parameters are used for training the SVR.

- Kernel: Radial basis function
- C: 100

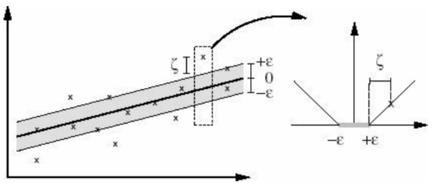


Fig. 3 ε-tube and ε-loss foundation of SVR

- Epsilon: 0.1

3.2 Sequence Modeling Approach

Sequence models are deep-learning algorithms which can traveling through time for processing sequence data, e.g., textual data, time-series data, etc. They have proven by many research works that are suitable for machine translation, video processing, sentiment analysis and sarcasm detection applications [13]–[16]. We select the following sequence models for this study:

3.2.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) that have memory mechanism first proposed by Hochreiter in 1997 [17]. A LSTM is composed of a cell, an input gate, an output gate and a forget gate as shown in Fig. 4. A cell remembers values in time-series data and the three gates control the flow of information into and out of the cell.

A LSTM was designed to model sequence that have long-range dependencies, which solved the exploding and vanish gradient problems. These problem can be normally founded when training the long sequential data such as text sequence in the traditional RNN. It is selected for this study due to this reason.

The equations of a LSTM cell are shown at Eq. (2) below.

$$\begin{aligned} i_t &= \sigma(x_t U^i + h_{t-1} W^i) \\ f_t &= \sigma(x_t U^f + h_{t-1} W^f) \\ o_t &= \sigma(x_t U^o + h_{t-1} W^o) \\ \tilde{C}_t &= \tanh(x_t U^g + h_{t-1} W^g) \\ C_t &= \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \\ h_t &= \tanh(C_t) * o_t \end{aligned} \tag{2}$$

where i_t, f_t, o_t represents an input gate, a forget gate, and an output gate respectively. W is the recurrent connection at the previous hidden layer and current hidden layer. U is the weight matrix connecting the inputs to the current hidden layer. $\tilde{C}$ is a candidate hidden state that is computed based on the current input and the previous hidden state. C is the internal memory of the unit. h is output hidden state, computed by multiplying the memory with the output gate.

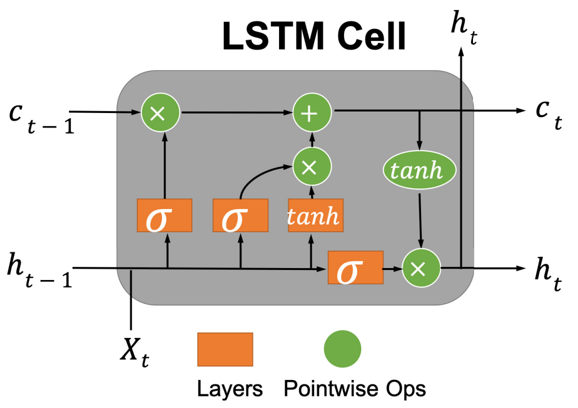


Fig. 4 LSTM architecture

In this study, we implemented a many-to-one LSTM with fully connected layers. We fed word vectors in each project description sentence into the LSTM network. The outputs of LSTM network are connected with the group of the Fully Connected (FC) networks to predict the man-month estimation. There is total approximately 20,145,883 trainable parameters. Its layers are listed as follows.

1. Input Layer (1, 143): Index of words respected to the GloVe [8] dataset of an input sentence describing the project characteristic from training set. The maximum length of input sentence is 143. For any sentences which have length less than 143 words, they are filled by 0 value.
2. Embedding Layer (1, 143, 50): The embedding 50 dimension vector representing 143 words, it is initialized from the GloVe [8] dataset.
3. Many-to-Many LSTM Layer (1, 143, 128): Unidirectional 128-dimensional hidden state many-to-many LSTM memory cells with *tanh* activation function.
4. Dropout Layer: Dropout Layer with 0, 0.2, 0.5, 0.8 dropout rate.
5. Many-to-One LSTM Layer (1, 143, 128): Unidirectional 128-dimensional hidden state many-to-one LSTM memory cells with *tanh* activation function.
6. Dropout Layer: Dropout Layer with 0, 0.2, 0.5, 0.8 dropout rate.
7. FC (1, 32): 32 fully-connected neurons with *relu* activation function.
8. Dropout Layer: Dropout Layer with 0, 0.2, 0.5, 0.8 dropout rate.

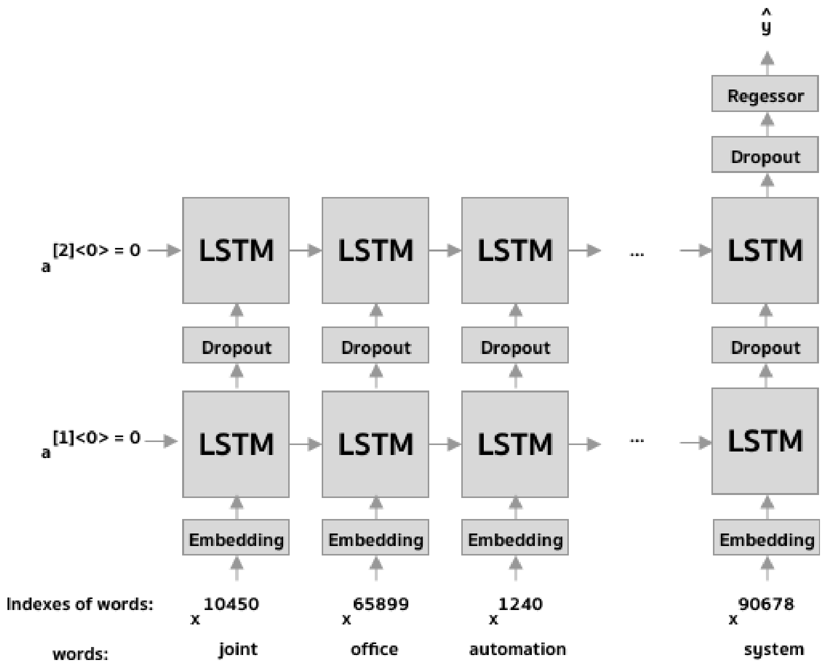


Fig. 5 Double LSTM layers with fully connected layers

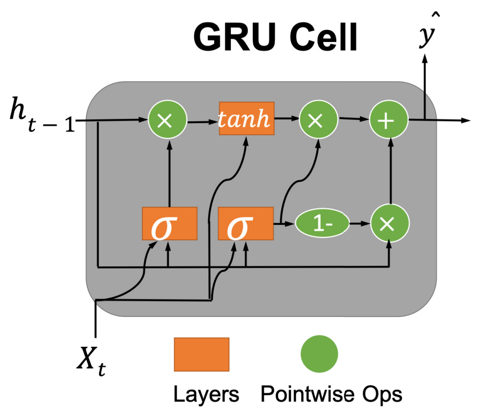


Fig. 6 GRU architecture

9. FC (1, 4): 4 fully-connected neurons with *relu* activation function.
10. Dropout Layer: Dropout Layer with 0, 0.2, 0.5, 0.8 dropout rate.
11. Output Layer (1, 1): Single output neurons without activation function.

We create variants LSTM structures based on the above architecture as follows.

1. **Single vs Double Layer(s)**: In single LSTM layer, we remove the Many-to-Many LSTM layer (layer 3) and the Dropout layer (layer 4). For double LSTM layers, we use the original architecture.
2. **Uni-directional vs Bi-directional**: For Bi-directional layer setting, we replace both Many-to-Many LSTM layer (layer 3) and Many-to-One LSTM layer (layer 5) with the bi-directional LSTM.
3. **With vs Without the Fully-Connected layer (FC)**: For without FC setting, we remove FC(1,32), Dropout, FC(1,4), and Dropout layers (layers 7-10). For with FC setting, we leave it as it.

3.2.2 Gated Recurrent Units (GRUs)

Gated Recurrent Units (GRUs) were first introduced by Kyunghyun Cho et al. [18] which is a simplified version of a LSTM network as shown in Fig. 6. Instead of having two separate gates, an input and an output gate is combined as the reset gate. The reset gate determines how to combine the new input with the previous memory. An forget gate was renamed to the update date which defines the size of the previous memory to keep around in the cell.

Although the number of parameters is much fewer, the performance of GRUs on various tasks such as speech signal modeling [19] was found similar to that of the LSTM. Compared with the LSTM, the GRUs have been exhibited better performance on some smaller datasets. The equations of a GRU cell are shown at Eq. (3) below.

$$\begin{aligned} z_t &= \sigma(x_t U^z + h_{t-1} W^z) \\ r_t &= \sigma(x_t U^r + h_{t-1} W^r) \\ \tilde{h}_t &= \tanh(x_t U^h + (r_t * h_{t-1}) W^h) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned} \tag{3}$$

where r and z is a reset gate, and a update gate respectively. W is the recurrent connection at the previous hidden layer and current hidden layer. U is the weight matrix connecting the inputs to the current hidden layer. h is output hidden state.

In this paper, we implemented a many-to-one GRU with fully connected layers. The GRUs structure for this study is exactly the same with the LSTM structure visualized in the previous section, except all LSTM cells (in layer 3, and layer 5) are replaced by GRU cells as described in Sect. 3.2.1. There is total approximately 20,145,883 trainable parameters. The variations of GRU structures for seeking the best configuration in the experiment are the same with the LSTM shown in Sect. 3.2.1.

4. Experimental Settings

We evaluate the performance of four architectures by finding the lowest mean absolute error (MAE) between 10 - 200 epochs on test datasets. MAE $= \frac{1}{n}\sum_{t=1}^{n} |(Y - \hat{Y})|$. Note that, all models were trained against the mean square error (MSE) as the loss function since it is outlier insensitive. The MAE was chosen as the evaluation metric since it is easier to understand. The experiments were early stopped if there is no improvement in the test loss after 10 epochs. The training and test datasets are generated by using ten-fold cross validation.

We chose Adam optimizer. It combines the best properties of the AdaGrad and RMSProp algorithms which can handle sparse gradients on noisy problems [20]. The dropout regularization is chosen since it is very computationally cheap and remarkably effective regularization method [21].

4.1 Hyper-Parameters

All experiments used the following hyper-parameters as follows..

- Epoch: 10-200
- Loss: Mean Square Error (MSE)
- Optimizer: Adam Optimization
- α: 0.001
- β_1: 0.9
- β_2: 0.999
- Fuzz factor: disabled
- Decay rate: 0.0
- AMSGrad: disabled
- Regularization: Dropout
- Dropout rate: 0, 0.2, 0.5, 0.8

The values of followings parameters: α, β_1, β_2, fuzz

Table 4 Experimental result on ISEM dataset

Method	Without Dropout	0.2-Dropout	0.5-Dropout	0.8-Dropout
Mean Average (Based Line)	1.983			
Support Vector Regression (SVR)	1.971			
Multi-Layer Perceptron (MLP)	1.758	1.626	1.433	1.129
Single Uni-LSTM without FC	1.310	1.212	1.276	1.236
Double Uni-LSTM without FC	1.148	1.208	1.261	1.183
Single Bi-LSTM without FC	1.091	1.182	0.991	1.100
Double Bi-LSTM without FC	0.724	**0.705**	0.832	0.803
Single Uni-LSTM with FC	1.333	1.567	2.253	1.850
Double Uni-LSTM with FC	1.414	1.628	2.164	1.878
Single Bi-LSTM with FC	1.168	1.499	2.369	2.943
Double Bi-LSTM with FC	1.298	1.256	2.599	2.703
Single Uni-GRU without FC	1.180	1.137	1.097	1.134
Double Uni-GRU without FC	0.989	0.940	1.041	1.092
Single Bi-GRU without FC	1.169	1.345	1.308	1.278
Double Bi-GRU without FC	0.859	0.846	0.815	1.013
Single Uni-GRU with FC	1.739	1.348	2.077	1.867
Double Uni-GRU with FC	1.612	1.873	2.039	1.911
Single Bi-GRU with FC	1.326	1.535	3.190	2.790
Double Bi-GRU with FC	1.172	1.182	2.887	3.265

Table 5 Experimental result on ISBSG dataset

Method	Without Dropout	0.2-Dropout	0.5-Dropout	0.8-Dropout
Mean Average (Based Line)	17.590			
Support Vector Regression (SVR)	16.413			
Multi-Layer Perceptron (MLP)	**14.069**	14.342	14.870	15.416
Single Uni-LSTM without FC	14.900	14.909	14.951	14.903
Double Uni-LSTM without FC	14.822	14.573	14.505	14.543
Single Bi-LSTM without FC	14.216	14.365	14.331	14.596
Double Bi-LSTM without FC	14.189	**14.077**	14.140	14.262
Single Uni-LSTM with FC	15.000	14.582	14.236	14.274
Double Uni-LSTM with FC	14.853	14.424	14.255	14.288
Single Bi-LSTM with FC	14.696	14.127	14.511	15.166
Double Bi-LSTM with FC	14.285	14.161	14.318	15.111
Single Uni-GRU without FC	14.998	15.023	15.131	14.876
Double Uni-GRU without FC	15.254	15.380	15.265	15.285
Single Bi-GRU without FC	14.691	14.822	14.584	14.616
Double Bi-GRU without FC	15.717	14.559	14.370	14.286
Single Uni-GRU with FC	14.762	14.495	14.253	14.297
Double Uni-GRU with FC	14.910	14.556	14.322	14.269
Single Bi-GRU with FC	14.977	14.278	15.125	15.712
Double Bi-GRU with FC	14.541	14.301	14.612	15.442

factor, decay rate, AMSGrad, and dropout rates are recommended in Adam Optimization and Dropout original papers [20], [21] and in the Keras deep-learning library website[†].

5. Experimental Results

The detailed experimental results for ISEM and ISBSG datasets are show in Table 4 and Table 5 respectively.

Compared between four proposed methods, the sequence models mostly obtained the lowest mean absolute error, followed by the the multi-layer perceptron, followed by the support vector regression, and finally the based line method.

[†]https://keras.io/optimizers/

For the ISEM dataset, the LSTMs architecture achieved the lowest mean absolute error at 0.705 man-months. The best configuration is the double-layers bi-directional LSTMs with single neuron as the output layer.

For the ISBSG dataset, the multi-layer perceptron (MLP) achieved the lowest mean absolute error at 14.069 man-months. Note that, the second lowest come from the double-layers bi-directional LSTMs with single neuron as the output layer like in the ISEM dataset at the mean absolute error of the 14.077 man-months. It is tiny difference.

After we examine on the experimental results, we see the effect of parameters as follows.

1. Sequence models generally give higher accuracy than Fully-connected layers.
2. Most of the time, the double-layer sequence models are

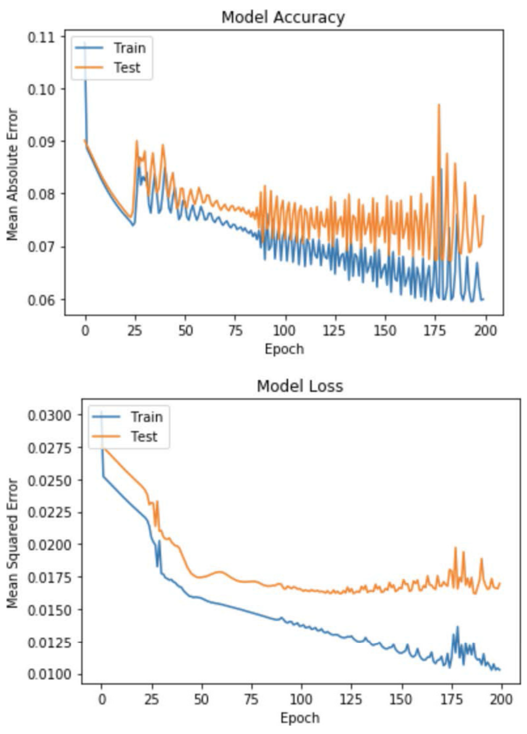


Fig. 7 Mean absolute errors and losses (mean square errors) of the LSTM model of training and test sets over 200 epochs

mostly obtained lower mean absolute errors than the single-layer sequence models.

3. Bi-directional sequence model obtained smaller mean absolute error compared to Uni-directional sequence model.
4. LSTM gave a slightly better performances than GRU.

Mean absolute errors and loss (mean square error) of a LSTM model of training and test sets over 200 epochs are visualized on Fig. 7. Most of the time, the models are early stopped learning about 75-125 epochs since there is no improvement in the test loss for 10 epochs.

We have investigated the effect of technical terms in software project description which is normally in the software development domain. We found small amount of the out-of-vocabulary (OOV) problem, since the GloVe [8] word embedding extracted vocabularies from the Wikipedia. It has a fair amount of technical terms such as MySQL, JSP, and Java. We have minor effect on the internal jargons of each software development company, since the ISEM and ISBSG are quite formal datasets. They are rarely using the self-developed jargons in the software text description.

6. Conclusion

This study explored the possibilities of helping the software companies to quickly estimate an accurate development efforts by artificial intelligence. This tool can greatly accelerate sale processes of software development businesses. Their customers can get a roughly price estimation of any software project instantly by just typing down the project text descriptions, which boost the great customer experience. Moreover, this tool will greatly support the project management process. The miscalculations from overconfident developers or diffident programmers normally create the money loss. Using this tool can create fair references for their estimations.

The comparisons between state-of-the-art traditional machine learning algorithms and deep learning sequence models for software efforts estimation on unstructured project description are evaluated. achieved the lowest mean absolute errors, which are 0.705 and 14.069 man-months using LSTM and MLP networks on the ISEM and the ISBSG software project datasets respectively. Several best practices for deep learning algorithms are discovered in this paper such as the usage of the fully-connected layers, the usage of the bi-directional layers, and the best number of layers of the sequence models.

For future works, we aimed to expand our studies in evaluating software efforts for small programming tasks which are helpful for software agile developments. The problems will be harder because of the shorter text. We found out that the language models from transforms such as the Bidirectional Encoder Representations from Transformers (BERT) are quite interesting [22].

Acknowledgments

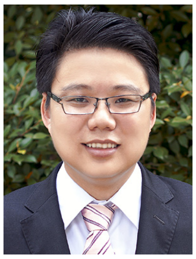
This research is financially supported under the Thammasat University's research fund, Center of Excellence in Intelligent Informatics, Speech and Language Technology and Service Innovation (CILS), and Intelligent Informatics and Service Innovation (IISI) Research Center, the Thailand Research Fund under grant number RTA6080013, the Thammasat University Fund on Research on Intelligent Informatics for Political Data Analysis, Royal Institute project of the e-test system, the Faculty Development Fund at Burapha University, Chanthaburi Campus, as well as the STEM workforce Fund by National Science and Technology Development Agency (NSTDA).

References

[1] L.H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," IEEE Trans. Softw. Eng., vol.SE-4, no.4, pp.345–361, 1978.
[2] C.F. Kemerer, "An empirical validation of software cost estimation models," Commun. ACM, vol.30, no.5, pp.416–429, 1987.
[3] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," IEEE Trans. Softw. Eng., vol.23, no.11, pp.736–743, 1997.
[4] J. Alzubi, A. Nayyar, and A. Kumar, "Machine learning from theory to algorithms: an overview," Journal of Physics: Conference Series, vol.1142, p.012012, IOP Publishing, 2018.
[5] A.L.I. Oliveira, "Estimation of software project effort with support vector regression," Neurocomputing, vol.69, no.13-15, pp.1749–1753, 2006.
[6] I..I. of Software Engineering improvement and quality Management, "About us, year = 2007, url = http://www.isem.co.th/home/index.php/about-us, urldate = 2007."

[7] M. Fernández-Diego and F. González-Ladrón-De-Guevara, "Potential and limitations of the isbsg dataset in enhancing software engineering research: A mapping review," Information and Software Technology, vol.56, no.6, pp.527–544, 2014.
[8] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," Proc. 2014 conference on empirical methods in natural language processing (EMNLP), pp.1532–1543, 2014.
[9] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective LSTMs for target-dependent sentiment classification," arXiv preprint arXiv:1512.01100, 2015.
[10] X. Chen, L. Xu, Z. Liu, M. Sun, and H. Luan, "Joint learning of character and word embeddings," 24th International Joint Conference on Artificial Intelligence, 2015.
[11] G. Cybenko, "Approximation by superpositions of a sigmoidal function," Mathematics of control, signals and systems, vol.2, no.4, pp.303–314, 1989.
[12] D. Basak, S. Pal, and D.C. Patranabis, "Support vector regression," Neural Information Processing-Letters and Reviews, vol.11, no.10, pp.203–224, 2007.
[13] C.-H. Wu, J.-M. Ho, and D.T. Lee, "Travel-time prediction with support vector regression," IEEE Trans. Intell. Transp. Syst., vol.5, no.4, pp.276–281, 2004.
[14] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang, "Hierarchical recurrent neural encoder for video representation with application to captioning," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.1029–1038, 2016.
[15] J. Wang, L.C. Yu, K.R. Lai, and X. Zhang, "Dimensional sentiment analysis using a regional cnn-lstm model," Proc. 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp.225–230, 2016.
[16] L.H. Son, A. Kumar, S.R. Sangwan, A. Arora, A. Nayyar, and M. Abdel-Basset, "Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network," IEEE Access, vol.7, pp.23319–23328, 2019.
[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol.9, no.8, pp.1735–1780, 1997.
[18] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," CoRR, vol.abs/1406.1078, 2014.
[19] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Improving speech recognition by revising gated recurrent units," CoRR, vol.abs/1710.00641, 2017.
[20] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," The Journal of Machine Learning Research, vol.15, no.1, pp.1929–1958, 2014.
[22] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

Tachanun Kangwantrakool is CMMI Lead Appraiser Certificate (Certificate of Lead Appraiser for Capability Maturity Model Integration): Standard for Project Management Process Category, Quality Management Process Category, Process Improvement Process Category, Engineering Process Category.



Kobkrit Viriyayudhakorn received a bachelor degree in Computer Science from Sirindhorn International Institute of Science and Technology in 2008, and master degree in Computer Engineer (Embedding System) from TAIST Tokyo Tech in 2010, and a doctoral degree in Knowledge Science from Japan Advanced Institute of Science and Technology in 2013. His research interests are artificial intelligence and machine learning such as natural language processing, deep learning, speech recognition, computer vision and creativity support systems.

Thanaruk Theeramunkong received a bachelor degree in Electric and Electronics, and master and doctoral degrees in Computer Science from Tokyo Institute of Technology in 1990, 1992 and 1995, respectively. He is currently a professor at School of Information, Computer and Communication Technology at Sirindhorn International Institute of Technology (SIIT) at Thammasat University, Bangkok, Thailand. He also is the Program Director of Information and Communication Technology for Embedded Systems (ICTES) at TAIST Tokyo Tech, National Science and Technology Development Agency (NSTDA). He is also an Associate Fellow, Academy of Science, the Royal Society of Thailand. As a professional society, He is the president of Artificial Intelligence Association of Thailand. He serves as an academic committee to the Industrial Section, National Research Council of Thailand (NRCT). His research interests are natural language processing, data mining, text mining, machine learning and applications to service science. He was an associate editor of the Institute of Electronics, Information and Communication Engineers (IEICE). He is a member of the Steering Committee of the Pacific-Asia Conferences on Knowledge Discovery and Data Mining (PAKDD) and a member of the Steering Committee of the Pacific Rim International Conferences on Artificial Intelligence (PRICAI). He is the author of more than 45 papers in a number of journals with impact factors and more than 130 conference papers.