Extreme Learning Machine in Thai Language Processing

Boonnithi Jiaramaneepinit
Satha Chaojaroenrat

Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2016
KMITL-2017-IC-B-003-006

Thesis – Academic Year 2016

Bachelor of Engineering in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title: Extreme Learning Machine in Thai Language Processing

Authors:

1. Mr. Boonnithi Jiaramaneepinit Student ID 56090002
2. Mr. Satha Chaojareonrat Student ID 56090024

Approved for submission

.....

(Assistant Professor Dr. Chaiwat Nuthong)

Advisor

Date 29 / 01 / 2017

Abstract

Natural language processing (NLP) has been huge challenges for many researchers in related fields over the past decades. One of these challenges includes how to make a machine understands a specific language comparable to human. A success of such task or alike in NLP would lead to many real-world applications. However, currently, such applications have been developed based on few languages, e.g. English, German, Japanese. Hence, there are many other languages, including Thai language, which have not been studied intensively for such research fields. One of the reasons is their scarcity in resources and studies.

The proposed work focuses on Thai language. As previously mentioned, the language itself has severe difficulties, for examples, indicating word boundaries, defining a minimal unit, etc. The goal of this work is to develop an effective system that processes Thai language. The system is expected to perform three subtasks, involving tokenization, structurization, and entity recognition by incorporating a novel learning algorithm, namely Extreme Learning Machine (ELM). Moreover, the performance of each subtask will be compared with well-known methods.

There are three modules in the proposed system. The first module is responsible for tokenization using longest matching approach. The module is tested based on ORCHID corpus. The results show 96.04% accuracy with average computational time of 0.11 s. The second module performs part-of-speech tagging in order to structurize text using neural networks (NN) with backpropagation and ELM as learning algorithms. It is tested on Open American National Corpus. The system achieves accuracy of 86.14% and 90.14% with backpropagation and ELM, respectively. The final module is designed for named-entity recognition, which is one of semantic analysis tasks. It incorporates word embedding using Continuous Bag of Words (CBOW) with NN for binary classification. The results show 95.43% accuracy for backpropagation and 94.43% for ELM. Nevertheless, ELM could provide high accuracy with significant lesser training time compared to backpropagation, in trading-off on memory consumption.

Acknowledgments

We would like to express our deep gratitude to Asst.Prof.Dr. Chaiwat Nuthong, our advisor, for providing us vast knowledge, enthusiastic encouragement, useful assessment, and many forms of support with his best abilities. We would like to thank Dr. Isara Anantavrasilp also for giving us criticism and guideline for making interesting and understandable presentation. Special thanks to Dr. Prachya Boonkwan for his professional guidance during his visit. Thanks to friends and professors at Applied Machine Learning Laboratory and Computer Vision Laboratory at the International College, King Mongkut's Institute of Technology Ladkrabang as well for giving us helpful advices, supportive discussions, and points from another perspective even with their tight schedule. Our grateful thanks are also extended to Sasawat Chanate for assisting in the editing of this thesis. Furthermore, we would like to thank both our families for unceasing encouragement and support. Lastly, we would like to express sense of gratitude to all, who directly or indirectly, have a part in this venture.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The idea of making machines understand human languages, or nowadays also known as natural language processing (NLP), has been with humans since even before the first computer was invented. It has been becoming one of the popular research fields in the past decades. In recent years, advancement in hardware and software researches provides substantially progress and development in many research fields, including NLP. Thus, hardware and software concerning NLP applications currently become more practical and easier to use. However, the difficulties of many NLP applications still persist both on languages themselves and their applied NLP methods. Since each language has its own characteristics, it is not entirely possible to use specific approaches for one language on another. In addition, for specific language, several approaches are needed to be investigated in order to achieve desired performance. Hence, the challenge in making machines understand every language in the world still persists even today.

Nowadays, there is only a small amount of NLP applications in few languages, e.g. English, Chinese, Japanese, etc. Although some NLP applications based on those languages are found to work well, it is not possible to apply the same methods with other languages, including Thai. Difficulties of applying such methods in Thai language come from various reasons, e.g. indicating word boundaries, defining a minimal unit, etc. In addition, Thai language itself has ambiguity in the *word*, which is the most primitive semantic unit in a language, reported by Sornlertlamvanich et al. [1]. According to work from Aroonmanakun [2], Thai words and sentences segmentation has

several major problems. The first problem is to indicate word boundaries. As Thai has no spaces to indicate the end of a word, there exists ambiguities in defining word boundaries. For example, a Thai phrase "ตู้เสื้อผ้าสีขาว" can be structurized as follows:

- (ตู้((เสื้อผ้า)สีขาว)) means 'a closet for white clothes'
- ((ตู้(เสื้อผ้า))สีขาว) means 'white closet'

Another problem is to find a minimal unit of words, such as, "แม่น้ำ" (river) which is already desirable in some cases even though it can be broken down further into "แม่" (mother) and "น้ำ" (water). Another challenge is the lack of explicit sentence boundary, which makes marking for sentence boundaries complicated.

The other difficult part is the process of optimal method selection to be used in teaching computers to understand human languages. This leads to the usage of machine learning. There are varieties of machine learning algorithms that could be applied in Thai language processing applications. These algorithms have different advantages and disadvantages. Hence, the investigation of each algorithm in order to achieve required performance is essential. In the past decades, one of the well-known methods found in NLP is neural network with backpropagation. However, backpropagation approach suffers various difficulties, for examples, intensive human intervention, slow learning speed, poor learning scalability, etc. Recently, there is a new promising learning algorithm, extreme learning machine (ELM), to be used for teaching neural network which claims that it can solve aforementioned difficulties [3]. To the authors' knowledge, such algorithm has not been applied to Thai language processing. This holds a potential to contribute advancement in Thai language processing.

1.2 Problem Descriptions

NLP for Thai language has not been widely developed compared to other languages, since Thai language is used only in restricted area and has its own difficulties. Furthermore, studies for Thai language processing in the past normally focus on one certain task which performs well on predetermined resources. This is due to complexities and complications presented in the language. As a

result, studies on Thai language processing pale in comparison to the others. From these reasons, many modern well-received approaches have not been well-adopted for Thai language processing.

In learning aspect of Thai language processing, selecting modern well-known approaches is still a difficult task. Algorithms, such as decision tree and neural network with backpropagation, have been found to be used on such problem. Recently, there is a novel algorithm, ELM, which is reported to have a promising result. However, results of applying ELM in Thai language processing have not been reported. Therefore, this research focuses on applying and comparing performances between ELM and backpropagation approach on Thai language processing.

1.3 Research Objectives

This research studies Thai language processing and applies ELM and backpropagation approaches on it. The objectives to be achieved are set as follows:

- To build a system that tokenizes, structurizes, and interprets Thai language
- To build a part-of-speech classifier module that is given a word and responds back part-of-speech tagging, with accuracy of at least 80%
- To build a named-entity classifier module that is given a word and decides whether named-entity or not, with accuracy of at least 80%
- To study the effectiveness of integrating ELM into natural language processing applications
- To compare the performance between ELM and backpropagation approach

1.4 Scope of the Study

This research focuses on tokenizer, part-of-tagging classifier, and named-entity classifier. It comprises of three main modules. The first module performs lexical analysis, which splits words from input text sequence. Then, multi-class classifier for part-of-speech is performed in the second module. Result from this module is part-of-speech of each word as grammatical structure. In the third module, binary classifier for named-entity is performed.

Performance measurement of each module is solely performed on Thai language with occasional use of other languages as a support of conceptual ideas. In addition, performance is measured in term of accuracy and computational time. The resources for Thai language processing in the study are dictionaries and corpora. The dictionary used in the study is the 1999 edition of the Royal Institute Dictionary, while corpora are ORCHID and BEST I corpora from NECTEC.

1.5 Expected Contribution

The expected contribution of this study is providing novel approaches in Thai language processing with better performance. More specifically, this study applies ELM in Thai language processing and also compares the performance with the well-known machine learning method, i.e. neural network with backpropagation. As a result, this can be beneficial for determining plausibility in using ELM for Thai language processing.

Chapter 2

Literature Review

This chapter is devoted for the literature review, which includes past studies and theories that related to this research. There are two main topics to be discussed: NLP and machine learning approaches. Detailed information can be found in following sections.

2.1 Natural Language Processing (NLP)

Natural language processing (NLP) is an ability of computer software to understand human language. In general, NLP comprises of five steps: lexical analysis, syntactic analysis, semantic analysis, disclosure integration, and pragmatic analysis [4]. However, due to limited time, this research only focuses on the first three steps. Detailed processes in each step is discussed in following subsections.

2.1.1 Lexical Analysis

This part focuses on segmenting input sequences of string into words, numbers, and/or groups of special characters. This is also known as string tokenization. Each language has different difficulty in doing tokenization. Thai language is a non-segmenting language, which means that there are no spaces between each word in a sentence. Thus, it has high complexity in tokenization process. This is one of problems in performing NLP in Thai language. There are several methods that are proposed for solving this problem, such as longest matching, maximal matching, part-of-speech

tri-gram modeling, machine learning, etc. [5]. Comparing the previously mentioned methods, machine learning approach is claimed to be the most accurate method [6].

2.1.2 Syntactic Analysis

This part focuses on structurizing sentences according to grammatical rules. It leads to part-of-speech (POS) tagging. POS tagging is a process which categorizes words into classes. In a past study, Viterbi algorithm is applied to compute the most probable sequence of POSs from tri-gram word segmentation model. From authors' knowledge, ORCHID is the only Thai POS tagset and corpus available publicly [7]. It is used in experiment of this study. From a work of Mittrapiyanuruk et al. [8], tokenizing a paragraph into sentences before perform POS tagging claims to reduce error rate by 11.3%. Moreover, another approach proposed by Levy et al. [9], which uses neural word embedding as implicit matrix factorization, claims to increase syntactic parsing accuracy in syntactic analysis. In addition, the model for creating word embedding is proposed with the study of Skip-Gram with Negative Sampling (SGNS). The objectives of using SGNS are two folds: maximizing similarity between words with similar contexts and maximizing dissimilarity between words with different contexts.

2.1.3 Semantic Analysis

Semantic analysis can be defined as a process of finding meanings of speech. This research selects named-entity recognition as a task to be processed for semantic analysis. Named-entity recognition or extraction in Thai language has, compared to many other languages, only few numbers of researches. One of them is named-entity recognition model based on Conditional Random Fields, which was reported by Tirasaroj et al. [10]. Their proposed system required word lists and dictionaries as clues to predict named entities. It achieved 92.54% precision and 72.06% recall. Another model proposed by Chanlekha et al. [11] incorporated Maximum Entropy model and achieved more consistent results with 87.6% precision and 87.8% recall. Both models required hand-picked features in order to classify named-entity, such as word features, dictionary features, lexical features, etc.

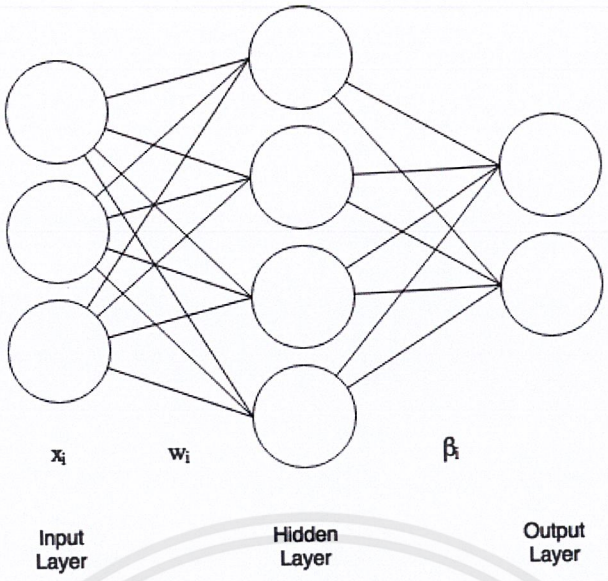

Figure 2.1: A single hidden layer feedforward neural network

2.2 Machine Learning

Machine learning is an ability of computer software to solve a problem without explicitly programmed. It is considered as part of artificial intelligence. There are various models of machine learning available. However, this research focuses solely on single-hidden layer feedforward neural network (SLFN) model with the usage of extreme learning machine and backpropagation approach as its learning algorithms. The details of these algorithms will describe in following subsections.

2.2.1 Single-Hidden Layer Feedforward Neural Network

A SLFN is the simplest artificial neural network as there are no cycles in it. In this type of neural network, data moves in only one direction, from the input layer to the output layer through the hidden layer, as shown in Figure 2.1. Each node in each layer is a summation of products of nodes' output in the previous layer and weights between layers. Moreover, before sending data out, the data passes through an activation function in order to create non-linearity in neural network model. This allows the network to learn more complex tasks.

An activation function, e.g. sigmoidal function, and a gradient descent learning algorithm, e.g. backpropagation, are components of a neural network [12]. Nevertheless, the execution of backpropagation on large neural networks has serious drawback regarding the performance due to

the need of iterative cycles for performing calculation [13].

2.2.2 Extreme Learning Machine

Extreme Learning Machine (ELM), the recent proposed learning algorithm, shows a better generalization performance for feedforward neural networks. It was proposed to solve intensive human intervention, slow learning speed, and poor learning scalability of feedforward neural networks [3]. Furthermore, this paper shows that the learning algorithm could significantly reduce the training time. In addition, there are many applications that could achieve high accuracy with ELM, i.e., Modified National Institute of Standards and Technology (MNIST) optical character recognition (OCR), traffic sign recognition, and three-dimensional shape classification.

For MNIST OCR, Deep Belief Networks and neural network with ELM were compared by Zhu et al. [14]. The results show that Deep Belief Network could achieve testing accuracy of 98.87% with the training time of 5.7 hours using GPU. While using ELM with multiple hidden layers and auto encoder results in the testing accuracy of 99.14% with the training time of 281.37 seconds using CPU. In this application, even using lower computational power, ELM still outperformed Deep Belief Networks.

For three-dimensional shape classification, Convolutional Deep Belief Net'work and neural network with ELM were compared by Wu et al. [15]. From the experiment, Convolutional Deep Belief Network achieves testing accuracy of 77.32% with the training time of more than 2 days using GPU. While ELM with multiple hidden layers and local receptive fields achieves the testing accuracy of 81.39% with the training time of 306.4 seconds using CPU. All in all, ELM outperformed in both accuracy and training time.

ELM shows significant outperformed results in many applications. However, from the authors' knowledge, there is no application for Thai language up until now. Thus, this project focuses on studying and applying ELM on the two analysis phases of Thai language processing, which are syntactic analysis and semantic analysis. For the lexical analysis, graph theory is used in order to solve tokenization problem. Details are discussed in the following chapter.

Chapter 3

Background Knowledge

This chapter introduces various concepts related to this study and briefly clarifies important points which are foundations that this study is based on. In general, NLP consists of five analysis steps. For this project, first three steps, i.e., lexical analysis, syntactic analysis, and semantic analysis, are considered. Algorithm for tokenization is designed and applied for lexical analysis. For syntactic analysis and semantic analysis, neural network classifiers for POS tagging and named-entity recognition are designed and implemented, respectively. In experiments on POS tagging and named-entity recognition using neural network, performance of ELM and backpropagation are measured and compared.

3.1 Directed Acyclic Graph

A DAG is a directed graph with a finite number of vertices and edges and no cycles. The definition of having no cycle is that for any vertex v in a DAG, there is no possible path p from v back to v. This implies that there is no loop in a DAG. One of the important properties of a DAG is that every DAG has a topological ordering. A topological ordering is a linear ordering of vertices which all edges from vertex u to vertex t, vertex u will always come before vertex t in the order. Figure 3.1 is an example of a DAG. Figure 3.2 is not a DAG because there exists at least one path which can traverse back to a start vertex. Here is one of such paths: $v \rightarrow x \rightarrow y \rightarrow v$.

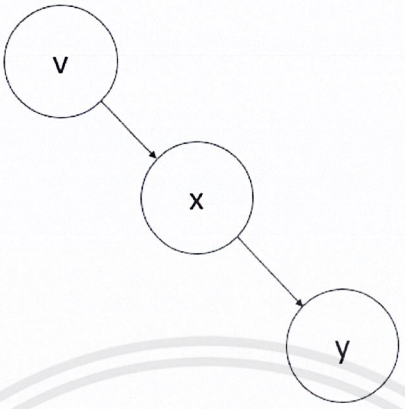

Figure 3.1: An example of a directed acyclic graph

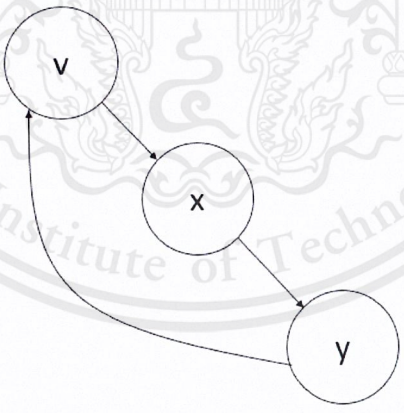

Figure 3.2: An example of a graph that is not a directed acyclic graph

3.2 Shortest-Paths Problem

A shortest-paths problem is one of common problems found in graph theory. According to Cormen et al. [16], given a weighted, directed graph $G = (\mathbf{v}, \mathbf{e})$, with vertors of vertices $\mathbf{v}$ and edges $\mathbf{e}$ mapped to weights $w \in \mathbb{R}$. The weight $w(\mathbf{p})$ is the sum of weight of edges in path $\mathbf{p}$. For any two vertices u and v, if there exists a path $\mathbf{p}$ from u to v, $\delta(u, v)$ defines as follow:

$$\delta(u, v) = min\{w(\mathbf{p}) : u \longrightarrow_{\mathbf{p}} v\} \tag{3.1}$$

A shortest path from u to v is defined as any path $\mathbf{p}'$ that satisfies following equation:

$$w(\mathbf{p}') = \delta(u, v) \tag{3.2}$$

The shortest-path problem will be used in lexical analysis. This helps in getting longest matching tokens from an input string. These will be used in further analysis steps.

3.3 Single-Hidden Layer Feedforward Neural Networks

A single-hidden layer feedforward neural network (SLFN) is a model of neural network with a single layer of hidden nodes. Similar to other types of artificial neural networks, SLFNs consist of a large number of simple processing units or nodes highly interconnected by directed weighted links. Each node outputs *an activation value* which propagates through the connected nodes.

Another two components of SLFNs are an activation function, e.g., sigmoid function, and a learning algorithm. At each node, products of weighted input values are summed and then passed through an activation function before propagating to other nodes. Before SLFNs are usable for classification or regression, they require a training session with training samples in order to adjust weights in weight vectors. Hence, learning algorithms in SLFNs are required to adjust weights of directed weighted links during training.

According to Huang et al. [3], for N samples $(\mathbf{x}_i, \mathbf{t}_i)$ where input vector $\mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{in}]^T \in \mathbb{R}^n$ and expected output vector $\mathbf{t}_i = [t_{i1}, t_{i2}, ..., t_{im}]^T \in \mathbb{R}^m$, SLFNs with $\hat{N}$ hidden nodes are mathematically modeled as

$$\mathbf{y}_j = \sum_{i=1}^{\hat{N}} \boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i), \quad \text{for} \quad j = 1, ..., N \tag{3.3}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ is the weight vector from input nodes to i^{th} hidden node, $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{in}]^T$ is the weight vector from the i^{th} hidden node to output nodes, and b_i is the threshold of the i^{th} hidden node.

In training with a learning algorithm, $\mathbf{y}_j$ in Equation (3.3) is substituted with expected output vector $\mathbf{t}_j$ and is written as

$$\mathbf{t}_j = \sum_{i=1}^{\hat{N}} \boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i), \quad \text{for} \quad j = 1, ..., N \tag{3.4}$$

Equation 3.4 can also be written as

$$\mathbf{T} = \mathbf{H}\boldsymbol{\beta} \tag{3.5}$$

where

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\hat{N}} \cdot \mathbf{x}_1 + b_{\hat{N}}) \\ \vdots & \ddots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_{\hat{N}} + b_1) & \cdots & g(\mathbf{w}_{\hat{N}} \cdot \mathbf{x}_{\hat{N}} + b_{\hat{N}}) \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_{\hat{N}}^T \end{bmatrix}, \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}.$$

$\mathbf{H}$ is called the hidden layer output matrix. The i^{th} column of $\mathbf{H}$ is the output of i^{th} hidden node with $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ as inputs.

This project uses SLFN as neural network model for classification purposes. The SLFN is used as a basis for both POS tagging and named-entity classifiers. Furthermore, backpropagation and ELM are selected as learning algorithms. The results are compared in experiments.

3.4 Learning Algorithm

A learning algorithm is an algorithm which is used for training a neural network model. The goal of training the model is to find proper weights that maximize accuracy of the model. There are various learning algorithm applied to the neural network, such as, gradient descent, newton's

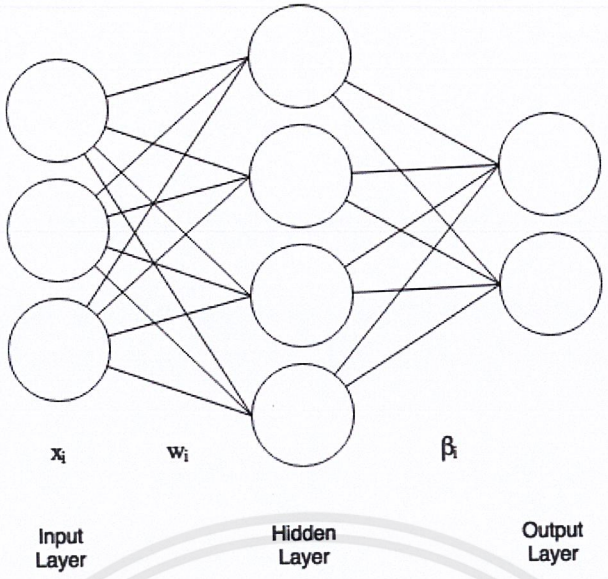

Figure 3.3: A single hidden layer feedforward neural network

method, conjugate gradient, etc. However, in this project, only two approaches are discussed, i.e., backpropagation with gradient descent and ELM.

3.4.1 Backpropagation Learning Algorithm

A backpropagation learning algorithm is one of the most popular learning algorithms. It is based on gradient descent optimization algorithm. The basic idea of gradient descent is to search for local minimum of a function by minimizing a cost function. Gradient-based algorithms usually are iterative and the backpropagation algorithm is not an exception. The backpropagation learning algorithm propagates the output to the input and iteratively adjusts weights and biases until the minimum of $||y_j - t_j||$ from Equation (3.3) and (3.4) is found.

Applying the backpropagation learning algorithm as shown in Equation (3.5), the vectors $\overline{\mathbf{w}}_i$, $\overline{\mathbf{b}}_i$, and $\overline{\beta}$ for $i = 1, ..., \hat{N}$ are needed to be adjusted during training such that

$$||\mathbf{H}(\overline{w}_1, ..., \overline{w}_{\hat{N}}, \overline{b}_1, ..., \overline{b}_{\hat{N}})\overline{\boldsymbol{\beta}} - \mathbf{T}|| = \min_{\mathbf{w}_i, \mathbf{b}_i, \boldsymbol{\beta}} ||\mathbf{H}(w_1, ..., w_{\hat{N}}, b_1, ..., b_{\hat{N}})\boldsymbol{\beta} - \mathbf{T}|| \tag{3.6}$$

which is equivalent to running gradient descent algorithm with the cost function E written as

$$E = \sum_{j=1}^{N} \left(\sum_{i=1}^{\hat{N}} \boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + \mathbf{b}_i) - \mathbf{t}_j \right)^2 \tag{3.7}$$

Suppose vector $\mathbf{W}$ is a set of weights and biases $\mathbf{w}_i$, β_i, and $\mathbf{b}_i$, the backpropagation learning algorithm finds the minimum of $||\mathbf{H}\boldsymbol{\beta} - \mathbf{T}||$ by iteratively adjusting $\mathbf{W}$ according to Equation (3.8).

$$\mathbf{W}_k = \mathbf{W}_{k-1} - \alpha \frac{\partial E(\mathbf{W})}{\partial \mathbf{W}} \tag{3.8}$$

where $\mathbf{W}_k$ and $\mathbf{W}_{k-1}$ are weight vectors at kth and $k-1$th iteration, respectively. While, α is a learning rate. There are several drawbacks on using backpropagation learning algorithm:

- α has to be manually set. If α is too small, learning algorithm progresses will be slow. However, if α is too large, the learning algorithm may not converge to the minimal point.
- The algorithm may converge and stop at local minima leading to undesirable results.
- Neural networks may be over-trained and become over-fitted by using backpropagation algorithm. Therefore, good validation and stopping methods are needed to prevent such problems.
- Gradient-based learning algorithms are time-consuming due to their iterative nature.

3.4.2 Extreme Learning Machine

Extreme learning machine (ELM) was at first proposed to be a new learning algorithm for single hidden layer feedforward neural networks [17]. The proposal of ELM points out and claims to solve two bottlenecks of the learning speed of feedforward neural networks: slow gradient-based learning and iterative parameters tuning approach.

ELM is based on the principle that not all parameters of SLFNs need to be adjusted. It claims that SLFNs can be used to approximate any continuous functions with arbitrary small error even if values for input weights $\mathbf{w}_i$ and hidden layer biases $\mathbf{b}_i$ are chosen arbitrarily at the beginning of training state[3].

In SLFNs, a learning algorithm has to find $\mathbf{w}_i$, $\mathbf{b}_i$, and $\boldsymbol{\beta}$ such that $||\mathbf{H}\boldsymbol{\beta} - \mathbf{T}||$ becomes minimum. Firstly, input weights $\overline{\mathbf{w}}_i$ and hidden layer biases $\overline{\mathbf{b}}_i$ are randomly assigned. Then, SLFNs are

trained mathematically using ELM, so that, Equation (3.6) is satisfied. The only value that needs an adjustment is $\overline{\beta}$. Then the smallest norm least-squares solution is calculated from

$$\overline{\beta} = \mathbf{H}^{-1}\mathbf{T} \tag{3.9}$$

where $\mathbf{H}^{-1}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$. Extreme learning machine holds several advantages over the backpropagation learning algorithm:

- Minimum training error as there is no issue dealing with local minima without using training iterations
- Smaller norm of weights which leads to better generalization performance [18]
- A unique minimum norm least-squares solution of $\mathbf{H}\beta = \mathbf{T}$

3.5 Word Embedding

Word embedding describes a set of feature extracting methods and a model of a language in natural language processing. There are several models that could be used in representing words. Mikolov et al. mentioned works of distributed representations of words, which claimed to be the most successful concept [19]. In another work, Bengio et al. proposed neural probabilistic language model [20]. Based on this concept and this model, Mikolov et al. improved a model and proposed new word representations using vector space. In addition to these representations, words are represented using vectors of real numbers called word vectors, which are used in this work.

In order to learn word vectors, predictive model is used. Two of the most well-known models are continuous bag-of-words model (CBOW) and continuous skip-gram model. They are claimed to have a better semantic accuracy compared to existing models, such as feedforward neural network language model and recurrent neural net language model [19].

Technically, CBOW and continuous skip-gram model are similar. The difference is that CBOW model predicts a word by using its surrounding words and context while skip-gram model predicts surrounding words by a given word, as shown in Figure 3.4 and 3.5, respectively.

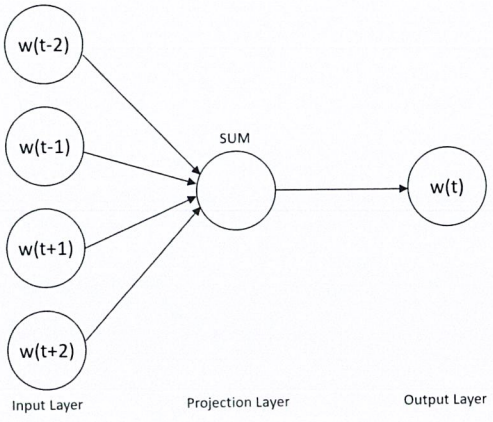

Figure 3.4: Continuous bag-of-words model

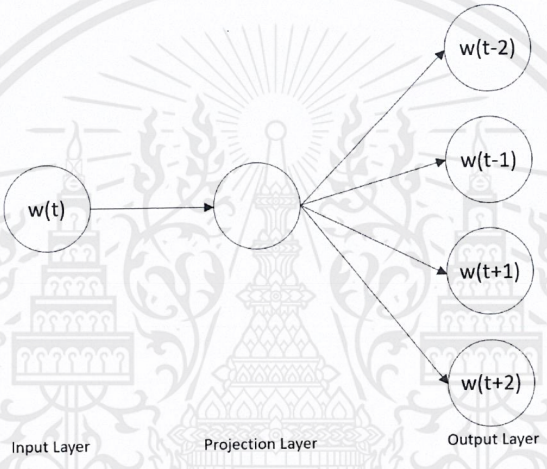

Figure 3.5: Continuous skip-gram model

This project applies DAG and shortest-paths algorithm on lexical analysis. For syntactic analysis, N-gram model and neural network are used. Furthermore, for semantic analysis, word embedding with neural network is applied. The details will be discussed in the following chapter.

Chapter 4

Methodology

This chapter introduces the methods proposed in this study. As mentioned previously, there are three major processes, i.e., tokenization, part-of-speech tagging, and named-entity recognition. The flow of the processes is shown in Figure 4.1.

4.1 Tokenization with Longest Matching

Tokenization in lexical analysis is a process to extract words, also called tokens, from an input text or string. There are several techniques that are used in solving a tokenization problem in different languages. An approach used in this study is to reformulate such problem into a graph theory problem, which will be shown in the following:

- First, the technique creates an ordered arrangement of the input text's characters where the number of the characters is l. Each character is then indexed starting from 0 to $l - 1$. This can be visualized as in Figure 4.2.

- Next, a DAG is constructed from the array of characters. A vertex on the graph represents each character. To construct edges, for any vertices u and v representing characters with indices i and j where $i < j$, there is an edge from a substring u to v if there is a word in a dictionary. The DAG is depicted in Figure 4.3.

- After completing a DAG, it results in a topological ordering according to property of DAG. The process of finding a longest match of the input text is searching for a shortest path on the

condition that the path starts from the starting vertex and ends at the ending vertex. The dark arrows in Figure 4.4 shows an example of the mentioned path.

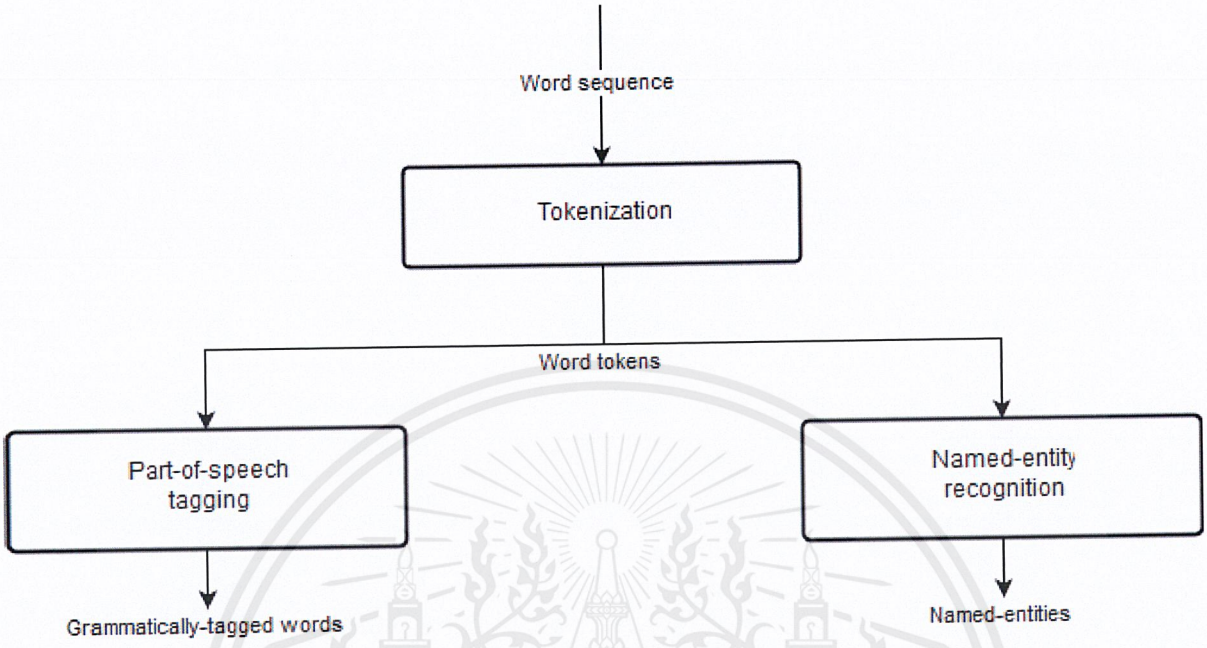

Figure 4.1: The flow of major processes

Figure 4.2: First step in tokenization

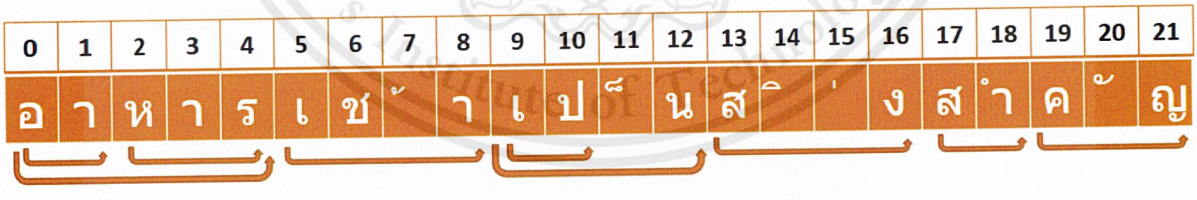

Figure 4.3: Second step in tokenization

The result after tokenization process with longest matching approach is a sequence of words. These words be will further used in part-of-speech tagging and named-entity recognition.

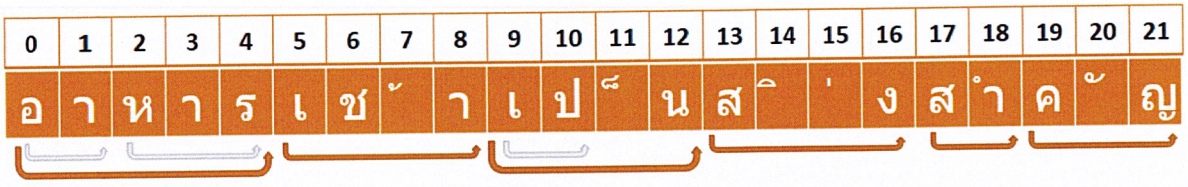

Figure 4.4: Third step in tokenization

4.2 Part-of-Speech Tagging using Neural Networks

Part-of-Speech tagging can be viewed as a classification problem in machine learning. Hence, using neural networks to solve is plausible. In this study, a model of SLFN previously shown in Figure 3.3 is applied. Figure 4.5 shows the structure of the input layer and output layer of the neural network.

In the output layer, each output unit y_j corresponds to a tag in the tagset. For training the neural network, an output unit that represents the correct tag is set to be activated while the others are set to be deactivated. In the trained neural network, the output unit with the highest activation value is considered the correct tag to the selected word.

As the neural network adopts n-gram model, classifying a word requires information from neighboring words. In the input layer, each input unit represents the lexical probability of each word w_i and each of part-of-speech tag p_j in the tagset. In other words, an input unit x_{ij} represents the probability that word w_i has part-of-speech p_j. Additionally, the total number of words in the input layer is $p + 1 + f$ words where p and f are the number of preceding words and the number . Mathematically, the lexical probability of a word can be represented as shown in Equation (4.1).

$$x_{ij} = P(p_j|w_i) \tag{4.1}$$

In the hidden layer, each hidden unit calculates a summation of product of input values and weights then passes the result through an activation function. In this model, each hidden unit has the sigmoid function as the activation function. A sigmoid function limits the value range of a resulting summation to $[0, 1]$. The mathematical formula of a sigmoid function is given as in Equation (4.2).

$$S(t) = \frac{1}{1 + e^{-t}} \tag{4.2}$$

For neural network training, a learning algorithm is needed to adjust weights of connections between units. In this study, two learning algorithms are applied separately to compare the performance. The first learning algorithm is backpropagation algorithm and the other is extreme learning machine.

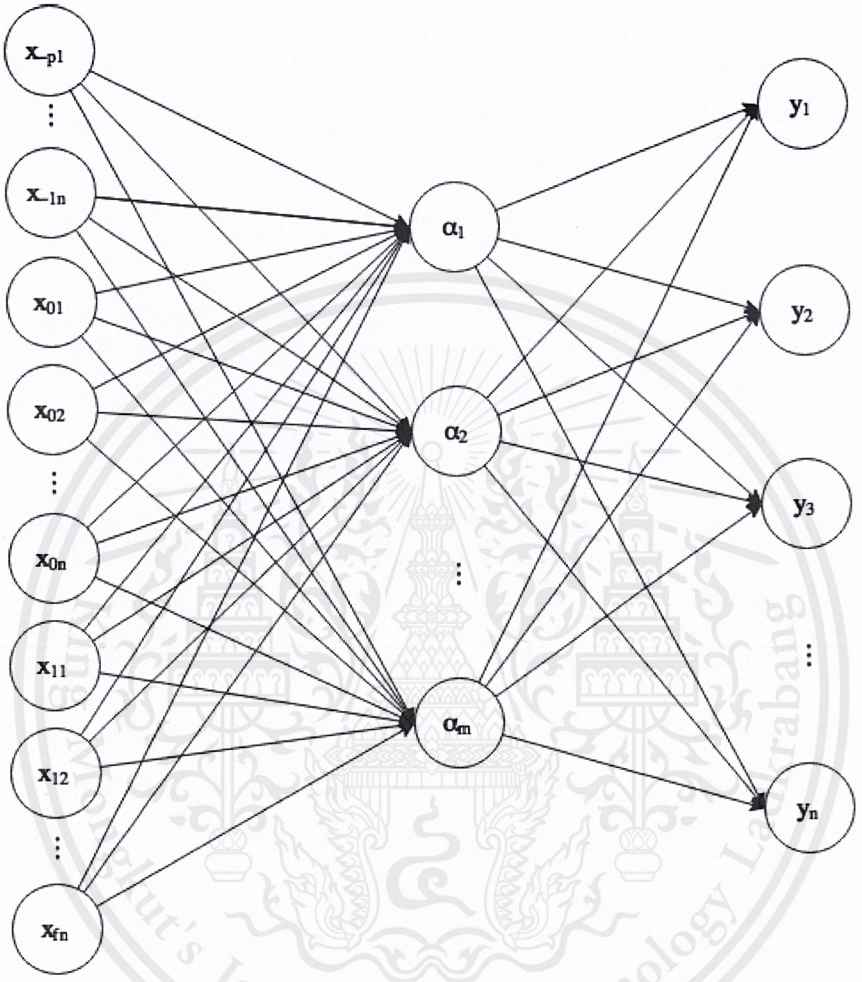

Figure 4.5: The structure of the input later and output layer

4.3 Named-Entity Recognition with Neural Networks

Named-entity recognition is a task in natural language processing to extract named entities from a given text. The keyword named entity has variety of meanings, such as, people, places, and time. In this study, the information to be extracted are subjects, actions, places, and time. The approach of this study is to use neural networks to learn from selected corpora to perform word classification.

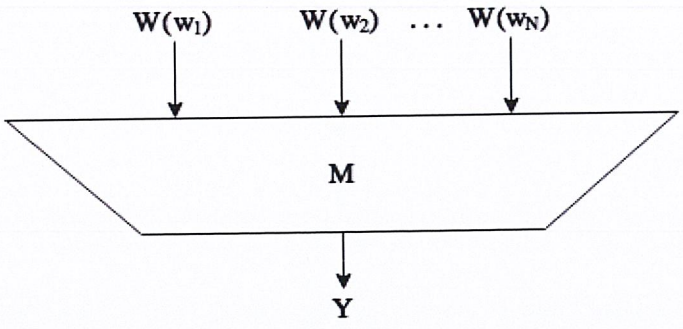

Figure 4.6: A simplified neural network model for named-entity recognition

In order to create a classifier model, features are required. These features are extracted from contextual windows of words in a corpus or corpora. The resulted features are word vectors, which are previously described in chapter 3.5. Words in a word vector W are then mapped to high-dimensional vectors. For example, words "ไก่" and "ไข่" may be represented as $W(\text{ไก่}) = (0.1, -0.2, 0.5, ...)$ and $W(\text{ไข่}) = (0.8, 0.1, -1.7, ...)$.

Suppose M is denoted a neural network model and Y is denoted a result from N total number of words, the model for this task, as illustrated in Figure 4.6, can be roughly defined as:

$$Y = M(W(w_1), W(w_2), ..., W(w_N)) \tag{4.3}$$

where $w_i,\ i = 1, 2, \ldots, N$ is a word.

The neural network used in this study is a SLFN. The number of input words, N, per instance is determined by n-gram word model. Hence, the total number of input nodes is $N \times d$, where d is a dimension of word vector. The model M in this study is a SLFN previously described in session 3.3, and the learning algorithms to be compared are backpropagation and extreme learning machine.

Tokenization will be applied for lexical analysis. Then, classifiers for part-of-speech tagging and named-entity will be experimented. Details of experiments and results will be discussed in following chapter.

Chapter 5

Experimentation and Results

This section first explains and lists out the development tools and environment used in this study. Moreover, it further elaborates the development of subprocesses individually according to the flow of information. Finally, the results of each experiment are discussed.

5.1 Development Environment

The system is implemented with Python 3.5.2. All the experiments are performed on a computer with specifications shown in Table 5.1. Additional Python libraries used in the system are shown in Table 5.2

Processor	Intel(R) Core(TM) i7-2600 CPU 3.40 GHz, 3401 MHz, 4 Cores, 8 Logical Processors
Physical Memory Capacity	16.0 GB
Physical Memory Type	DDR3
Physical Memory Speed Bus	1600 MHz
Operating System	Microsoft Windows 10 Pro Version 1607 10.0.14393

Table 5.1: Specifications of the experimental computer

Library Name	Version	Purpose
gensim	2.0.0	Word vector generation
NumPy	1.12.0	Operations on n-dimensional arrays
Matplotlib	2.0.0	Data visualization
Tensorflow	1.0.1	Neural network modeling
hpelm	1.0.4	ELM modeling
scikit-learn	0.18	Data preprocessing

Table 5.2: List of additional Python libraries

5.2 Experiment: Lexical Analysis

5.2.1 Objective

The experiment is conducted to measure the correctness of the longest matching algorithm.

5.2.2 Experiment Setup

The experiment is set up as follows:

Dataset: This experiment uses Orchid corpus as data. According to Guffey et al. [21], sentence's length has an effect on reading comprehension. The authors reported that a sentence with length of 8 words can be comprehended with comprehension rate of 100 percent while other lengths are less comprehensive. Thus, sentences with the length of 8 words is selected for this experiment benchmark. Moreover, all words in sentence must exist in Thai Royal Institute Dictionary 1999 [22]. By following the previously mentioned constraints, 253 sentences are selected for performance measurement. The experiment applies longest matching algorithm to the mentioned dataset. Results are shown in followings.

5.2.3 Results

The results show the accuracy of 96.04% with the average computation time of 0.1133 s.

5.3 Experiment: Part-of-Speech Tagging

5.3.1 Objective

This experiment is conducted to measure and compare between accuracies and time consumptions of two learning algorithms in part-of-speech tagging. The two learning algorithms are backpropagation and ELM which apply in part-of-speech tagging.

5.3.2 Experiment Setup

The experiment is set up as follows:

Dataset: This experimentation uses data from Open American National Corpus (OANC) [23]. The total number of tagged words available is approximately 14 million. Due to limitations of the equipment, the experiment cannot afford to be conducted on the whole corpus. As sentences in OANC's novel section are comparatively more complex than the rest, the experiment is conducted with 74,386 samples from novel section alone.

Parameters: There are 37 tags or word classes in OANC. A 5-gram model is used in the experiment. This results in a SLFN with 185 input nodes from 5-gram model (37×5), 37 output nodes corresponding to the tags, and several different numbers of hidden nodes. During the test, neural networks with backpropagation and ELM are applied on novel section of OANC. While the other parameters are kept unchanged, number of hidden nodes is varied in seven different values, i.e. 500, 1000, 2000, 3500, 5000, 7000, and 8000.

5.3.3 Results

The results over increasing number of hidden nodes are shown in Figure 5.1 and Table 5.3. In Figure 5.1, vertical axis shows test accuracy, while horizontal axis shows number of hidden nodes in SLFN or neural nodes. The highest accuracy achieved with backpropagation approach is 57.2% at 7,000 neural nodes while ELM provides 89.48% accuracy at 5,000 neural nodes. As depicted by Figure 5.1, ELM results in higher accuracy over all number of hidden nodes. Therefore, according to the results, ELM outperforms backpropagation in window-based part-of-speech tagging.

Moreover, from Table 5.3, ELM uses at most twice less time consumption comparing to back-propagation in worst case.

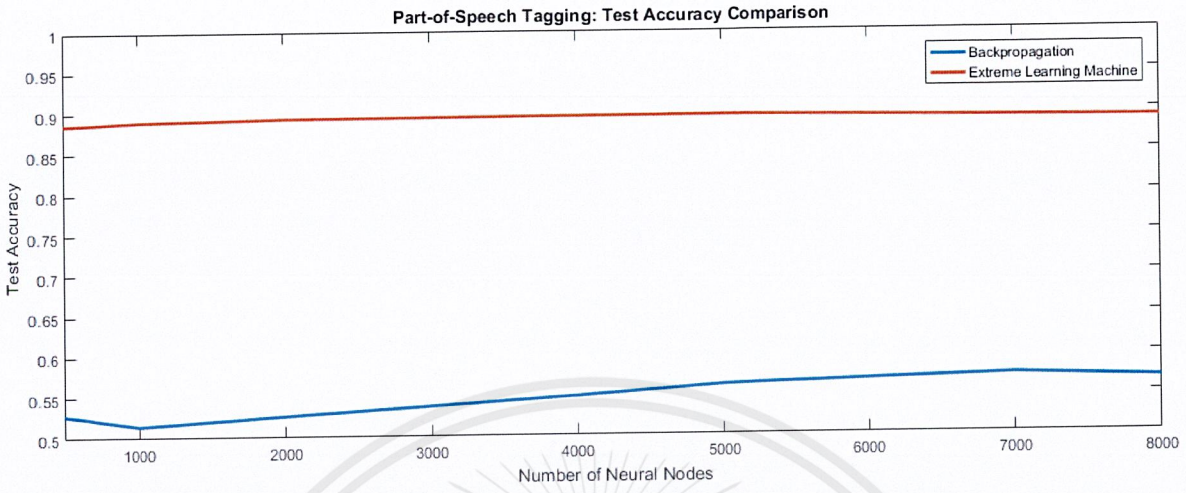

Figure 5.1: Comparison between backpropagation and ELM in part-of-speech tagging

	Training Time (seconds)		**Test Accuracy (%)**	
HN	Backpropagation	ELM	Backpropagation	ELM
500	50.77	1.98	52.74	88.57
1000	38.33	4.33	51.37	88.96
2000	69.59	10.88	52.13	89.29
3500	68.60	25.71	54.82	89.39
5000	86.22	46.13	56.14	**89.49**
7000	241.48	82.40	**57.21**	88.96
8000	279.52	91.66	56.67	88.92

Table 5.3: Comparison of time consumption and test accuracy between backpropagation and ELM in POS tagging

5.4 Experiment: Semantic Analysis

5.4.1 Objective

The experiment is conducted to measure and compare accuracies and time consumptions between two learning algorithms, i.e. backpropagation and ELM, in named-entity recognition task.

5.4.2 Experiment Setup

The experiment is set up as follows:

Dataset: BEST I corpus from NECTEC is used to conduct this experiment. There are 300,000 instances used in the experiment, where half of them is name entity and the other half is not. Moreover, the instances are divided into two portions of 80% and 20% for training and testing purpose, respectively.

The experiment is conducted with the following procedures:

1. Create a word vector from BEST I
2. Create a sample set of features and expected classes with 5-gram model
3. Proportion 80% of data for training dataset and the other 20% for testing dataset
4. Train neural networks with backpropagation and ELM as learning algorithms and record the training time
5. Test the neural networks and record accuracies from the two learning algorithms
6. For ELM, repeat step 3 to 5 for ten times and use their average

It should be noted that the chose neural network model is SLFN. This is done in order to reduce variations which may cause biases in comparison between the learning algorithms. Moreover, ELM requires repetitions to reduce noise caused by random distribution.

5.4.3 Results

The results are shown in Table 5.4, 5.5, and Figure 5.2. The tables show how the two learning algorithms progress as the number of hidden nodes increase. HN, P, and R stand for the number of hidden nodes, precision, and recall, respectively. The first interesting point is the difference in training time between the two algorithms. Time consumption is usually increased as the number of hidden nodes increases. On the other hand, the number of hidden nodes is usually translated to the complexity of mathematical function of neural network. In this regard, ELM gives a better result. Another interesting point from the results is that the accuracy of ELM seems to increase in

HN	Training Time (seconds)	Train Accuracy(%)	Test Accuracy(%)	P(%)	R(%)
5	90.77	90.59	90.56	86.83	95.57
10	92.95	90.73	90.70	87.04	95.59
100	179.02	90.93	90.88	87.31	95.63
500	639.82	91.11	91.57	88.18	96.10
1000	1320.83	91.25	95.43	**95.99**	94.87
1500	1787.41	91.32	91.31	87.81	95.89
1750	2085.38	91.35	91.32	87.83	95.88
2000	2387.01	91.37	91.34	87.85	95.91
3500	4283.85	91.45	91.42	87.95	95.94
5000	8261.25	91.50	91.48	88.02	95.97
7500	18923.54	91.56	91.52	88.05	96.03

Table 5.4: Named-entity recognition performance using backpropagation

HN	Training Time (seconds)	Train Accuracy(%)	Test Accuracy(%)	P(%)	R(%)
5	0.19	59.18	59.16	58.96	61.88
10	0.22	65.16	65.12	64.07	68.71
100	1.38	81.98	81.96	79.78	85.61
500	7.95	90.12	90.10	86.69	94.75
1000	17.90	92.06	92.03	88.51	96.61
1500	29.73	92.87	92.80	89.28	97.30
1750	35.41	93.07	93.03	89.50	97.49
2000	41.77	93.27	93.22	89.74	97.60
3500	100.34	93.95	93.75	90.40	97.92
5000	159.05	94.32	94.18	91.05	98.00
7500	318.84	94.82	94.36	91.38	97.99
8000	325.80	94.88	94.43	91.37	98.11

Table 5.5: Named-entity recognition performance using ELM

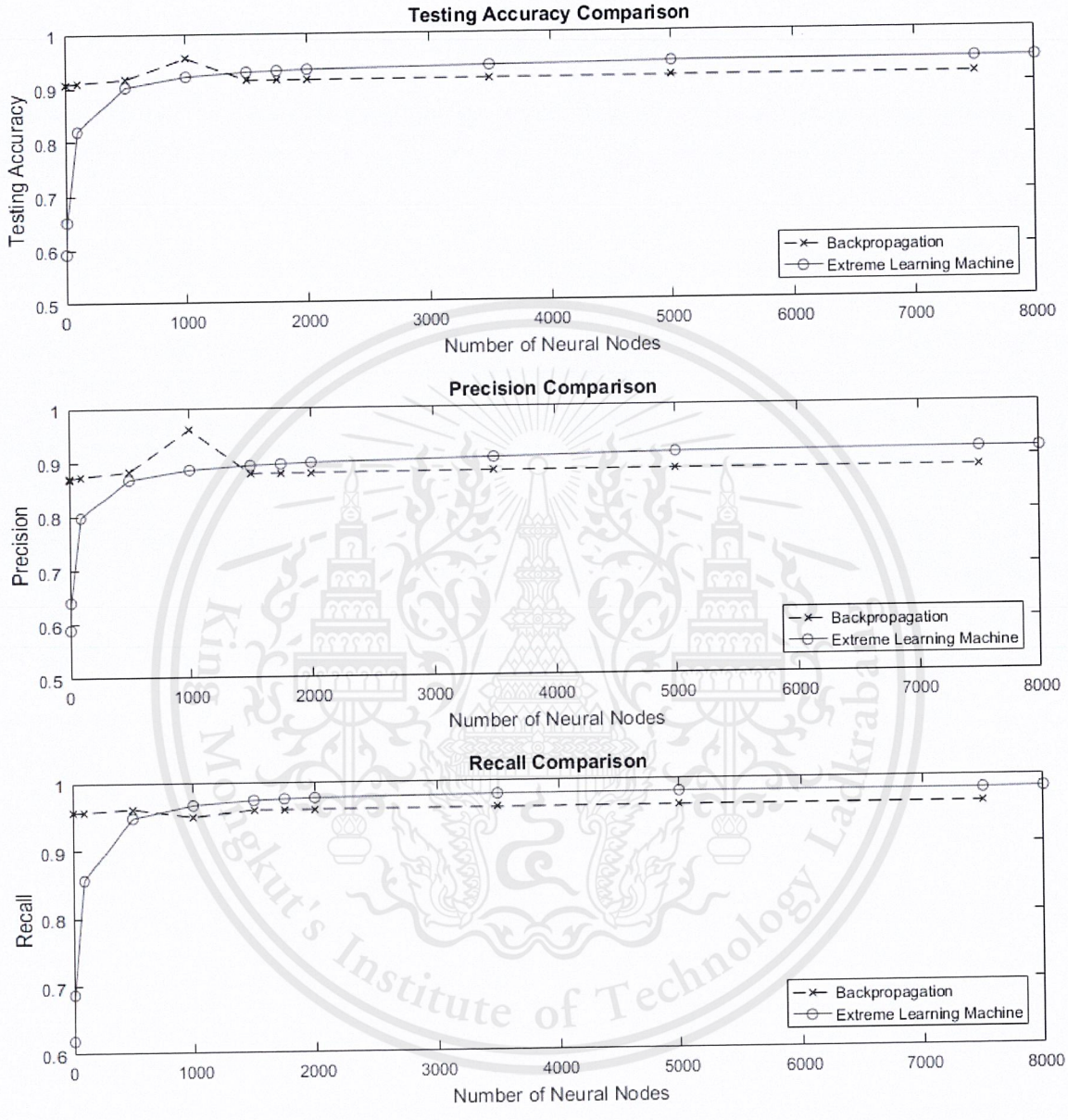

Figure 5.2: Comparison graphs between backpropagation and ELM over number of neural nodes

a more consistent manner. As seen in Table 5.4, the test accuracy of backpropagation algorithm is at its peak when the number of hidden nodes is 1000 before dropping to about 91%. On the contrary, the ELM's accuracy does not show such behavior. Hence, according to Table 5.4 and 5.5, ELM seems to be less prone to overfitting. However, there is a disadvantage in using ELM to be mentioned. While backpropagation is obviously a more time-consuming algorithm, ELM requires more memory to perform. This results from the computation of ELM which is in non-iterative fashion and requires significantly larger matrices in calculation. For this reason, the experimental computer cannot afford to compute the peak performance of ELM.

5.5 Named-Entity Recognizer Application

An application is built to test the proposed system on real-world data. The application combines the proposed lexical analyzer and named-entity recognizer. The application operates as follows:

1. Receive text input from a user
2. Perform tokenization on the text input
3. Map the resulting word tokens from step 2 to the pre-trained word vectors
4. Perform named-entity recognition with the word vectors from step 3
5. Highlight the resulting named-entities

Example images of the application are shown in Figure 5.3, 5.4, and 5.5. Figure 5.3 shows the start window of an application. Lower part of the window is where a user can input text. Figure 5.4 depicts a window after the user inputs the text. After the user clicks "Search" button, the application will highlight all found name-entities, which illustrated in Figure 5.5.

In this chapter, the experiment applies graph theorem solution for tokenization task. It also compares two aspects, i.e. time consumption and accuracy, between ELM and backpropagation by varying number of hidden nodes in classification task. The detailed conclusion will be discussed in following chapter.

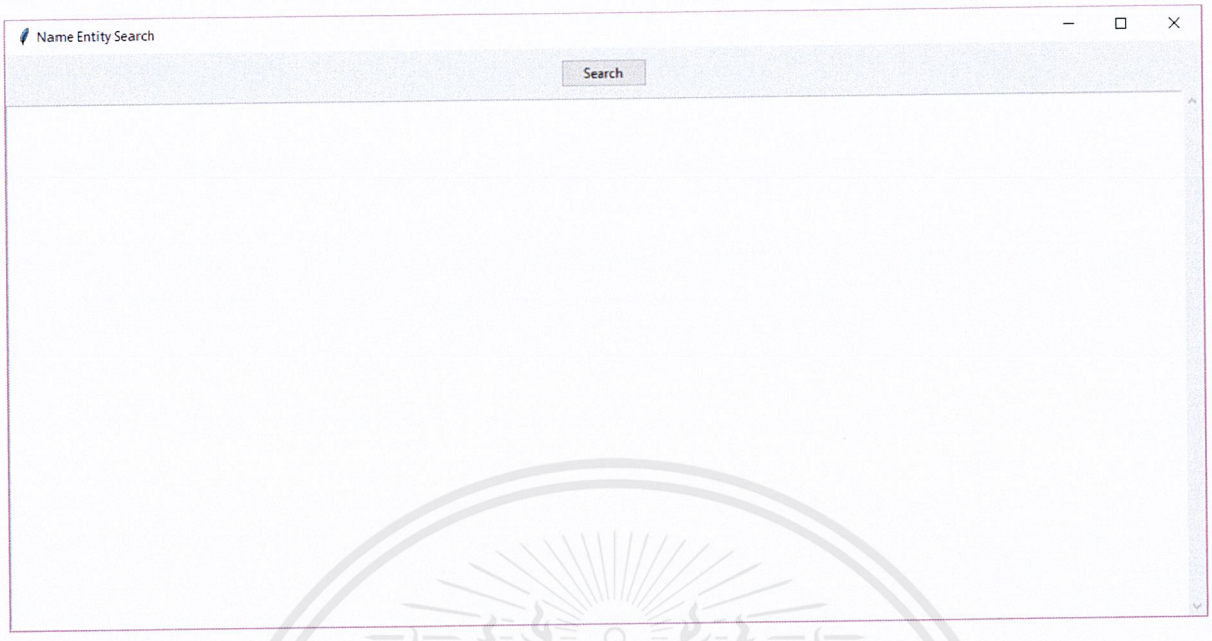

Figure 5.3: The main page of the application

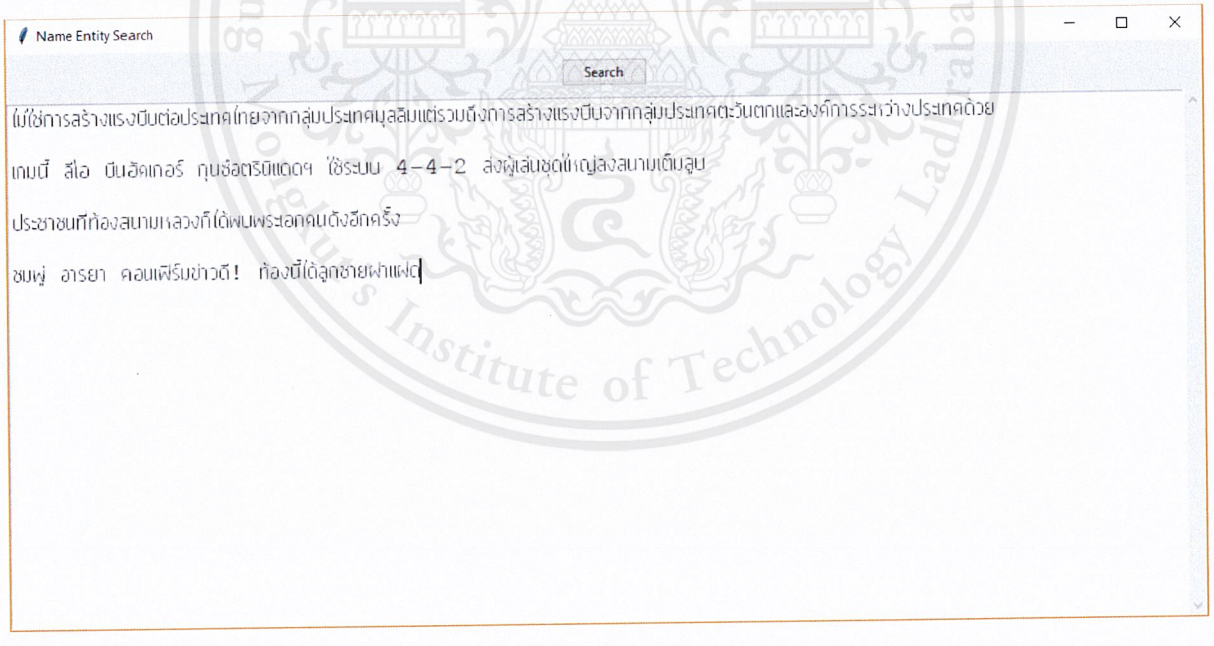

Figure 5.4: The application with input text

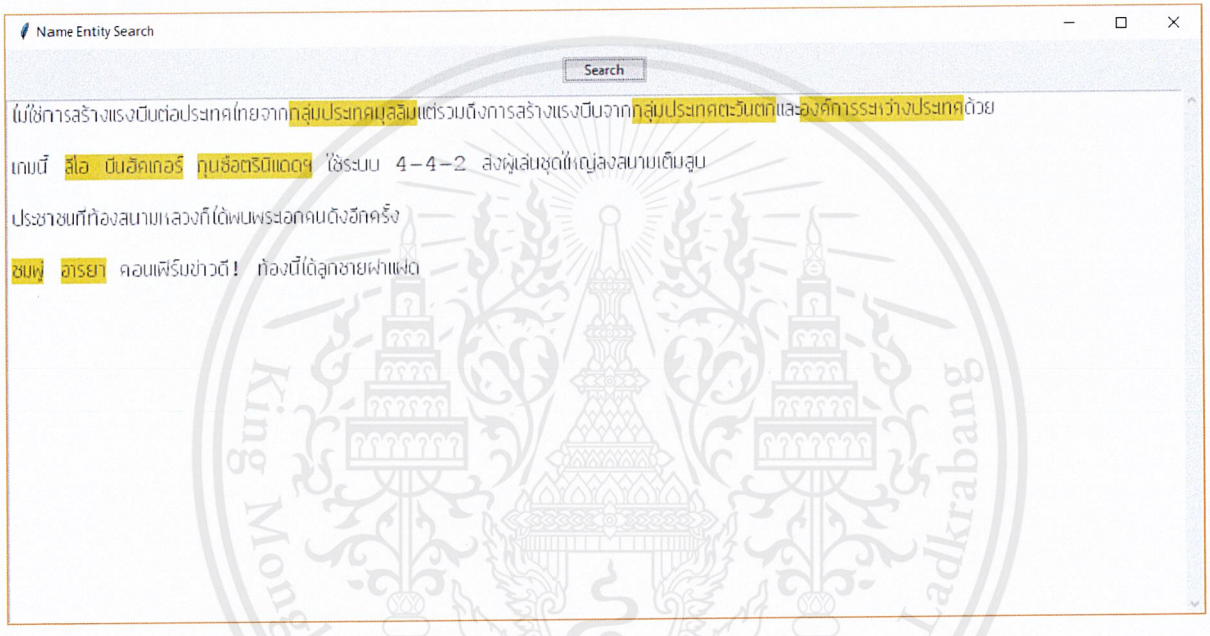

Figure 5.5: The application after performing named-entity recognition

Chapter 6

Conclusion and Discussion

This study proposes three systems for three tasks in natural language processing. These tasks refer to tokenization, part-of-speech tagging, and named-entity recognition. The tokenization system is used in data preparation. This system results in 96.04% accuracy. The systems dealing with part-of-speech tagging and named-entity recognition are based on neural network model with two different training algorithms, i.e. backpropagation and extreme learning machine. The systems are experimented on two corpora, which are Open American National Corpus and BEST I. For part-of-speech tagging, extreme learning machine has significantly better performance with the highest accuracy of 89.48%. For named-entity recognition, both learning algorithms achieve comparable accuracy. It is found that backpropagation gives a slightly higher accuracy than extreme learning machine in one of the experimental trials. However, in most conditions, extreme learning machine gives more overall consistent accuracy. The results also show advantages and disadvantages of the two algorithms. In summary, backpropagation algorithm requires many user-defined parameters, consumes high computational time, and is susceptible to overfitting. Nevertheless, it consumes less amount of memory compared to extreme learning machine. On the other hand, extreme learning machine requires less human intervention. Furthermore, it uses much more lower computational time and is less prone to overfitting. These advantages come with the trade-off in significant amount of memory usage.

This study completes a proposal of three systems. The proposed systems with extreme learning machine achieve higher average accuracy. Based on these experiments, it is found that extreme learning machine is viable in natural language processing even despite its limitation in parameters'

adjustment.

Although the proposed systems seem to work well, they can be improved in both the data and model aspects. Concerning the data aspect, the authors plan to extend the system to support more Thai corpora when they are available. This might lead to better comparison result with more confident. Moreover, the model can be further improved by varying network framework and algorithms in initial weight randomization.

References

[1] V. Sornlertlamvanich, T. Potipiti, C. Wutiwiwatchai, and P. Mittrapiyanuruk, "The state of the art in thai language processing," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2000, pp. 1–2.

[2] W. Aroonmanakun, "Thoughts on word and sentence segmentation in thai," in *Proceedings of the Seventh Symposium on Natural language Processing, Pattaya, Thailand, December 13–15*, 2007, pp. 85–90.

[3] G. B. Huang, Q. Y. Zhu, and C. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proceedings of International Joint Conference on Neural Networks 2014*, Jul. 2004.

[4] Ai - natural language processing. [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_natural_language_processing.htm

[5] V. Sornlertlamvanich, "Three nlp key challenges in big data," http://sites.ieee.org/thailand-cis/files/2016/05/1-Virach.pdf, 2016, accessed: October 2016. [Online]. Available: http://sites.ieee.org/thailand-cis/files/2016/05/1-Virach.pdf

[6] V. Sornlertlamvanich, T. Potipiti, and T. Charoenporn, "Automatic corpus-based thai word extraction with the c4.5 learning algorithm," in *Proceedings of the 18th International Conference on Computational Linguistics (COLING2000)*, 2000, pp. 802–807.

[7] V. Sornlertlamvanich, N. Takahashi, and H. Isahara, "Building a thai part-of-speech tagged corpus (orchid)," *Journal of the Acoustical Society of Japan (E)*, vol. 20, no. 3, pp. 189–198, 1999.

[8] P. Mittrapiyanuruk and V. Sornlertlamvanich, "The automatic thai sentence extraction," in *Proceedings of the fourth symposium on Natural Language Processing*, 2000, pp. 23–28.

[9] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2177–2185. [Online]. Available: http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf

[10] N. Tirasaroj and W. Aroonmanakun, "Thai named entity recognition based on conditional random fields," in *2009 Eighth International Symposium on Natural Language Processing*, Oct 2009, pp. 216–220.

[11] H. Chanlekha and A. Kawtrakul, "Thai named entity extraction by incorporating maximum entropy model with simple heuristic information."

[12] P. Auer, H. Burgsteiner, and W. Maass, "A learning rule for very simple universal approximators consisting of a single layer of perceptrons," *Neural Networks*, vol. 21, no. 5, pp. 786–795, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608007002730

[13] P. D. Moerland and E. Fiesler, "Handbook of neural computation: E1.2. neural network adaptations to hardware implementations," in *Proceedings of the fourth symposium on Natural Language Processing*. New York: Institute of Physics Publishing and Oxford University Publishing, 1996, pp. 1–13.

[14] W. Zhu, J. Miao, L. Qing, and G. B. Huang, "Hierarchical extreme learning machine for unsupervised representation learning," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[15] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1912–1920.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[17] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[18] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, Mar 1998.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[20] P. V. C. J. Y. Bengio, R. Ducharme, "A neural probabilistic language model," *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pp. 1137–1155, 2003.

[21] M. E. Guffey and D. Loewy, *Essentials of Business Communication*, 10th ed. Cengage Learning, 2014.

[22] Office of the Royal Society, "พจนานุกรม ฉบับราชบัณฑิตยสถาน พ.ศ. ๒๕๔๒," http://rirs3.royin.go.th/, accessed: March 2017. [Online]. Available: http://rirs3.royin.go.th/

[23] "Open american national corpus | open data for language research and education," http://www.anc.org/, accessed: March 2017. [Online]. Available: http://www.anc.org/