Abstract—In order to save time in extracting specific information from high volume of data in web documents, this paper proposes an architectural model of generic web document classification system using design patterns for classifying web documents. This work implements two classification techniques for classifying Thai web documents, namely centroid classification and neural network classification, based on the proposed model and compares their classification effectiveness empirically. The training data sets in this experiment consist of 500 web documents of the following five categories (100 documents for each category): mobile phone sales, book sales, travel sales, education information and company profile. Another two hundred and fifty web documents were then used to test the two classifiers. The experiment results showed that the centroid classifier outperforms the neural network classifier both in term of efficiency and effectiveness.

Index Terms—centroid, neural network, document analyzer, text classification, web classification modeling

I. INTRODUCTION

With the current explosive growth of computer and internet usage, the demand for fast and useful access to online data is increasing. Users can find the web pages they want in the enormous database of web pages represented by the internet, using many search engines available to them, including the well known Google, Yahoo, and MSN. Tools like search engines assist users in locating information on the internet. They perform excellently in locating information but provide limited ability in organizing the search results. A good web page classification method is thus of importance in facilitating user searches. Various learning techniques for web page classification have been developed, such as centroid classifier [1] and neural network classifier [2]. Most research works have used standard data sets such as Reuters, OHSUMED, 20-Newsgroups for evaluating the experiments. These datasets are published in English. However, there is only a few systematic research works on web document classification in Thai language, especially for Thai web document collection. Unlike English, Thai written language is a language without word boundary delimiters. Automatic web document classification of Thai documents introduces a new challenge. For Thai web page classification, it is necessary to cope with a problem called word segmentation since the language has no explicit word boundary delimiter. For languages without word boundary, word segmentation plays an important role to construct a set of terms for classification process. Moreover, most Thai Web documents use both Thai and English terms including transliteration and transcription. This characteristic implies a large number of unique terms in the web collection. Besides, the cultures may play into an important role on how information on the web is organized to attract interests from respective readers. Classification techniques that work well for web documents written in English may not work well for Thai web documents. Good classification systems should work well for collections of documents of mixed languages. This work presents a design of a generic, extensible web document classification system that can handle different written languages and host various classification techniques. It also maneuvers two different classification techniques and tests them on Thai web documents as well as comparing their effectiveness.

The rest of this paper is organized as follows. Section II describes a literature review on classification techniques. Section III introduces an architectural design of an extensible web document classification system. Section IV presents the proposed Thai web document classification approaches and Section V evaluates them empirically. The conclusion is made in Section VI.

II. LITERATURE REVIEW

Several web categorization models were developed in different schemes, such as probabilistic models [3][4], decision trees and rules [5][6], regression models [7][8], example-based models (e.g., k-nearest neighbor or KNN) [8][11], linear models [12][16], support vector machine (SVM) [8][17], neural networks [18][19] and so on. Among these models, a variant of linear models called a centroid-based method is attractive since it has relatively less computational complexity than other methods in both the learning and classification stages. The traditional centroid-based method [1] can be viewed as a specialization of so-called Rocchio method [20] and has been used in several works on web categorization [13][14][21][22].

Rocchio algorithm is a classic algorithm for document routing and filtering in information retrieval [13][20][23]. Rocchio algorithm employs Tfidf feature weighting method to create a feature vector for each document. Previous works using the Rocchio algorithm in text classifi-
Text classification is the problem of automatically assigning electronic text documents to pre-specified categories. Typically, text classification systems learn models of categories using a large training corpus of labeled data in order to classify new examples. Due to the tedious and often subjective nature of manual labeling, labeled examples are difficult and expensive to obtain, whilst unlabeled documents are plentiful and easy to obtain [3]. A common method for utilizing this unlabeled data for classification is co-training [41]. In this method, the input data is used to create two predictors that are complementary to each other. Each predictor is then used to classify unlabeled data which is used to train new corresponding complementary predictors. Typically, the complementary predictor is achieved either through two redundant views of the same data [41] or through different supervised learning algorithms [42]. Thus, these co-training methods are applicable only to a limited class of problems.

For research in Thai Web Classification, reference [43] investigated a set of methods to classify Thai medicinal Web documents in a systematic way and analyzed the results by statistical methods. Three factors are taken into account i.e. classification algorithm, word segmentation algorithm and term modeling. Normally, this collection has some special properties i.e. a large number of terms in both Thai and English; several terms are represented in higher-gram; and many typing errors. From the experimental results, classification performance depended on these factors with the major one being the classification algorithm factor. In a model without Thai word segmentation, the number of unique features is greater than those of a unigram model with Thai word segmentation. For this case, TFIDF with term distributions efficiently utilize these unique features. It outperformed other classifiers including SVM. Furthermore, the performance of TFIDF with term distributions in the bigram model was higher than the unigram and only less than SVM by a margin by less than 2 percents. The results suggested that the bigram model of TFIDF with term distributions was a good model. [44] improved Thai-language academic web page classification by using inverse class frequency and web link information. They suggest that inverse class frequency should be used instead of inverse document frequency for centroid based text categorization. With their experiments, the simple term weighting of $t \times idf$ was not sufficient for classifying a small set of web documents when they were categorized by the sources of information. Typically, the $idf$ can be applied to eliminate the impact of frequent terms that exist in almost all documents. They suggest that an important term of the specific category seems to exist in only few classes. They instead use inverse class frequency in both Thai and English; several terms are represented in higher-gram; and many typing errors. From the experimental results, classification performance depended on these factors with the major one being the classification algorithm factor. In a model without Thai word segmentation, the number of unique features is greater than those of a unigram model with Thai word segmentation. For this case, TFIDF with term distributions efficiently utilize these unique features. It outperformed other classifiers including SVM. Furthermore, the performance of TFIDF with term distributions in the bigram model was higher than the unigram and only less than SVM by a margin by less than 2 percents. The results suggested that the bigram model of TFIDF with term distributions was a good model.

K-nearest-neighbors (KNN) [37][38] is a learning method that delays the learning process until a new document must be classified. KNN classifier has been successfully applied for document classification [4][7][8][26][39][40]. KNN classifier compares a new document directly with the given training documents. It uses cosine metric to compute the similarity between two document vectors. It ranks, in a descend order, the training documents based on their similarities with the new document. The top $k$ training documents are $k$-nearest neighbors of the new document and the $k$-nearest neighbors are used to predict the categories of the new document. A major drawback of the similarity measure used in $k$-NN is that it uses all features equally in computing similarities. This can lead to poor similarity measures and classification errors, when only a small subset of the words is useful for classification.

Text classification could be found in [4][24]-[27]. More interestingly, [14] proposed a probabilistic analysis of the Rocchio algorithm, which he called PrTFIDF classifier and it showed improvement compared to the original Rocchio algorithm.

Support Vector Machines (SVMs) have shown to yield good performance on a wide variety of classification problems, most recently on text classification [8][17][26][28][31]. They are based on Structural Risk Minimization principle from computational learning theory [32][33]. The idea of structural risk minimization is to find a hypothesis which is defined as the decision function with maximal margin between the vectors of positive examples and the vectors of negative examples [34]. The major drawback of this algorithm is that it can only solve two-class pattern recognition problems.

Neural network approaches to text classification were evaluated by many researchers, such as [18][28][34]-[36]. Reference [34] employed a perceptron approach (without a hidden layer) and a three-layered neural network (with a hidden layer), while [18] evaluated only perceptrons. Since neural networks are among the top ranking classifiers [4][26], Perceptrons and Least Mean Square rules will be briefly described in terms of web page classification. For web page classification, a perceptron is used for a given category. It takes a feature vector representing a web page as input, calculates a linear combination of the features of the input vector, then outputs a +1 if the result is greater than a threshold (that is automatically learned during the training process) or −1 otherwise, which indicates whether the web page belongs to the category or not, respectively. Least Mean Square (LMS) training rule, also known as Widrow-Hoff rule, was employed and showed good performance for text classification [4][25]. Although the perceptron rule finds a successful weight vector when the training examples are linearly separable, it may fail to converge if the examples are not linearly separable. LMS training rule is designed to overcome this difficulty [25], [26].

The bigram model of TFIDF with term distributions was a good model. [44] improved Thai-language academic web page classification by using inverse class frequency and web link information. They suggest that inverse class frequency should be used instead of inverse document frequency for centroid based text categorization. With their experiments, the simple term weighting of $t \times idf$ was not sufficient for classifying a small set of web documents when they were categorized by the sources of information. Typically, the $idf$ can be applied to eliminate the impact of frequent terms that exist in almost all documents. They suggest that an important term of the specific category seems to exist in only few classes. They instead use inverse class frequency to represent importance of that term. It is defined as the inverse ratio of the number of classes that contain a term to the total number of classes. It works in a class level in text collection instead of the document level.
III. SYSTEM ARCHITECTURAL MODEL

This paper presents an architectural model of a generic web classification system that can handle mixed language documents and embed different classification techniques. There are many powerful web search engines available. So, instead of building new search engines, this work will use existing search engines to retrieve initial search results and perform the classification process on the search results. This section describes the architectural design of the underlying system. The system is modeled as a set of loosely-coupled components interacting with one another using a set of well-known design patterns [45] to ensure reusability and extensibility of the model. Each component performs specific functions and can be replaced or enhanced with additional components. There are 7 components as follows: Front Interface Component, Search Engine Connector, Collection Manager, Text Preprocessor, Document Analyzer, Feature Extractor, and Classification Component. The user or the client of this system can be a human being or just another computer program; hereafter, referred to as a client. Fig. 1 shows the system architectural model. The functionalities of each component of the system are as follows.

A. Front Interface Component

Front Interface Component is for communicating with the client of the system to make it easy to use. It acts as a façade [45] for all other system components to let its client configure and use the system conveniently and effectively without having to know the dependencies among the rest of system components. However, if the client needs an access to complex functionalities of the system, it can bypass this component and interact with other components of the system directly. There are three main functionalities of this component:

- allows the client to read and configure overall system parameters and parameters of other components.
- allows the client to send input parameters and receive output parameters; e.g., setting keyword queries.
- allows the client to choose to turn on/off other components and/or rearrange computing sequences among different components when applicable.

B. Search Engine Connector

Search Engine Connector is for communicating with search engines to send keyword queries and get back the results. Since there are various search engines available and they are different in both query interfaces and result formats, this component consists of many sub-components, one for each search engine. Each sub-component is a wrapper of a specific search engine. All wrappers provide a uniform interface for sending queries and fetching pages of result sets. The search engine connector allows its client to select which search engines to be used (through the front interface component). Queries will be sent to those selected search engines via their corresponding wrappers, and the search engine connector will merge the result sets receiving from multiple search engines through their wrappers. The search engine connector acts as a bridge [45] between its client and the wrappers/search engines with additional functionalities that a bridge usually does not have. Generally, the Bridge pattern decouples the abstraction from its implementation. In this case, the client only performs its task through the search engine connector (the abstraction) without having to handle each wrapper (the implementation) specifically. Additional functionalities of this search engine connector that is not common in general bridges is that the client can select multiple implementations (i.e., wrappers, in this case) to perform the search task at the same time. When this is the case, the search engine connector will also need to merge results from those selected wrappers/search engines. The wrapper itself uses another design pattern, called Adapter pattern [45]. There are many renowned search engines available free of charge. They however have different advantages. Some are good at searching for specific domains while others have a huge collection of data. There is no search engine that is better than all other search engines in all aspects. Building another search engine with all features and advantages of all existing search engines is a time consuming process and it is almost impossible. So, using multiple existing search engines to find information would benefit from all of them without having to rebuild a new one. However, existing search engines have different interfaces and characteristics. In order to be able to use them all the same way, an adapter is built to wrap each search engine to make sure that they all respond to the same command the same way and return the search results in the same format. The wrappers are responsible for adapting the inputs and the outputs of their corresponding searching to the same format for their client.
C. Collection Manager

Collection Manager is for handling data collections. All data are stored by the collection manager so that the client can conveniently manage data collections. The client can browse through the document collection and tag each document with a category term so that this tag, called category, can be used to construct and train classifiers. It also keeps track of intermediate results from various steps of the document processing. Without the collection manager, each component of the system has to communicate to the file system by itself to maintain its own data collection and the intermediate results. The collection manager also facilitates the communication between various components of the system. One component may read one data collection from the collection manager, produce another data collection and store the result as another data collection by using the service provided by the collection manager. Different data collections are not always independent of one another. A data item (e.g., a downloaded web document) in one data collection may be processed by a system component and the result is stored as another data item (e.g., a processed web document) in another data collection. The relationships among data items in different data collections are maintained by the collection manager. These relationships do not need to be one-to-one relationships. They may be one-to-one, one-to-many, or many-to-many relationships. For instance, all web documents in a collection may be processed and represented as a collection of all words or terms in that collection. The collection manager handles the data management and storage management task of the system. It provides storage transparency for all other components of the system. Different components of the system may communicate with one another asynchronously through the collection manager.

D. Text Preprocessor

Text Preprocessor is for extracting text from web documents. It removes html tags and scripts, which are irrelevant to the text contents. It consists of sub-components which take care of the language-dependent contents. For instance, unlike English sentences, Thai sentences have neither word boundary nor explicit sentence boundary. Section IV will describe how Thai text is handled. This process also removes insignificant words, so-called stopwords, as well as performing other language-dependent tasks; like stemming, e.g., changing the word “changing”, “changed”, and “changes” to “change”. After finishing this process, each document is essentially a list of terms where each term is a single word or a phrase. The input of this process is called a web document and the output is called a term document. The input of this process comes from the output of the search engine connector. The text preprocessor component may consist of many different sub-components to perform each step of text preprocessing, such as tag removal, stopword removal, stemming, Thai word segmentation, and other language-dependent processes. These steps may be in different sequences. Based on this, the Builder pattern [45] is used to model this component where different sub-components may be selected and concatenated to form the sequence of text preprocessing steps depending on the needs. The client has choices to pick which sub-components should process the input data first or which sub-components should not be used at all. This design promotes extensibility of the system. In the future, there may be a better word segmentation technique to replace the old one or another language-dependent process for another language. These processing units can be added to the system without redesigning the whole system.

E. Document Analyzer

Document Analyzer is for computing document representations. Given a term document, which is a list of terms, the document analyzer computes a vector representation for the document. After this process, each document is represented as a numerical vector where each dimension of the vector indicates how important is the term represented by that dimension to that document. There are many ways to represent a document as a vector. Different approaches yield different results. This paper proposes an approach to represent a document vector. The detail process is described in Section IV. The input of this process comes from the output of the text preprocessing process.

F. Category Feature Extractor

Category Feature Extractor consists of two processes. One is to find common characteristics of documents (in the vector forms) in the same category. These common characteristics are called the feature of the category. Essentially, this step picks a number of terms that have significant impact on the category as the feature of that category. The other process is to use this category feature to transforms each document vector to a document feature vector of that category. Essentially, for each category consideration, it removes less important terms to the considered category from the document vectors so that the remaining part is the feature vectors with respect to that category. The input of this process comes from the output of the document analysis process. Since the document analysis process may generate a very high dimension document vector for each document. Not all dimensions are equally significant. Besides data processing in a very high dimension space is very time consuming. Hence, this category feature extraction process helps reducing the number of dimensions by choosing those that are significant to each particular category. The detail of this process is described in Section IV. After these processes, each document is represented by one feature vector for each category. This document feature vector is used by the classification component to determine whether the document is in that category or not.

G. Classification Component

Classification Component is for classifying each document into categories. There can be many classification techniques. This paper implements, evaluates, and compare two classification techniques. The detail of this process is described in the Section IV.
IV. THAI WEB DOCUMENT CLASSIFICATION

The previous section describes the architecture of the underlying system. This section focuses on two classification techniques.

A. Thai Text Preprocessor

Unlike English, Thai word does not have word boundary, i.e. there is no space between words. In this system, SWATH [46] which is an automatic Thai word segmentation tool is chosen. It is developed in the Text Processing Section, Division of Research and Development on Information, National Electronics and Computer Technology Center, Thailand. It has two automatic functions: automatic word segmentation and automatic Part of Speech (POS) tagging. It provides longest matching, maximal matching, and POS bigram techniques. SWATH also supports POS bigram technique for automatic POS tagging.

B. Document Analyzer

The output from text pre-processing process is a term document. Each term may appear more than once. This is a good indicator of its importance. To represent each document appropriately, document analysis process is applied. In this implementation, $tf\times df$ weight is used to calculate importance of each term. The term weights are determined as follow:

1) Find unique terms and their raw term frequency of each document. Then sort them by their frequency.
2) Calculate $tf$ of each term (define as term frequency divided by total words).
3) Calculate term weighting ($w$) by multiply $tf$ with $df/N$, where document frequency ($df$) is the number of documents in each category under which a term occurs, $i$ is term number, $j$ is document number and $N_i$ is total number of Web documents of category $k$. Equation (1) shows each document vector in category $k$.

$$d_{jk} = (w_{i_1j}, w_{i_2j}, \ldots, w_{i_j}, w_{i_k})$$

C. Category Feature Extractor

The category feature identification process chooses only dominant words that have a significant influence on the category. It results in a small set of words which are really significant for the examined category. The process not only can remove noise or unimportant words but also can improve the speed performance of the next processes since the size of each data item is reduced. The process is done by examining all documents in the category in order to find common patterns found in most documents in the same category. This process will identify $s$ terms with highest average weighting scores in each category. The procedure is given as follows:

1) Find category feature vector $\hat{c}_k$ using (2)

$$\hat{c}_k = (V_{1j}, V_{2j}, \ldots, V_{sj})$$

where $V_{ijk} = \frac{\sum w_{ijk}}{N_k}$

2) From (2), sort and select the best $s$ terms as selected feature vector. Equation (3) shows selected feature vector for each category.

$$\hat{F}_k = (u_{1k}, u_{2k}, \ldots, u_{sk})$$

where $u$ is selected term or word in category $k$ and $s$ is number of selected term

D. Classifications

The objective of this process is to learn the characteristics of each category from document representations in the category and to use that characteristic to classify new documents. This process consists of two tightly-coupled sub-processes: Learning process and Classifier process. The learning process attempts to learn from the sample dataset while the classifier process uses the result from the learning process to classify new documents.

Two classification techniques were selected, centroid classifier and neural network classifier.

1) Centroid classifier

Learning process: This process calculates a Centroid vector for each category. From (3), the best $s$ terms are used to represent document vector in each category. Equation (4) shows each representative document vector or term weighting using $s$ terms.

$$d_{jk} = (w_{u_{1jk}}, w_{u_{2jk}}, \ldots, w_{u_{sjk}})$$

For a category, outputs of category extraction process which are training document feature vector are fed into the centroid computation process. Equation (5) shows each centroid classifier $C\hat{T}_k$ in category $k$

$$C\hat{T}_k = \frac{1}{N_k} \sum_{d \in N_k} d_{jk}$$

Classifier process: This process calculates similarity between vector of testing Web document $\hat{x}$ and each of category Centroid vectors $(C\hat{T}_1, C\hat{T}_2, \ldots, C\hat{T}_k)$. Cosine measure is used to calculate similarity in this system by

$$\cos(\hat{x}, C\hat{T}_k) = \frac{\hat{x} \cdot C\hat{T}_k}{\| \hat{x} \| \| C\hat{T}_k \|}$$

2) Neural Network classifier

Learning process: In order to train a neural network to perform classification, there is a need to adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network computes the error derivative of the weights (EW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. Inputs are term weightings with category feature by Eq. 4 and outputs are predicted function for a particular category. Only one and two hidden nodes were used to prevent the model from overfitting. S-shaped function was used as the activation function.

Classifier process: This process predicts which category a testing document may belong to. For a
predicted function of category i, its inputs are term weight vectors of category i. Each term weight vector is placed into an appropriate variable of the predicted function. The output value of the predicted function is a real number between 0 and 1. A testing document is classified into a category with highest output value. Fig. 2 show neural network diagram for one category.

V. EXPERIMENT

This section presents the characteristic of experimental data in experiment, the detail in experiment and results and analysis.

A. Thai Web Document Dataset

Five categories are defined (Mobile phone sales, Book sales, Travel sales, Education Information and Company Profile). A set of 100 initial Web documents per category was gathered. These documents were used to find a N-dimension category feature vector, a reduced representation of a category. Five hundred training documents (100 training documents per category) were used to create five centroid vectors and create five neural network predicted functions. Set of 250 testing documents (50 testing documents per category) were used to test the effectiveness of centroid and neural network classifiers. Accuracy rate were then calculated.

B. Experimental Process

The relevant documents were placed into a corresponding pre-determined category. However, they were typically in a HTML or similar tagged format with full-text contents. There is a need to transform retrieved documents into a more appropriate format for efficient computation. At this phase, the system performs an automatic text pre-processing, and text transformation. The goal of text pre-processing of a web document is to scan and clean it. At the text transformation, a procedure is executed to identify some category features (terms). A set of s terms (where s are 5, 10, 15, 20, 25, 30, 35 and 40 terms), with one set per category, was identified.

Next step was to train a selected classification model. Two classifiers were tested in this work. In the following experiment, we trained the system with 500 pre-determined web documents (100 documents per category). In general, better results were achieved when more training documents were used. Again it was a tradeoff between time and effectiveness. At the last step, effectiveness of centroid and neural network classification techniques were measured against another set of 250 pre-determined Web documents (50 documents per category).

C. Experimental Result

The classification precision and recall of the two classifiers on the five experiment data sets are shown in Fig. 3-6. These results correspond to the average classification precision/recall of our experiments. In each experiment, 100 web documents per category were randomly selected as the training set and 50 web documents per category were used as the test set.

From Fig. 3 and 4, we found that the lowest precision in centroid classifier and neural network classifier was Book sales category. The more words in feature vector used, the lower the precision. It is because the added keywords draw web documents in other categories into the Book category. These misclassified documents affected the precision of this category.

From Figs. 5 and 6, we found that the lowest recall in both centroid classifier and neural network classifier was the Company profile category. When we looked at the results in detail, we found that the web contents in this category varied. That is, the increase in the number of words used in the feature vector did not help to improve the recall because there was too much variation of document in the category. The more words added, the less important the words were. To solve this problem, the category should be refined into more specific subcategories where there is less variation.

Looking at the results of Figs. 7 and 8, we can see that the centroid classifier outperformed the neural classifier in four out of the five categories. The surprisingly good performance of the centroid-based classification scheme suggests that it employs a sound underlying classification model. The advantage of the summarization performed by the centroid vectors is that it combines multiple prevalent features together, even if these features are not simultaneously present in a single document. That is, if we look at the prominent dimensions of the centroid vector (i.e., highest weight terms), these will correspond to terms that appear frequently in the documents of the class, but not necessarily all terms would appear in the same set of documents. This is particularly important for high dimension-al data sets for which the coverage of any individual feature is often quite low. Moreover, in the case of documents, this summarization has the additional benefit of addressing issues related to synonyms, as commonly used synonyms will be represented in the centroid vector.
VI. CONCLUSION

In this paper, we proposed an architectural model of a generic web document classification system that can handle different languages. The model employs design patterns to ensure reusability and extensibility of the system. The system consists of multiple loosely-coupled components where each component can be extended and enhanced with newly implemented features without having to redesign the system. Existing search engines can be incorporated into the system seamlessly. A new search engine can also be integrated into the system by just implementing a wrapper for that search engine.

We implemented two Thai web document classification techniques, namely centroid-based document classification and neural network-based document classification. The experimental evaluation has shown that the centroid-based classifier consistently and substantially outperformed the neural network classifier on a wide range of data sets. We have shown that the power of this classifier is due to the function that it uses to compute the similarity between a test document and the centroid vector of the class. This similarity function can account for both the term similarity between the test document and the documents in the class, as well as for the dependencies between the terms present in these documents. For further work, it would be helpful to improve classification performance in some categories in which the contents vary significantly. The categories could be refined into more specific subcategories where there is less variation. New classification techniques can be integrated. One may extend the system further by integrating multiple classification techniques together to help improving effectiveness of the whole classification system.
REFERENCES


