Content-Adaptive Feature Selection for Classifying Class-Imbalanced Data

Part Pramokchon and Punpiti Piamsa-nga

Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Bangkok 10900, Thailand, g4785036@ku.ac.th, pp@ku.ac.th

Abstract

In multi-class text classification, ranking based feature selection usually obtains low performance when number of data in each class is highly imbalanced. The larger class has much dominant influence on the selected features and the smaller one seems to be ignored. To reduce the influence of class-imbalanced distributions, concept of separating the larger classes into several smaller subclasses according to their proximities is proposed. We analyze data in each main class by hierarchical agglomerative clustering (HAC) then statistical outlier of distances among its subclasses are used to cluster data. Without a predefined number of clusters, this method can cluster data more statistically-balanced in each main class. Then more-balanced subclass data are used to select a feature set for text classification. Performances of this method are evaluated on SVM and Naïve Bays by classifying the RCV1v2 dataset. The experimental results show that subclasses clustered by the proposed method achieve better classification performance than main classes and equally comparable to subclasses from ground-truth.

Keywords: Ranking method, Text categorization, Class imbalance distribution, Feature score, Hierarchical Agglomerative Clustering

1. Introduction

In the problem of high dimensionality of the features in text classification (TC) [1], most words used as features in text documents are irrelevant to the classes [2]. Irrelevant features not only slow down the classification process, but it can also degrade the accuracy [1, 3-5]. Feature selection is a popular method used to select a small number of relevant features in order to reduce the dimension of feature space and improve efficiency for machine learning [5, 6]. Therefore, in the TC tasks, a small set of the most relevant words to the classes is selected for the classification process, while accuracy is still acceptable [1-6].

Filter-based method is one of the most frequently-used methods as the preprocessing for the task of text classification to select features. The method tries to evaluate the “usefulness” of each individual feature for classification [4]. The usefulness score is determined by measurement the intrinsic characteristics of the training data (such as distance, information, dependency, or consistency between features and class) without involving any learning algorithm [1, 5, 7, 8]. Typically, a feature that has a higher score is more relevant and ones that have higher scores than a cut-off threshold are selected. Filter-based method is fast and available and the selected features are independent of a classifier; therefore, it is easy to scale up for machine learning research with high-dimensional datasets [2, 4, 7, 9, 10].

However the most favorable measures of feature score, such as Information Gain and Chi-square [3, 4, 6, 7, 11], often fail to produce good performance when it is applied to the problem of classifying instances into more than two classes (such as the multi-class TC). The performance of the classification, especially the recall rate, is usually dropped due to the number of features and number of text documents in each class is drastically different in real life [7, 12-14] (called as the skewed class distribution problem or the class imbalance problem). The feature ranking methods tend to rank features in large classes (majority), while those in small classes (minority), which are difficult to be learned, are rarely considered [7, 12-15]. This not only leads to a low recall rate for the smaller classes but also reduces the overall performance of the multi-class TC.

Many recent research projects have proposed solutions for dealing with the imbalance data distribution problem. They can be categorized into three approaches: data preprocessing, cost-sensitive learning, and algorithm modification [14, 16]. There is no single best approach to deal with imbalance
problems, the re-sampling is data preprocessing which has been shown to be very successful in many researches [16, 17]. However, the re-sampling methods may also introduce problems of the important concept missing, overfitting, and over-generalization [14]. In this paper, we focus on data preprocessing method and avoid these problems of re-sampling simultaneously.

We hypothesize that if each subclass in all main classes has comparably equal number of instances, rare but important features of the small main classes would have more chances to be selected as members of optimal feature set which improve the classification performance. In [18], we proposed a technique to reduce effects of imbalanced class distribution using creating intermediate subclasses by clustering documents in each large class and then use these subclasses to be analyzed along with small main classes. Though it has shown promising results, the method based on K-mean clustering also has its biggest drawback; predetermining the number of K subgroups of any datasets is difficult and nondeterministic [19-21]. The method has to get an appropriate K value by expertise determination or iteratively empirical experiments or sophisticated approaches.

In order to enhance the idea about applying clustering as the data processing before classification, in this work, we propose the method based on hierarchical agglomerative clustering (HAC) [19, 20, 22] integrated with class-sensitive weighting for determination cut-off for imbalance text classification problems. Technically, main classes of training dataset are clustered by HAC method into many subclasses and those K subclasses are used to be replaced with their main class for computing feature scores. The HAC method does not require a predefined number of clusters as input. For partitioning data into clusters, the method measures the dissimilarity between instances and then uses outlier of distance to determine the clusters. Also, this method does not require expertise knowledge or iterative empirical experiments to define the number of K clusters. Unlike K-means clustering, clustering results by this method are deterministic. Also, unlike most of the re-sampling paradigm (under-sampling, over-sampling, and hybrids), our proposed method avoids the problem of the important concept missing, overfitting, and over-generalization [14] by considering both within-class and between-class relationships based on the clustering method.

The performance of the proposed algorithm is evaluated on the RCV1v2 dataset [23]. Using macro-averaged F1 in the classification of the dataset on Support Vector Machine (SVM) [2, 24] and Naïve Bayes (NB) [1, 6, 13], experimental results show that a feature set selected by proposed algorithm is significantly better than the one selected by only main classes. Compared macro-averaged F1 on SVM at 1300 features, results from features selected by our method is 0.931 while result by main class is 0.918. Similarly, Compared macro-averaged F1 on Naïve Bayes at 400 features, results from features selected by our method is 0.836 while result by main class is 0.743. The feature selected by the proposed method not only increases overall classification result but also improves recall of the smallest class from 0.809 to 0.829 on SVM and from 0.849 to 0.873 on NB.

The rest of the paper is organized as follows. The details of our proposed method including the background of text classification, feature selection and problem of class distribution imbalance are described in the next section. After that, the experiments are explained and then the results are discussed. The final section is the conclusion of this work.

2. Background

2.1. Multi-class text classification

From the perspective of machine learning, Multi-class text classification can be regarded as a supervised learning problem which is an automatic process for assigning a class in a finite set of predefined classes, \( C = \{c_1, \ldots, c_{|C|}\} \), to text documents. Given a supervised training set of labeled text documents, \( Tr = \{d_1, \ldots, d_{|Tr|}\} \), the goal is to induce a classifier based on the assessment of their content to correctly label a class of a new unlabeled document (also known as testing set, \( Test = \{d_1, \ldots, d_{|Test|}\} \)).

Let \( F = \{f_1, \ldots, f_{|F|}\} \) be a set of original distinct features that occur at least once in at least one document in the data collection. Generally, a document \( d_j \) is represented as a feature weight vector \( \tilde{d}_j = [w(f_1, d_j), \ldots, w(f_{|F|}, d_j)] \) [1, 23]. This feature weight \( w(f_i, d_j) \) quantifies importance of feature \( f_i \) for describing semantic content of document \( d_j \). All documents in the training set are
represented by these feature weight vectors which are used in the learning classifier steps. Finally, a classifier is then built by a machine-learning algorithm.

2.2. Feature scoring measure

Feature scoring measure is an approach to evaluating a usefulness of an individual feature by analyzing general characteristics of the training examples such as information, dependency, distance, consistency, etc. [5]. In this research, we used information gain in our experiments. Information Gain (IG) \[1, 3, 6, 11\] is used to measure an amount of information that a feature contribute to a classification result. The Information Gain of feature \(f_i\) is defined as:

\[
IG(f_i) = - \sum_{k=1}^{\text{|C|}} P(c_k) \cdot \log P(c_k) + \sum_{k=1}^{\text{|C|}} P(c_k | f_i) \cdot \log P(c_k | f_i)
\]

where \(P(c_k | f_i)\) is the conditional probability that the feature \(f_i\) occurs in category \(c_k\), and \(P(c_k | f_i^c)\) is the conditional probability that \(f_i\) does not occur in \(c_k\). IG is commonly applied in TC which uses the class information to compute the score [25]. Then, features are ranked features by their scores. Finally, an optimal feature is selected with respect to the feature scoring.

2.3. Class imbalance and its effect on feature selection

Class imbalance of dataset is a situation that number of entities in each class is drastically different [14, 16]. Typically, classifiers are optimized for overall accuracy without taking distribution of class sizes into account. Thus, the larger classes are dominated by high frequencies of supporting data, and the smaller classes become statistically unimportant. Many research projects on classifications in imbalance domains have shown the significant effect of the between-class imbalance on the performance [14, 16].

Feature ranking methods also fail when being applied to multiple classes with non-uniform class distributions (class skew) as well. The method pays more attention to the large classes to compensate for their weakness that tends to ignore small classes. Thus, the relevant features for large classes can be selected but features for small classes are ignored. In [7], it has been shown that Information Gain assigns higher scores to the terms (features) correlated with large classes and such that small classes are under-represented in top-rank features. Their experimental result also showed that the recall rate for small classes is significantly low even though averaged recall is high. The experimental results in [12] have also concluded that other scores have suffered to these kinds of problems as well.

There are three approaches to handling class imbalance problem. [14, 16]. The first approach is dataset preprocessing before learning task. Most of the approaches are re-sampling techniques in order to balance the class distribution. These works have proved empirically that is a useful solution and the further advantage of these techniques is that they are independent of the underlying classifier. The second approach is the cost-sensitive learning which assumes high misclassification costs for samples in minority classes and seek to minimize the high-cost errors. The precise value of misclassification cost is necessarily achieved by domain experts, or can be learned via other approaches. The third approach is ensemble-based method, also known as multiple classifier system. The basic idea is to construct several classifiers from the original data and then aggregate their predictions with the unknown instances entered. The ensemble-based classifier can improve the performance of single classifiers by inducing several classifiers and combining them to obtain a new classifier that outperforms every one of them.

All groups described above have different particular drawbacks which make them appropriate for a given application. The ensemble methodologies have shown very accurate result, but their learning time may be high and the output model can be difficult to understand by the final user. Cost-sensitive approaches have also shown very precise result, but the necessity of defining an optimal cost-matrix
imposes hard restrictions to their use. Finally, the re-sampling algorithms have shown their robustness and obtained very good global results, and therefore, they can be viewed as a standard approach for imbalanced datasets [16, 17]. However, re-sampling paradigm (such as down-sampling, over-sampling and hybrid [14]) has to face with their critical problems, such as the important concept missing, overfitting, and over-generalization, which reduces its performance.

In our previous work [18], we apply K-mean clustering algorithm as the data preprocessing before classification to reduce the effect of class imbalanced distribution. The K-mean clustering based approach considers both within-class and between-class relationships of whole dataset without using re-sampling paradigm thus it can avoid the problem of the important concept missing, overfitting, and over-generalization. Although the experimental results have shown promising outcome, the method based on K-mean clustering also has its biggest drawback; it requires the predefined number of K subgroups of any datasets in which is difficult and non-deterministic. The method has to get an appropriate K value by expertise determination or iteratively empirical experiments or sophisticated approaches.

3. Proposed Method

We use the hierarchical agglomerative clustering to separate data from both larger and smaller class into several subclasses which are used instead of the main classes for feature scoring and the ideal is demonstrated in Figure 1.

3.1. Hierarchical Agglomerative Clustering (HAC)

Text clustering is about to combine similar text documents into a group (cluster). Some classes of text are large enough to be further clustered and then all data can be organized as hierarchical cluster. Hierarchical Agglomerative (bottom-up) clustering [19, 20] is about to consider each document as a singleton cluster in the beginning and then successively merge (or agglomerate) pairs of clusters until all documents have been merged into a single binary tree.

An example of document clustering is visualized by a dendrogram shown in Figure 2. In the dendrogram, element on the x-axis is document and distance marks that two elements (either document or cluster) are merged in on the y-axis. Thus, a binary tree can be plotted so that the height of each node is proportional to the values of the intergroup distances between its two child nodes. Possible clustering results depend on cut-off distance. In Figure 2, there are three examples of clustering: at height 0.84 into 3 clusters, at 0.72 into 4 clusters, and at 0.61 into 6 clusters. The higher cut-off value, the smaller number of clusters is.

![Figure 1. Concept of proposed method](image-url)
In order to determine the best cut-offs, a measure of distance between sets of observations is required. In most methods of hierarchical clustering, this is achieved by use of an appropriate metric (a measure of distance between pairs of documents), and a linkage criterion which specifies the distance of sets as a function of the pair-wised distance between two clusters. Many research projects argue that using angles to measure similarity is more suitable for document clustering which has highly-dimensional sparse data [8]. Therefore, absolute cosine is used as geometric view in our study.

HAC merges pairs of elements that are in close proximity into binary clusters using the linkage function. There are three popular linkage criterion [19, 20], such as the single-link, complete-link, and UPGMA schemes. However, we use the complete-link clustering in our work because the approach can be more efficient than the other approaches for large dimensional data [22]. The overall complexity of the algorithm is \( O(N^2 \log N) \) which \( N \) is the number of document in the dataset. The HAC algorithm is inferior in terms of time complexity compared to the other clustering algorithms; however, it is the deterministic algorithm which does not require a predefined number of clusters. The HAC algorithm is processed in one time while the K-mean clustering has to empirically repeat until the best number of clusters is reached, that is more exhaustive task.

### 3.2. Proposed Algorithm

The proposed algorithm is shown in Algorithm 1 (Cluster-based feature scoring measure algorithm). In Algorithm 1, given a set of training data with known main classes, \( \mathcal{T}_r = \{(d_t, c_j)\}_{t=1, \ldots, |\mathcal{T}_r|, j=1, \ldots, |\mathcal{C}|} \), the proposed method starts by separately clustering each class \( c_j \) into \( k_j \) clusters using the Hierarchical Agglomerative Clustering algorithm (lines 1-2). Then, in line 3, resulting data become \( \mathcal{T}_{r'} = \{(d_t, c_j^t)\}_{t=1, \ldots, |\mathcal{T}_r|, j=1, \ldots, |\mathcal{C}|, t=1, \ldots, |K|} \) and \( |K| = \sum_{j=1}^{|\mathcal{C}|} k_j \) where \( t \) is the label of a specific cluster (subclass) of main class \( C \). Then, we have new training sets \( \mathcal{T}_{r'} \), which are passed as input to measure the feature score with considering subclasses instead of the main classes (line 4-5): after looping for computing scores of each feature, then used it for ranking all features, decreasingly (line 6). The result, which is a ranked feature by its scoring, is returned (line 7).

#### Algorithm 1 Cluster-based feature scoring measure algorithm

**Input:** Original feature set \( (F) \)
- Training data \( (\mathcal{T}_r = \{(d_t, c_j)|\}_{t=1, \ldots, |\mathcal{T}_r|, j=1, \ldots, |\mathcal{C}|}) \)
- Main class \( (C = \{c_1, ... , c_{|\mathcal{C}|}\}) \)
- Predefined number of cluster \( (K = \{k_1, ... , k_{|\mathcal{C}|}\}) \)

**Output:** Feature score \( (S) \)
- Sorted feature score \( (S_{sorted}) \)
Output: Feature set ranked by score (FScore)

1: for all classes in C do
2: cluster all data \( \{d, c_j\} \) within the class \( c_j \) into \( \{d, c_j'\} \) \( t = 1, \ldots, k_j \) using HAC using cut-off value
3: \( T_r = \{d, c_j', \ldots, d|_{t=1}, c_j|_{c_j}\} \) // new training set
4: for each feature \( f_i \) in \( F \) do
5: \( F_{\text{Score}} \leftarrow \text{ComputeScore} \left(t_i, T_r\right) \)
6: \( F_{\text{Score}} \leftarrow \text{RankedFeatureByScore} \left(s_i\right) \)
7: return \( F_{\text{Score}} \)

Proposed method considers relevance between each feature and each class in the whole dataset by using the resulted subclasses to re-labeling each document without data over- or under-sampling process. Therefore, unlike the re-sampling paradigm, this method does not face problems of the important concept missing, overfitting, and over-generalization etc.

3.3. Class-sensitive weighting for determining the cut-off

After creating the hierarchy, we need a cutting point to partition data into clusters. We propose a method for determining an appropriate clustering cut-off based on normalized height of nodes in the cluster tree. It is about to compare the height of each node with the heights of neighboring nodes below it in the tree. A node that is approximately the same height as the nodes below it indicates that there is no distinct between objects joined at this level of the hierarchy. These nodes are to exhibit a high level of consistency because the distance between the objects being joined is approximately the same as the distances between the objects they contain. On the other hand, a node, whose height differs noticeably from the height of the nodes below it, is indicated that the objects joined at this level in the cluster tree are much farther apart from each other than their components were when they were joined. This node is said to be inconsistency with the nodes below it, except the leaf node’s inconsistency which is zero. The relative consistency of each node in the hierarchical cluster tree can be quantified and expressed as the inconsistency coefficient. This value compares the height of a node in a cluster hierarchy with the average height of nodes below it.

The inconsistent coefficient of each cluster node \( n_i \) in the hierarchy is given by

\[
I(n_i) = \frac{h(n_i) - \mu(h(n_i))}{\sigma(h(n_i))}
\]

where \( h(n_i) \) is the height of the node \( n_i \), \( \mu(h(n_i)) \) is mean of the heights of all the nodes included in the calculation (\( n_i \) and other nodes below it), and \( \sigma(h(n_i)) \) is the standard deviation of the height of all the node included in the calculation. Nodes, which join different clusters, have a high inconsistency coefficient, and nodes, which join same clusters, have a low inconsistency coefficient.

Next, we proposed the computing cut-off threshold, \( T_{\text{cut}} \), based on the statistical method of outlier detection defined as

\[
T_{\text{cut}} = \mu_i + \sigma_i \cdot \left( Z_{1-\frac{\alpha}{2}} + w_{c_j}\right)
\]

where \( \alpha \) is a confidence coefficient which is only one statistical parameter in the proposed method, \( Z_{1-\frac{\alpha}{2}} \) is the \( 1 - \frac{\alpha}{2} \) quantize of the \( N(0,1) \), \( \mu_i \) and \( \sigma_i \) are mean and standard deviation of all cluster node in hierarchy, respectively and \( w_{c_j} \) is the class-sensitive weight of each main class (described below). Outlier detection is a statistical approach that indicates the observations which have very different behavior from others in data collection [26]. The cluster node \( n_i \), that has inconsistent coefficient value \( I(n_i) \) more than \( T_{\text{cut}} \), this node is inconsistent when compared with its child nodes in the hierarchy, then its child nodes are grouped in the same cluster. The lower cut-off value means that the
class will be divided into many subclasses while the higher cut-off value will make the clustering
divides the class into the less numb of subclasses.

Separating the main classes into the subclass before computing feature score not only reduces the
influence of large classes but also increases the chance that important but rare features of small classes
can be selected. In order to avoid the impact of class separating with small classes, we assign a more
value for small classes and small value for large classes. The \( w_{c_j} \) value of the main class, \( c \in C \)
is obtained by following formula:

\[
\frac{1}{(|c_j|/m \cdot \frac{m}{|c_j|} \in C)^{1/2}}
\]

where \( |c_j| \) is a number of documents in the class \( c_j \), \( m \cdot \frac{m}{|c_j|} \in C \) represented the size of the
smallest class in training dataset. Actually, \( w_{c_j} \) value of the smallest class is 1, the values of the other
classes are descending with respect to their size and the weight of largest classes is the lowest value.

4. Experiments

To evaluate the performance of proposed method, the experiments on multi-class imbalanced
datasets have been conducted. The aim of these experiments is to verify that data preprocessing by
clustering technique can reduce the effect of imbalanced data in terms of performance metrics for
classifying both overall classes and minority class.

4.1. Dataset

In the experiment, the RCV1v2 benchmark dataset is used to evaluate our algorithm [23]. It is
organized in four “Topic Codes” hierarchical classes: CCAT, ECAT, GCAT, and MCAT (called as the
main classes). Each class has differently defined subclasses, which we used it as ground truth. Table 1.
shows sizes of main classes and number of their ground truth subclasses. Totally, dataset contains
19,806 training documents and 16,886 testing documents. The largest class (CCAT) is almost five
times as large as the smallest class (ECAT).

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of subclasses</th>
<th>Number of documents</th>
<th>training</th>
<th>testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAT</td>
<td>18</td>
<td>8410</td>
<td>6993</td>
<td></td>
</tr>
<tr>
<td>GCAT</td>
<td>23</td>
<td>4959</td>
<td>4580</td>
<td></td>
</tr>
<tr>
<td>MCAT</td>
<td>4</td>
<td>4747</td>
<td>3846</td>
<td></td>
</tr>
<tr>
<td>ECAT</td>
<td>10</td>
<td>1690</td>
<td>1467</td>
<td></td>
</tr>
</tbody>
</table>

4.2. Support Vector Machine (SVM) and Naïve Bayes Classifiers

We evaluate quality of our selected features by using them to train classifiers. Since our dataset has
many features and sparse examples, we adopt the Support Vector Machine (SVM) with linear kernel
and the Naïve Bays (NB) as two classifiers in the experiments because of their characteristics. SVM
with linear kernel is the favorite and robust learning algorithm for highly dimensional data. Although
The Naïve Bays classifier suffers from lower accuracy compared to the SVM classifier, but it easy to
apply with the document-based application. The experiment is performed using the LibSVM package
and Naïve Bayes package of WEKA, with the default values of parameters [27].
4.3. Assessment Metrics for Imbalanced Learning

To evaluate the performance of the proposed method in the multi-class text classification, we employ the Accuracy defined by $\text{Accuracy} = \frac{\sum_{j=1}^{C} TP_j}{\sum_{j=1}^{C} (TP_j + FN_j)} \times 100$. True positive $TP_j$ is a number of test documents that are classified into class $C_j$ correctly, and false negative $FN_j$ is a number of test documents incorrectly classified under other categories.

We also use $F1$ ($F_1$) as the local performance measure, which is a combination of precision $P_j$ and recall $R_j$. The definition of $F1_j$ is $F1_j = \frac{2 \times P_j \times R_j}{P_j + R_j}$, where $P_j = \frac{TP_j}{TP_j + FP_j}$ and $R_j = \frac{TP_j}{TP_j + FN_j}$, which False positive $FP_j$ is numbers of test documents that are classified into class $C_j$ incorrectly. To exhibit the performance of the proposed method on the smaller class, the $F1$ measures over the multiple classes are summarized using the macro-averaged $F1$ [1] defined by $M F1 = \frac{\sum_{j=1}^{C} F1_j}{|C|}$. The $M F1$ provides equal weights to all classes, regardless of the number of documents in each class; therefore, it is not mainly dependent on large classes.

4.4. Results

In the experiment, we assumed that reducing size of large classes by clustering to smaller subclasses will make features in the small classes statistically more visible to the learning machine than results without clustering. We determined effectiveness of feature scoring measures among four feature sets from: 1) the main class (M), 2) the ground truth subclass from the dataset (G), and 3) resulted K subclass from the HAC algorithm with baseline cut-off value (HAC_cutoff1), and 4) resulted K subclass from the Algorithm 1 with class-sensitive cut-off value (HAC_cutoff2).

We generated a vector model for training data by using 3,000 top-ranked features instead of all features in order to reduce training time and preliminary empirical experiments showed that 3,000 features are sufficient for acceptable clustering results. These features are ranked by Document Frequency (DF) in this process.

We applied Information Gain (IG) to measure feature scores on training data which is labeled with four different class labels. First, main class (M) is used as the baseline method. Second, ground truth subclass (G) is extracted from metadata of RCV1v2. Third, we used HAC algorithm onto the vector model. We manually analyze several arbitrary values of cut-off threshold for each main class and evaluate the results of HAC algorithm and empirically select the best-resulted value as the baseline cut-off. The baseline cut-off (HAC_cutoff1) can cluster the main classes into 161 result subclasses which consist of 15, 29, 48 and 69 subclasses of main class CCAT, ECAT, GCAT and MCAT respectively. The result, which is a set of clusters, was used as a subclass for measuring feature score.

After finding the baseline cut-off, the Algorithm 1 has been applied on the vector space of training data. By setting the value of a parameter $\alpha = 0.01$, Algorithm 1 (HAC_cutoff2) obtains the resulted cluster which consisted of 284 subclasses (134, 37, 35 and 81 subclasses from main classes CCAT, ECAT, GCAT and MCAT, respectively.) The resulted clusters were used as subclasses for measuring the score as well.

Then, the traditional feature ranking method proceeded. Feature ranking method is executed with numbers of selected features $|F|$, ranged from 100 to 2000, stepped by 100 and from 2000 to 3000, stepped by 500. We adopt SVM with linear kernel and Naïve Bayes as the classifiers in the experiments. Using traditional feature ranking method, the selected feature subsets were used to train the classifier; then the classification performance was evaluated on training documents. The performance of the classifier is estimated by four-fold cross-validation. To eliminate random variation, the performances were averaged over the three runs. We use Student’s paired two-tailed t-test in order to evaluate the statistical significance (at the significant level 0.05) of the difference between the various results. Details of results on each classifiers are discussed as follows.
4.4.1. Results on Naïve Bayes

Figure 3 shows the ranking accuracy of Naïve Bayes classifier with feature ranking method using clustering-resulted subclass, ground truth subclass and main class on the RCV1v2 training dataset. We observe that 1) the ground truth subclass (G) outperformed the main class (M) when the number of feature further increases, thus, using subclass for feature scoring measure can improve accuracy substantially of low-effective classifier such as Naïve Bayes; 2) HAC clustering using baseline cut-off value (HAC_cutoff1) and our algorithm using class-sensitive cut-off (HAC_cutoff2) achieved accuracy which is similar trend to the ground truth subclass (with correlation coefficient > 0.9); and 3) HAC_cutoff2 achieved higher accuracy that is superior to the main classes at lower number of feature from 100 to 1100 but inferior to the ground truth subclasses, and main class at higher number of feature from 1100 to 3000. However, this work has focused on selection small number of feature for classification. The accuracy of HAC_cutoff2 is maximal at 400 features, which is higher than the maximum accuracy of the main class. By the ranking wrapper-based paradigm, thus the 400 top-ranked features are selected as the optimal feature subset for Naïve Bayes classification for class variants.

Next, Naïve Bayes classifiers from the training process were used to classify testing documents and the classification performance was then evaluated. Figure 4 illustrates the performance comparison at selected numbers of features (400 features) of main class and subclass. First, original main classes (red bar in Figure 4) produce the lowest performances (F1 and recall) of Naïve Bayes. Second, we can see that using any subclasses (blue, green and orange bars) instead of main class (red bar) can improve both F1 and recall measures. Third, in the smallest class (ECAT), Figure 4 (a) shows that subclass resulted from the Algorithm 1 with class-sensitive cutoff (HAC_cutoff2) makes the classification obtain the F1-measures increased from 0.641 (red bar) to 0.666 (orange bar). The macro-averaged F1 (MF1) are increased from 0.743 to 0.811. Especially, Figure 4 (b) shows that the recall of ECAT is increased from 0.849 to 0.873 and the averaged recall is increased from 0.805 to 0.850.

Figure 3. Accuracy of NB on the RCV1v2 dataset.

Figure 4. Results of NB with 400 features for class variants; (a) F1; (b) Recall
4.4.2. Results on SVM

Figure 5 shows the ranking accuracy of SVM classifier with feature ranking method using clustering-resulted subclass, ground truth subclass and main class on the RCV1v2 training dataset. We observe that 1) the ground truth subclass (G) also outperformed the main class (M) when the number of feature further increases; 2) compared with main class (M), using ground truth subclasses (G) for feature scoring measure can improve classification accuracy substantially; 3) using baseline cut-off value, resulted subclasses (HAC_cutoff1) achieved equal accuracy to the ground truth subclasses (G); 4) our algorithm (HAC_cutoff2) achieved higher accuracy than the main class (M) at number of feature from 100 to 3000 and approximately equal to the ground truth subclass (G); and 5) all lines in the Figure 5 are similar trend (with correlation coefficient > 0.9). The slope of resulted subclass line by the proposed method (HAC_cutoff2) is zero at 1300 features. Basically, the feature ranking method will select 1300 top-ranked features as the optimal feature subset for SVM classifier for class variants.

![Figure 5](image_url)  
**Figure 5.** Accuracy of SVM on the RCV1v2 dataset.

Next, SVM classifiers from the training process were used to classify testing documents and the classification performance was then evaluated. Figure 6 illustrates performance comparison at selected numbers of features (1300 features) of main class and subclass. First, Figure 6 shows that original main class (red bar) also produces the lowest performance in spite of using the high-effective classifier such as SVM. Second, we can see in all columns that using any subclasses (blue, green and orange bar) instead of main class (red bar) can improve both F1 and Recall, which is the same as the Naïve Bayes classifier. Third, in the smallest class (ECAT), Figure 6 (a) shows that subclass resulted from the Algorithm 1 with class-sensitive cutoff (HAC_cutoff2) also increases the F1-measures from 0.862 (red bar) to 0.872 (orange bar). The macro-averaged F1 (MF1) is increased from 0.918 to 0.931. Figure 6 (b) shows that the recall of ECAT is increased as well, from 0.809 to 0.829 and the averaged recall is increased from 0.904 to 0.923.

![Figure 6](image_url)  
**Figure 6.** Results of SVM with 1300 selected features for class variants: (a) F1; (b) Recall
4.5. Discussion

By the experimental results, we can summarize that using balanced subclasses instead of imbalanced main class can produce features that has less effects from class-imbalanced distribution in the dataset. Classification from those features on smaller main classes is more accurate than features from standard method. Proposed method based on the HAC algorithm with class-sensitive threshold (HAC_cutoff) can be used to automatically re-cluster efficient subclasses. Selected features are effective when they are evaluated on both SVM and Naïve Bayes. Clustering-resulted subclasses from proposed method are also compared with the ground truth subclass (G) extracted from metadata of the dataset. Macro-averaged F1 (MF1) of resulted subclass produced by the proposed algorithm is slightly lower than of the ground truth subclasses. Since clustering technique is unsupervised learning method which has no prior knowledge; thus, a number of errors normally occurs. However, if the ground truth subclass does not exist, particularly in the real application, the proposed algorithm is still available and it yields the sufficient results.

5. Conclusion

In this paper, we proposed a method to select features from class-imbalanced dataset by only analyze statistics of dataset, without using any predefined value or preliminary empirical experiments. We applied the Hierarchical Agglomerative Clustering (HAC) algorithm to separate data from original main classes into many subclasses according to their proximities, then all subclasses are considered for feature scoring measure instead of the main classes. The proposed method aim to reduce the influence of large classes to feature scoring measure, and increase chances of small classes to be selected its rare and important feature for improving the effectiveness of multi-class classification in case that these class-size are imbalanced. Unlike the other data preprocessing paradigms, such as re-sampling based approach, our proposed method avoids the problem of the important concept missing, overfitting, and over-generalization by considering both within-class and between-class relationships via the clustering method. Moreover, including a method for determining appropriate cut-off threshold based on a statistical parameter and class-sensitive weighting can automatically, efficiently and effectively separate the imbalance dataset. Our algorithm is developed and tested on the RCV1v2 dataset which has predefined hierarchies of classes and we used as the ground truth subclasses for performance comparison using two classifiers, SVM and Naïve Bayes. Experiments show that proposed method works well and can separate main classes to new more balanced subclasses. We also found that considering relevance between features and subclasses, instead of main classes for feature scoring measure improves the performance of multi-class classification with class-imbalanced dataset. The proposed method presents slightly lower performance than results of the ground-truth subclasses and results of manually-tuned baseline cluster. However, its results which clearly outperform using only the main classes led us to conclude that our proposed method is suitable for the data sets with high imbalanced class distribution and the sub-hierarchies class is unavailable.

In the future work, we will perform a quantitative analysis with the state-of-the-art methods to justify the advantage of the proposed approach. An extended study has been done to explore further performance on other imbalanced class dataset both text and non-text domains.

6. Acknowledgement

This research is supported by a grant from the Kasetsart University Research and Development Institute. Part Pramokchon is supported by the Kasetsart Graduate School for his graduate research.

7. References