Chapter - 6

EXPERIMENTAL SETUP

6.1 Introduction

The main motivation of this chapter is to identify and recognise the various inductive learning algorithms and datasets using different configuration settings in order to analyse the practical discriminative power of classifiers and discover the potential configuration pattern which is informative and beneficial for text classification task. The configuration with optimal performance is extensively analysed and improved in the next chapter.

This chapter elaborates the experimental methodology of this thesis. Section 6.1 describes the learning algorithms used in this thesis. Section 6.2 presents the benchmark corpora on which the experiments have been conducted. Section 6.3 introduces evaluation metrics to measure the performance of various methods.

6.2 Machine Learning Algorithms adapted in this Thesis

Among the most prominent machine learning algorithms, it has intentionally chosen SVM and kNN, because of the reported experimental achievement by these two methods in the literature [YL98, Joa98, DPHS98]. Other reason is that although other algorithms such as Decision Tree and Naive Bayes are also widely used, they are not included because their inability to handle real numbers of term weights except for the binary text representation. Even if it makes them to handle real numbers with some constraints [MN98], these learning algorithms reportedly have shown worst performance than the two profound algorithms. One more reason to choose these two state-of-the-art algorithms is the scalability issue, because these two algorithms are reportedly to have scaled to thousands of features in some classification problems.

6.2.1 Support Vector Machine

SVM has increasingly been used in wide variety of classification tasks, due to its ability to handle large scale datasets efficiently without hurting classification accuracy. It has been proven as profound method in terms of better performances than other methods by many
researchers [Joa98, DPHS98, LK02, DVW99]. One can classify the SVM into two broad categories namely linear and non-linear with respect to the kernel function adapted by the SVM. In this thesis, it adopted the linear SVM rather than non-linear SVM. There are many reasons behind this decision. They are as follows. First and foremost, linear SVM is simple and fast [Vap00, DPHS98]. Second, theoretical analysis carried out by [Joa98] on text categorization task suggest that the linear classifier itself is the best fit model guaranteed to perform better than the nonlinear models. Third, since the present focus is on comprehensive comparison of feature selection methods rather than tuning the parameters of kernel functions used in SVM. Due to these well established reasons, it adapted an SVM with linear kernel function as the baseline classifier in this thesis.

With regard to SVM software package, it uses the *SVM light* system version 6.02 which is an very user friendly and efficient implementation by Joachims [Joa97] and has been widely used in previous reported research works. The implementation has three main sub-modules; they are pre-learning phase where input files are formatted with respect to SVM style. Learning phase, where command line SVM program is executed for each category in a set of all classes. In this phase, class comparisons are automatically done by the one-versus-all style in which every category is compared against the aggregation of all the possible categories for the assignment of each document. Thus, batch program is called for each different category independently. Post-learning phase, where SVM output data is prepared for evaluation measures such as precision and recall.

### 6.2.2 k Nearest Neighbours

Due to its simplicity and effectiveness, the kNN algorithm still remains as a powerful learning algorithm in the research community. However, it has much received criticism because of its inefficiency in handling relatively high dimensional data sets. This criticism comes from the inherent nature of this lazy learning algorithm as it doesn’t seem to have a true learning phase and thus incur a high computational complexity at the classification time. Moreover the parameter $k$ plays an important role in tuning the performance of $k$NN. The performance of $k$NN is unmodifiable if the value of is fixed. Estimating the value for this parameter requires a separate validation set in addition to the training dataset. Quite often researchers used a part of training dataset itself as validation set to predict the parameter $k$ and rest of the training set is meant for training the classifier. [LC96] used $k = 20$. The previous reported work in [YL99] set $k$ as between 30 and 45 because constant effectiveness
was observed in that particular range. [Joa98] also observed similar performance effectiveness for \( k \)NN when \( 30 \leq k \leq 45 \). In experimental research, the work experimented with varying \( k \) values where \( k \in \{1, 10, 15, 20, 35, 45\} \) for the \( k \)NN classifier. The experimental results based on this parameter \( k \) along with performance on datasets are reported in the results sections in the next chapter.

6.3 Benchmark Datasets

In order to obtain a meaningful comparison between observed experimental results with the published results on text categorization, it conducts experiments on standard widely-used benchmark data collections. It next describes these datasets and various pre-processing task designed for these text corpora.

6.3.1 Text Pre-processing

First and foremost, a list of stop words is created, which are assumed to have no information content and these words are commonly called as functional or connective words. Appendix A lists the 530 stop words. After filtering the stop words and punctuation characters, the Porter's stemming was done to find the morphological root of the words based on porter stemming algorithm [Por80]. As for as feature selection is concerned, this work adapted some of the most popular and powerful metric as discussed in Section 4.4 such as information gain, odd ratio and chi square. Many of the reported work observed these metric [YP97] are found to be the most effective feature selection metrics in their experiments. However, it is learnt from previous study that SVM doesn’t require any sort of feature selection metric due to its capability to handle high dimensional features and thus even if it uses any such metric leads to not improving or even slightly degradation in SVM performance [LK02] and [DL04], but this work has not conducted any such experiments by using all features without any feature selection.

6.3.2 Reuters-21578 Corpus

Reuters is one of the most widely used text corpus in text categorization tasks. It consist of numerous mews articles concerned with economics and these articles are filed under different categories. Most of the experimental works related to text categorization mention the use of Reuters as the viable dataset to validate the performance of the inductive learning classifier. Because of above cited reasons, experimenting with this kind of text corpus surly leads to meaningful comparison with existing work. Though there is numerous
numbers of categories in Reuters dataset, most of the existing reports used only those
documents from top ten largest categories of the Reuters-21578 document collection. So it
followed this selection strategy of using top ten largest categories. As for as document
representation is concerned, the most straight forward bag-of-words approach is adapted in
this thesis. With regard to the widely-used ModApte split (it is now a standard split in
Reuters-21578 corpus), this work used a standard R8 split which has around 7,674 news
stories which have been partitioned into a training set of 5,485 documents and a test set of
2,189 documents. The distribution of training and test documents for each class is presented
in the table 6.1.

Table 6.1 Description of Reuters 21578 Dataset

<table>
<thead>
<tr>
<th>Class</th>
<th># train docs</th>
<th># test docs</th>
<th>Total # docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acq</td>
<td>1596</td>
<td>696</td>
<td>2292</td>
</tr>
<tr>
<td>Crude</td>
<td>253</td>
<td>121</td>
<td>374</td>
</tr>
<tr>
<td>Earn</td>
<td>2840</td>
<td>1083</td>
<td>3923</td>
</tr>
<tr>
<td>Grain</td>
<td>41</td>
<td>10</td>
<td>51</td>
</tr>
<tr>
<td>Interest</td>
<td>190</td>
<td>81</td>
<td>271</td>
</tr>
<tr>
<td>money-fx</td>
<td>206</td>
<td>87</td>
<td>293</td>
</tr>
<tr>
<td>Ship</td>
<td>108</td>
<td>36</td>
<td>144</td>
</tr>
<tr>
<td>Trade</td>
<td>251</td>
<td>75</td>
<td>326</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>5485</strong></td>
<td><strong>2189</strong></td>
<td><strong>7674</strong></td>
</tr>
</tbody>
</table>

All pre-processing step such as stop word removal, special character and punctuation
removal were done. Morphological root word were obtained by the Porter's stemmer [Por80].
The minimal threshold value for term length is fixed as 3 (i.e. every term has at least 3
characters). It eliminated the vectors with null values (i.e. vectors with all attributes valued
0). After performing all pre-processing tasks, the resulting controlled vocabulary has around
15937 words (terms or features). The dimensionality of feature space is further reduced by
feature selection methods such as IG, Chi square and Odd ratio. This filtering process
selected the top p features per category from the training dataset. For the sake of experiment
purpose, it sets $p \in \{50, 100, 150, 200, \ldots \ldots, 2000\}$ respectively. A remarkable property of Reuters text corpus is the document distribution is skewed across categories.

### 6.3.3 WebKB Corpus

The WebKB data set contains web pages gathered from four different college Web sites, namely Cornell, Texas, Washington, Wisconsin, and some miscellaneous Web pages. The pages are divided into seven categories: student, faculty, staff, course, project, department and other. WebKB contains 8,282 Web pages. The average document length in WebKB dataset is 130 terms. In this thesis, the four most popular entity-representing categories are used: student, faculty, course and project, all together containing 4199 pages. The distribution of training and test webpage’s for each class is presented in the table 6.2.

**Table 6.2. Description of WebKB Dataset**

<table>
<thead>
<tr>
<th>Class</th>
<th># train docs</th>
<th># test docs</th>
<th>Total # docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>project</td>
<td>336</td>
<td>168</td>
<td>504</td>
</tr>
<tr>
<td>course</td>
<td>620</td>
<td>310</td>
<td>930</td>
</tr>
<tr>
<td>faculty</td>
<td>750</td>
<td>374</td>
<td>1124</td>
</tr>
<tr>
<td>student</td>
<td>1097</td>
<td>544</td>
<td>1641</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2803</strong></td>
<td><strong>1396</strong></td>
<td><strong>4199</strong></td>
</tr>
</tbody>
</table>

All pre-processing step such as stop word removal, special character and punctuation removal were done. Morphological root words were obtained by the Porter's stemmer [Por80]. It is fixed that the minimal threshold value for term length is 3 (i.e. every term has at least 3 characters). After performing all pre-processing tasks, the resulting controlled vocabulary has around 7095 words (terms or features). The dimensionality of feature space is further reduced by feature selection methods such as Information Gain, Chi square and Odd ratio. This filtering process selected the top $p$ features per category from the training dataset. For experiment purpose, it sets $p \in \{50, 100, 150, 200, \ldots \ldots, 2000\}$ respectively.
6.3.3 20 News Groups Corpus

Among the text corpus used in text categorization, 20 News groups is the most sought dataset after Reuters. This corpus consist of news group articles of size around 20,000 which are evenly fallen into twenty blogs in which every document is categorised as one of the 20 categories with respect to the category name of the blogs that the document was assigned to. Though it contains 20 different categories, some articles have close resemblance to other categories. For instance, the articles assigned to category of \textit{comp.sys.ibm.pc.hardware} have close resemblance to those in category of \textit{comp.sys.mac.hardware}. However, this resemblance is restricted to limited number of categories and other categories are highly not correlated, for instance, the category of \textit{misc.forsale} and category of \textit{soc.religion.christian}. The distribution of training and test webpage’s for each class is presented in the table 6.3.

After having done all pre-processing tasks like removing duplicates and headers, the resultant 18821 documents are sorted and are partitioned into 11293 training documents (about 60%) and 7528 test documents (about 40%). As a result of this portioning, the training and test documents are now evenly distributed across the given 20 categories. While comparing with the skewed category distribution in the Reuters corpus, the 20 categories in the 20 Newsgroups corpus seems to follow a uniform distribution at large. The controlled vocabulary, after doing all pre-processing takes such as stop words removal (530 stop words) and filtering word frequency of less than 3 to 6 times in both positive and negative categories respectively, has yielded 50088 words. The dimensionality of feature space is then reduced by applying feature selection methods such as IG, Chi square and Odd ratio. This filtering process selected the top $p$ features per category from the training dataset. For experiment purpose, it sets $p\in\{50,100,150,200,250,\ldots,2000\}$ respectively.

Table 6.3. Description of 20 News Group Dataset

<table>
<thead>
<tr>
<th>Class</th>
<th># train docs</th>
<th># test docs</th>
<th>Total # docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>alt.atheism</td>
<td>480</td>
<td>319</td>
<td>799</td>
</tr>
<tr>
<td>comp.graphics</td>
<td>584</td>
<td>389</td>
<td>973</td>
</tr>
<tr>
<td>comp.os.ms-windows.misc</td>
<td>572</td>
<td>394</td>
<td>966</td>
</tr>
<tr>
<td>comp.sys.ibm.pc.hardware</td>
<td>590</td>
<td>392</td>
<td>982</td>
</tr>
<tr>
<td>comp.sys.mac.hardware</td>
<td>578</td>
<td>385</td>
<td>963</td>
</tr>
</tbody>
</table>
6.4 Evaluation Metrics

An evaluation of text categorization performance has always been centred on two main issues; they are computational efficiency and categorization effectiveness [MN98]. The computational efficiency of the learning algorithms is actually not a concern of researchers because of the currently available processing ability and memory speed. Therefore here it deals with categorization effectiveness only since it is believed as more reliable one when it comes to experimentally comparing different learning algorithms.

6.4.1 Precision and Recall

By convention, effectiveness of the classifier system is scaled by using two predominant IR criteria such as precision and recall. Precision \((p)\) with respect to \(c_i\), the percentage of documents deemed to belong to \(c_i\) that in fact belong to the category and recall \((r)\) with respect to \(c_i\), the percentage of documents belonging to \(c_i\) that are in fact deemed to belong to the category. The two IR measures are defined in following equations:
\[ p_i = \frac{TP_i}{TP_i + FP_i} \quad 6.1 \]
\[ r_i = \frac{TP_i}{TP_i + FN_i} \quad 6.2 \]

At this point, it would be meaningful to consider a confusion matrix given in Table 6.4 which helps in computing true positive, false positive and false negative in order to understand the intuition behind the standard measures *precision* and *recall*.

### Table 6.4. Confusion Matrix

<table>
<thead>
<tr>
<th>Category ( i )</th>
<th>Expert Judgement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE</td>
</tr>
<tr>
<td>Classifier Judgement</td>
<td>( TP_i )</td>
</tr>
<tr>
<td>FALSE</td>
<td>( FN_i )</td>
</tr>
</tbody>
</table>

Where

- \( TP_i \) *True Positives*. Those assessments where classifier and expert agree for a label assignment.
- \( FP_i \) *False Positives*. Those labels assigned by the classifier that does not agree with expert judgement.
- \( FN_i \) *False Negatives*. Those labels classifier failed to assign as they were by expert.
- \( TN_i \) *True Negatives*. Those non assigned labels that also were discarded by the expert.

To compute the right estimate for *precision* and *recall*, researchers adopted two different techniques, *micro-averaged*: where the *precision* and *recall* are obtained by summing over all individual decisions; *macro-averaged*: where *precision* and *recall* are first evaluated locally for each category, and then globally by averaging over the results of the different categories; It is observed from previous research that these two methods may contradict and may give surprising results, in some situations where the different categories
have distinct generality. It means that the classifier which perform well on categories with low generality (i.e. categories with few positive training instances) will be stressed by macro-averaged and on the other hand it will be stressed much less by micro-averaged. It is quite evident from this that the two methods will be stabilised (equalized) on the datasets which follows uniform category distribution. The question of choosing one measure over another is purely rest with the requirement of application. Either precision or recall makes no sense in isolation from each other as it is widely accepted phenomenon in IR rule that higher levels of precision may be obtained at the price of low values of recall and vice versa. Hence, a classifier must be evaluated by means of a measure which takes into account both precision and recall. Toward achieving this, various criteria have been proposed in the literature. Among them, the two most common measures adopted by the researchers are, $F_1$ measure and breakeven point.

6.4.2 Micro and Macro Averaging

To measure the overall effectiveness on precision and recall and $F_1$ measure over all categories, micro and macro averaging are used. Micro averaging calculates the total true positives, false positives and false negatives over all categories and uses the same precision, recall and $F_1$ formulas given above to calculate each of the measure over all categories. Table 6.2 describes the method of computing micro and macro averaged precision/recall over all categories in the given dataset.

Macro averaging uses the individual categories precision, recall and $F_1$ measure to build the average of each of these measures over all categories. The macro averaging may lead to inaccurate evaluation if the categories have skew distribution, as the number of categories is considered in building the average. Thus a category with small number of documents and high precision may mislead the overall precision.

Table 6.5 Micro and Macro averaged precision/recall measures

<table>
<thead>
<tr>
<th>Micro</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{micro} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} TP_i + FP_i}$</td>
<td>$P_{macro} = \frac{1}{C} \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} TP_i + FP_i}$</td>
</tr>
<tr>
<td>$R_{micro} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} TP_i + FN_i}$</td>
<td>$R_{macro} = \frac{1}{C} \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} TP_i + FN_i}$</td>
</tr>
</tbody>
</table>
6.4.3 $F_1$ Measure

When it comes to single quantity that measures the performance of the classifier, $F_1$-measure is the most sensible option. It is defined as the harmonic mean of precision and recall; that is

$$F_\beta = \frac{(\beta^2 + 1) \cdot p \cdot r}{\beta^2 \cdot p + r} \quad 6.3$$

Where $p$ – precision

$r$ - recall

$\beta$ – control parameter

In equation 6.3, the parameter $\beta$ gives relative degree of importance to both precision and recall. When $\beta = 0$, the function $F_1$ coincides with precision, on the contrary when $\beta = \infty$, $F_1$ coincides with recall. However, to attribute equal importance to precision and recall, researchers often set the value of $\beta$ as 1. Thus, the $F_1$ function is transformed as:

$$F_1 = \frac{2 \cdot p \cdot r}{p + r} \quad 6.4$$

Like precision and recall, the $F_1$ function also can be computed as micro-averaged (where the precision and recall are obtained by summing over all individual decisions) and macro-averaged (where precision and recall are first evaluated locally for each category, and then globally by averaging over the results of the different categories). Moreover, these two methods also give quite contradictory results, especially when categories of datasets have very distinct generality.

6.4.4 Break Even Point

Breakeven point is another important evaluation measure aimed at value in which precision equals recall. To compute the breakeven point, a plot of precision as a function of recall is calculated by repeatedly varying the threshold $\rho$; thus the breakeven point value is the value of precision or recall for which the plot intersects the precision = recall line. This idea behind this approach is that, by decreasing the parameter $\rho$, recall steadily increases monotonically from 0 to 1 and precisions usually decreases monotonically from a value near 1 to lower. When precision and recall are not exactly equal for any such value, the $\rho$ is set to...
the value for which precision and recall are closest, or an interpolated break-even is computed as the average of the values of precision and recall. It must be noted that there may be no parameter setting that yields the break-even; in this case the final break-even value obtained by interpolation is artificial. Compared with the $F_1$ function, [YL99] showed that the break-even point of a classifier is always less or equal than its $F_1$ value.

6.4.5 Accuracy

Accuracy is generally viewed as weak indicator in terms of evaluating the classifier effectiveness. Though it is widely used in the machine learning literature, it is not adapted in TC. The primary reason for this is that, as [YL99] reported that the large number of documents in the whole corpus belongs to negative class makes them much more insensitive to variations in the number of correct decisions than precision and recall. This kind of property makes the classifier behave very much like a trivial rejecter.

$$\text{Acc}_i = \frac{TP_i + TN_i}{TP_i + FP_i + FN_i + TN_i}$$

6.5

Where

- $\text{Acc}_i$ – accuracy
- $TP_i$ – true positives
- $FP_i$ – false positives
- $TN_i$ – true negatives
- $FN_i$ – false negatives

Before doing experiments, the evaluation measure on which classifier is to be evaluated must be finalised. Then, a classifier can be fine tuned by according to predetermined operational setting such as thresholds and other parameters in order to achieve the best possible effectiveness by that classifier. Tuning any such control parameter is usually done via experiments. It means that performing the same experiment on the validation set repeatedly with various values for control parameters while values for parameter $\rho$ remains fixed. The value for which the experiments have produced the best effectiveness is retained for the parameter $\rho$. In general, all experiments adopted the micro-averaged precision, recall and $F_1$ functions as the measures of performance for effectiveness.
6.5 Text Categorization Framework

In order to conduct experiments to test the validity of the proposed approach, this work has built a comprehensive text categorization platform of its own design and development. This work has opted to build a comprehensive new framework for text categorization because of many state-of-the-art classifiers and tools are freely available for researchers, while those software tools are restricted control over their operations. The proposed framework automates the entire life-cycle of text categorization tasks including text pre-processing (such as stop word removal, stemming), feature extraction, construction, selection and valuation, followed by actual classification. The system currently support parsing of tokens, sentence boundary detection, stemming [Por80], controlled dictionary construction, a variety of feature selection metrics (IG, GHI, OR, etc), and feature weighting schemes (tf.idf, df, etc.). This framework has more than 100 configurable parameters that control its operations spread across the modules. Moreover, it interfaces with SVM, KNN and Naïve Bayes text categorization algorithms, and computes all standard measures of categorization performance such as accuracy, precision, recall, breakeven point and $F_1$ measure. It was designed with a particular emphasis on object orientation and computational efficiency, and portably implemented in the Java programming language using NetBeans IDE. The system has built-in support to load all standard bench mark dataset such Reuters-21578 [Lew97], RCV1 [RLYY04], 20 Newsgroups [Lan95], and WebKB, whereas other additional datasets can be easily integrated in to the system in a modular fashion.

Figure 6.1 shows the architectural diagram of the proposed text categorization framework. Each document undergoes the following processing tasks. Unstructured document text is first tokenized, and category of the document is kept aside as first token to emphasize its importance. Then, stop words, numbers and mixed alphanumeric strings are eliminated, and the morphological root words are obtained for the remaining words. In addition to this, rare words whose occurrence is less than three in documents are eliminated. Since earlier studies indicated that the merits of Bag of Word kind of representation is indeed useful for SVM like text classifier, the bag of word approach is used to represent documents with the set of features generated for the document collection by analyzing its content as explained in Section 3.4. The generated features, however, undergo feature selection using several metric such as information gain, odd ratio and chi square etc. Finally, feature weighting is done using the standard criterion as explained in Section 4.3.2.
6.6 Summary

This chapter described the various state-of-the-art inductive learning algorithms adapted in this thesis for text categorization. Moreover, it analysed different benchmark datasets and evaluation criteria using different configuration settings for comprehensive experimental research. In order to analyze the discriminative ability of classifiers and feature selection methods, it has built a computational framework and described the practical advantages of the implementation. The proposed framework is informative and beneficial for text classification task from various perspectives.