CHAPTER 4

METHODS AND TOOLS

In this chapter, the classification methods and tools used are discussed in detail. The feature reduction method used is discussed in section 4.1. Then followed by a description of the classification methods employed (section 4.2) and various sampling methods used to handle imbalanced datasets are discussed in section 4.3. The datasets and tools used for implementation are described in section 4.4 and 4.5 respectively.

4.1. Feature Reduction

One major difficulty of the sentiment classification problem is the high dimensionality of the features used to describe texts, which raises big hurdles in applying many sophisticated learning algorithms to text sentiment mining. The aim of feature reduction methods is to obtain a reduction of the original feature set by removing some features that are considered irrelevant for sentiment classification to yield improved classification accuracy and decrease the running time of learning algorithms (Abbasi.A. et al., (2008a), Tan.H. et al., (2008), Wang et al., (2007)).

Principal Components Analysis (PCA) is applied to reduce the dimensions of the inputs when the dimensions of the input are large and the components are highly correlated. PCA determines a smaller set of artificial variables which will represent the variance of a set of observed variable. The artificial variables calculated are called principal components. The variables are orthogonalized by the PCA. The principal components with largest variation are chosen and components with less
variation are eliminated from the dataset. The PCA is applied as follows on a set of data (Kambhatla.N. et al. (1997)),

**Step 1:** Get the input data.

**Step 2:** Subtract the mean from each of the data dimensions.

**Step 3:** Calculate the covariance matrix to find out how much the dimensions vary from the mean with respect to each other.

**Step 4:** Calculate the eigen vectors and eigen values of the covariance matrix.

**Step 5:** Choosing components and forming a feature vector using the eigen vectors and eigen values.

**Step 6:** Deriving the new dataset by taking the transpose of the vector and multiply it on the left of the original dataset.

### 4.2. Classification Methods

Supervised machine learning techniques implicate the use of a labelled training corpus to learn a certain classification function and involve learning a function from examples of its inputs and outputs. The output of the learning function is either a continuous value or discrete. In this work, sentiment mining task is formulated as a supervised learning problem with two classes, positive and negative. Product reviews are used as training and testing data. The following Fig 4.1 shows the general framework used by supervised classification. The supervised algorithms used in this work are discussed in the following sub sections.
Naive bayes classifier is a simple probabilistic model based on the bayes rule along with a strong independence assumption. Naive bayes model involves a simplifying conditional independence assumption. That is, given a class (positive or negative), the words are conditionally independent of each other. This assumption does not affect the accuracy in text classification by much, but makes really fast classification algorithms applicable to the problem. In sentiment classification, the maximum likelihood probability of a word belonging to a particular class is given by the expression (Eq.4.1),

$$P(x_i|c) = \frac{\text{Count of } x_i \text{ in documents of class } c}{\text{Total no of words in documents of class } c}$$ (4.1)

The frequency counts of the words are stored in hash tables during the training phase. According to the bayes rule, the probability of a particular document belonging to a class $c_i$ is given by (Eq.4.2),
\[ P(c_i|d) = \frac{P(d|c_i)P(c_i)}{P(d)} \]  

(4.2)

If we use the simplifying conditional independence assumption, that given a class (positive or negative), the words are conditionally independent of each other. Due to this simplifying assumption (Eq.4.3) the model is termed as “naive”.

\[ P(c_i|d) = \frac{(H_{c}P(x|c_i))P(c_i)}{P(d)} \]  

(4.3)

Here the \( x_i \)'s are the individual words of the document. The classifier outputs the class with the maximum posterior probability. If the classifier encounters a word that has not been seen in the training set, the probability of both the classes would become zero and there will not be anything to compare between. This problem can be solved by laplacian smoothing (Eq.4.4),

\[ P(x_i|c_j) = \frac{\text{Count}(x_i)+k}{(k+1)\times\text{No of words in class } c_j} \]  

(4.4)

Usually, \( k \) is chosen as 1. This way, there is equal probability for the new word to be in either class (Xia.R. et al., (2011), Melville.P. et al., (2009), Ye.Q. et al., (2009), Tan.S. et al., (2008)).

### 4.2.2. Support vector machine (SVM)

SVM is a widely used supervised classifier that has a solid theoretical foundation and performs classification more accurately than most other algorithms in many applications. Many researchers have reported that SVM is perhaps the most accurate method for text classification (Liu, 2011). It is also widely used in sentiment classification (Xia.R. et al., (2011), Prabowo.R. et al., (2009), Ye.Q. et al., (2009), Tan.S. et al., (2008)). SVM is a linear learning method that finds an optimal hyperplane to separate two classes. As a supervised
A generalization of the linear binary SVM classifier, \( \gamma (x) \), classifying instances of \( x \in \mathbb{X} \) described by a vector of features \((x_1, x_2)\), into classes \( C = \{ \text{positive, negative} \} \) can be expressed as follows (Eq. 4.5)

\[
\gamma(x) = w \cdot x + b
\]  

(4.5)

Where \( w \) represents a vector that is perpendicular to the boundary and \( b \) determines the offset of the boundary from the origin. Based on this formalization, instances of \( x \) will be classified as positive if \( \gamma(x) \geq 0 \) or negative if \( \gamma(x) \leq 0 \).

When classes cannot be linearly separated, as shown in Fig. 4.2 (b), the input data space is transformed into a higher dimensional feature space in order to make data linearly separable and suitable for the linear SVM formulation. Usually, this transformation is achieved by
using a kernel function. It makes possible to determine a nonlinear decision boundary, which is linear in the higher dimensional feature space, without computing the parameters of the optimal hyper plane in a feature space of possibly high dimensionality. Therefore, the solution can be written as a weighted sum of the values of certain kernel function evaluated at the support vectors.

### 4.2.3. Logistic regression (LR)

Logistic regression is an approach to learning functions of the form \( f : \mathbf{X} \rightarrow \mathbf{Y} \), or \( P(Y|X) \) in the case where \( Y \) is discrete valued and \( X = \{X_1 \ldots X_n\} \) is any vector containing discrete or continuous variables. For example consider the case where \( Y \) is a boolean variable. Logistic regression assumes a parametric form for the distribution \( P(Y|X) \), then directly estimates its parameters from the training data (Peduzzi.P. et al., (1996), Neslin.S.A. et al., (2006)). The parametric model assumed by logistic regression in the case where \( Y \) is boolean in (Eq.4.6),

\[
P(Y=1|X) = \frac{1}{1+\exp\left(w_0+\sum_{i=1}^n w_i X_i\right)}
\]

and

\[
P(Y=0|X) = \frac{\exp\left(w_0+\sum_{i=1}^n w_i X_i\right)}{1+\exp\left(w_0+\sum_{i=1}^n w_i X_i\right)}
\]

Notice that equation (4.7) follows directly from the equation (4.6), because the sum of these two probabilities must equal 1. One highly convenient property of this form for \( P(Y|X) \) is that it leads to a simple linear expression for classification. To classify any given \( X \) assign the value \( y_k \) that maximizes \( P(Y = y_k|X) \). Put another way, assign the label \( Y = 0 \) if the following condition holds (Eq. 4.8),

\[
1 < \frac{P(Y=0|X)}{P(Y=1|X)}
\]
substituting from equations (4.6) and (4.7), this becomes (Eq.4.9),

\[ 1 < \exp (w_0 + \sum_{i=1}^{n} w_i X_i) \]  

(4.9)

and taking the natural log of both sides we have a linear classification rule that assigns label \( Y = 0 \) if \( X \) satisfies (Eq. 4.10)

\[ 0 < (w_0 + \sum_{i=1}^{n} w_i X_i) \]  

(4.10)

and assigns \( Y = 1 \) otherwise.

### 4.2.4. Neural networks

The central idea of a neural network is to derive features from linear combinations of the input data and then model the output as a nonlinear function of these features. The structure of the most used neural networks consists of three layers such as an input, a hidden and an output layer of nodes. A neuron is a simple mathematical model that produces an output value in two steps. First, the neuron computes a weighted sum of its inputs and then applies an activation function to this sum to derive its output. The activation function is typically a nonlinear function and it ensures that the entire network can estimate a nonlinear function, which is learned from the input data.

#### 4.2.4.1. Back propagation neural network (BPN)

The BPN is a multilayered, feed forward neural network and is by far the most extensively used. It is also considered one of the simplest and most general methods used for supervised training of multilayered neural networks. Back propagation works by approximating the nonlinear relationship between the input and the output by adjusting the weight values internally. It can further be generalized for the input that is not included in the training patterns. Generally, the back propagation network has two stages, training and testing. The following
While error is too large

Step 1. For each training pattern presented in random order do the following

a. The inputs are applied to the network.
b. Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer.
c. Calculate the error at the outputs.
d. Use the output error to compute error signals for pre output layers.
e. Use the error signals to compute weight adjustments.
f. Apply the weight adjustments.

Step 2. Periodically evaluate the network performance.

Fig 4.3 BPN topology

4.2.4.2. Probabilistic neural network (PNN)

A probabilistic neural network is based on statistical bayesian classification algorithm. The functions are organized into multilayered feed forward network with four layers such as input layer, pattern layer, summation layer and output layer. The input layer consists of input nodes, which are the set of measurements. The pattern layer is fully connected to the input layer, with one neuron for each pattern in the training set. The pattern layer outputs are selectively connected to the summation units depending on the class of patterns (Fig 4.5).

PNN is a nonlinear classification technique and essentially a parallel algorithm based on bayesian minimum risk criteria.

Fig 4.5  PNN topology

Although SVM performs well in sentiment classification in most cases, its drawback is high computational cost in finding the best parameter combinations. But PNN is simple and is easily trained, because it has only one parameter to be optimized in the experiment.

Given an unknown sample $x$, we can compute its posterior probability $P(c_i|x)$ to determine which class label the sample $x$ belongs to. According to Bayesian rule, $P(c_i|x)$ is proportional to the multiplication of all prior probability $\pi_i$ by probability density function $f_i(x)$. That can be represented as: $P(c_i|x)\alpha \pi_i f_i(x)$. Let $m$ be the number of training samples, $n$ the number of genes, $x_{ij}$ the $j$th training sample for class $i$, and $k_i$ the number of samples of class $i$. The Parzen estimate probability density function for class $i$ can be written as (Eq. 4.11)

$$f_i(x) = \frac{1}{(2\pi)^{m/2}\sigma^m} \frac{1}{k_i} \sum_{j=1}^{k_i} \exp\left[-\frac{(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}\right]$$ \quad (4.11)$$

where $\sigma$ is a smoothing parameter. The choice of $\sigma$ has a great effect on the estimation error of the PNN classifier and is determined experimentally by comparing their corresponding classification accuracy rates. If there is no way to obtain the prior probability, the prior probability can be evaluated by the following formula (Eq. 4.12)

$$\pi_i = \frac{k_i}{\sum_{j=1}^{c} k_j}$$ \quad (4.12)$$

where $c$ denotes the number of subclasses in dataset. The datasets used in this work has only two sub classes i.e. positive and negative. The steps involved in the PNN model are shown in Fig 4.6 (Savchenko.A.V. et al, (2013)).
4.2.5. Ensemble methods

Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem. In contrast to single machine learning approaches that try to learn one hypothesis from the training data, ensemble methods try to construct a set of hypotheses and combine them for use. The generalization ability of an ensemble method is usually much stronger than that of a single learner, which makes ensemble methods very attractive (Li.W. et al., (2012), Tsutsumi.K. et al., (2007)).

4.2.5.1. Bagging

The main idea is to construct each member of the ensemble from a different training dataset, and to predict the combination by uniform averaging over class labels (Whitehead M. et al.,(2008)). A bootstrap sample of S items is selected uniformly at random with replacement. This means each classifier is trained on a sample of examples taken with a replacement from the training set, and each sample size is equal to the size of the original training set. Then, they are aggregated into to make a collective decision using majority voting. Therefore, bagging

---

**Fig 4.6 Steps in PNN**

**Step 1.** The input layer neurons distribute input measurements to all the neurons in the pattern layer.

**Step 2.** The second layer has the Gaussian kernel function formed using the given set of data points.

**Step 3.** The third layer performs an average operation of the outputs for each review class.

**Step 4.** The fourth layer performs a vote, selecting the largest value and class label is then determined.
produces a combined model that often performs better than the single model built from the original single training set (Fig. 4.7). The final classifier $H(x)$ is formed by aggregating the “n” classifiers to classify an instance $x$, a vote for class $y$ is recorded by every classifier $H_i(x)= y$. $H(x)$ is the class with the majority voting.

| Input: Training set $T=(x_i,y_i), i=1$ to $n$: Integer $n$ (iteration number). |
| Output: Classifier $H(x)$. |
| for each iteration $i=1$ to $n$ |
| { |
| Select a subset $T_i$, of size $N$ form the original training examples $T$. |
| The size of $T_i$ is the same with the $T$ where some instances may not appear in $T_i$, while other appear more than ones. |
| Generate a classifier $H_i(x)$ from the $T_i$ |
| } |

**Fig 4.7 Pseudocode of bagging**

### 4.2.5.2. Bayesian boosting

Boosting is an iterative process, which adaptively changes the distribution of training examples so that the base classifiers will focus on examples that are hard to classify. Boosting has become one of the alternative frameworks for classifier design, together with the more established classifier like bayesian classifier (Polikar.R. (2006), Zhou.Z. H. (2012), Da Silva.N.F. et al., (2014)). The pseudocode of bayesian boosting method is shown in Fig 4.8.
**Input**: D, Training data D of labeled examples d_i.

**Output**: A classification model.

**Procedure**:

i. Initialize the weight w_i=1/n of each example d_i in D, where n is the total number of training examples.

ii. Generate a new dataset D_i with equal number of examples from D using selection with replacement technique.

iii. Calculate the prior probability P(C_j) for each class C_j in dataset D_i.

iv. Calculate the class conditional probabilities P(A_i|C_j) for each attribute values in dataset D_i.

v. Classify each training example t_i in training data D with maximum posterior probabilities.

vi. Updates the weights of each training examples d_i in D, according to how they were classified. If an example was misclassified then its weight is increased, or if an example was correctly classified then its weight is decreased. To updates the weights of training examples the misclassification rate is calculated, the sum of the weights of each of the training example d_i in D that were misclassified.

\[
error(M_i) = \sum_{i} w_i * err(d_i)
\]

vii. Where err(d) is the misclassification error of example d_i. If the example d_i was misclassified, then is err(d_i) 1. Otherwise, it is 0. If a training example was correctly classified, its weight is multiplied by error(M_i)/(1- error(M_i)). Once the weights of all of the correctly classified examples are updated, the weights for all examples including the misclassified examples are normalized so that their sum remains the same as it was before.

viii. To normalize a weight, the algorithm multiplies the weight by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified examples are increased and the weights of correctly classified examples are decreased.

ix. Repeat steps 2 to 6 until all the training examples d_i in D are correctly classified.

x. To classify a new/unseen example use all the probability set in each round and considers the class of new example with highest classifier’s vote.

**Fig 4.8 Pseudocode of bayesian boosting**
4.3. Sampling Methods

A dataset is imbalanced if the classification categories are not approximately equally represented. Generally, the prediction accuracy of the minority class becomes worse since the prediction accuracy of the majority class is dominant in satisfying objective functions of the models. The machine learning community has addressed the issue of class imbalance by assigning distinct costs to training examples or by resampling the original dataset, either by oversampling the minority class or under sampling the majority class. One of the concerns in using sampling methods is that over sampling might cause over fitting to added cases. Another concern is that under sampling might eliminate useful cases. Therefore, experimental evaluation of sampling methods is needed. In this work, the following sampling methods are used.

4.3.1. Random under sampling (RUS)

Random under sampling balances the minority class through the random elimination of some samples belonging to the majority class (Li.S et al. (2011a, 2011b)).

4.3.2. Synthetic minority over sampling (SMOTE)

SMOTE is the popular oversampling method. The basic idea of this algorithm is to place the new samples along the lines connecting existing rare samples. In SMOTE over sampling approach, the minority class is over sampled by creating synthetic' examples rather than by over sampling with replacement. The minority class is over sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any of the $k$ minority class nearest neighbors.
Depending upon the amount of over sampling required, neighbors from the $k$ nearest neighbors are randomly chosen. Synthetic samples are generated in the following way. Take the difference between the feature vector under consideration and its nearest neighbor.

```
Algorithm SMOTE (T, N, k)

Input: Number of minority class samples T, Amount of SMOTE N %; Number of nearest neighbors k
Output: ($N/100$) * T synthetic minority class samples

if $N < 100$
    then Randomize the T minority class samples
        $T = (N/100)\ast T$;
        $N = 100$;
endif
$N = (\text{int})(N/100)$;
$k = \text{Number of nearest neighbors} , \text{numattrs} = \text{Number of attributes}$
Sample[ || ]: array for original minority class samples
newindex: keeps a count of number of synthetic samples generated, initialized to 0, Synthetic[ || ]: array for synthetic samples
for $i \leftarrow 1$ to $T$
    Compute $k$ nearest neighbors for $i$, and save the indices in the nnarray
    Populate($N$, $i$, nnarray)
endfor

Populate($N$, $i$, nnarray)
while $N \neq 0$
    Choose a random number between 1 and $k$, call it nn. This step chooses one of the $k$ nearest neighbors of $i$.
    for attr $\leftarrow 1$ to numattrs
        Compute: $\text{dif} = \text{Sample}[\text{nnarray}[\text{nn}][\text{attr}]] - \text{Sample}[i][\text{attr}]$
        Compute: $\text{gap} = \text{random number between 0 and 1}$
        $\text{Synthetic}[\text{newindex}][\text{attr}] = \text{Sample}[i][\text{attr}] + \text{gap} \ast \text{dif}$
    endfor
    newindex++
    $N = N - 1$
endwhile
return
```

Fig 4.9 Pseudocode of SMOTE
Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general (Chawla. N. et al., (2002), Sun. A. et al., (2009)). The pseudocode of SMOTE is given in Fig 4.9.

4.4. Description of Datasets

Throughout this work, the two balanced datasets (Section 3.2) will be denoted as the balanced dataset-1 (BD1) and balanced dataset-2 (BD2) and imbalanced datasets will be denoted as the imbalanced dataset-1 (IBD1) and imbalanced dataset-2 (IBD2). All of the datasets considered in this work have a binary class (positive and negative), and consideration of multi class domains (positive, negative and neutral) is left to future work. In the past few years, a great attention has been received by web documents as a new source of individual opinions and experience. This situation is producing increasing interest in methods for automatically extracting and analyzing individual opinion from web documents such as customer reviews, weblogs and comments on news.

The widely used data for research on sentiment analysis are product review data downloaded from the Amazon website. Current available product review dataset for feature level sentiment mining was released by Hu. M et al., (2004). In addition to Hu’s dataset, customized datasets are constructed for BD2 and IBD2. For BD2 and IBD2, the review sentences are extracted for two different types of camera.

4.4.1. Balanced dataset

The polarity dataset (BD1) used is a set of product review sentences which were labelled as positive, negative or neutral. The review
sentences are collected from the publicly available customer review dataset (http://www.cs.uic.edu/~liub/FBS/FBS.html). This dataset contains annotated customer reviews of digital cameras. Out of 937 reviews of camera, 272 are negative, 355 are positive and 310 are neutral reviews. Outliers are performed as suggested in Briand.L.C. et al., (2000) and are not considered for further processing. In order to obtain a balanced data distribution for the binary classification problem, only 250 positive and 250 negative (500 reviews) reviews are considered. For each of the positive and negative review sentences the product attributes annotated in the review sentences are collected manually. Unique product features are grouped, which results in a final list of product attributes (features) of size 115. Among 115 product attributes, 96 are unigram attributes, 12 are bigrams and 7 are trigram attributes. In terms of these, the descriptions of review dataset models to be used in the experiment are given in Table 4.1.

A sample annotated sentence from the dataset BD1 is as given below,

“picture quality[+1]##a few of my work constituents owned the g2 and highly recommended the canon for picture quality.”

Where, picture quality[+1]: picture quality is a product feature.
+ denotes Positive opinion, 1 is the opinion strength and
##: start of review sentence.

The balanced dataset-2 (BD2) used is a set of product review sentences which were labeled as positive or negative. The review sentences from the publicly available customer review website www.amazonreviews.com are collected. The domain chosen is digital camera reviews. A java based web crawler was used to download 970
positive reviews and 710 negative reviews randomly. In the crawled reviews, it is found that, there are borderline and neutral reviews in between along with the clear positive and negative reviews. A review is discarded if it is not clearly aligned towards positive or negative sentiment. Outliers analysis is performed (Briand.L.C. et al., (2000)).

Twenty five sentences are identified as outliers and are not considered for further processing. The sentiment dataset obtained is a set of product review sentences of product which were labelled as positive or negative. As a result, there are 950 positive and 705 negative reviews.

<table>
<thead>
<tr>
<th>Table 4.1 Properties of dataset BD1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model I</strong></td>
</tr>
<tr>
<td><strong>Product</strong></td>
</tr>
<tr>
<td><strong>No.of. reviews</strong></td>
</tr>
<tr>
<td><strong>Positive reviews</strong></td>
</tr>
<tr>
<td><strong>Negative reviews</strong></td>
</tr>
<tr>
<td><strong>Feature</strong></td>
</tr>
<tr>
<td><strong>No.of.attributes</strong></td>
</tr>
<tr>
<td><strong>Attributes type</strong></td>
</tr>
<tr>
<td><strong>Class attribute</strong></td>
</tr>
<tr>
<td><strong>Vector space</strong></td>
</tr>
</tbody>
</table>
For our binary classification problem, to avoid the imbalanced class distribution, 600 positive and 600 negative reviews are selected randomly to establish the dataset. Features in this study of sentiment classification are the words, terms or phrases representing the product attributes that strongly express the opinions as positive or negative.

Each product has its own set of attributes (features). For each of the positive and negative review sentences, the product attributes discussed in the review sentences are collected. The unique characteristic of words representing the product attributes in a review sentence is that they are mostly nouns by POS tagging. POS tagger (Stanford) is efficiently applied to find out the product features. The words representing product attributes in review sentences can be unigram, bigram or trigram.

<table>
<thead>
<tr>
<th>Table 4.2 Properties of dataset BD2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model I</strong></td>
</tr>
<tr>
<td><strong>Product</strong></td>
</tr>
<tr>
<td><strong>No.of .reviews</strong></td>
</tr>
<tr>
<td><strong>Positive reviews</strong></td>
</tr>
<tr>
<td><strong>Negative reviews</strong></td>
</tr>
<tr>
<td><strong>Feature</strong></td>
</tr>
<tr>
<td><strong>No.of.attributes</strong></td>
</tr>
<tr>
<td><strong>Attributes type</strong></td>
</tr>
<tr>
<td><strong>Class attribute</strong></td>
</tr>
<tr>
<td><strong>Vector space</strong></td>
</tr>
</tbody>
</table>
Product attributes selected as features for classification model have a higher impact on the orientation of the text than the other words in the same text. In terms of these, the descriptions of review dataset model BD2 to be used in the experiment are given in Table 4.2.

4.4.2. Imbalanced dataset

Although the corpus is artificially balanced, it is preferred using the reviews just as they were extracted, as it is interesting in measuring the accuracy of this approach when applied to real user-generated texts. In this work, two datasets are used to investigate the performance of the approaches used on imbalanced sentiment classification. The first one is a widely used public dataset collected by Hu.M. et al., (2004). For this experiment, the camera domain of Hu’s dataset (IBD1) is selected which contains 135 negative samples (randomly selected from original negative samples) and 375 positive samples. A second dataset IBD2 is created by extracting sentences from Amazon website of two different types of camera. These datasets are different in their sizes as well as in the proportion between the two classes, so as to give several domains for the resampling method.

The polarity dataset IBD1 used is a set of product review sentences which were labeled as positive, negative or neutral. The review sentences from the publicly available customer review dataset (http://www.cs.uic.edu/~liub/FBS/FBS.html) are used. This dataset contains annotated customer reviews of 5 products. Of those five products, reviews of digital camera are selected. From camera reviews, 139 are negative, 371 are positive and 478 are neutral reviews. Outliers are performed as suggested in Briand.L.C. et al., (2000). Ten sentences are identified as outliers and are not considered for further processing (Appendix B). Thus the review dataset is of size 500 (365 positive and
135 negative reviews). For this binary classification problem, 365 positive and 135 negative (500 reviews) reviews are only considered. For each of the positive and negative review sentences the product attributes annotated in the review sentences are collected. Unique product features are grouped, which results in a final list of product attributes (features) of size 115. Among 115 product attributes 96 are unigram attributes and 19 are bigrams and trigram combinations. In terms of these, the descriptions of review dataset IBD1 to be used in the experiment is given in Table 4.3.

Table 4.3 Properties of dataset IBD1

<table>
<thead>
<tr>
<th></th>
<th>Model I</th>
<th>Model II</th>
<th>Model III</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
<td>Camera</td>
<td>Camera</td>
<td>Camera</td>
</tr>
<tr>
<td><strong>No.of.reviews</strong></td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td><strong>Positive reviews</strong></td>
<td>365</td>
<td>365</td>
<td>365</td>
</tr>
<tr>
<td><strong>Negative reviews</strong></td>
<td>135</td>
<td>135</td>
<td>135</td>
</tr>
<tr>
<td><strong>Feature</strong></td>
<td>unigrams</td>
<td>unigrams and bigrams</td>
<td>unigrams,bigrams and trigrams</td>
</tr>
<tr>
<td><strong>No.of.attributes</strong></td>
<td>96 attributes , 1 class label (sentiment)</td>
<td>108 attributes , 1 class label (sentiment)</td>
<td>115 attributes , 1 class label (sentiment)</td>
</tr>
<tr>
<td><strong>Attributes type</strong></td>
<td>integer</td>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td><strong>Class attribute</strong></td>
<td>binomial</td>
<td>binomial</td>
<td>binomial</td>
</tr>
<tr>
<td><strong>Vector space</strong></td>
<td>500 x 97</td>
<td>500 x 109</td>
<td>500 x 116</td>
</tr>
<tr>
<td><strong>Imbalance ratio</strong></td>
<td>3:1 (approx)</td>
<td>3:1 (approx)</td>
<td>3:1 (approx)</td>
</tr>
</tbody>
</table>
The dataset IBD2 used contains product review sentences which were labelled as positive, negative or neutral. The review sentences are collected from the Amazon reviews website. This dataset contains reviews of two different digital cameras (Canon G3 and Nikon Coolpix 4300). There are 1800 annotated reviews and the data is presented in plain text format. For this binary classification problem, only 900 positive reviews and 125 negative reviews are considered. The product attributes discussed in the review sentences are collected for each of the positive and negative review sentences as mentioned for dataset BD2.

Table 4.4 Properties of dataset IBD2

<table>
<thead>
<tr>
<th></th>
<th>Model I</th>
<th>Model II</th>
<th>Model III</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
<td>Camera</td>
<td>Camera</td>
<td>Camera</td>
</tr>
<tr>
<td><strong>No. of reviews</strong></td>
<td>1025</td>
<td>1025</td>
<td>1025</td>
</tr>
<tr>
<td><strong>Positive reviews</strong></td>
<td>900</td>
<td>900</td>
<td>900</td>
</tr>
<tr>
<td><strong>Negative reviews</strong></td>
<td>125</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td><strong>Feature</strong></td>
<td>unigrams</td>
<td>unigrams, bigrams</td>
<td>unigrams, bigrams and trigrams</td>
</tr>
<tr>
<td><strong>No. of attributes</strong></td>
<td>245 attributes, 1 class label (sentiment)</td>
<td>352 attributes, 1 class label (sentiment)</td>
<td>400 attributes, 1 class label (sentiment)</td>
</tr>
<tr>
<td><strong>Attributes type</strong></td>
<td>integer</td>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td><strong>Class attribute</strong></td>
<td>binomial</td>
<td>binomial</td>
<td>binomial</td>
</tr>
<tr>
<td><strong>Vector space</strong></td>
<td>1025 x 246</td>
<td>1025 x 353</td>
<td>1025 x 401</td>
</tr>
<tr>
<td><strong>Imbalance ratio</strong></td>
<td>9:1 (approx)</td>
<td>9:1 (approx)</td>
<td>9:1 (approx)</td>
</tr>
</tbody>
</table>

Unique unigram product features are grouped, which results in a final list of product attributes (features) of size 245. Unique bigram and
trigram product features are grouped, which results in a final list of product attributes (features) of size 107 and 48 respectively. In terms of these, the descriptions of review dataset IBD2 is given in Table 4.4.

4.5. Tools

This section gives a brief description about various tools used in this analysis. The tools used are as follows,

✓ **RapidMiner**: RapidMiner is an open source data mining and knowledge discovery tool written in java, incorporating most well known mining algorithms for classification, clustering and regression; it also contains plugins for specialized tasks such as text mining and analysis of streamed data. RapidMiner is a graphical user interface based tool, but mining tasks can also be scripted for batch mode processing. In addition to its numerous choice of operators, RapidMiner also includes the data mining library from the Weka toolkit. RapidMiner supports various plugins. The plugin for text mining is used. The text mining plugin contains tasks specially designed to assist in the preparation of text documents for mining tasks, such as tokenization, stop word removal and stemming.

✓ **Weka**: Weka toolkit is one of the best known and most widely available machine learning packages. It supports a wide range of supervised learning techniques. Weka comes with both a graphical user interface and a command line interface, as well as a Java API. One of the most interesting features of weka is its flexibility for text classification.

✓ **Knime**: Knime is a visual data mining tool with easy to use and intuitive data flow user interface and powerful data mining elements. The Knime text processing feature enables to read,
process, mine and visualize textual data in a convenient way. It provides functionality for natural language processing (NLP), text mining and information retrieval. One key behind the success of knime is its inherent modular workflow approach, which documents and stores the analysis process in the order it was conceived and implemented, while ensuring that intermediate results are always available.

☑ **Stanford parser:** A natural language parser is a program that works out the grammatical structure of sentences, for instance, which groups of words go together and which words are the subjects or object of a verb. This package is a java implementation of probabilistic natural language parsers. The parser provides stanford dependencies output as well as phrase structure trees.

☑ **Web crawler:** The task of the crawler is to keep on getting information from the internet into the database of the search engine. It literally crawls over the internet from page to page, link by link and downloads all the information to the database. Web crawler is a program that traverses the hyperlinks structure of web pages. Web crawler visits each web page by following the hyperlinks on previous page it visits. While it is crawling web pages to extract hyperlinks, a web crawler also saves each page it has visited.