CHAPTER 7

The problem of imbalanced data sets

A general goal of classifier learning is to learn a model on the basis of training data which makes as few errors as possible when classifying previously unseen test data. Many factors can affect the success of a classifier: the specific ‘bias’ of the classifier, the selection and the size of the data set, the choice of algorithm parameters, the selection and representation of information sources and the possible interaction between all these factors. In the previous chapters, we experimentally showed for the eager learner RIPPER and the lazy learner TIMBL that the performance differences due to algorithm parameter optimization, feature selection, and the interaction between both easily overwhelm the performance differences between both algorithms in their default representation. We showed how we improved their performance by optimizing their algorithmic settings and by selecting the most informative information sources.

In this chapter, our focus shifts, away from the feature handling level and the algorithmic level, to the sample selection level. We investigate whether performance is hindered by the imbalanced class distribution in our data sets and we explore different strategies to cope with this skewedness. In Section 7.1, we introduce the problem of learning from imbalanced data. In the two following sections, we discuss different strategies for dealing with skewed class distributions. In Section 7.2, we discuss several proposals made in the machine learning
literature for dealing with skewed data. In Section 7.3, we narrow our scope to the problem of class imbalances when learning coreference resolution. In the remainder of the chapter, we focus on our experiments for handling the class imbalances in the MUC-6, MUC-7 and KNACK-2002 data sets.

7.1 Learning from imbalanced data sets

The problem of learning from data sets with an unbalanced class distribution occurs when the number of examples in one class is significantly greater than the number of examples in the other class. In other words, in an unbalanced data set the majority class is represented by a large portion of all the instances, whereas the other class, the minority class, has only a small part of all instances. For a multi-class classification task, it is also possible to have several minority classes.

One of the major reasons for studying the effect that class distribution can have on classifier learner, is that we are confronted with unbalanced data sets in many real-world applications. For all these applications it is crucial to know whether class imbalances affect learning and if so, how. Example applications include vision (Maloof 2003), credit card fraud detection (Chan and Stolfo 1998), the detection of oil spills in satellite radar images (Kubat, Holte and Matwin 1998) and language applications, such as text categorization (Lewis and Gale 1994), part-of-speech tagging, semantic class tagging and concept extraction (Cardie and Howe 1997). These studies and many others show that imbalanced data sets may result in poor performance of standard classification algorithms (e.g. decision tree learners, nearest neighbour and naive bayes methods). Some algorithms will find an acceptable trade-off between the false positive and true positive rates. Other algorithms often generate classifiers that maximize the overall classification accuracy, while completely ignoring the minority class.

The common approach in detection tasks such as credit card fraud detection, the detection of oil spills in satellite radar images, and NLP tasks such as text categorization and also coreference resolution is to define these tasks as two-class classification problems. This implies that the classifier labels instances as being “fraudulent” or “non-fraudulent” (credit card fraud detection), “oil spilling” or “non oil spilling” (oil spills in satellite radar images), “coreferential” or “non-coreferential” (coreference resolution), etc. But in all these tasks, we are only interested in the detection of fraud, oil spills or coreferential relations. From that perspective, we might consider these tasks as one-class classification (see for example Manevitz and Yousef (2001) and Tax (2001) for a discussion of one-class classification) problems.
The motivation to consider coreference resolution as a one-class classification task is that we are only given examples of one class, namely of coreferential relations between NPs and we wish to determine whether a pair of NPs is coref-erential. But the negative “non-coreferential” class can be anything else, which makes the choice of negative data for this task arbitrary, as shown in Section 7.3. The number of possible candidates for building negative instances is so huge, that finding interesting instances, or instances near the positive instances, is challenging. To train a standard two-class classification algorithm will probably result in a high number of false negative detections. However, considering the coreference resolution task as a one-class classification task requires the use of an entirely different classification strategy (such as one-class support vector machines (Tax 2001)) as the one being used in this thesis.

Since the difference between one-class and two-class classification is beyond the scope of this work, we will restrict the discussion to the task of coreference resolution as a two-class classification task. The positive class (“coreferential”) will always correspond to the minority class and the negative class (“non-coreferential”) to the majority class.

7.2 Machine learning research on imbalanced data sets

A central question in the discussion on data sets with an imbalanced class distribution is in what proportion the classes should be represented in the training data. One can argue that the natural class distribution should be used for training, even if it is highly imbalanced, since a model can then be built which fits a similar imbalanced class distribution in the test set. Others believe that the training set should contain an increased number of minority class examples. In the machine learning literature, there have been several proposals (see Japkowicz and Stephen (2002)) for adjusting the number of majority class and minority class examples. Methods include resizing training data sets or sampling, adjusting misclassification costs, learning from the minority class, adjusting the weights of the examples, etc. We will now discuss these approaches in more detail. In Subsection 7.2.1, we discuss two commonly used methods to adapt machine learning algorithms to imbalanced classes: under-sampling and over-sampling. We continue with a discussion on cost-sensitive classifiers. Subsection 7.2.3 covers the approaches in which the examples are weighted in an effort to bias the performance to the minority class.
7.2.1 Sampling

Two sampling methods are commonly used to adapt machine learning algorithms to imbalanced classes: under-sampling or down-sampling and over-sampling or up-sampling. In case of under-sampling, examples from the majority class are removed. Examples removed can be randomly selected, or near miss examples, or examples that are far from the minority class examples. In case of over-sampling, examples from the minority class are duplicated. Both sampling techniques can also be combined. Examples of this type of sampling research include Kubat, Holte and Matwin (1997), Chawla, Bowyer, Hall and Kegelmeyer (2002), Drummond and Holte (2003) and Zhang and Mani (2003).

The primary motivation of the use of sampling for skewed data sets is to improve classifier performance. But under-sampling can also be used as a means to reduce training set size.

Especially the sensitivity of the C4.5 decision tree learner to skewed data sets and the effect of under-sampling and over-sampling on its performance has been intensively studied. Drummond and Holte (2003), Domingos (1999), Weiss (2003), Japkowicz and Stephen (2002) and Joshi, Kumar and Agarwal (2001) all investigate the effect of class distribution on the C4.5 classifier. The conclusions are similar: under-sampling leads to better results, whereas over-sampling produces little or no change in performance. None of the approaches, however, consistently outperforms the other and it is also difficult to determine a specific under-sampling or over-sampling rate which consistently leads to the best results. We will come to similar conclusions for our experiments.

Both over-sampling and under-sampling have known drawbacks. The major drawback from under-sampling is that it disregards possibly useful information. This can be countered by more intelligent under-sampling strategies such as those proposed by Kubat et al. (1997) and Chan and Stolfo (1998). Kubat et al. (1997), for example, consider majority examples which are close to the minority class examples as noise and discard these examples. Chan and Stolfo (1998) choose for an under-sampling approach without any loss of information. In a preliminary experiment, they determine the best class distribution for learning and then generate different data sets with this class distribution. This is accomplished by randomly dividing the majority class instances. Each of these data sets then contains all minority class instances and one part of the majority class instances. The sum of the majority class examples in all these data sets is the complete set of majority class examples in the training set. They then learn a classifier on these different data sets and integrate all these classifiers (meta-learning) by learning from their classification behaviour.

---

1Except for Japkowicz and Stephen (2002) who come to the opposite conclusion.
One of the problems with over-sampling is that it increases the size of the training set and the time to build a classifier. Furthermore, in case of decision tree learning, the decision region for the minority class becomes very specific through the replication of the minority class and this causes new splits in the decision tree, which can lead to overfitting. It is possible that classification rules are induced which cover one single copied minority class example. An over-sampling strategy which aims to make the decision region of the minority class more general and hence aims to counter overfitting has been proposed by Chawla et al. (2002). They form new minority class examples by interpolating between minority class examples that lie close together.

Although most of the sampling research focuses on decision tree learning, this does not imply that other learning techniques are immune to the class distribution of the training data. Also support vector machines (Raskutti and Kowalczyk 2003), kNN methods (Zhang and Mani 2003) (see also 7.2.3), neural networks (Zhang, Mani, Lawrence, Burns, Back, Tsoi and Giles 1998), etc. have been shown to be sensitive to the class imbalances in the data set.

### 7.2.2 Adjusting misclassification costs

Another approach for coping with skewed data sets is the use of cost-sensitive classifiers. If we consider the following cost matrix, it is obvious that the main objective of a classifier is to minimize the false positive and false negative rates.

<table>
<thead>
<tr>
<th>Actual negative</th>
<th>Actual positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predict negative</td>
<td>true negative</td>
</tr>
<tr>
<td>Predict positive</td>
<td>false positive</td>
</tr>
</tbody>
</table>

If the number of negative and positive instances is highly unbalanced, this will typically lead to a classifier which has a low error rate for the majority class and a high error rate for the minority class. Cost-sensitive classifiers (Pazzani, Merz, Murphy, Ali, Hume and Brunk 1994, Domingos 1999, Kubat et al. 1998, Fan, Stolfo, Zhang and Chan 1999, Ting 2000, Joshi et al. 2001) have been developed to handle this problem by trying to reduce the cost of misclassified examples, instead of classification error. Cost-sensitive classifiers may be used for unbalanced data sets by setting a high cost to the misclassifications of a minority class example.

The MetaCost algorithm of Domingos (1999) is an example of such a cost-sensitive classifier approach. It uses a variant of bagging (Breiman 1996), which
makes bootstrap replicates from the training set by taking samples with replacement from the training set. In MetaCost, multiple bootstrap samples are made from the training set and classifiers are trained on each of these samples. The class’s probability for each example is estimated by the fraction of votes that it receives from the ensemble. The training examples are then relabeled with the estimated optimal class and a classifier is reapplied to this relabeled data set. Domingos (1999) compared his approach with under-sampling and over-sampling and showed that the MetaCost approach is superior to both.

Other cost-sensitive algorithms are the boosting\(^2\) algorithms CSB1, CSB2 (Ting 2000) and AdaCost (Fan et al. 1999). In order to better handle data sets with rare cases, these algorithms take into account different costs of making false positive predictions versus false negative predictions. So in contrast to AdaBoost (Freund and Schapire 1996), in which a same weight is given to false and true positives and false and true negatives, the CSB1, CSB2 and AdaCost algorithms update the weights of all four types of examples differently. All three algorithms assign higher weights to the false negatives and thus focus on a recall improvement.

### 7.2.3 Weighting of examples

The ‘weighting of examples’ approach has been proposed from within the case-based learning framework (Cardie and Howe 1997, Howe and Cardie 1997). It involves the creation of specific weight vectors in order to improve minority class predictions. The commonly used feature weighting approach is the use of so-called task-based feature weights (as for example also used in TIMBL), in which the feature weights are calculated for the whole instance base.

In order to increase the performance on the minority class, however, Cardie and Howe (1997) and Howe and Cardie (1997) propose the use of class-specific and even test-case-specific weights. The class-specific weights are calculated per class whereas the test-case-specific weights are calculated for each single instance. The creation of class-specific weights (Howe and Cardie 1997) is as follows: the weights for a particular class on a given feature are based on the distribution of feature values for the instances in that class and the distribution of feature values for the instances in the other class(es). Highly dissimilar distributions imply that the feature can be considered useful and will have a high weight.

---

\(^2\)Boosting is a machine learning method in which learning starts with a base learning algorithm (e.g. C4.5 (Quinlan 1996)), which is invoked many times. Initially, all weights over the original training set are set equally. But on each boosting round, these weights are adjusted: the weights of incorrectly classified examples are increased, whereas the weights of the correctly classified examples are decreased. Through these weight adjustments, the classifier is forced to focus on the hard training examples.
During testing, all training instances with the same class value are assigned the weight associated with that particular class value. Howe and Cardie (1997) describe different techniques for the creation of class-specific weights. These techniques represent different levels of locality in feature weighting, ranging from the calculation of feature weight vectors across all classes to get a single global weight vector to a fine-grained locality by assigning different weights for each individual feature value. They show that the use of class-specific weights globally leads to better classification accuracy.

Cardie and Howe (1997) describe the use of test-case-specific weights, which are determined on the basis of decision trees. The weight vector for a given test case is calculated as follows: (1) Present the test case to the decision tree and note the path that is taken through the tree, (2) Omit the features that do not appear along this path, (3) calculate the weights for the features that appear along the path by using path-specific information gain values, (4) use this weight vector in the learning algorithm to determine the class of the test case. Cardie and Howe (1997) show that example weighting leads to a significant increase of the recall.

In the experiments described in the remainder of this chapter in which we investigate the effect of class distribution on classifier performance, we decided not to introduce additional learning techniques, such as decision tree learning or different boosting techniques in this discussion on methodology. Instead, we chose for a straightforward resampling procedure and a variation of the internal loss ratio parameter in RIPPER.

7.3 Imbalanced data sets in coreference resolution

As already frequently mentioned before, coreference resolution data sets reveal large class imbalances: only a small part of the possible relations between noun phrases is coreferential (see for example Table 3.1). When trained on such imbalanced data sets, classifiers can exhibit a good performance on the majority class instances but a high error rate on the minority class instances. Always assigning the “non coreferential” class will lead to a highly ‘accurate’ classifier, which cannot find any coreferential chain in a text.
7.3.1 Instance selection in the machine learning of coreference resolution literature

In the machine learning of coreference resolution literature, this problem of class imbalances has to our knowledge not yet been thoroughly investigated. However, the different methodologies for corpus construction show that at least the problem of instance selection has been acknowledged. Soon et al. (2001), for example, only create positive training instances between anaphors and their immediately preceding antecedent. The NPs occurring between the two members of each antecedent-anaphor pair are used for the creation of the negative training examples. Imposing these restrictions on corpus construction still leads to high imbalances: in their MUC-6 and MUC-7 training data, only 6.5% and 4.4%, respectively, of the instances is positive. Strube et al. (2002) use the same methodology as Soon et al. (2001) for the creation of positive and negative instances, but they also first apply a number of filters, which reduce up to 50% of the negative instances. These filters are all linguistically motivated, e.g. discard an antecedent-anaphor pair (i) if the anaphor is an indefinite NP, (ii) if one entity is embedded into the other, e.g. if the potential anaphor is the head of the potential antecedent NP, (iii) if either pronominal entity has a value other than third person singular or plural in its agreement feature. But Strube et al. (2002) do not report results of experiments before and after application of these linguistic filters. And Yang et al. (2003) use the following filtering algorithm to reduce the number of instances in the training set: (i) add the NPs in the current and previous two sentences and remove the NPs that disagree in number, gender and person in case of pronominal anaphors, (ii) add all the non-pronominal antecedents to the initial candidate set in case of non-pronominal anaphors. But also here, no comparative results are provided of experiments with and without instance selection.

Ng and Cardie (2002a) propose both negative sample selection (the reduction of the number of negative instances) and positive sample selection (the reduction of the number of positive instances), both under-sampling strategies aiming to create a better coreference resolution system. Ng and Cardie (2002a) use a technique for negative instance selection, similar to that proposed in Soon et al. (2001) and they create negative instances for the NPs occurring between an anaphor and its farthest antecedent. Furthermore, they try to avoid the inclusion of hard training instances. Given the observation that one antecedent is sufficient to resolve an anaphor, they present a corpus-based method for the selection of easy positive instances, which is inspired by the example selection algorithm introduced in Harabagiu et al. (2001). The assumption is that the easiest types of coreference relationships to resolve are the ones that occur with high frequencies in the training data. Harabagiu et al. (2001) mine by hand three sets of coreference rules for covering positive instances from the training data by
finding the coreference knowledge satisfied by the largest number of anaphor-
antecedent pairs. The high confidence coreference rules, for example, look for
(i) repetitions of the same expression, (ii) appositions or arguments of the same
copulative verb, (iii) name alias recognitions, (iv) anaphors and antecedents
having the same head. Whenever the conditions for a rule are satisfied, an
antecedent for the anaphor is identified and all other pairs involving the same
anaphor can be filtered out. Ng and Cardie (2002a) write an automatic positive
sample selection algorithm that coarsely mimics the Harabagiu et al. (2001)
algorithm by finding a confident antecedent for each anaphor. They show that
system performance improves dramatically with positive sample selection. The
application of both negative and positive sample selection leads to even better
performance. But they mention a drawback in case of negative sample selection: it
improves recall but damages precision.

All these approaches concentrate on instance selection, on a reduction of the
training material and they aim to produce better performing classifiers through
the application of linguistically motivated filters on the training data before ap-
plication of the classifier. Through the application of these linguistic filters, part
of the problem to be solved, viz. coreference resolution, is solved beforehand.

Our instance selection approach differs from these approaches on the following
points:

• We investigate whether both learning approaches we experiment with are
  sensitive to class imbalances in the training data. None of the above
described approaches investigates the effect of class imbalances on classifier
performance.

• In case of sensitivity to class imbalances, we investigate whether classifier
  performance can be improved through a rebalancing of the data set. This
  rebalancing is done without any a priori linguistic knowledge about the
  task to be solved.

7.3.2 Investigating the effect of skewedness on classifier
performance

In Section 3.1.2, we described the selection of positive and negative instances
for the training data. For the construction of these instances, we did not impose
any limitations on the construction of the instance base. For English, we did
not take into account any restrictions with respect to the maximum distance
between a given anaphor and its antecedent. Due to the presence of documents
exceeding 100 sentences in the KNACK-2002 data, negative instances were only
made for the NPs in a range of 20 sentences preceding the candidate anaphor. For both languages, we did not apply any linguistic filters (such as gender and number agreement between both nominal constituents) on the construction of the positive and negative instances. The main “restriction” was that, since we investigate anaphoric and not cataphoric relations, we only looked back in the text for the construction of the instances. The instances were made as follows:

- **Positive instances** were made by combining each anaphor with each preceding element in the coreference chain.
- The **negative instances** were built (i) by combining each anaphor with each preceding NP which was not part of any coreference chain and (ii) by combining each anaphor with each preceding NP which was part of another coreference chain.

Table 7.1: $F_{\beta=1}$ and recall results on the cross-validation data in relation to the share of minority class examples in the data sets.

<table>
<thead>
<tr>
<th></th>
<th>MUC-6</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>% of min.</td>
<td>Timbl</td>
<td>Ripper</td>
<td>Timbl</td>
<td>Ripper</td>
</tr>
<tr>
<td>All</td>
<td>6.6</td>
<td>56.15</td>
<td>62.59</td>
<td>55.50</td>
<td>49.65</td>
</tr>
<tr>
<td>Pronouns</td>
<td>7.0</td>
<td>31.97</td>
<td>28.70</td>
<td>27.42</td>
<td>19.44</td>
</tr>
<tr>
<td>Proper nouns</td>
<td>7.9</td>
<td>65.37</td>
<td>71.04</td>
<td>67.53</td>
<td>61.60</td>
</tr>
<tr>
<td>Common nouns</td>
<td>5.0</td>
<td>53.62</td>
<td>65.44</td>
<td>53.53</td>
<td>55.55</td>
</tr>
<tr>
<td>MUC-7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>5.80</td>
<td>48.68</td>
<td>49.36</td>
<td>46.09</td>
<td>36.21</td>
</tr>
<tr>
<td>Pronouns</td>
<td>8.54</td>
<td>39.25</td>
<td>32.86</td>
<td>36.60</td>
<td>22.70</td>
</tr>
<tr>
<td>Proper nouns</td>
<td>6.00</td>
<td>59.49</td>
<td>64.83</td>
<td>56.87</td>
<td>52.56</td>
</tr>
<tr>
<td>Common nouns</td>
<td>4.24</td>
<td>41.03</td>
<td>49.24</td>
<td>39.17</td>
<td>36.76</td>
</tr>
<tr>
<td>KNACK-2002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>6.31</td>
<td>46.78</td>
<td>46.49</td>
<td>44.93</td>
<td>34.92</td>
</tr>
<tr>
<td>Pronouns</td>
<td>8.58</td>
<td>47.31</td>
<td>50.57</td>
<td>44.81</td>
<td>43.14</td>
</tr>
<tr>
<td>Proper nouns</td>
<td>6.18</td>
<td>58.11</td>
<td>60.21</td>
<td>54.04</td>
<td>49.49</td>
</tr>
<tr>
<td>Common nouns</td>
<td>3.92</td>
<td>30.51</td>
<td>36.52</td>
<td>30.37</td>
<td>25.92</td>
</tr>
</tbody>
</table>

As shown in Table 7.1, this approach leads to an instance base with a highly skewed class distribution for all three data sets. In the MUC-6 training data, for example, 159,815 instances out of 171,081 are negative and merely 11,266 (6.6% of the total) are positives. Furthermore, the number of instances in both training and test set is large comparing to the number of references (in MUC-6 1644 and
7.4 Balancing the data set

In order to investigate the effect of class distribution on classifier performance, it is necessary to compare the performance of the classifier on training sets with a variety of class distributions. One possible approach to create this variety of class distributions is to decrease the number of instances in the majority class. We investigated the effect of random down-sampling and down-sampling of the true negatives for both TIMBL and RIPPER. For RIPPER, we also changed the ratio false negatives and false positives in order to improve recall. We did not perform any up-sampling experiments, since creating multiple copies from one instance can only guide the choice of classification in memory-based learning if there is a conflict among nearest neighbours. Furthermore, as already discussed
earlier, up-sampling can lead to rules overfitting the training data. For example, when a certain instance is copied ten times, the rule learner might quite possibly form a rule to cover that one instance.

### 7.4.1 Random

In order to reduce the number of negative training instances, we experimented with two randomized down-sampling techniques. In a first experiment, we gradually down-sampled the majority class at random. We started off with no down-sampling at all and then gradually downsized the number of negative instances in slices of 10% until there was an equal number of positive and negative training instances.

With respect to the accuracy results, we can conclude for both classifiers that the overall classification accuracy decreases with a decreasing rate of negative instances. The precision, recall and $F_{\beta=1}$ results from the down-sampling experiments for both TIMBL and RIPPER are plotted in Figure 7.1, Figure 7.2 and Figure 7.3. At the X-axis, the different down-sampling levels are listed, ranging from 1 (no down-sampling at all) to 0 (an equal number of positive and negative instances). The plots for both learning methods show that a decreasing rate of negative instances is beneficial for the recall or the classification accuracy on the minority class instances. The plots also reveal that down-sampling is harmful for precision. By reducing the number of negative instances, it becomes more likely for a test instance to be classified as positive. This implies that more negative instances will be classified as being positive (the false positives). But the plots also reveal more subtle tendencies.

For TIMBL, the $F_{\beta=1}$ values for the “Pronouns” data set in the whole down-sampling process remain rather constant (i.e. there are no significant differences compared to the default scores) or they even significantly increase. On MUC-6, for example, TIMBL obtains a default $F_{\beta=1}$ score of 31.97% and decreasing the number of negative instances with 40% leads to a top $F_{\beta=1}$ score of 34.42%. For MUC-7, TIMBL obtains $F_{\beta=1}$ scores ranging between 38.21% and 41.41%. Only the last down-sampling step in which the training set contains an equal number of positive and negative instances leads to a larger drop in $F_{\beta=1}$ scores (34.21%). For KNACK-2002, the default $F_{\beta=1}$ score of 47.31% is raised to 50.02% at a down-sampling level of 0.5. For the other three data sets (“All”, “Proper nouns” and “Common nouns”), however, down-sampling does not lead to a significant increase of $F_{\beta=1}$ values.

With respect to the RIPPER $F_{\beta=1}$ values, we can conclude the following. For MUC-6, the $F_{\beta=1}$ values for the “Pronouns” data set are all significantly better
7.4 Balancing the data set

than the default result during the whole down-sampling process. RIPPER obtains a default $F_{\beta=1}$ score of 28.70% and decreasing the number of negative instances with 90% leads to a top $F_{\beta=1}$ score of 37.04%. A similar tendency can be observed for MUC-7: a default $F_{\beta=1}$ score of 32.86% and a top $F_{\beta=1}$ score of 45.00% when down-sampling with 80%. For KNACK-2002, the default $F_{\beta=1}$ score of 50.57% is raised to 63.00% at a down-sampling level of 0.5. For the other three data sets, the $F_{\beta=1}$ values only significantly deteriorate at higher down-sampling levels. For MUC-6, for example, the $F_{\beta=1}$ values for the “Proper nouns”, “Common nouns” and “All” data sets only significantly deteriorate when down-sampling with 30%, 70% and 50% respectively. For the KNACK-2002 data, down-sampling even leads to a significant improvement on the default $F_{\beta=1}$ score: from a default 46.49% to a top score 58.84% at level 0.3 (“All”), from 36.52% to 42.84% at level 0.3 (“Common nouns”) and from 60.21% to 64.60% at level 0.3 (“Proper nouns”).

For all these random down-sampling experiments we can conclude that TIMBL and RIPPER behave differently. In Table 7.1, we showed that RIPPER is more sensitive to the skewedness of the classes. A comparison of results in the Figures 7.1, 7.2 and 7.3 shows that down-sampling can be beneficial for the RIPPER results. Furthermore, down-sampling only starts being harmful at a high down-sampling level. TIMBL has shown this tendency only on the “Pronouns” data set. All these observed tendencies hold for both English data sets and the Dutch data set.

7.4.2 Exploiting the confusion matrix

In the previous sampling experiments, all negative instances (both the true negatives and false positives) qualified for down-sampling. In a new set of experiments, we also experimented with down-sampling of the true negatives. This implies that the falsely classified negative instances, the false positives, were kept in the training set, since they were considered harder to classify. The true negatives were determined in a leave-one-out experiment on the different training folds and then down-sampled. However, due to the highly skewed class distribution, the level of true negatives is very high, e.g. Timbl falsely classifies merely 3% of the negative instances in the KNACK-2002 “All” data set. This implies that these down-sampling experiments reveal the same tendencies as the random down-sampling experiments.

As already touched upon in Section 4.3, RIPPER incorporates a loss ratio (Lewis and Gale 1994) (see also Section 4.3) parameter which allows the user to specify the relative cost of two types of errors: false positives and false negatives. It thus controls the relative weight of precision versus recall. In its default version,
Figure 7.1: Cross-validation results in terms of precision, recall and $F_{\beta=1}$ after application of TIMBL and RIPPER on the MUC-6 data with a randomly downsampled majority class. The test partitions keep their initial class distribution.
Figure 7.2: Cross-validation results in terms of precision, recall and $F_{\beta=1}$ after application of TIMBL and RIPPER on the MUC-7 data with a randomly down-sampled majority class. The test partitions keep their initial class distribution.
Figure 7.3: Cross-validation results in terms of precision, recall and $F_{\beta=1}$ after application of Timbl and Ripper on the KNACK-2002 data with a randomly down-sampled majority class. The test partitions keep their initial class distribution.

<table>
<thead>
<tr>
<th>Level of downsampling</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>0.8</td>
<td>80</td>
</tr>
<tr>
<td>0.6</td>
<td>60</td>
</tr>
<tr>
<td>0.4</td>
<td>40</td>
</tr>
<tr>
<td>0.2</td>
<td>20</td>
</tr>
<tr>
<td>0.1</td>
<td>10</td>
</tr>
</tbody>
</table>

- **Timbl Pronouns**
- **Ripper Pronouns**
- **Proper Nouns**
- **Common Nouns**
- **All**
7.4 Balancing the data set

RIPPER uses a loss ratio of 1, which indicates that the two errors have equal costs. A loss ratio greater than 1 indicates that false positive errors (where a negative instance is classified positive) are more costly than false negative errors (where a positive instance is classified negative). Setting the loss ratio above 1 can be used in combination with the up-sampling of the positive minority class in order to counterbalance the overrepresentation of the positive instances. But this is not what we need. In all previous experiments with RIPPER we could observe high precision scores and rather low recall scores. Therefore, we decided to focus the loss ratio on the recall. For our experiments, we varied the loss ratio in RIPPER from 1 (default) to 0.05 (see for example also Chawla et al. (2002) for similar experiments). The motivation for this reduction of the loss ratio is double: (i) improve on recall and (ii) build a less restrictive set of rules for the minority class.

We can conclude from these experiments that, as also observed in the down-sampling experiments, a change of loss ratio is generally at the cost of overall classification accuracy. The cross-validation precision, recall and $F_\beta=1$ results of these experiments are displayed in Figure 7.4, Figure 7.5 and Figure 7.6. Similar tendencies as for the down-sampling experiments can be observed: the focus on recall is harmful for precision. With respect to the $F_\beta=1$ values, we can conclude that the $F_\beta=1$ values for the “Pronouns” can significantly increase when decreasing the loss ratio value. On MUC-6, for example, RIPPER obtains a default $F_\beta=1$ score of 28.70% and decreasing the loss ratio value to 0.09 leads to a top $F_\beta=1$ score of 38.80%. On MUC-7, the $F_\beta=1$ score is raised up to 13% when changing the loss ratio parameter. With respect to the MUC-6 and MUC-7 $F_\beta=1$ values for the “Proper nouns”, “Common nouns” and “All” data sets we can observe a small increase of performance. As shown in Figure 7.6 and Table 7.2, a change of the loss ratio parameter leads to a large performance increase for the different KNACK-2002 data sets. Furthermore, for three out of the four data sets, the default class distribution returns the lowest $F_\beta=1$ score.

<table>
<thead>
<tr>
<th></th>
<th>default</th>
<th>high</th>
<th>low</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>46.49</td>
<td>60.33 (loss ratio: 0.2)</td>
<td>46.49</td>
</tr>
<tr>
<td>Pronouns</td>
<td>50.57</td>
<td>63.49 (loss ratio: 0.4)</td>
<td>50.57</td>
</tr>
<tr>
<td>Proper nouns</td>
<td>60.21</td>
<td>63.69 (loss ratio: 0.06)</td>
<td>58.61</td>
</tr>
<tr>
<td>Common nouns</td>
<td>36.52</td>
<td>42.68 (loss ratio: 0.07)</td>
<td>36.52</td>
</tr>
</tbody>
</table>

The general conclusion from these experiments with loss ratio reduction is that decreasing the loss ratio leads to better recall at the cost of precision. For both
the English and Dutch data sets overall $P_{\beta=1}$ increases can be observed. Furthermore, loss ratio reduction also leads to a less restrictive set of rules for the minority class, reflected in an increasing recall. With respect to the specific loss ratio values, we conclude that no particular value leads to the best performance over all data sets. This confirms our findings in the parameter optimization experiments (Chapter 5), which also revealed that the optimal parameter settings of an algorithm for a given task have to be determined experimentally for each new data set.

7.5 Summary and discussion

In this chapter we focused on the problem of imbalanced data sets. In Section 7.2, we discussed several proposals made in the machine learning literature for dealing with skewed data sets and we continued with a discussion on class imbalances when learning coreference resolution. In the remainder of the chapter, we presented results for the MUC-6, MUC-7 and KNACK-2002 data sets. We first compared the share of minority class examples in the data sets with the percentage of the total test errors that can be attributed to misclassified minority class test examples. There, we could observe a large number of false negatives or a large error rate for the examples from the minority class. These results confirm earlier results on both machine learning and NLP data sets, e.g. by Weiss (2003) or Cardie and Howe (1997). Furthermore, we showed that although RIPPER performs better on the data set as a whole, it exhibits a poorer performance on the recall for the minority class than TIMBL does.

In order to investigate the effect of class distribution on the performance of TIMBL and RIPPER, we created a variety of class distributions through the use of down-sampling and by changing the loss ratio parameter in RIPPER. For the down-sampling experiments we could conclude for the two learning methods that a decreasing rate of negative instances is beneficial for recall. The same conclusion could be drawn in the experiments in which the loss ratio parameter was varied for RIPPER. These conclusions confirm earlier findings of Chan and Stolfo (1998), Weiss (2003) and others. Another general conclusion is that both down-sampling and a change of the loss ratio parameter below 1 is harmful for precision. This implies that more false positives will be produced.

However, both learning approaches behave quite differently in case of skewedness of the classes and they also react differently to a change in class distribution. TIMBL, which performs better on the minority class than RIPPER in case of a largely imbalanced class distribution, mainly suffers from a rebalancing of the data set. In contrast, the RIPPER results are sensitive to a change of class
Figure 7.4: Cross-validation results in terms of precision, recall and $F_{\beta=1}$ when changing the RIPPER loss ratio parameter on the MUC-6 data.

Figure 7.5: Cross-validation results in terms of precision, recall and $F_{\beta=1}$ when changing the RIPPER loss ratio parameter on the MUC-7 data.

Figure 7.6: Cross-validation results in terms of precision, recall and $F_{\beta=1}$ when changing the RIPPER loss ratio parameter on the KNACK-2002 data.
distribution or loss ratio. We believe that this different behaviour of TIMBL and RIPPER can be explained by the nature of both learning approaches (see also Daelemans et al. (1999) for a discussion on this topic). In a lazy learning approach, all examples are stored in memory and no attempt is made to simplify the model by eliminating low frequency events. In an eager learning approach such as RIPPER, however, possibly interesting information from the training data is either thrown away by pruning or made inaccessible by the eager construction of the model. This type of approach makes abstraction from low-frequency events. Applied to our data sets, this implies that RIPPER prunes away possibly interesting low-frequency positive data. A decrease of the number of negative instances counters this pruning.

This chapter also concludes our discussion on different factors which can influence a (comparative) machine learning experiment. Throughout the previous chapters we experimentally showed that apart from algorithm bias, many other factors such as data set selection, the selection of information sources and algorithm parameters and also their interaction potentially play a role in the outcome of a machine learning experiment. We showed that changing any of the architectural variables can have great effects of the performance of a learning method, making questionable many conclusions in the literature based on default settings of algorithms or on partial optimization only.

In the algorithm-comparing experiments using all information sources and default parameter settings, we could observe some clear tendencies with respect to the precision and recall scores. We saw that the precision scores for TIMBL were up to about 30\% lower than the ones for RIPPER, which implies that TIMBL falsely classifies more instances as being coreferential. Furthermore, with respect to the recall scores, the opposite tendency could be observed: TIMBL generally obtained a higher recall than RIPPER. In the feature selection experiments, we observed the large effect feature selection can have on classifier performance. Especially TIMBL showed a big sensitivity to a good feature subset. In the parameter optimization experiments we observed that the performance differences within one learning method are much larger than the method-comparing performance differences, which was also confirmed in the experiments exploring the interaction between feature selection and parameter optimization. Furthermore, with respect to the selected features and parameter settings, we observed that no particular parameter setting and no particular feature selection is optimal. This implies that the parameter settings which are optimal using all features are not necessarily optimal when performing feature selection. We also showed that the features considered to be optimal for TIMBL can be different than the ones optimal for RIPPER. In the experiments varying the class distribution of the training data, we showed that this was primarily beneficial for RIPPER. Once again, we showed that no particular class distribution nor loss ratio value was
optimal for all data sets. Therefore, this resampling should also be subject to optimization. This additional optimization step was already incorporated in the joint feature selection and parameter optimization experiments reported in the previous chapter, where we also varied the loss ratio parameter for RIPPER. These experiments revealed that a loss ratio value below one was selected in 97% of the optimal individuals found over all experiments. An illustration of these selected loss ratio values is given in Figure 7.7.

Figure 7.7: Selected loss ratio values in the individuals found to be optimal after GA optimization.