Over-Sampling Methods for Polarity Classification of Imbalanced Microblog Texts

Kiyoaki Shirai **Yunmin Xiang**
Japan Advanced Institute of Science and Technology
1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan
{kshirai,s1710250}@jaist.ac.jp

Abstract

Polarity classification is the task of classifying sentiments or opinions shown in a given text into positive, negative or neutral. Most previous studies have developed and evaluated their methods on balanced datasets. However, in Twitter, the polarity distribution is highly imbalanced since most tweets are neutral. This paper proposes novel methods to train an accurate classifier from imbalanced data for polarity classification of tweets. They are kinds of synthesizing over-sampling methods that newly generate minority samples to balance the polarity distribution. In our approach, since sentiment words are effective features, minority samples are synthesized more from a sample that includes sentiment words. Furthermore, the number of synthesized minority samples is carefully determined by measuring the performance on development data. According to our experiments using an imbalanced dataset of tweets, the F1-measure of the polarity classification is much improved when our proposed methods are combined with two existing over-sampling methods SMOTE and ADASYN.

1 Introduction

Sentiment analysis is process of analyzing the emotions or opinions in texts. Polarity classification is one of the fundamental techniques in sentiment analysis. It is the task of classifying a given text into polarity classes, such as positive, negative or neutral. In particular, the polarity classification of texts in a microblog such as Twitter received much research attention. Since users actively express their opinions on social media, microblog texts are valuable resources for sentiment analysis and opinion mining.

Supervised machine learning is a major approach for polarity classification. In past studies, polarity classifiers have usually been trained and evaluated on balanced datasets, i.e. those in which the number of samples of each polarity class is almost the same. However, in real social media, the distribution of the polarity of texts is actually imbalanced, since there are many more neutral samples than positive or negative ones. Machine learning usually performs poorly on imbalanced data, since a classifier tends to judge a sample of a minority class as belonging to a majority class. On the other hand, the detection of minority samples (i.e. positive and negative samples) is important because they provide useful information in sentiment analysis.

This paper proposes several methods to train an accurate classifier to determine the polarity of texts in Twitter from an imbalanced dataset. Our methods are an extension of existing over-sampling methods. Over-sampling is a technique to increase the number of minority samples artificially to make a balanced dataset. In our approach, sentiment words are taken into account in the generation of the minority samples, since sentiment words are obviously useful features for polarity classification.

2 Related work

2.1 Sentiment analysis

Supervised machine learning has been widely applied to sentiment analysis and polarity classification. Pang et al. (2002) used Naive Bayes, Maximum Entropy and Support Vector Machine (SVM) to classify the polarity of a given movie review. In their experiment on the classification of positive or negative classes, SVM outperformed Naive Bayes and Maximum Entropy. The accuracy was relatively high, 82.9%. However, the classifiers were trained and evaluated on a balanced dataset consisting of 700 positive and 700 negative movie reviews.

Early work on polarity classification of three classes (positive, negative or neutral) has been done by Koppel and Schler (2006). They claimed that a precise three-class classifier could not be trained from positive and negative samples, since the neutral samples would not simply be located at somewhere near a boundary between the positive and negative classes. Thus neutral samples were necessary for training. A stack of three kinds of binary classifiers (positive vs negative, positive vs neutral, and negative vs neutral) achieved 74.1% accuracy on a TV domain and 85.5% on a shopping domain. The datasets of both domains were completely balanced.

Recently, deep neural networks have been introduced to polarity classification. CharSCNN (Character to Sentence Convolutional Neural Network) employed two convolutional neural layers (Dos Santos and Gatti, 2014). One was to obtain abstract representations of the words from character embedding, which enabled the model to use character-level features. The other was to obtain abstract representations of sentences from word vectors, which were concatenations of word embedding including word-level features and the output of the first network including character-level features. Finally, two feed forward networks were used to obtain a score for each polarity class. CharSCNN achieved 85.7% accuracy on the Stanford Sentiment Treebank (Socher et al., 2013) and 86.4% on the Stanford Twitter Sentiment corpus (Go et al., 2009). The distributions of polarity classes in these two datasets were balanced.

Similar to the above papers, most of the past studies of polarity classification evaluated methods using balanced data that consists of almost equal numbers of samples for all polarity classes. However, as we will report in Section 3, in social media, the number of neutral texts is much greater than the numbers of positive and negative texts. Thus the distribution of the polarity is higly skewed. The datasets of SemEval 2016 task 4 (Nakov et al., 2016) and SemEval 2017 task 4 (Rosenthal et al., 2017) are widely used for research on sentiment analysis in Twitter. However, neutral samples are not overwhelmingly dominant in these datasets. Considering a real application to a microblog, this paper focuses on polarity classification in imbalanced data where there are many more neutral samples than the others.

2.2 Over-sampling methods

Over-sampling and under-sampling are commonly used to improve the performance of supervised machine learning on an imbalanced dataset. The core idea of these methods is to increase the number of minority samples or to decrease the number of majority samples so that the number of samples of each class becomes balanced. This research has focused on the over-sampling approach.

Chawla et al. (2002) proposed Synthetic Minority Oversampling TEchnique (SMOTE), an over-sampling method that synthesizes new minority samples from existing ones. Figure 1 shows its pseudocode. Let us suppose that an imbalanced dataset consists of a large number of majority samples and a small number of minority samples, and each sample is represented as a feature vector in a vector space. For each minority sample $\vec{x_i}$, its k nearest neighbours of minority samples are chosen at line 7. Then, one minority sample $\vec{n}$ is chosen randomly at line 9. A random point somewhere on the line between $\vec{x_i}$ and $\vec{n}$ is chosen as indicated in lines 10-12. It is the newly synthesized minority sample $\overrightarrow{syn}$ and is added to the dataset at line 13. bal is a balance parameter to control the number of synthesized samples. It is defined as the proportion of the minority samples to the majority samples in the new (over-sampled) dataset. For example, $bal = 1$ means that the new training data contains equal numbers of majority and minority samples, whereas $bal = 0.5$ means that number of minority samples becomes 50% of the number of majority samples. g_{all} at line 3 denotes the total number of samples to be synthesized, while g at line 4 denotes the number of samples to be synthesized from one minority sample. The synthesis of the minority samples from $\vec{x_i}$ is repeated g times, as indicated at line 8.

ADAptive SYNthetic sampling (ADASYN) (He et al., 2008) is another over-sampling method. The key idea is to synthesize more samples from minority samples that are located near a borderline between minority and majority classes. It can enable a trained classifier to more easily discriminate between minority and majority samples. Figure 2 shows the pseudocode of ADASYN. At line 6, $r[i]$ is the ratio of the majority samples in the k nearest

```
Input: $X$(original training data), $bal$(balance parameter), $k$(number of nearest neighbours)
Output: $X'$ (new training data)
1:  $S_{min} \leftarrow$ a set of minority samples in $X$
2:  $S_{maj} \leftarrow$ a set of majority samples in $X$
3:  $g_{all} \leftarrow |S_{maj}| \times bal - |S_{min}|$
4:  $g \leftarrow int(g_{all}/|S_{min}|)$
5:  $Syn \leftarrow \phi$
    /* a set of synthesized minority samples */
6:  for each $\vec{x_i} \in S_{min}$ do
7:      $K_i \leftarrow k$ nearest neighbours of $\vec{x_i}$ in $S_{min}$
8:      for $j = 1$ to $g$ do
9:          $\vec{n} \leftarrow$ a sample randomly chosen from $K_i$
10:         $\overrightarrow{diff} \leftarrow \vec{n} - \vec{x_i}$
11:         $gap \leftarrow$ random value between $[0, 1]$
12:         $\overrightarrow{syn} \leftarrow \vec{x_i} + gap \times \overrightarrow{diff}$
13:         $Syn \leftarrow Syn \cup \{\overrightarrow{syn}\}$
14:     end for
15: end for
16: return $X' = X \cup Syn$
```

Figure 1: Pseudocode of SMOTE

neighbours of a minority sample $\vec{x_i}$. It evaluates how likely $\vec{x_i}$ is to be close to the borderline. It is normalized in line 9 to calculate the density distribution $\hat{r}$, then $g[i]$, the number of samples to be synthesized from $\vec{x_i}$, is calculated in line 10. The following procedures are almost the same as SMOTE. An important difference between SMOTE and ADASYN is that equal numbers of synthetic samples are generated for each minority sample in SMOTE, whereas in ADASYN, more samples are generated from minority samples near a borderline.

This paper extends SMOTE and ADASYN to improve the accuracy of polarity classification of imbalanced data.

3 Survey of the polarity distribution in Twitter

A preliminary survey was conducted to briefly investigate the distribution of the polarity of texts in Twitter. It is expected that neutral tweets are the overwhelming majority in the real world and thus polarity distribution is highly imbalanced.

Tweets are collected by searching a keyword with Twitter API. Eight topics including electronic products, celebrities, movies and so on, are chosen as keywords, which are shown in Table 1. One hundred tweets are retrieved for each topic. Thus 800 tweets are retrieved in total. Note that advertisement tweets are excluded. These tweets are manually classified in terms of their polarity toward a topic.

```
Input: $X$(original training data), $bal$(balance parameter), $k$(number of nearest neighbours)
Output: $X'$ (new training data)
1:  $S_{min} \leftarrow$ a set of minority samples in $X$
2:  $S_{maj} \leftarrow$ a set of majority samples in $X$
3:  $g_{all} \leftarrow |S_{maj}| \times bal - |S_{min}|$
4:  for each $\vec{x_i} \in S_{min}$ do
5:      $NN_i \leftarrow k$ nearest neighbours of $\vec{x_i}$ in $X$
6:      $r[i] \leftarrow \frac{|NN_i \cap S_{maj}|}{k}$
7:  end for
8:  for each $\vec{x_i} \in S_{min}$ do
9:      $\hat{r}[i] \leftarrow \frac{r[i]}{\sum_i r[i]}$
10:     $g[i] \leftarrow int(\hat{r}[i] \times g_{all})$
11: end for
12: $Syn \leftarrow \phi$
13: for each $\vec{x_i} \in S_{min}$ do
14:     $K_i \leftarrow k$ nearest neighbours of $\vec{x_i}$ in $S_{min}$
15:     for $j = 1$ to $g[i]$ do
16:         $\vec{n} \leftarrow$ a sample randomly chosen from $K_i$
17:         $\overrightarrow{diff} \leftarrow \vec{n} - \vec{x_i}$
18:         $gap \leftarrow$ random value between $[0, 1]$
19:         $\overrightarrow{syn} \leftarrow \vec{x_i} + gap \times \overrightarrow{diff}$
20:         $Syn \leftarrow Syn \cup \{\overrightarrow{syn}\}$
21:     end for
22: end for
23: return $X' = X \cup Syn$
```

Figure 2: Pseudocode of ADASYN

Table 1 shows the number of positive, negative and neutral tweets about 8 topics as well as the total numbers. It shows that the ratio of neutral tweets is quite high, 86%. It is found that users usually write a fact or statement about a topic and do not express their emotions or opinions.

4 Proposed method

This section first explains how to train a classifier for polarity classification, then describes our proposed over-sampling methods.

Table 1: Distribution of classes for each topic

	pos.	neg.	neu.
iPhone X	20	12	68
HUAWEI	10	7	83
SAMSUNG	3	1	96
Morgan Freeman	3	2	95
Gabe Newell	5	3	92
Star Wars	6	3	91
Harry Potter	7	4	89
Monster Hunter : World	11	2	87
Total	65	34	701

4.1 Polarity classifier

Each tweet is represented as a feature vector as follows. After preprocessing, including conversion from upper to lower case, removal of stopwords, and replacement of URL and @+user_id with special tokens, the vector of a tweet is obtained by Equation (1).

$$\text{tweet vector} = \frac{1}{\sum_i w_i^2} \sum_i w_i \times \vec{v_i} \tag{1}$$

where $\vec{v_i}$ is the vector representation of the ith word, and w_i is the weight of TF-IDF for the ith word. Word vectors are word embedding, which was pretrained by a skip-gram model (Mikolov et al., 2013) from the English Wikipedia corpus. The dimension of the word embedding was set to 250.

The SVM was trained using sklearn[1]. The square of the hinge loss function was chosen as the loss function for the training. The penalty parameter C of the error term was set to 0.5. The kernel of the SVM is the linear kernel.

4.2 Quantity Control Over-Sampling (QCO)

In a synthetic over-sampling strategy, minority samples are newly synthesized. The quality of such synthesized samples is questionable, since they are not real samples at all. The more samples are synthesized to balance the distribution of the classes, the more unreliable samples are likely to be added in the dataset. The generation of too many samples may cause a decline in the classification performance. However, in the papers of SMOTE (Chawla et al., 2002) and ADASYN (He et al., 2008), the number of synthesized samples was given by the user, and there was no discussion how to determine it appropriately.

The number of synthesized samples can be empirically determined. More specifically, the balance parameter bal can be optimized using the development data.[2] First, we prepare the training data and development data. We also prepare a set of balance parameters B. Next, for each balance parameter $bal_i \in B$, we train a classifier from the training data balanced by SMOTE or ADASYN, and apply it for the development data. Finally, the optimized balance parameter is chosen so that the F1-measure on the development data becomes the highest.

In this paper, we call this method Quantity Control Over-Sampling (QCO). It is not a novel method as it is common to optimize parameters using the development data. However, we will demonstrate that the optimization of the number of synthesized samples is crucial in the experiment in Section 5.

4.3 Over-sampling methods considering sentiment words

4.3.1 SMOTE with Sentiment Oriented Over-Sampling (SOO)

We propose an over-sampling method that takes sentiment words into account in the synthesis of minority samples. It is well known that sentiment words are important and effective features for polarity classification. We expect that a superior classifier could be trained from a dataset including many sentiment words. The key idea of our method is to generate more samples including sentiment words.

We introduce a sentiment weight parameter, sen. It is defined as the weight of the samples including sentiment words. Minority samples from a minority sample including a sentiment word are synthesized sen times more often than a sample that does not include a sentiment word. Note that sen is supposed to be greater than 1.

Figures 3 and 4 show the pseudocode. $y =$

[1]https://scikit-learn.org/

[2]The number of synthesized samples is chosen in different ways in SMOTE (Chawla et al., 2002) and ADASYN (He et al., 2008). In the pseudocodes in Figure 1 and 2, SMOTE and ADASYN are slightly modified so that the number of synthesized samples is defined in the same way, i.e. controlled by bal. Note that these pseudocodes are completely equivalent to the original algorithms.

```
Input: $X$(original training data), $bal$(balance parameter), $k$(number of nearest neighbours)
Output: $X'$ (new training data)
 1: $S_{min} \leftarrow$ a set of minority samples in $X$
 2: $S_{maj} \leftarrow$ a set of minority samples in $X$
 3: $g_{all} \leftarrow |S_{maj}| \times bal - |S_{min}|$
 4: $\boxed{y \leftarrow \text{SYNTHESISWEIGHTS}(S_{min})}$
 5: for each $\vec{x_i} \in S_{min}$ do
 6:     $\hat{y}[i] \leftarrow \frac{y[i]}{\sum_i y[i]}$
 7:     $g[i] \leftarrow int(\hat{y}[i] \times g_{all})$
 8: end for
 9: $Syn \leftarrow \phi$
10: for each $\vec{x_i} \in S_{min}$ do
11:     $K_i \leftarrow k$ nearest neighbours of $\vec{x_i}$ in $S_{min}$
12:     for $j = 1$ to $g[i]$ do
13:         $\vec{n} \leftarrow$ a sample randomly chosen from $K_i$
14:         $\overrightarrow{diff} \leftarrow \vec{n} - \vec{x_i}$
15:         $gap \leftarrow$ random value between $[0, 1]$
16:         $\overrightarrow{syn} \leftarrow \vec{x_i} + gap \times \overrightarrow{diff}$
17:         $Syn \leftarrow Syn \cup \{\overrightarrow{syn}\}$
18:     end for
19: end for
20: return $X' = X \cup Syn$
```

Figure 3: Pseudocode of our proposed over-sampling algorithm

$\{y[0], \cdots, y[n]\}$ is a list of synthesis weights. Each $y[i]$ controls how many minority samples are newly synthesized from the ith minority sample $\vec{x_i}$. SYNTHESISWEIGHTS is a function to determine y, which is described in Figure 4. $y[i]$ is set to be sen if t_i (a tweet of the ith minority sample) contains a sentiment word, otherwise 1. SentiWordNet (Baccianella et al., 2010) is used to judge whether a word in a tweet is a sentiment word or not. Note that the algorithm of Figure 3 is the same as SMOTE when $y[i]$ is set to 1 for all i. In SMOTE, all $\vec{x_i}$ receive the same number of synthesized samples, whereas in our method, sen times as many samples are generated from $\vec{x_i}$ with a sentiment word.

After y is determined at line 4 in Figure 3, the procedures are almost the same as ADASYN. $y[i]$ is normalized to be $\hat{y}[i]$ at line 6, similar to $\hat{r}[i]$ in ADASYN in Figure 2. Then $g[i]$ is calculated in line 7. The minority samples are generated $g[i]$ times

```
 1: function SYNTHESISWEIGHTS($S_{min}$)
 2:     for each $\vec{x_i} \in S_{min}$ do
 3:         if $t_i$ includes a sentiment word then
 4:             $y[i] \leftarrow sen$
 5:         else
 6:             $y[i] \leftarrow 1$
 7:         end if
 8:     end for
 9:     return $y$
10: end function
```

Figure 4: SYNTHESISWEIGHTS of SMOTE+SOO

from $\vec{x_i}$ in lines 12-18.

Finally, the sentiment weight parameter sen is optimized on the development data. Hereafter, 'Sentiment Oriented Over-Sampling' (SOO) refers to the proposed technique that synthesizes more samples from samples including sentiment words. SMOTE combined with SOO is referred to as SMOTE+SOO.

4.3.2 ADASYN with Sentiment Oriented Over-Sampling (SOO)

SOO can be combined with ADASYN. The pseudocode of ADASYN+SOO is presented in Figure 3 where the function SYNTHESISWEIGHTS is defined as in Figure 5. The only difference between this algorithm and the original ADASYN is lines 5-9 in Figure 5: $y[i]$ is always set to $r[i]$ in ADASYN, but in ADASYN+SOO, it is multiplied by sen if t_i contains a sentiment word. Thus ADASYN+SOO is able to not only generate more synthetic samples near a borderline but also create more samples including sentiment words. Similar to SMOTE+SOO, the parameter sen is optimized using the development data.

4.3.3 Sentiment Intensity Oriented Over-Sampling (SIOO)

Another extension of ADASYN made in the present paper is ADASYN with Sentiment Intensity Oriented Over-Sampling (SIOO). In SOO, although more samples are generated from a sample including a sentiment word, the number of synthesized samples is the same for all samples with a sentiment word. However, it is supposed that samples showing intense emotion heavily contribute to polarity classification. In SIOO, more samples are generated from

```
1: function SynthesisWeights($S_{min}$)
2:     for each $\vec{x_i} \in S_{min}$ do
3:         $NN_i \leftarrow k$ nearest neighbours of $x_i$ in $X$
4:         $r[i] \leftarrow \frac{|NN_i \cap S_{maj}|}{k}$
5:         if $t_i$ includes a sentiment word then
6:             $y[i] \leftarrow sen \times r[i]$
7:         else
8:             $y[i] \leftarrow r[i]$
9:         end if
10:    end for
11:    return $y$
12: end function
```

Figure 5: SynthesisWeights of ADASYN+SOO

a minority sample that expresses strong sentiment.

The sentiment intensity score $s[i]$ of the ith tweet t_i is defined by

$$s[i] = \frac{\sum_{w_i \in SW(t_i)} score(w_i)}{|SW(t_i)|} \quad (2)$$

$$score(w_i) = max(s_{pos}(w_i), s_{neg}(w_i)) + 1 \quad (3)$$

where $SW(t_i)$ is the set of sentiment words in t_i. $score(w_i)$ is the sentiment score of w_i defined by Equation (3), and s_{pos} and s_{neg} are the averages of the positive and negative scores of w_i in SentiWordNet.[3] Thus $s[i]$ evaluates the intensity of the sentiment of the ith sample regardless of its polarity orientation.

The pseudocode of ADASYN+SIOO is presented in Figure 3, where the function SynthesisWeights is defined as in Figure 6. The only difference between SOO and SIOO is that the sentiment weight parameter sen is replaced with $s[i]$ in SIOO, as indicated in lines 6 and 7. Note that $s[i]$ should be greater than one to give importance to samples with polarity words in the generation of the minority samples. That is the reason why we add 1 in Equation (3). Note that the positive and negative scores in SentiWordNet are between 0 and 1, so $s[i]$ can be less than 1 if we do not add 1 in $score(w_i)$.

Another advantage of SIOO is its lesser computational cost. SOO requires trial and error in its training and applying classifiers for the optimization of

[3]In SentiWordNet, positive and negative scores are given for each sense of a word. Therefore, polysemous words have several positive and negative scores. We take their average.

```
1: function SynthesisWeights($S_{min}$)
2:     for each $\vec{x_i} \in S_{min}$ do
3:         $NN_i \leftarrow k$ nearest neighbours of $x_i$ in $X$
4:         $r[i] \leftarrow \frac{|NN_i \cap S_{maj}|}{k}$
5:         if $t_i$ includes a sentiment word then
6:             $s[i] = \frac{\sum_{w_i \in SW(t_i)} score(w_i)}{|SW(t_i)|}$
7:             $y[i] \leftarrow s[i] \times r[i]$
8:         else
9:             $y[i] \leftarrow r[i]$
10:        end if
11:    end for
12:    return $y$
13: end function
```

Figure 6: SynthesisWeights of ADASYN+SIOO

the parameter sen, but SIOO can easily calculate $s[i]$ using the sentiment lexicon.

5 Evaluation

5.1 Experimental setting

The SemEval 2017 task 4 (Rosenthal et al., 2017) dataset was used for the experiment. It is a collection of tweets about several topics with manually annotated polarity labels. The polarity of each tweet is represented by 5-point labels from 1 (very negative) to 5 (very positive). In this experiment, we defined three polarity classes by converting 1 and 2 to "negative", 3 to "neutral" and 4 and 5 to "positive". Our preliminary survey showed that in Twitter, 86% of tweets are neutral. To make the distribution of the polarity labels of the dataset closer to the actual distribution, we added neutral tweets to the dataset by the following procedures.

1. Retrieve tweets via Twitter API by searching the keywords of the topics in the SemEval 2017 dataset.

2. Classify the retrieved tweets by AYLIEN[4], which is a web toolkit for polarity classification. Only tweets classified as neutral are kept, the other are discarded.

3. Add neutral tweets to the dataset until the proportion of neutral tweets reaches 86%.

[4]https://aylien.com/text-api/sentiment-analysis/

Table 2: Statistics of training, development and test data

	positive	negative	neutral
training	5,748	1,637	46,534
development	1,642	468	13,298
test	822	234	6,649
total	8,212 (11%)	2,339 (3%)	66,491 (86%)

Table 3: Optimized balance parameter *bal*

	positive	negative
SMOTE+QCO	0.6	0.4
ADASYN+QCO	0.6	0.5

Finally, the dataset was divided into 70% training data, 20% development data, and 10% test data. The numbers of samples in the three classes are shown in Table 2.

Two binary classifiers have been evaluated. The first classifier judges whether a tweet is positive or not. To train and evaluate it, the negative and neutral tweets were merged into "not positive" tweets as majority samples, while the positive tweets remained as minority samples. The second classifier judges whether a tweet is negative or not. Similarly, the positive and neutral tweets were merged into "not negative" tweets as majority samples. Precision, recall, and F1-measure have been used as the evaluation criteria. Throughout these experiments, the parameter of the number of nearest neighbour k was set to 7.

5.2 Results of QCO

The balance parameter bal was changed from 0.2 to 1 in steps of 0.1. Figure 7 shows the F1-measure of the positive and negative classification on the development data by SMOTE+QCO and ADASYN+QCO with different bal. It is confirmed that the F1-measure drastically changes with bal. This indicates that the optimization of the number of synthesized samples is important. The optimized parameters bal are summarized in Table 3.

Next, the performance of the methods with QCO as well as the baselines was measured on the test data. Table 4 presents the precision (P), recall (R), and F1-measure (F) of the positive and negative classification on the test data. SMOTE and ADASYN are the original algorithm with $bal = 1$. The baseline is a classifier trained from the original imbalanced dataset. All over-sampling methods outperform the baseline. Comparing SMOTE+QCO or ADASYN+QCO with methods without QCO, QCO greatly improves the precision with a little deterioration of the recall. The F1-measures of SMOTE+QCO and ADASYN+QCO are better than those of SMOTE and ADASYN in both positive and negative classification, respectively.

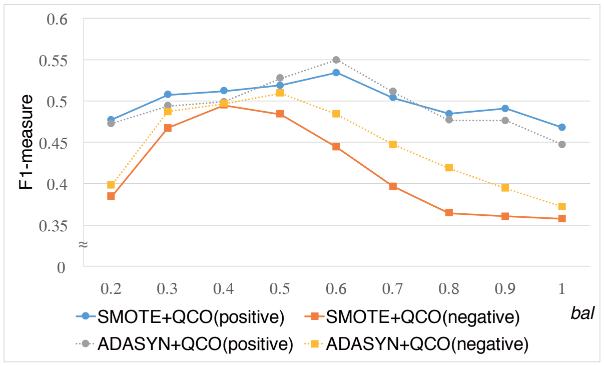

Figure 7: F1-measure of SMOTE+QCO and ADASYN+QCO on development data

Table 4: Results of methods with QCO on test data

	positive			negative		
	P	R	F	P	R	F
Baseline	.3064	.7037	.4271	.1886	.7012	.2973
SMOTE	.3437	.7155	.4652	.2529	.7170	.3739
SMOTE+QCO	.4297	.7098	.5279	.3763	.6917	.4874
ADASYN	.3136	.7335	.4374	.2788	.7315	.4037
ADASYN+QCO	.4461	.6778	.5378	.4003	.7016	.5144

5.3 Evaluation of SOO and SIOO

We conducted experiments to evaluate the proposed over-sampling methods considering sentiment words. Thoughout these experiments, bal was set to the optimized value in Table 3. As for SOO, the sentiment weight parameter sen was changed from 1 to 7. Figure 8 shows the F1-measures of positive and negative classification on the development data by SMOTE+SOO and ADASYN+SOO with different sen. When sen is greater than 1, i.e. more minority samples are synthesized from a sample including sentiment words, the F1-measure is improved. However, the performance deteriorates when sen becomes too large. The optimized parameters sen

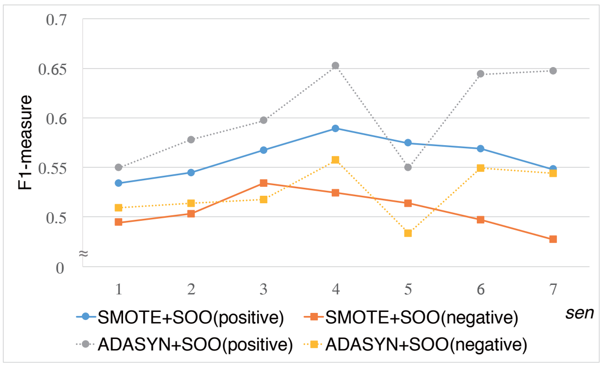

Figure 8: F1-measure of SMOTE+SOO and ADASYN+SOO on development data

Table 5: Optimized sentiment weight parameter *sen*

	positive	negative
SMOTE+SOO	4	3
ADASYN+SOO	4	4

Table 6: Results of methods with SOO and SIOO on test data

	positive			negative		
	P	*R*	*F*	*P*	*R*	*F*
SMOTE+QCO	.4297	.7098	.5279	.3763	.6917	.4874
SMOTE+SOO	.4752	.7421	.5794	.4466	.6851	.5407
ADASYN+QCO	.4461	.6778	.5378	.4003	.7016	.5144
ADASYN+SOO	.6037	.7047	.6503	.4314	.7559	.5493
ADASYN+SIOO	.5676	.7255	.6369	.4096	.7443	.5284

are summarized in Table 5.

Table 6 presents the results of methods with SOO on the test data. Comparing SMOTE+SOO or ADASYN+SOO with SMOTE+QCO or ADASYN+QCO, the former outperforms the latter for all criteria except for the recall of negative classification by SMOTE. This proves that our method, which puts more weight on samples including sentiment words in the synthesis of the minority samples, is effective at improving the performance of the polarity classification.

We now present the evaluation of ADASYN+SIOO. The results of ADASYN+SIOO are shown in the last row of Table 6. Contrary to our expectations, ADASYN+SIOO is worse than ADASYN+SOO. This indicates that to determine the sentiment weight parameter by the sentiment scores of the words in a tweet is not as good as empirical optimization using the development data.

In the experiments as a whole, ADASYN mostly outperforms SMOTE. ADASYN+SOO achieves the best F1-measure, 0.65 and 0.55 for the positive and negative classification, respectively.

6 Conclusion

The contributions of this paper are summarized in what follows. First, the effectivness of the Quantity Control Over-Sampling (QCO) was empirically investigated. It was found that QCO could improve the F1-measure drastically. It indicates that the optimization of the number of synthesized minority samples is quite important. QCO is general and applicable to any classification task on imbalanced data. Second, we proposed the Sentiment Oriented Over-Sampling (SOO) method that synthesizes more minority samples containing sentiment words. SOO is a method only for polarity classification, but could be combined with any supervised machine learning algorithm. We also proposed the Sentiment Intensity Oriented Over-Sampling (SIOO) method that considers the intensity of the sentiment in the generation of the minority samples. The results of experiments showed that SOO could greatly improve the classification performance of SMOTE and ADASYN. On the other hand, the effectiveness of SIOO was not confirmed by the experiments in this paper.

In the future, in order to improve SIOO, the way to measure the intensity of the sentiment in tweets will be carefully investigated. For example, a combination of SOO and SIOO is worth assessing. The combination of SIOO with SMOTE (i.e. SMOTE+SIOO) should also be evaluated. In addition, since only the extended SemEval 2017 dataset was used in the experiments, our proposed methods should be applied to other microblog datasets for more precise evaluation. Another important line of future research is to extend our method to multiclass classification, since the current methods are only applicable to binary classification. This would enable us to classify a tweet into positive, negative, or neutral by a single system.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation*, pages 2200–2204.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Cicero Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical report, Stanford University.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328.

Moshe Koppel and Jonathan Schler. 2006. The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2):100–109.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3111–3119.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 1–18. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–86. Association for Computational Linguistics.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 502–518. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.