Differential Voting in Case Based Spam Filtering

Deepak P, Delip Rao, Deepak Khemani

Department of Computer Science and Engineering
Indian Institute of Technology Madras, India
deepakswallet@gmail.com, delip@cse.iitm.ernet.in, khemani@iitm.ac.in

Abstract. Case-based reasoning (CBR) has been shown to be of considerable utility in a spam-filtering task. In the course of this study, we propose that the non-random skewed distribution of the cases in a case base is crucial, especially in the context of a classification task like spam filtering. In this paper, we propose approaches to improve the performance of a CBR spam filter by making use of the non-random nature of the case base. We associate each case in the case base with a voting power, which is essentially a function that incorporates the knowledge of the local neighborhood of the case. We show that the performance of the spam filter can be considerably improved by making use of such techniques that incorporate the voting powers.

1 Introduction

Spam or unsolicited email is a major concern to the industry and the end users. Finding a legitimate email in a deluge of spam mails can become a daunting task. Automated techniques for spam filtering have been used with considerable success to distinguish spam messages from the non-spam ones. One of the dangers of using spam filtering is to classify a legitimate email as spam. A pessimistic approach is usually adopted, i.e., when in doubt leave the message as legitimate. This however lowers the accuracy of such systems. Automatic spam filtering systems are continuously striving to improve accuracy. Case-based reasoning has been shown to be of considerable value in a spam-filtering task [2, 3]. It is interesting to note that the distribution of the spam cases in the case space is not uniform. Data mining techniques can be used to exploit this skewed distribution (hereafter referred to as patterns) for our advantage. We associate every case in the case base with a voting power, which incorporates the knowledge of the neighborhood of the case, which can then be made use of to classify a test case. We propose various algorithms for computing the voting power of a case, and show that many of them perform better in comparison with the traditional techniques. Further, the complexity of re-computing the voting powers when an addition or deletion occurs to the case base is linear in the number of cases in the case base.

Section 2 reviews the related work in the area. Section 3 lays down the motivation behind the work and justifies the motivation by some experiments. Section 4 describes the techniques that we propose (and the intuition behind them) and the techniques with which we compare our approaches. Section 5 lists the performance

measures used and Section 6 lists the experiments conducted, the results and their implications.

2 Related Work

The work on spam filtering (as a classification task) has warranted lot of investigation in the past using several classifiers with Support Vector Machines giving the best performance [9]. In this paper we describe a memory based approach to spam filtering. It has been shown [1] that memory based approaches for spam filtering work significantly better than well studied naïve Bayesian approaches. Further, they go on to say that it might probably due to the fact that there are many more types of messages rather than just spam and legitimate. For instance, the common classes of spam mails that the authors receive include mortgage mails, free university degree mails and so on and so forth. A more recent work [2] proposes new methods of feature selection based on spam and non-spam vocabularies and asserts that a CBR approach to spam filtering can effectively track and adapt to the changing behavior of spammers and legitimate mails (concept drift) and provides methods for the same. It uses the conventional and intuitive CBR approach of majority voting (the voters being the neighbors of the test case in the case base) to flag a message as either spam or legitimate. It may be noted that the tracking of concept drift can be easily incorporated in a CBR system, and that the incremental (online) model acquisition is a natural process in a CBR system. This may be contrasted with other conventional classification systems, where the model is built based on only the training data. Another work [3] incorporates various ideas specific to spam filtering. Firstly, it incorporates the notion that a legitimate message classified as spam is much costlier than a spam message labeled legitimate. Conventional spam filtering systems, either label a mail as spam (labeling systems), or send it to a special folder such as "bulk" or "spam" (redirection systems). Thus a legitimate mail is classified as spam may well escape the attention of the user in systems that employ the latter method. It has been estimated empirically estimated that a false positive (legitimate mail classified as spam) is 9 times costlier than a false negative in a labeling system, whereas the ratio is 999 for a redirection system. This error disparity is hereafter referred to in the paper as "Severity Disparity". Further, the same work presents a significant departure from the conventional CBR model in that it incorporates differential weighting of votes, viz., the closer a neighbor is, to the test case, the higher would be the value of it's vote (we call this technique Diff-CBR hereafter in this paper). It also incorporates differential weighting of features in the vector space. In this paper, we compare our techniques to the approaches used in the latter two papers.

3 Motivation and Justification

In the day-to-day life, an average web user comes across a wide range of spam and legitimate mails. Intuitively, most mails fall into more categories than just two classes as spam and legitimate. Many of the spam mails that the authors receive fall into categories such as "interest free home loans", "mortgage", "easy university degree"

and pornographic spam mails. We will later show that these spam mail fall into well defined clusters or patterns in the case-base of the CBR spam filter. Our hypothesis is that by making use of such patterns we should be able to improve the performance of a CBR spam filter considerably. It may well be re-emphasized here that the need for incremental model acquisition (to tackle concept drift) and the presence of more (logical) classes than the ones to which a case is to be classified as, makes CBR the natural choice for a spam filtering system. In the following subsections, we describe the dataset used and justify our assertion that spam mails form clusters experiments on the corpus.

3.1 Dataset Used

We choose to use the SpamBase Database (hereafter referred to as the corpus) compiled by George Forman of HP Labs. It is available through the University of California Irvine Machine Learning Repository[1]. SpamBase is a collection of 4601 pre-processed messages, each message represented as a labeled (as either spam or legitimate) vector of 57 selected features and contains 39.4% spam messages.

3.2 Justification

We now show the existence of clusters in the mail corpus and for suitable values of k we can get sufficiently "pure" clusters that we can use for our filtering purpose. Purity can be defined as the ratio of the sum of the cardinalities of the maximally represented class for each cluster (across all clusters) to the total number of cases clustered [8]. We applied the traditional k-means clustering algorithm with k = 15 with varying initial cluster centers on the corpus. Several values of k were used and we observed empirically that a large value of k yielded clusters of high purity. Note that our method is parameterized on k. It may be noted that fixing k at 15 is motivated by the need to choose a high value of k to illustrate the clustering in the corpus and is not related to any background knowledge of the corpus whatsoever. The presence of clusters would reveal a skewed distribution of messages among clusters and that is exactly what we ran into. Among the 15 clusters that we obtained, one of them had 42% of the corpus and only 5 clusters had more than 5% of the corpus. The percentages of the corpus in each of the 15 clusters are listed in table 1.

Table 1. Distribution in k-Means Clusters with k = 15

Percentages of the Corpus in Clusters														
11	1	20	3	6	4	3	1	1	2	42	1	1	0	6

Having justified our claim that there are clusters in the corpus, we now show that the clusters are pure enough (using the labels in the corpus). The overall purity of a clustering is the weighted average of the purities of the different clusters. Figure 1 shows the weighted average of the purity of the k-means clusters against k. As can be seen, the purity plot meanders around 0.78, which shows that the clusters are pure

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

enough. We now propose various techniques to make use of the skewed distribution in the CBR framework and compare it with the state-of-the-art techniques such as simple CBR and diff-CBR.

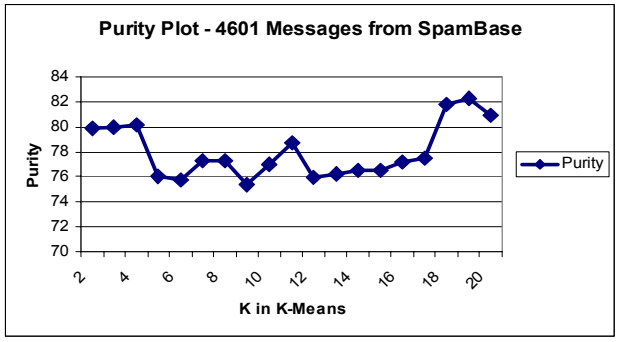

Figure 1. Total Purity of Clusters with varying k

3.3 Broad Methodology

In each of the techniques that we propose (in sections that follow), we use the concept of voting powers for a case in a case base. Given a case base, we can associate each case with a voting power depending on the cases in its vicinity and their labels. In order to compute voting power, we can either consider

- The k nearest neighbors (k-NN) of the case (and their labels) in the case base (k may be an input parameter)
- All cases which are no farther from the case than a given distance (which would be taken as an input parameter)

Given a test case and the ultimate aim of classifying it as either spam or legitimate, we need two functions:

- Confidence_Spam(Test_Case T, k-NN of the test case in the Case Base) which returns the confidence of the test case being spam and
- Confidence_Legitimate(Test_Case T, k-NN of the test case in the Case Base) which returns the confidence of the test case being legitimate

The intuitive algorithm for filtering is given by:

- Flag(Test_Case T) = Confidence_Spam(T,k-NN) > Confidence_Legitimate(T,k-NN) ? “Legitimate” : ”Spam”

In cases where the confidences are equal, we argue that we should “play safe” and label it as legitimate.

The Confidence functions that we propose make use of the voting powers of each of the k nearest neighbors, and optionally their distances from the test case. We use Confidence_X to denote two functions Confidence_Spam and

Confidence_Legitimate. Both the confidence functions use the same algorithm, except for the fact that one considers only spam elements in the k-NN and vice versa. We present a broad framework of the classifier algorithm.

```
Confidence_X(Test_Case T, k-NN of T in the Case Base)
{
    confidence = 0.0;
    for each C among the k-NN neighbors
    {
         if(label(C) == X)
         {
           add_pwr = Voting_Power(C);
optionally,
           add_pwr = add_pwr / distance(Test Case, C);
           confidence = confidence + add_pwr;
         }
     }
     return confidence;
}
```

Having introduced as many primitives as has been done, each algorithm can be specified by the voting power computation function and as to whether it involves the optional step in the algorithm. We call algorithms that include the optional step as *distance-weighting algorithms* for the sake of brevity hereafter in this paper.

4 Techniques for Spam Filtering using CBR

We present two techniques that have already appeared in literature followed by six techniques that we propose, to improve the performance of the CBR Spam Filter. All the descriptions use the primitives introduced in the preceding section. The intuition behind each technique that we propose has been detailed therein.

Simple CBR. A simple CBR [2] is a non-distance-weighting algorithm that uses a constant voting power function. Presenting it in another fashion, it takes the majority vote for classification.

Diff-CBR. This technique [3] which has been shown to be much better than Simple CBR is a distance-weighting algorithm that uses a constant voting power function.

C1 CBR. A case in the case base which is part of a spam cluster would have mostly spam cases among its k-NN (and vice versa). Such a case surrounded by spam cases being among the k-NN of a test case, intuitively gives a higher confidence that the test case is part of or in the near vicinity of the spam cluster (and vice versa). There is a host of clustering algorithms which rely on finding elements with a dense neighborhood and use them as seed points for identifying clusters [5,6,7]. C1 CBR is a variation of Simple-CBR which incorporates this notion in a straightforward manner. It is a non-distance-weighting algorithm in which the voting power of a case

is the number of cases with the same label as the case (in question) among its k-NN. In order that no case is assigned a voting power of zero, we include the case itself among the k-NN neighbors of the case to compute its voting power.

C2 CBR. This is a variation of Diff CBR along the same lines as C1 CBR and is a distance weighting algorithm, where the voting power function is exactly the same as in C1 CBR.

C3 CBR. C1 and C2 CBR are suspected to suffer from a serious drawback. Consider a dense spam cluster and a singleton point in an isolated area of the case base. All points in the spam cluster would get a voting power of k when k nearest neighbors are considered (unsurprisingly), and the singleton point would also get a voting power of k (surprisingly!) if all its k-NNs are spam (possibly, they are part of the dense cluster) although they are a considerable distance away compared to the former case. C3 CBR tries to rectify this problem by introducing an additional parameter, which we hereafter refer to as the radius. The voting power of a case in the case base is computed as 1 + (number of cases with the same label as the case in question, and which fall within a distance of radius from the case). The addition of 1, once again is to ensure that no case gets a zero voting power. This is a non-distance-weighting algorithm. We would like to clarify at this point that determining an optimal value for radius is a non-trivial task.

Better Voting Power Function (BVPF). A bit of thought is more than sufficient to come up with the insufficiencies of the C3 CBR voting power function. Consider a highly noisy space where a spam case has 50 spam cases and 50 legitimate cases within its radius. On the contrary, consider a pure space which has a sparse cluster where a spam case has just 5 spam neighbors within its radius. Intuitively, the second case deserves a better voting power whereas the C3CBR voting power function assigns a voting power of 51 to the former and 6 to the latter; a huge disparity indeed. Although the frequency of such hostile cases have to be studied, the disparity introduced by the C3CBR voting power function is so high that we can't let it go unattended. In this context, we choose to lay down some of the more intuitive desiderata for a voting power function.

Assume that the total number of cases in the radius of the case in question is t, let the number of cases with a matching label among them be m_1 and those with mismatching labels among the t cases be m_2. Firstly, the voting power function should be directly related to m_1. Secondly, the voting power function should be directly related to t. Thirdly, it should be inversely related to m_2. Given that t is $(m_1 + m_2)$, one might reasonably argue that any two of the above relations should be sufficient. Such a function is highly non trivial. We propose a voting power function, hereafter referred to as BVPF which we define as follows.

$$BVPF(t, m_1, m_2) = (m_1 - m_2) * log(t) / t$$

A closer look at the function would reveal that it favors dense areas compared to sparse ones. To illustrate the aspect, $BVPF(70,50,20) > BVPF(7,5,2)$. Although it is easy to create a hostile situation to BVPF, we argue that a hostile situation for C3CBR function is much more probable than one for BVPF. In the remaining algorithms that we propose, we consistently use BVPF as the voting power function.

C4 CBR. This is a non-distance-weighting algorithm which uses the BVPF as the voting power function.

C5 CBR. This is the distance-weighting algorithm which uses BVPF as the voting power function.

C6 CBR. As mentioned earlier, determining an optimal value for radius is a non-trivial task. C6 CBR tries to make the process as much insensitive to the value of the radius parameter. We distort the confidence function a bit and redefine it as following:

$$\text{Confidence\_X\_C6CBR}_{\text{radius=r}}\text{(T, k-NN)} = \sum_{i=1}^{n} \text{Confidence\_X}_{\text{radius=i*r}}\text{(T, k-NN)}$$

$\text{Confidence\_X}_{\text{radius=r}}$ (T, k-NN) denotes Confidence_X(T, k-NN) computed with BVPF as the voting power function and r taken as the radius for the BVPF computations. We consistently set n = 5 (the number of terms in the right-hand-side of the above equation) in the course of our experiments with C6 CBR.

5 Performance Measures Used

We use various performance measures for evaluating the different techniques for spam filtering described above. *Spam precision* is the percentage of messages classified as spam that truly are. *Spam recall* (interchangeably referred to as spam accuracy) is the proportion of actual spam messages that are classified as spam. Non-spam messages are usually called solicited messages or legitimate messages. *Legitimate precision*, analogously, is the percentage of messages classified as legitimate that truly are. *Legitimate recall* (interchangeably referred to as legitimate accuracy) is the proportion of actual legitimate messages that are classified as legitimate [4]. The *total accuracy* is the total number of messages classified correctly. Intuitively, as discussed earlier, the severity of the error of classifying a spam message as legitimate is much less than the severity of classifying a legitimate message as spam. Taking these into account, we define an *error cost function* as the sum of the errors with differential weighting for the two kinds of errors. The obvious parameter to this cost function would be the difference in severities. It has been shown [1] that the cost of the latter error is 999 times that of the former in a setting where spam messages are blocked from the user. In a scenario where messages are just flagged as spam by the filter, the disparity in severity comes down to 9. The error cost is hence calculated as (Severity Disparity)*(# of false positives)+(# of false negatives). We analyze the techniques with both values for the severity disparity parameter.

6 Experiments, Results and Implications

In this section, we walk through (in chronological order) the results of the various experiments that have been conducted. As is typical in any experiment in this context,

we divide the corpus into the seed case base (the training set) and the test set (which in the real world context, would be a stream of incoming mail messages). The case base forms the training set and each message in the test set is classified by the CBR making use of the case base. In cases where we choose the seed case base to have a size of 50% of the corpus, we start off putting every other message from the corpus into the seed case base. As the order of the corpus is not representative of the order of messages coming in (the SpamBase corpus does not have time-stamped messages), we choose not to add processed test cases to the case base. It may well be appreciated that adding processed test cases to the case base would be helpful only in cases where the order of consideration of cases is representative of the order of arrival of messages. Given that the entire corpus (and hence the test set too) is labeled, we can get a feel of the performance of the algorithms in this regard.

6.1 Performance of non-radius based techniques

As explained in the preceding section, Diff-CBR, Simple CBR, C1 and C2 CBRs don't require a radius parameter. We experimented with them on varying case base sizes. We present the spam precision and total accuracy charts for those techniques.

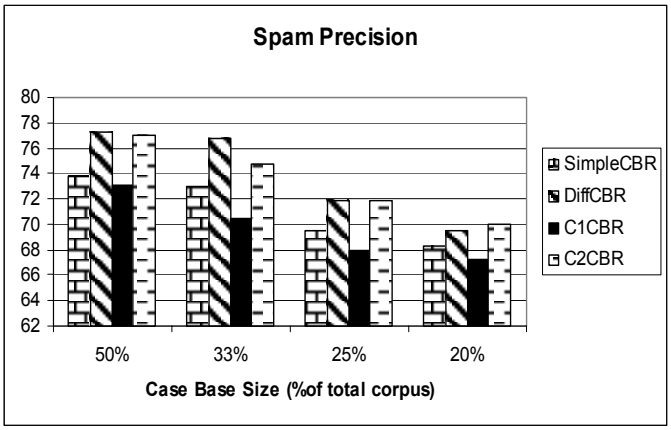

Figure 2. Spam Precision Chart

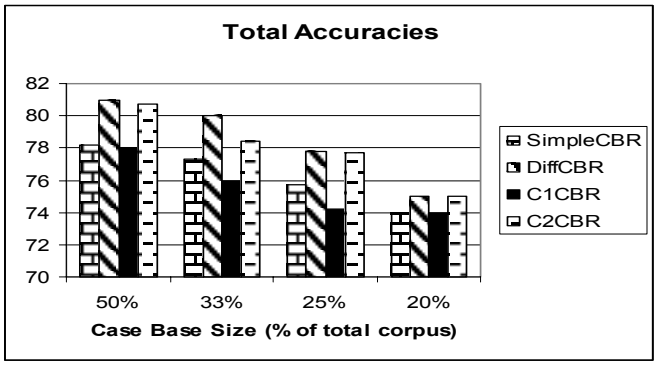

Figure 3. Total Accuracy Chart

As can be seen, Diff-CBR works the best in each of the cases, whereas C2 CBR does approach it closely in performance compared to other methods. Given that C2 CBR seems to be brushing shoulders with Diff-CBR and applies the common technique of distance weighting, we decided to look at the number of common errors that they make to gather an insight as to how much influence the voting power function actually had. The number of common errors was a surprisingly high 88% on the average which indicates that the performance of C2 CBR was more due to the distance-weighting component than the voting power function. As the comparison between C1 and Simple-CBR shows, the voting power function is actually worse than the constant power function used by the latter. Although these results are clearly disheartening, we choose to examine the extent of the effect caused by the drawback of the C1 and C2 voting power functions as mentioned in an earlier section.

6.2 C3CBR

We choose to analyze C3 CBR separately as all others to follow use the same voting power function. We chose to use a 50% case base, and increasing values of radius. We present the accuracy chart as below.

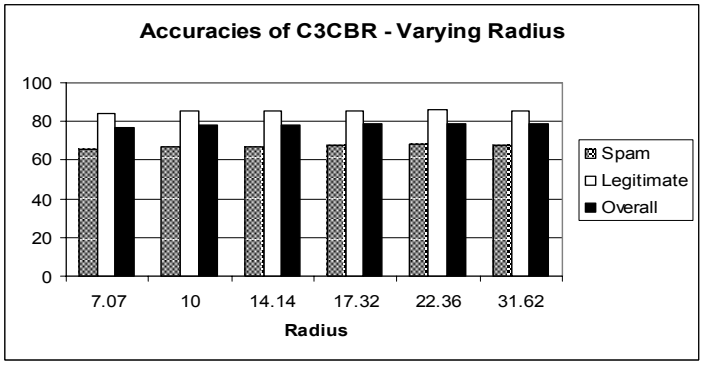

Figure 4. Accuracies of C3CBR, Varying Radius

As is evident from the results, the accuracies don't approach that of the conventional techniques such as Diff and Simple CBR. But one interesting thing worth mentioning in this context is that, although C3 CBR accuracies are (slightly) lesser than Diff-CBR, the fraction of common errors between C3 and Diff were 75% (compared to the figure of 88% for the C2-Diff pair) on the average. This gives us enough confidence that we are not searching in the dark and hence, we proceed to quantify the extent of the drawback discussed in the previous section.

6.3 Performance of Techniques that use BVPF

In our experiments with C4, C5 and C6 CBR, we were able to arrive at significantly better results (with varying values of radius). Even C4 CBR, the non-distance weighting BVPF CBR, gave much better accuracies compared to earlier techniques. C5 and C6 CBRs performed exceedingly well in comparison with others on spam precision. Although we do not include the charts of all the experiments that were conducted, we present a list of representative charts to show the performances of each of the techniques discussed so far so as to assert that BVPF works exceedingly well as a voting power function. All these were done with 50% of the corpus as the case base.

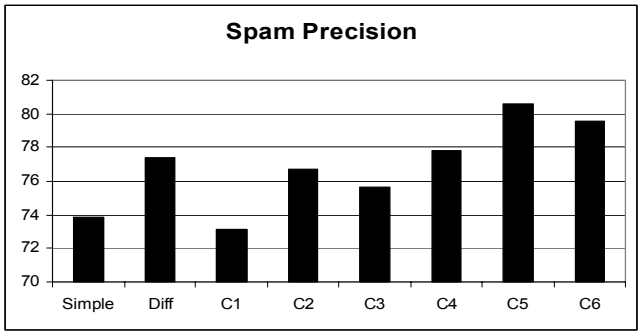

Figure 5. Spam Precision Chart

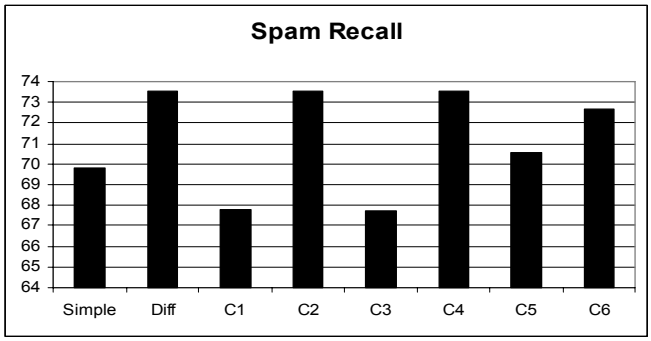

Figure 6. Spam Recall

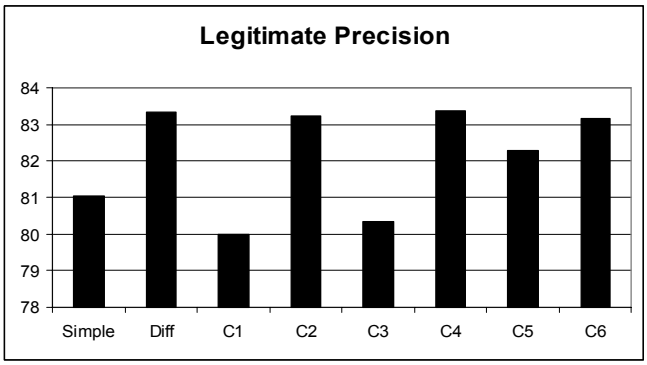

Figure 7. Legitimate Precision

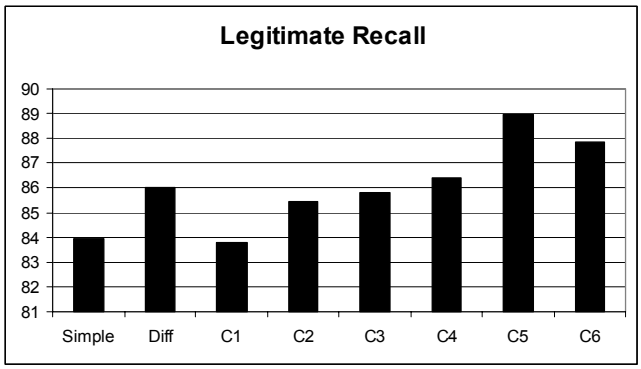

Figure 8. Legitimate Recall

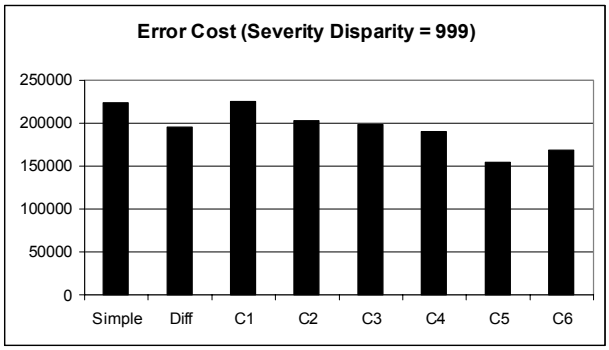

Figure 9. Error Cost Chart (Severity Disparity = 999)

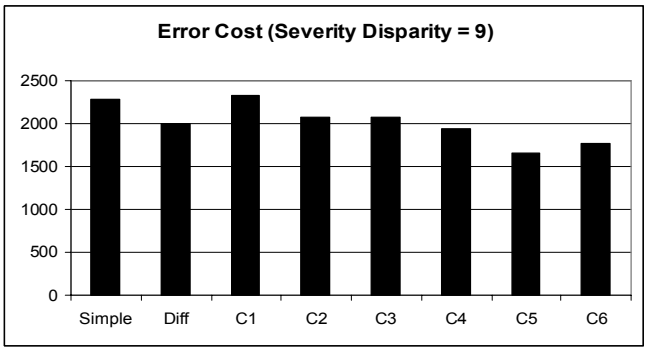

Figure 10. Error Cost Chart (Severity Disparity = 9)

As can be seen, all techniques that use BVPF, viz., C4, C5 and C6 CBRs work much better than the others on the parameters that count the most, i.e., spam precision, legitimate recall, and hence error costs. C6 CBR performs better than Diff on all measures except for Spam Recall where Diff-CBR works slightly better. C5 CBR gives the lowest error costs, and gives the highest Spam Precision and Legitimate Recall. On the whole, C5 and C6 techniques are way ahead of the conventional techniques. This proves our point that making use of the patterns in the case base is does improve performance very much.

7 Voting Powers and a Dynamic Case Base

Given that all our experiments have been on static case bases, it is reasonable enough to devote a section on how the computation of voting powers would be in a real-world scenario where cases get added and deleted from the case base. We provide a straightforward method to update the voting powers when a case gets added or deleted from the case base. We describe an algorithm to show how the BVPF powers can be updated on addition of a case base and omit other details as they would be a straightforward modification of the algorithm. We propose storing the <m_1, m_2, *BVPF* power> triplets for each case in the case base. The linear algorithm described below updates these triplets for relevant cases in the case base when a case gets added.

```
Update_On_Addition(NewCase n, Case Base C)
{
   m1 = m2 = 0;
   for(each case c in C)
   {
      if(distance(c,n) < radius)
      {
         if(label(c) == label(n))
         {
            increment m1;
            increment m1 of case n;
            re-compute BVPF of n;
```

```
            }
            else
            {
              increment m2;
              increment m2 of case n;
              re-compute BVPF of n;
            }
          }
      }
      Store <m1,m2,BVPF(m1+m2,m1,m2)> for the new case n;
}
```

8 Contributions and Future Work

We have, by means of this paper, provided approaches to make use of the patterns in the case base by means of associating each case with a voting power to improve spam filtering using CBR. This is, to the best of our knowledge, the first work on making use of the skewed distribution in the case base for a classification task. We have laid down the concerns on the design of a voting power function. Further, we have experimented exhaustively and made the implications of the voting power functions explicit.

As a part of future work in this regard, we propose to look deeper into the BVPF function and hostile cases to it. Further, as mentioned earlier, BVPF favors dense clusters over sparser ones. The implications of such a bias have to be examined in detail. BVPF is just our first approach in satisfying the desiderata for a voting power function and we have to look to find better variants of BVPF. Secondly, clustering is a data mining task which has been receiving a lot of attention of late. We would like to explore the feasibility of actually clustering the case base and making use of the clusters for the CBR classification task at hand. Further, we would like to look into other domains and test the applicability of BVPF and variants for classification tasks therein.

References

1. Androutsopoulos, Paliouras, Karkaletsis, Sakkis, Spyropoulos and Stamatopoulos, 2000, *Learning to filter spam e-mail: a comparison of a naive Bayesian and a memory-based approach*, PKDD Workshop on Machine Learning and Textual Information Access, 2000
2. Cunningham, Nowlan, Delany and Haahr, 2003, *A case-based approach to spam filtering that can track concept drift*, Proceedings of the ICCBR Workshop on Long-Lived CBR Systems, Norway, 2003

3. Sakkis, Androutsopoulos, Paliouras, Karkaletsis, Spyropoulos and Stamatopoulos, 2003, *A memory-based approach to anti-spam filtering for mailing lists*, Journal of Information Retrieval, Kluwer, 2003
4. Sahami, Dumais, Heckerman & Horvitz, 1998, *A bayesian approach to filtering junk e-mail*, AAAI-98 Workshop on Learning for Text Categorization, 1998
5. Ester, Kriegel, Sander, Xu, 1996, *A Density based algorithm for discovering clusters in large spatial databases with noise*, International Conference on Knowledge Discovery in Databases, KDD-1996
6. Hinneburg and Keim, 1998, *An efficient approach to clustering in large multimedia databases with noise*, International Conference on Knowledge Discovery in Databases, KDD-1998
7. Ankerst, Breunig, Kriegel and Sander, 1999, *OPTICS: Ordering Points to Identify the Clustering Structure*, Proceedings of the ACM SIGMOD Conference
8. Zhao, Karypis, "Criterion Function for Document Clustering: Experiments and Analysis", Department of Computer Science, University of Minnesota, TR#01-40
9. Drucker, H.D., Wu, D., Vapnik, V.: Support Vector Machines for spam categorization. IEEE Transaction on Neural Networks, Vol. 10 (5). 1999 1048-1054