2007-01-01

Catching the Drift : Using Feature-Free Case-Based Reasoning for Spam Filtering.

Sarah Jane Delany
Dublin Institute of Technology, Sarahjane.Delany@dit.ie

Derek Bridge
University College Cork

Recommended Citation

Delany, S. & Bridge, D. (2007) Catching the Drift: using Feature-Free Case-Based Reasoning for Spam Filtering. *7th. International Conference on Case-Based Reasoning (ICCBR),* Belfast, Northern Ireland, 13-16 August.

Catching the Drift: Using Feature-Free Case-Based Reasoning for Spam Filtering

Sarah Jane Delany[1] and Derek Bridge[2]

[1] Dublin Institute of Technology, Dublin, Ireland
sarahjane.delany@comp.dit.ie
[2] University College Cork, Cork, Ireland
d.bridge@cs.ucc.ie

Abstract. In this paper, we compare case-based spam filters, focusing on their resilience to concept drift. In particular, we evaluate how to track concept drift using a case-based spam filter that uses a feature-free distance measure based on text compression. In our experiments, we compare two ways to normalise such a distance measure, finding that the one proposed in [1] performs better. We show that a policy as simple as retaining misclassified examples has a hugely beneficial effect on handling concept drift in spam but, on its own, it results in the case base growing by over 30%. We then compare two different retention policies and two different forgetting policies (one a form of instance selection, the other a form of instance weighting) and find that they perform roughly as well as each other while keeping the case base size constant. Finally, we compare a feature-based textual case-based spam filter with our feature-free approach. In the face of concept drift, the feature-based approach requires the case base to be rebuilt periodically so that we can select a new feature set that better predicts the target concept. We find feature-free approaches to have lower error rates than their feature-based equivalents.

1 Introduction

Spam filtering is a classification task. The filter must predict whether an incoming email is 'ham' (legitimate) or 'spam' (illegitimate).

Different filters work in different ways. In procedural approaches users deploy whitelists, blacklists and authentication protocols[3]; in collaborative approaches users share signatures computed from the spam they receive[4]; and in content-based approaches the filter inspects the header, body and attachments of each email, or features computed from these, for content that is indicative of spam. Of course, filters may also combine the different approaches.[5]

In content-based filters, the classifier may make its decisions using, for example, rules, decision trees or boosted trees [2], Support Vector Machines [3],

[3] E.g. www.email-policy.com/Spam-black-lists.htm, www.emailauthentication.org/
[4] E.g. http://razor.sourceforge.net/, http://www.rhyolite.com/anti-spam/dcc/
[5] E.g. http://spamassassin.apache.org/

probabilities [4, 5] or exemplars [5–8]. Except in the case of rules, which are most often human-authored, a learning algorithm usually induces the classifier from a set of labelled training examples. This is a form of concept learning. But spam has the following characteristics that make this form of concept learning especially challenging. First, there is a subjective and personal aspect to spam: what is spam to one person may not be spam to another [9]. Second, spam is a heterogeneous concept: spam that advertises replica watches shares little content with pornographic spam. Third, there is a high cost to false positives: it is unacceptable to most users if ham is incorrectly classified as spam. Finally, there is on-going concept drift: spam is constantly changing.

We have been taking a Case-Based Reasoning (CBR) approach to spam filtering [6, 7, 10], as have others [8], in the belief that this approach can overcome the challenges. First, individual users can maintain their own case bases to represent their personal, subjective interests. Second, instance-based approaches, such as CBR, often perform well when learning complex target concepts, including heterogeneous and disjunctive ones [11]. Third, as we explain in Section 2.1, by using a unanimous voting mechanism we can bias a k-nearest neighbours classifier away from false positives. Finally, lazy learners including CBR can easily be updated incrementally to cope with concept drift [12].

We recently introduced a 'feature-free' distance measure into our case-based spam filter, resulting in significant improvements in classification accuracy [10, 13]. In this paper, we show that the feature-free distance measure is also more resilient to concept drift.

In Section 2, we describe our case-based approach to spam filtering. We describe both the feature-based distance measure that we used in earlier work and the feature-free distance measure, based on text compression, which we have been using in our more recent work. In Section 3, we describe concept drift in more detail and we review techniques for handling concept drift especially in instance-based learners. In Section 4, we present our new experimental results. These include a comparison of two different distance measures that are based on text compression; a comparison of instance selection and instance weighting approaches to tracking concept drift; and a comparison of feature-based and feature-free approaches to handling concept drift, based on periodically rebuilding the case base.

2 Case-Based Spam Filtering

2.1 Email Classification Using Examples (ECUE)

Our case-based spam filter is called ECUE [7, 14]. Its case base is a set of labelled training examples, both ham and spam. ECUE retrieves an incoming email's k nearest-neighbours (k-NN) from the case base and uses unanimous voting in reaching the classification. In other words, ECUE classifies the incoming email as spam only if all k of the nearest-neighbours are spam. The case base is a Case Retrieval Net [15], which speeds up the retrieval process.

ECUE incorporates case base editing algorithms, which, prior to classification, can remove redundant or noisy cases from the case base. Such algorithms aim to reduce the size of the case base and hence reduce retrieval time, while endeavouring to maintain or even improve the generalisation accuracy [16–18].

The case base editing technique that we use is called Competence-Based Editing (CBE) [6]. CBE builds a competence model of the case base by identifying for each case its usefulness (represented by the cases that it contributes to classifying correctly) and also the damage that it causes (represented by the cases that it causes to be misclassified). A two step process uses these case properties to identify the cases to be removed. The first step, Blame-Based Noise Reduction, removes noisy cases that adversely affect classification accuracy. The second step, Conservative Redundancy Reduction, removes redundant cases that are not needed for correct classification. CBE has been shown to conservatively reduce the size of an email case base while maintaining and even improving its generalisation accuracy. We describe CBE in detail in [6].

We have recently been experimenting with two variants of the ECUE system, one which uses a feature-based case representation and distance measure (Section 2.2), and another which takes a feature-free approach, based on text compression (Section 2.3).

2.2 The Feature-Based Distance Measure

In the feature-based version of ECUE, we represent each email e_j as a vector of feature values, $e_j = (f_{1j}, f_{2j}, \ldots f_{nj})$. We use binary-valued features only: if the feature exists in the email the feature value $f_{ij} = 1$, otherwise $f_{ij} = 0$. We do not use numeric-valued features (e.g. occurrence frequencies) because we found that they resulted in only minor improvements in overall accuracy, no significant decrease in false positives, and much increased classification and case base editing times [7].

The distance between a target case (an incoming email) e_t and a case from the case base e_c is simply a count of features on which they disagree:

$$FDM(e_t, e_c) =_{\text{def}} \sum_{i=1}^{n} |f_{it} - f_{ic}| \tag{1}$$

We found it better, especially from the point of view of false positives, not to use *feature weighting* on the binary representation [7].

We compute features from some of the header fields and the body of the emails, with no stop-word removal or stemming. We have three types of features: word, character and structural. For word features, the feature is set to 1 if and only if the word appears in the email. For character features, whether the feature is set to 1 or 0 depends on the frequency of occurrence of the character in the email. The Information Gain (IG) is calculated for each character over the full dataset, where the character is represented as a continuous, and not binary, feature [19] (i.e. the value in each email is the frequency of the character normalised by the maximum character frequency). The normalised frequency that returns the highest IG value in this calculation is used as a threshold for that character across the dataset. The character feature value in each email is set to 1 if and only if the normalised frequency of the character in the email is greater than or equal to this threshold. For structural features (e.g. the proportion of uppercase characters, lowercase characters or white space in the email), we again use Information Gain as a threshold to give a binary representation.

Feature extraction on the training examples finds a large set of candidate features; *feature selection* on these candidates uses Information Gain to identify a subset that is predictive of ham and spam. Based on the results of preliminary cross-validation experiments, we chose to use 700 features for the evaluations in this paper. One observation, which we will return to in Section 3, is that to handle concept drift in spam filtering it is advantageous to periodically re-run the feature extraction and feature selection processes using the most recently received emails.

2.3 The Feature-Free Distance Measure

In the feature-free version of ECUE, we compute distance directly on the textual content of some of the header fields and the bodies of the emails using a distance measure that is based on text compression.

Distance measures based on data compression have a long history in bioinformatics, where they have been used, e.g., for DNA sequence classification [20]. Outside of bioinformatics, compression-based distance measures have been applied to *clustering* of time-series data [21] and languages [22, 23]. They have also been applied to *classification* of time series data [21]. But, to the best of our knowledge, ours is the first application of these distance measures to text classification in general and spam filtering in particular[6] [10, 13]. There have, however, been other classifiers based on text compression. In these classifiers, for each class an adaptive statistical compressor builds a compression model from training examples belonging to that class. The classifier assigns a target document to the class whose compression model best accounts for that document [24–26]. Bratko *et al.* have recently used classifiers of this kind for spam filtering [27, 28]. Rennie and Jaakkola, on the other hand, propose using text compression to discover features indicative of spam [29].

Keogh *et al.* [21] and Li *et al.* [1] have both presented generic distance measures based on data compression and inspired by the theory of Kolmogorov complexity. The *Kolmogorov complexity* $K(x)$ of a string x can be defined as the size of the smallest Turing machine capable (without any input) of outputting x to its tape. The *conditional Kolmogorov complexity* $K(x|y)$ of x relative to y can be defined as the size of the smallest Turing machine capable of outputting x when y is already on its tape. This can be the basis of a distance measure. Informally, if $K(x|y) < K(x|z)$, then y contains more information content that is useful to outputting x than z does, and so y is more similar to x than z is.

[6] But see the discussion of the possibility of using compression in instance-based classification of email at www.kuro5hin.org/story/2003/1/25/224415/367

Unfortunately, Kolmogorov complexity is not computable in general, and so we must approximate it. Since the Kolmogorov complexity of a string is in some sense the size of the smallest description of the string, one way of thinking of $K(x)$ is that it is the length of the best compression we can achieve for x. So, we can approximate $K(x)$ by $C(x)$, the size of x after compression by a data compressor.

We can define useful distance measures by comparing $C(x)$, $C(y)$ and $C(xy)$, where $C(xy)$ is the size after compression of y concatenated to the end of x. The intuition here is that compression of xy will exploit not only the redundancies within x and within y but also inter-document redundancies (similarities) between x and y too. If there are inter-document redundancies, then the amount of compression of xy should be greater than we obtain by compressing x and y separately. This still leaves the question of how to combine these into a normalized measure of distance.

Keogh *et al.* [21] define the Compression-Based Dissimilarity between strings x and y as follows:

$$CDM(x,y) =_{\text{def}} \frac{C(xy)}{C(x)+C(y)} \tag{2}$$

CDM produces values in the range $(0.5, 1]$. Even with the best possible compression algorithm, the lowest value it can produce is slightly above 0.5 because, even if $x = y$, $C(xy)$ will be slightly greater than $C(x)$. In principle CDM's maximum value is 1. This would occur when x and y are so different that $C(xy) = C(x) + C(y)$. In other words, it occurs when there is no inter-document redundancy.

Li *et al.* [1] offer an extensive theoretical analysis of their definition, which normalizes differently. They define the Normalized Compression Distance between strings x and y as follows:

$$NCD(x,y) =_{\text{def}} \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))} \tag{3}$$

NCD produces values in the range $[0, 1+\epsilon]$, where the upper bound allows for imperfections in the compression algorithm. Li *et al.* say that values of ϵ above 0.1 are unlikely [1]. In fact, in a leave-one-out validation on a case base of 1000 emails, the range we obtained was $[0.02, 0.93]$.

In our previous work [10, 13], we used CDM. In Section 4.2, we report the first empirical comparison of CDM and NCD, and we find in fact that NCD generally gives superior results. It is then NCD that we use in the remaining experiments in this paper.

Note that the properties expected of distance metrics do not in general hold for CDM and NCD. In general, it is not the case that $CDM(x,x) = 0$ iff $x = y$; $CDM(x,y) \neq CDM(y,x)$, i.e. CDM is not symmetric; and $CDM(x,y) + CDM(y,z) \not\geq CDM(x,z)$, i.e. the triangle-inequality does not hold. Similarly, these do not hold in general for NCD.

None of this prevents use of CDM or NCD in, for example, classification tasks, provided the classification algorithm does not rely on any of these properties. For example, an exhaustive implementation of k-NN (in which the algorithm finds the k nearest neighbours to the query by computing the distance between the query and *every* case in the case base) will work correctly. But retrieval algorithms that rely on these properties to avoid computing some distances (e.g. k-d trees [30] and Fish and Shrink [31]) are not guaranteed to work correctly.

CDM and *NCD* give us feature-free approaches to computing distances between textual cases. They can work directly on the raw text. Hence, this feature-free approach has negligible set-up costs. Cases are represented by raw text: there is no need to extract, select or weight features; there is no need to tokenise or parse queries or cases. This is a major advantage, especially if each user is to have a personalised case-based filter. By contrast, in the feature-based approach, we must extract and select features for each individual user case base.

We also believe that the feature-free approach should have an advantage in tracking concept drift. We explore and test this in the remainder of this paper.

3 Concept Drift

In some tasks, including spam filtering, the target concept is not static. It changes over time, and the characteristics that are predictive of the target concept change also. Spam changes according to the season (e.g. an increase after Christmas of spam that advertises weight loss products) and according to world events (e.g. the surge in spam related to religion following the death of Pope John Paul II). What people regard as spam also changes: their interests change (e.g. a person's interest in emails that advertise replica watches may cease after buying one) and their tolerances change (e.g. reminders of a conference or seminar can become increasingly unwelcome). But, above all, spam changes because there is an arms race between spammers and those who produce filters: each continually tries to outwit the other.

For many spam filters, this is a losing battle. But where classifiers are induced from examples, we can retrain the classifier using new examples of ham and spam, especially misclassified examples. Lazy learners, including the case-based approach, have the advantage that they can easily learn incrementally. The filter can insert into the case base new emails, along with their correct classifications, in the hope that this improves the competence of the system. However, incremental update is not enough. On its own, it results in an ever larger case base and ever longer classification times. Importantly also, because of the concept drift, some cases in the case base may no longer contribute to correct classification.

There are two main solutions: *instance selection* and *instance weighting*. The goal of instance selection is to identify, among all instances in the case base, those instances that define the current target concept. Other instances are deleted or, less commonly, are made ineligible for retrieval. Most instance selection algorithms are window-based. A window slides over the training examples, using, for example, only the most recent examples for prediction. Examples of window-based algorithms include the FLORA family of algorithms [32], FRANN [33] and Time-Weighted Forgetting [34]. Some algorithms use a fixed-size window, while others adjust the window size based on the rate and amount of drift [35, 36, 32].

In instance weighting, instances are weighted according to their age or performance. Instances with low weights are less likely to be used for classification and, if their weights become low enough, may even be deleted. Klinkenberg describes how to use Support Vector Machines in this way [36]. Instance-based approaches to instance weighting include Locally-Weighted Forgetting (LWF) and Prediction Error Context Switching (PECS) [34]. In LWF, each instance is weighted by a combination of its age and whether the learner has subsequently encountered new examples that are similar to it; new examples eventually oust older similar examples. In PECS, instances are weighted by their predictiveness. Specifically, if a stored example begins to disagree often with the correct classifications of its neighbours, then it is moved to an inactive set, from where it is no longer eligible for retrieval. This kind of performance weighting, and the use of confidence interval tests in deciding when to move instances between the active and inactive sets, gives PECS strong similarities with IB3 [37].

For completeness, we mention an altogether different way of handling concept drift: the use of ensembles of classifiers induced from different subsets of the training examples [38]. Further consideration of ensemble approaches is beyond the scope of this paper.

The research we have described above on instance selection, instance weighting and ensembles tracks concept drift by trying to use just the subset of examples that are predictive of the current target concept. But, it is important to realise that, additionally in feature-based approaches, the features that were once predictive may no longer be so. Hence the case-based filter must periodically extract and select a new feature set from the most recent examples and rebuild the case base so that it uses these new features. We investigated this in [14]. A related approach has subsequently been reported in [8], where a case-specific set of features is computed separately for each email when it arrives.

Of course, in this paper we are comparing a feature-based approach with a feature-free approach. We believe that the feature-free approach, which we have already found to be more accurate and to have lower set-up costs, will also be more resilient to concept drift precisely because its definition of the concept depends only on the current contents of the case base, and not on any features.

4 Spam Filtering Experiments

4.1 Evaluation Setup

The focus of the experiments reported in this paper is to investigate the effect of feature-free, compression-based distance measures on concept drift. To that end, we used two large datasets of date-ordered ham and spam.[7] Each dataset was collected by an individual from the email that they received over a period of approximately one year. For each dataset, we set up an initial case base

[7] The datasets are available for download at www.comp.dit.ie/sjdelany/dataset.htm

using a training set of one thousand cases, five hundred consecutively received ham emails and five hundred consecutively received spam emails. This left the remainder of the data for testing and updating the case base. Table 1 shows the profile of the test data across each month for both datasets.

Table 1. Profile of the test data in Datasets 1 and 2

	Feb '03	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan '04	**Total**
Dataset 1													
spam		629	314	216	925	917	1065	1225	1205	1830	576		**8902**
ham		93	228	102	89	50	71	145	103	85	105		**1076**
Dataset 2													
spam	142	391	405	459	406	476	582	1849	1746	1300	954	746	**9456**
ham	151	56	144	234	128	19	30	182	123	113	99	130	**1409**

We presented each test email for classification to the case-based classifiers in date order. Results were accumulated over each month.

Since False Positive (FP) classifications (ham classified incorrectly as spam) are much more serious than False Negative (FN) classifications (spam classified incorrectly as ham), accuracy (or error) as a measure of performance does not present the full picture. Two filters with similar accuracies may have very different FP and FN rates. In addition, as the amount of ham is considerably lower than the amount of spam in the datasets, the actual error figure would follow the FN rate and not give adequate emphasis to FPs. The measure we use is the average within-class error rate, $\mathit{Err} = \frac{\mathit{FPRate}+\mathit{FNRate}}{2}$, rather than the actual error rate ($\mathit{Err} = \frac{\mathit{number\ misclassified}}{\mathit{total\ emails}}$). We also report the FP rate ($\mathit{FPRate} = \frac{\mathit{number\ of\ false\ positives}}{\mathit{total\ negative\ examples}}$) and FN rates (defined analogously) separately. Where differences exist, we use a two-tailed, paired t-test across the monthly results to test for significance.

The compression algorithm we use in all experiments reported here is GZip, a variant of Lempel-Ziv compression, in which a repetition of a string within a text may be replaced by a pointer to an earlier occurrence.

From the point of view of Kolmogorov complexity, one should use the best compression algorithm because the better the compression rate, the closer the compression algorithm will approximate the Kolmogorov complexity. But this does not necessarily mean that using the better compressor in a compression-based distance measure such as those defined in Equations (2) and (3) will result in a better approximation of the distance as it may effect different terms in the formulae differently [1]. We compared compression algorithms in [13], specifically GZip with PPM, an adaptive statistical compressor that is considered to achieve some of the best compression rates. We found GZip to compress the emails in our datasets slightly more than PPM with little difference in classification accuracy when used in a distance measure. PPM is considerably slower than GZip and by truncating each email to 8000 characters (exploiting the fact that substitutions in GZIP are confined to a 32 Kbytes sliding window) we were able to further speed up GZip with no appreciable loss of accuracy.

4.2 A Comparison of *CDM* and *NCD*

To compare the two distance measures, *NCD* and *CDM*, we presented each email in date order for classification against the full case base of 1000 training emails. The results are in Figure 1 which show that *NCD* performs better overall than *CDM*. The differences in overall error are significant in both datasets at the 99% level, but the differences in FP rate are not. Figure 1 also includes the results of adding each misclassified email as it is found into the case base: the error rates are considerably lower, with no significant difference between the two distance measures. These results show that even relatively simple attempts to track concept drift (in this case, retaining misclassified examples) can work well.

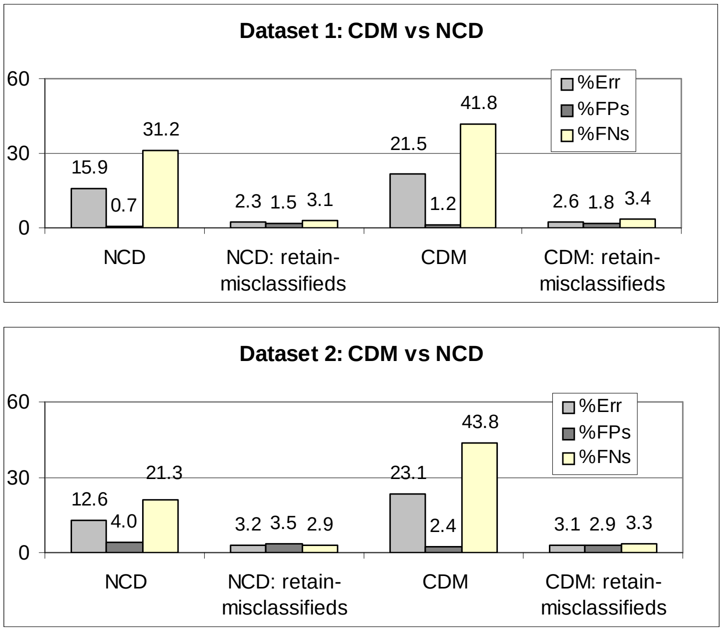

Fig. 1. A comparison of *CDM* and *NCD*

4.3 Handling Concept Drift with *NCD*

As we described in Section 3, one of the difficulties of handling concept drift by retaining examples is that the size of the case base increases. In the experiments performed in Section 4.2 above, when the case bases are updated with misclassified emails they increased in size by 32% and 30% respectively over the duration of the experiment.

It is necessary therefore to include a *forgetting* mechanism to remove instances that may no longer be useful in classification. We present results here of a variety of different approaches to retention and forgetting within an *add-1-delete-1* policy. The approaches to retention can be categorised as either *retain-all*, where all examples are added to the case base, or *retain-misclassifieds*, where just those examples that are misclassified are added. The forgetting mechanisms we considered were *forget-oldest* and *forget-least-accurate*. *Forget-oldest* is a simple form of *instance selection*, sliding a fixed-size window over the examples and deleting the oldest. *Forget-least-accurate* can be considered as a simple form of *instance weighting*, inspired by IB3 and PECS. In *forget-least-accurate*, the accuracy of a case is measured as the proportion of times the case is successfully retrieved. More formally, $accuracy = \#successes/\#retrievals$, where $\#retrievals$ is the number of times a case is retrieved as a nearest neighbour in k-NN classification and $\#successes$ is the number of times the case is both retrieved as a neighbour and has the same class as the target. In the case of ties, we delete the oldest of the least accurate cases. So that we compare like with like as much as possible, we use the accuracy records only for forgetting; they do not, as in IB3, influence which cases are retrieved as neighbours.

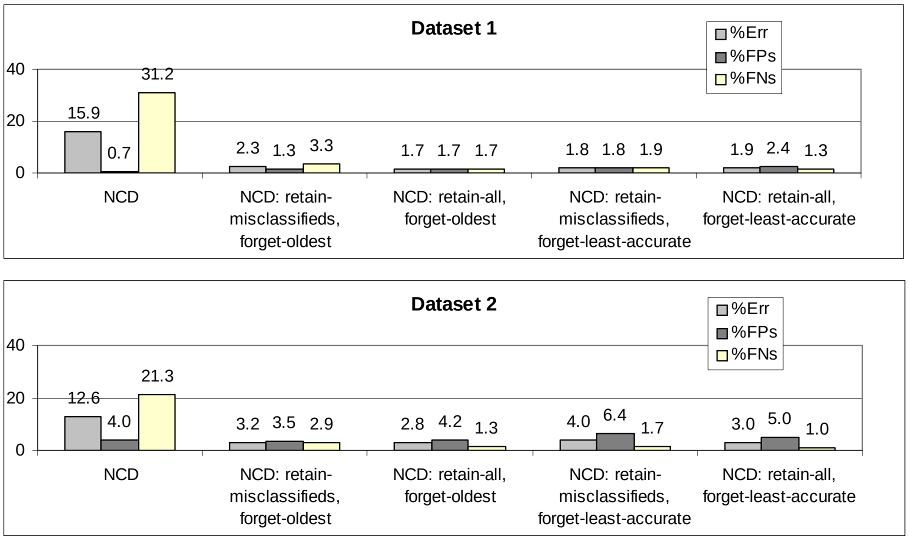

Fig. 2. *NCD* with various concept drift tracking mechanisms

It is evident from Figure 2 that all the approaches are successful at tracking the drift. It is difficult to identify any one approach that is the best, but *forget-*

least-accurate, in spite of having a significant beneficial effect on FNs, appears to have a negative effect on FPs which is not desirable.

We also investigated using Competence-Based Editing on the initial case bases but found that it did not significantly improve on the results shown in Figure 2. The consequence of using an *add-1-delete-1* policy is that the case base size remains constant. If case base editing is also used then the sizes of the case bases will differ across systems.

4.4 Feature-Free versus Feature-Based

Given that we have found that *NCD* can track concept drift, this section investigates how well it works compared with an approach that uses a feature-based representation and distance measure, which we designate *FDM*. To track concept drift with our feature-based version of ECUE, we incorporate two levels of learning [14]. Firstly, we retain misclassified emails as they are found. Secondly, the feature set needs to be periodically updated from the more recent examples of email. Specifically, at regular intervals (monthly for the purposes of our evaluation), we re-run feature extraction and selection on the most recent emails; we rebuild the case base using only more recent examples of email, so that each included case is now represented using the new set of features; and we run Competence-Based Editing (CBE) on the rebuilt case base. Since CBE deletes noisy and redundant cases, it gives us an implicit form of forgetting, thereby limiting case base growth. Note that previously reported experiments with *FDM* show this two-level form of learning to outperform a window-based instance selection system [14].

Although *NCD* does not use features, in order to compare like with like we implemented a version of ECUE that uses *NCD*, retains misclassified examples and achieves a degree of forgetting by rebuilding the case base from the most recent examples and running CBE on the rebuilt case base every month.

Figure 3 shows the results of comparing the *FDM* approach with the equivalent *NCD* approach. The Figure shows that when neither the feature-based nor the feature-free system incorporates any mechanism for handling concept drift, *NCD* has significantly lower FP rates; its FN rates are higher for one dataset and lower for the other.

However, the Figure also shows that when both systems retain misclassified examples and rebuild their case bases every month (in the way described above), the feature-free system tracks the concept drift better than the feature-based system with the differences in overall error significant at the 95% level for Dataset 2 and the 90% level for Dataset 1. There is, however, no significant difference in the FP rate.

The differences between the *NCD* tracking results presented in Figure 3 and those presented for the *add-1-delete-one* policies in Figure 2, although better in some cases and worse in others, are for the most part not significant. In any case, the case base rebuild and edit approach to forgetting offers advantages over the *add-1-delete-1* policies. Firstly it allows the spam filter to use smaller case bases, which speeds up classification. Secondly, it facilitates bootstrapping

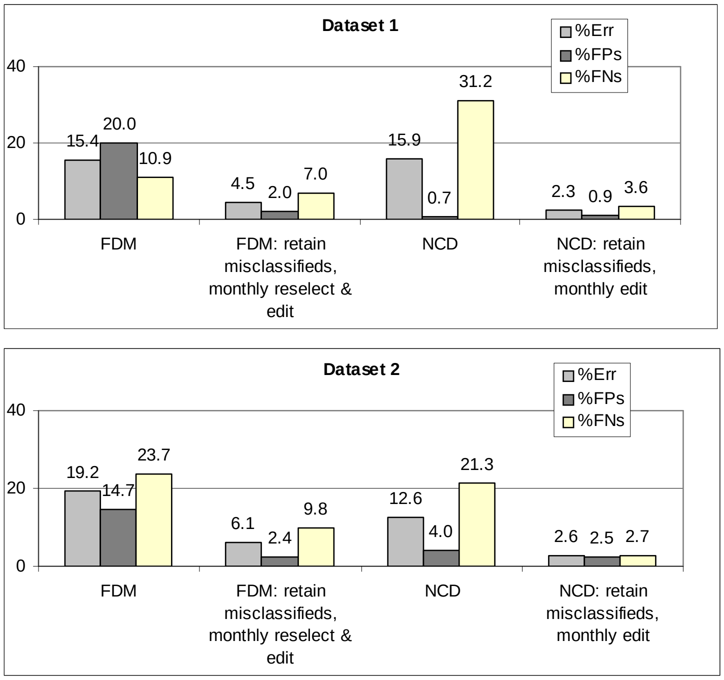

Fig. 3. *NCD* compared with *FDM*

of a personalised spam filter. The system can be installed with small amounts of training data available from a user's email but over time will stabilise to appropriate amounts of training data as periodic rebuild and edit takes place.

A graphical representation of the average monthly error across classifications is shown in Figure 4. Although *NCD* is better overall than *FDM* (Figure 3), Figure 4 shows that, when the case base is never updated, *NCD* does not consistently outperform *FDM* in all months, in particular in Dataset 1. But when misclassified examples are retained and the case base is rebuilt and edited every month, the *NCD* results are as good as or better for all months in Dataset 2 and for all except one month in Dataset 1.

5 Conclusion

This paper continues our investigation of a feature-free distance measure based on text compression for case-based spam filtering. We have new results in which Li *et al.*'s *NCD* outperforms Keogh *et al.*'s *CDM*. But the real focus of the paper has been on concept drift.

We have shown that concept drift in spam is very real and, without a way of handling it, accuracy will be much the lower. We have shown too (Figures 1 and 2) that even quite simple retention and forgetting policies can be very effective.

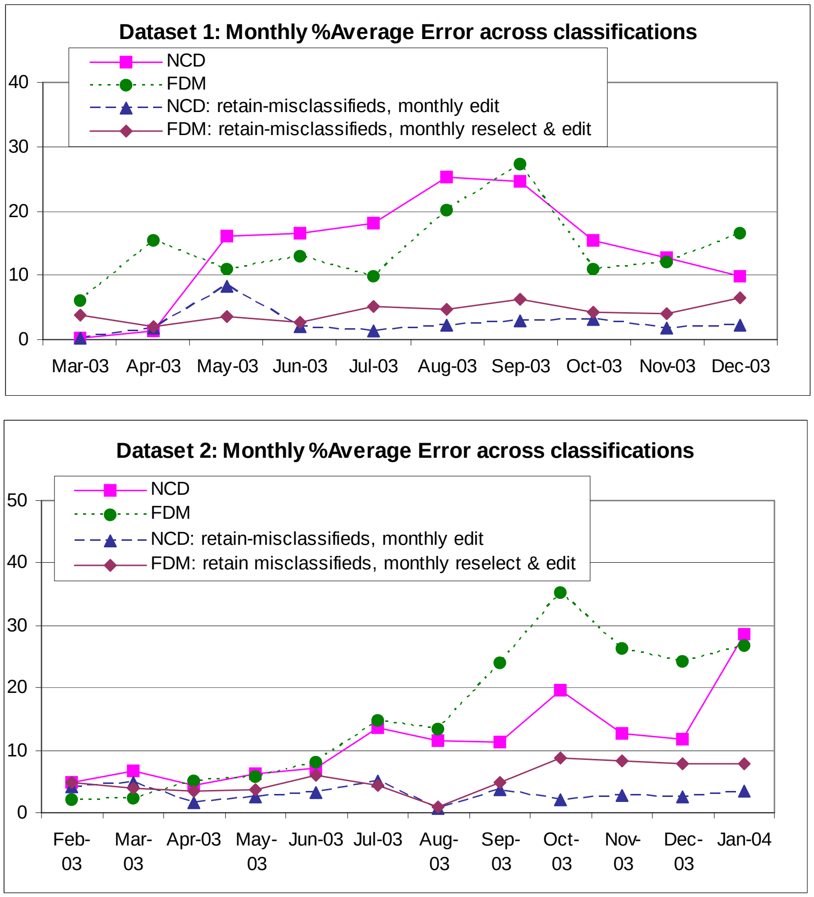

Fig. 4. Monthly average error across classifications

Finally, we have shown (Figures 3 and 4) that the feature-free approach can obtain accuracy that is better than or comparable to that of the feature-based approach but with lower set-up costs and simpler periodic maintenance demands.

In the future we would like to experiment with feature-free case-based spam filters using more sophisticated forms of instance weighting, perhaps closer to IB3 or PECS, and combining recency of use, amount of use and degree of success in the weighting formula. Encouraged by the success of the periodic case base rebuild and edit experiments, we would like to develop and evaluate a forgetting policy based on a competence model, that we hope would be cheaper than running a full case base edit. We would also like to extend the application of *NCD* to texts other than emails, to tasks other than classification, and to text other than raw text, e.g. text that has undergone POS-tagging.

References

1. Li, M., Chen, X., Li, X., Ma, B., Vitanyi, P.: The similarity metric. In: Procs. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, Maryland (2003) 863–872
2. Carreras, X., Marquez, L.: Boosting trees for anti-spam filtering. In: Procs. of the 4th International Conference on Recent Advances in Natural Language Processing, Tzigov Chark, Bulgaria (2001) 58–64
3. Drucker, H., Wu, D., Vapnik, V.: Support vector machines for spam categorization. IEEE Trans. on Neural Networks **10** (1999) 1048–1054
4. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian approach to filtering junk email. In: Procs. of the AAAI-98 Workshop for Text Categorisation, Madison, Wisconsin (1998) 55–62
5. Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C., Stamatopoulos, P.: Learning to filter spam e-mail: A comparison of a naive Bayesian and a memory-based approach. In: Procs. of the PKDD-2000 Workshop on Machine Learning and Textual Information Access. (2000) 1–13
6. Delany, S.J., Cunningham, P.: An analysis of case-based editing in a spam filtering system. In: Procs. of the 7th European Conference on Case-Based Reasoning, Madrid, Spain (2004) 128–141
7. Delany, S., Cunningham, P., Coyle, L.: An assessment of case-based reasoning for spam filtering. Artificial Intelligence Review **24** (2005) 359–378
8. Méndez, J.R., Fdez-Roverola, F., Iglesias, E.L., Díaz, F., Corchado, J.M.: Tracking concept drift at feature selection stage in spamhunting: An anti-spam instance-based reasoning system. In: Procs. of the 8th European Conference on Case-Based Reasoning, Fethiye, Turkey (2006) 504–518
9. Gray, A., Haahr, M.: Personalised, collaborative spam filtering. In: Procs. of 1st Conference on Email and Anti-Spam, Mountain View, CA (2004)
10. Delany, S.J., Bridge, D.: Feature-based and feature-free textual CBR: A comparison in spam filtering. In: Procs. of the 17th Irish Conference on Artificial Intelligence and Cognitive Science, Belfast, Northern Ireland (2006) 244–253
11. Aha, D.W.: Generalizing from case studies: A case study. In: Procs. of the 9th International Conference on Machine Learning, Aberdeen, Scotland (1992) 1–10
12. Delany, S.J., Cunningham, P., Smyth, B.: ECUE: A spam filter that uses machine learning to track concept drift. In: Procs. of the 17th European Conference on Artificial Intelligence, (PAIS stream), Riva del Garda, Italy (2006) 627–631
13. Delany, S.J., Bridge, D.: Textual case-based reasoning for spam filtering: A comparison of feature-based and feature-free approaches. Artificial Intelligence Review (Forthcoming)
14. Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L.: A case-based technique for tracking concept drift in spam filtering. Knowledge-Based Systems **18** (2005) 187–195
15. Lenz, M., Auriol, E., Manago, M.: Diagnosis and decision support. In Lenz, M., et al., eds.: Case-Based Reasoning Technology, From Foundations to Applications. LNAI 1400, Springer-Verlag (1998) 51–90
16. McKenna, E., Smyth, B.: Competence-guided case-base editing techniques. In: Procs. of the 5th European Workshop on Case-Based Reasoning, Trento, Italy (2000) 186–197
17. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Machine Learning **38** (2000) 257–286

18. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery **6** (2002) 153–172
19. Quinlan, J.R.: C4.5 Programs for Machine Learning. Morgan Kaufmann (1997)
20. Loewenstern, D., Hirsh, H., Yianilos, P., Noordewier, M.: DNA sequence classification using compression-based induction. Technical Report 95-04, Rutgers University, Computer Science Department (1995)
21. Keogh, E., Lonardi, S., Ratanamahatana, C.: Towards parameter-free data mining. In: Procs. of the 10th ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining, New York, NY, USA (2004) 206–215
22. Benedetto, D., Caglioti, E., Loreto, V.: Language trees and zipping. Physical Review Letters **88** (2002) 048702/1–048702/4
23. Cilibrasi, R., Vitanyi, P.: Clustering by compression. IEEE Transactions on Information Theory **51** (2005) 1523–1545
24. Frank, E., Chui, C., Witten, I.H.: Text categorization using compression models. In: Procs. of the IEEE Data Compression Conference, Utah, USA (2000) 200–209
25. Teahan, W.J.: Text classification and segmentation using minimum cross-entropy. In: Procs. of the 6th International Conference on Recherche d'Information Assistee par Ordinateur, Paris, France (2000) 943–961
26. Teahan, W.J., Harper, D.J.: Using compression-based language models for text categorization. In: Procs. of the Workshop on Language Modeling for Information Retrieval, Carnegie Mellon University (2001) 83–88
27. Bratko, A., Filipič, B.: Spam filtering using character-level Markov models: Experiments for the TREC 2005 spam track. In: Procs. of the 14th Text REtrieval Conference, Gaithersburg, MD (2005)
28. Bratko, A., Cormack, G.V., Filipič, B., Lynam, T.R., Zupan, B.: Spam filtering using statistical data compression models. Journal of Machine Learning Research **7** (2006) 2673–2698
29. Rennie, J.D.M., Jaakkola, T.: Automatic feature induction for text classification. In: MIT Artificial Intelligence Laboratory Abstract Book, Cambridge, MA (2002)
30. Wess, S., Althoff, K.D., Derwand, G.: Using *k-d* trees to improve the retrieval step in case-based reasoning. In: Procs. of the 2nd European Workshop on Case-Based Reasoning, Chantilly, France (1994) 167–181
31. Schaaf, J.W.: Fish and shrink. A next step towards efficient case retrieval in large-scale case bases. In: Procs. of the 3rd European Workshop on Case-Based Reasoning, Lausanne, Switzerland (1996) 362–376
32. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning **23** (1996) 69–101
33. Kubat, M., Widmer, G.: Adapting to drift in continuous domains. In: Procs. of the 8th European Conference on Machine Learning, Heraclion, Crete (1995) 307–310
34. Salganicoff, M.: Tolerating concept and sampling shift in lazy learning using prediction error context switching. Artificial Intelligence Review **11** (1997) 133–155
35. Klinkenberg, R., Joachims, T.: Detecting concept drift with support vector machines. In: Procs. of the 17th International Conference on Machine Learning, San Francisco, CA (2000) 487–494
36. Klinkenberg, R.: Learning drifting concepts: Example selection vs. example weighting. Intelligent Data Analysis **8** (2004) 281–300
37. Aha, D.W., Kibler, D., Albert, M..K.: Instance-based learning algorithms. Machine Learning **6** (1991) 37–66
38. Kuncheva, L.I.: Classifier ensembles for changing environments. In: Procs. of the 5th International Workshop on Multiple Classifier Systems, Italy (2004) 1–15