Classifying High-Dimensional Text and Web Data using Very Short Patterns

Hassan H. Malik and John R. Kender
Department of Computer Science, Columbia University
New York, NY 10027
{hhm2104,jrk}@cs.columbia.edu

Abstract

In this paper, we propose the "Democratic Classifier", a simple, democracy-inspired pattern-based classification algorithm that uses very short patterns for classification, and does not rely on the minimum support threshold. Borrowing ideas from democracy, our training phase allows each training instance to vote for an equal number of candidate size-2 patterns. Similar to the usual democratic election process, where voters select candidates by considering their qualifications, prior contributions at the constituency and territory levels, as well as their own perception about candidates, the training instances select patterns by effectively balancing between local, class, and global significance of patterns. In addition, we respect "each voter's opinion" by simultaneously adding shared patterns to all applicable classes, and then apply a novel power law based weighing scheme, instead of making binary decisions on these patterns.

Results of experiments performed on 121 common text and web datasets show that our algorithm almost always outperforms state of the art classification algorithms, without requiring any dataset-specific parameter tuning. On 100 real-life, noisy, web datasets, the average absolute classification accuracy improvement was as great as 9.4% over SVM, Harmony, C4.5 and KNN. Also, our algorithm ran about 3.5 times faster than the fastest existing pattern-based classification algorithm.

1. Introduction and Motivation

Machine learning algorithms like SVM, C4.5 and kNN are among the most successful and widely used classification algorithms. Additionally, a number of rule-based (i.e., rule-induction-based, association-based, or frequent-pattern-based) algorithms have achieved initial success on a variety of classification problems. We identified two major problems with existing rule-based classification algorithms. These problems are discussed in the next two sections.

1.1. Minimum support and long patterns

The classification model in a typical rule-based classification algorithm consists of frequent patterns that form classification rules. The patterns are obtained by applying various pruning heuristics to reduce a very large search space. Minimum support threshold is the most common of these heuristics, and is widely used [5, 7, 9, 10] as the primary means to filter a significant percentage of candidate patterns (i.e., with a second measure such as confidence, *Information Gain*, or *Chi-Square* used for further filtration). Setting a good value for this threshold is non-trivial. A high minimum support may miss important patterns, and may also risk having some training instances unrepresented altogether (i.e., with no rules in the resulting classification model to cover such training instances), especially on unbalanced datasets. On the other hand, a small value may result in discovering a large number of noisy patterns. Considering these issues, one must question the usefulness of minimum support as the primary means to identify patterns for classification.

Additionally, on high-dimensional datasets, the number of candidates considered, as well as the number of frequent patterns found may significantly increase with the pattern size, especially when a low minimum support is used. Consequently, mining long patterns might take significantly more computational time as compared to mining short patterns. Since long patterns are always derived from short patterns, we are motivated to explore if high-dimensional datasets can be effectively classified using only the short patterns.

1.2. Three levels of pattern significance

A number of existing rule-based classification algorithms [2, 3, 4] follow a greedy rule-induction process to discover classification rules. In these algorithms, rules are discovered one rule at a time, and instances covered by the newly discovered rule are eliminated from the training set, which may degrade the quality of discovered rules as the training process advances [9], because of incomplete information. Furthermore, this process may need to be repeated for each class, negatively impacting the runtime of these algorithms. On the other hand, association-rule-based classification algorithms like [5, 7] first mine globally significant patterns, and then follow a sequential covering paradigm to select the final set of rules. Because of their inherent dependencies on minimum support and confidence thresholds, these algorithms may find too many or too few rules, and may still not cover some of the training instances. Therefore, we consider these algorithms "non-democratic".

With Harmony, Wang and Karypis [9] proposed a more effective, instance-centric approach to mine classification rules. Harmony builds the classification model by directly mining some user-defined number of highest-confidence rules for each training instance that satisfy minimum support. Furthermore, rules for all classes are mined simultaneously, and one of the user-configurable, local item ranking schemes (i.e., correlation coefficient ascending order) takes both the class and global item supports in to account. Experimental results in [9] show that Harmony, when further tuned with a suitable minimum support value for each dataset, outperformed existing rule based classification algorithms, and achieved classification accuracies that are comparable to SVM. These findings are consistent with our own classification experiments.

We observe that a labeled collection of training instances provides three important pieces of information about each pattern in a categorical dataset: first, the global frequency of the pattern; second, the frequency of the pattern in each applicable class; and third, the frequencies of atomic sub-patterns in individual training instances that contain the whole pattern. These three pieces of information can be used to evaluate the pattern significance at various levels. Unfortunately, none of the existing rule-induction-based, association-based, and frequent-pattern-based classification algorithms fully utilize all of these three levels of information. As discussed above, most of the existing algorithms only consider global significance (i.e., global support, confidence, entropy, or *Information Gain*), while others may estimate the global significance using incomplete information. Note that even though some of the widely used measures like entropy select patterns that are significant across all classes, they might not help in selecting a pattern with respect to a specific class (i.e., Section V(A) of [9] provides an example). Harmony fully utilizes the global significance, and partially utilizes the class significance of each pattern, but does not utilize the local significance. Since Harmony ensures that each training instance is covered by the selected patterns, but does not consider pattern significance with respect to individual training instances while selecting patterns, we consider Harmony "semi-democratic".

1.3. The "Democratic Classifier"

We observe that the problem of finding patterns for classification shares some similarities with the problem of electing public representatives in a human society. The typical election process in a human society involves dividing the territory (i.e., a country) into smaller constituencies (i.e., states or provinces). Each "voter" is allowed to cast their vote(s) in the constituency (or constituencies, in some cases) of its residence. The voter is presented with a list of candidates, and the voter selects a candidate (or candidates) from the list. Through a formal election campaign, the candidates communicate their qualifications and prior achievements at both the constituency and the territory level, hoping to influence their voters' decision. Still, an individual voter may be biased by its own perception about each candidate.

In terms of finding patterns for classification, the training phase may be considered analogous to the election process, where the training set is the territory, divided in to smaller constituencies (i.e., classes). Each training instance represents a voter, and the set of candidates in a constituency consists of all patterns that exist in any instance that belongs to the constituency. Each candidate's prior contributions and qualifications at the territory and constituency levels are represented by a pattern's global and class significance values, respectively. Finally, a pattern's local significance represents the voter's perception about the candidate. A local significance value of zero means that the voter does not have any opinion about the candidate (i.e., the instance does not contain the pattern).

Considering that in spite of its problems (the details of which are out of scope of this paper), democracy [18] is the most widely adopted method of electing public representatives, we adopt a democratic pattern selection scheme in this paper. More specifically, we adopt the "open list" method in the "proportional representation" scheme [20], which allows each voter to select up to *k* candidates from the candidate list.

After pre-processing training instances (Section 2.1) to eliminate less-significant features, the "Democratic Classifier" builds a classification model (Section 2.2) that contains a list of very short (i.e., size-1 and 2) patterns for each class, and allows patterns to appear in multiple classes. On an instance by instance basis, each training instance contributes to the classification model by first adding all of its size-1 patterns to the pattern-lists of its classes (i.e., the voter's constituencies of residence), and then, by "voting" for *k* (where *k* is a user-defined value) size-2 patterns, each of which is also added to the pattern-lists of applicable classes. The "voting" process selects top *k* patterns for each training instance in a way that provide an effective balance between local (i.e., voter's perception of the candidate), class (i.e., candidate's qualifications and prior contributions at the constituency level), and global (i.e., candidate's qualifications and prior contributions at the territory level) significance. We use the local pattern frequencies to determine local significance, and a contingency table-based interestingness measure to calculate class and global significance values.

All patterns in the classification model (i.e., pattern lists for each class) are then assigned an initial "pattern weight". For this purpose, we use the global support values for atomic (i.e., size-1) patterns, and the global interestingness values of size-2 patterns. These weights are first normalized using z-score standardization (with more "importance" given to size-2 patterns), and then adjusted with respect to pattern significance within the class, using a novel, power law based weight adjustment scheme. These weights are later used to calculate class scores in the classification phase.

Test instances are classified (Section 3) by first identifying all patterns in the test instance that also exist in the classification model, and then applying a scoring function to calculate class scores. Our scoring function considers both the pattern weights in the classification model, and the local pattern significance in the test instance. For single-label problems, the class with the highest score is selected, and for multi-label problems a weighted dominant factor-based scheme similar to [9] is used to select multiple classes.

In ten-fold cross-validated results of experiments performed on 121 common benchmark datasets, we show in Section 4.1 that our algorithm resulted in classification accuracies that are better than, or comparable to state of the art classifiers. On 100 real-life web datasets, our algorithm significantly outperformed all existing classification algorithms, and achieved classification accuracies that rival human experts (i.e., 95% as an average). Furthermore, unlike existing classification algorithms, where dataset-specific parameter tuning is necessary to achieve high classification accuracies, we show that our fixed parameters are robust across datasets. As an example, results reported in Section 4.1 use the same parameter values across all 121 datasets. Still, we achieve overall classification accuracies that are comparable to, or better than fully tuned existing classification algorithms. Finally, we show in Section 4.2 that our algorithm ran about 3.5 times faster than the state of the art pattern-based classification algorithm.

For completeness, we note in passing that with this research, we always intended to find a pattern-based classification algorithm that yields superior classification results on text and web data. Much like the evolution of human history itself, our research converged to a simple democratic solution after many iterations and optimizations, allowing us to connect our final solution to the powerful analogy of democracy.

2. Training the Classifier

In this section, we provide details on training the "Democratic Classifier". We first discuss our simple-yet-effective dimensionality reduction scheme, and then describe various steps involved in building the classification model.

2.1. Dimensionality reduction

Studies [16, 17] show that reducing the dimensionality of the feature space may significantly improve the effectiveness and scalability of traditional classification algorithms, especially on high-dimensional datasets. Furthermore, dimensionality reduction tends to reduce overfitting [17]. Pattern-based classification algorithms equally benefit from dimensionality reduction, as both the quality and the number of non-atomic patterns discovered directly depends on the initial, atomic patterns (i.e., 1-itemsets).

Typically, features are selected by first sorting all available features in terms of their significance, and then selecting top-*n*, or top-*n*-percent features (with a caveat that selecting a suitable value for *n* is not straightforward). A recent study [16] evaluated various measures to calculate feature significance and concluded that *Information Gain*, *Chi-Square* and *Bi-normal Separation* worked equally well on a number of datasets, with no statistically significant difference.

Unfortunately, selecting top-*n* features alone may leave some training instances with no features, which also eliminates all aize-2 pattern candidates available to these instances for later selection, as in Section 2.2. Furthermore, the optimal number (or percentage) of features (i.e., the value of *n*) needed to achieve good classification results remains unclear. The literature

[17] is inconclusive on *n*: some studies suggest that the number of selected features should be same as the number of training examples, while others suggest that feature selection may make matters worse, especially when the number of available features is small.

Considering these issues, in a way similar to [32], we adopt a three-step heuristic feature selection method that uses the number of training instances, and the number of available features to automatically estimate *n*, and also ensures that the final set of selected features covers all training instances.

***Step 1 (calculate n)*:**

$$n = i + \left(i \times \log \frac{f}{i} \right)$$

Where i = number of training instances, and f = total number of available features. This empirically derived formula ensures a reasonable base amount for low dimensional datasets, while moderately growing this number for high dimensional datasets.

***Step 2 (select globally significant features)*:** Sort all features in decreasing order of their *Information Gain* values, and then add the resulting top-*n* features to set *S* (i.e., the set of "selected" features).

***Step 3 (ensure local coverage)*:** First find all training instances with less than *t* features in *S* (i.e., instances not properly covered by the selected features), and then process these instances, on an instance by instance basis. Sort all features in the current instance in the decreasing order of their (TF * *Information Gain*), where TF = Term Frequency, calculated in the usual way. This "balances" the local significance (i.e., TF) and the global significance (i.e., *Information Gain*). Finally, add the resulting top-*t* features to set *S*. The experiments in this paper used an empirically selected fixed value of $t = 10$.

2.2. Building the classification model

Typical rule-based classification algorithms associate each selected rule (or pattern) to a single class. In reality, a large percentage of patterns may appear in many training instances that might not be associated with the same class. Table 1 contains a training dataset used as a running example throughout this section. Pattern {b, d} appears in six training instances in this example. Two of these training instances (i.e., T1 and T10) are associated with class 0 whereas the other four are associated with class 1. Associating this pattern to only one of these classes might not fully capture its significance in the training set. Instead of making such a binary decision, or eliminating these "shared" patterns as "confusing" or "insignificant", we borrow ideas from some of the world's democracies that allow candidates to be elected from multiple constituencies, and allow patterns to appear in multiple classes, with weights (i.e., described below) representing their significance in each applicable class.

Table 1. An example training set, feature selected

Instance ID	*Feature-frequency pairs*	*Class ID*
T1	(a:2), (b:4), (d:1)	0
T2	(a:3), (c:1), (d:6), (e:1)	0
T3	(b:2), (c:3), (d:1)	1
T4	(b:3), (c:1), (d:2), (e:4)	1
T5	(b:7), (c:2), (d:1)	1
T6	(a:1), (b:1), (c:1), (e:1)	0
T7	(b:9), (c:3), (f:4)	1
T8	(c:6), (d:2)	0
T9	(b:3), (d:2), (e:6)	1
T10	(a:4), (b:2), (d:7), (f:3)	0
T11	(c:1), (e:1), (f:1)	1

Additionally, training instances in real-life text and web datasets may contain a feature (i.e., atomic pattern) more than once. These local feature frequency counts are largely ignored by existing algorithms (such as Harmony [9]) that only considers binary presence or absence of features in training instances to select patterns used for classification. Similar to the democratic election process where a voter's perception about each candidate may significantly impact their selection, these local feature frequencies may provide useful insights about a pattern's significance with respect to a training instance. As an example, we consider a recent news article at cnn.com about certain types of dinosaurs that are believed to be good swimmers. The word "dinosaurs" occurs 19 times in the entire article whereas the word "marine" occurs only once. Clearly, considering both of these words with equal importance can be problematic. Therefore, by accommodating local frequencies, our training algorithm achieves a balance between global, class, and local significance. Note that considering features with high local frequencies is not the same as considering features with high support.

Figure 1 presents our training algorithm. After selecting features and initializing the classification model, training instances are processed, one instance at a time. Each training instance first adds all of its size-1 patterns (i.e., patterns remaining after feature selection) to the pattern-lists of all of its applicable classes (i.e., the voter's constituencies of residence), with global support used as the initial pattern weight (line 6).

Next, each size-2 pattern is processed (lines 10-23) to compute the "overall" pattern significance with respect to the current training instance, considering the pattern significance at all three (i.e., local, class, and global) levels. We determine the local pattern significance (line 11) by averaging the TF values of both the atomic patterns (i.e., p_1 and p_2) in the size-2 pattern (i.e., p).

```
01) build-model(training_set, k, measure)
02)  {select features as explained in Section 2.1}
03)  model = $\Phi$
04)  forall training instances $t \in$ training_set do begin
05)      forall atomic patterns $p \in t$ do begin
06)          weight(p) = support(p, training_set)
07)          append (p, each applicable class in model)
08)      end
09)      list = $\Phi$
10)     forall size-2 patterns $p \in t$ do begin
11)          $significance_{local}$ = average(TF($p_1$), TF($p_2$))
12)          class_significance_list = $\Phi$
13)          forall class c $\in$ p do begin
14)              append(class_significance_list,
15)                      interestingness(p, measure, training_set, c)
16)          end
17)          $significance_{class}$ = average(class_significance_list)
18)          $significance_{global}$ =
19)                      interestingness(p, measure, training_set)
20)          significance(p) = $significance_{local}$*
21)                      $significance_{class}$ * $significance_{global}$
22)          append (list, p)
23)      end
24)      {sort list in decreasing order of significance(p)}
25)      for (i = 1; i <= k; i ++) do begin
26)          weight($list_i$) =
27)                      interestingness($list_i$, measure, training_set)
28)          append ($list_i$, each applicable class in model)
29)      end
30)  end
31)  {apply z-score standardization on all weights in model}
32)  forall classes $c \in$ model do begin
33)     forall patterns $p \in c$ do begin
34)          weight(p) = weight(p) * mono(support(p, c)/size(c))
35)      end
36)  end
37)  return model
38) end
```

Figure 1. Method build-model

Next, in order to determine the pattern significance at class and global levels, we use a common 2 x 2 contingency-table-based interestingness measure. A recent study [19] evaluated most of the interestingness measures found in [12, 13], in the context of hierarchical document clustering, and reported that only a small number of interestingness measures generalize well to datasets with varying characteristics. Coincidently, we found that the same measures (in a slightly different order) are useful to determine class and global significance values for pattern-based classification. Since training instances may belong to more than one class in multi-label classification problems, we determine the class significance (lines 13-17) by averaging the pattern interestingness values of all classes applicable to the current training instance.

All size-2 patterns are then sorted according to their significance values (line 24), and top-k patterns are selected (lines 25-29) to represent the training instance in the classification model, with global pattern significance used as initial pattern weight (line 26).

Example: Considering the training instance T1 in Table 1, and pattern {a, b}, we calculate the local pattern significance by averaging the TFs of atomic patterns 'a' (i.e., 2/7 = 0.285) and 'b' (i.e., 4/7 = 0.571), i.e., 0.428. The class significance of pattern {a, b} is obtained by calculating the value of the selected interestingness measure using a contingency table, formed using the frequencies of atomic patterns 'a' (i.e., 4) and 'b' (i.e., 3) in class 0, where $N = 5$ (i.e., number of instances in class 0), in the usual way [12]. Similarly, the global significance of pattern {a, b} is obtained by calculating the value of the selected interestingness measure using a contingency table, that considers the frequencies of atomic patterns 'a' (i.e., 4) and 'b' (i.e., 8) in the whole training set, where $N = 11$ (i.e., the total number of instances in the training set).

It is important to note that weights assigned to size-1 and size-2 patterns do not lie on the same scale. This is an artifact of their methods of calculation, rather than their relative importance. We investigated ways of normalizing these weights, and found that the simplest way is to use z-score standardization. Realizing that z-score standardization assumes a normal distribution, which might not be true in some cases, we leave investigating a more robust technique for future work.

Furthermore, based on our empirical observation that size-2 patterns are more important than size-1 patterns, we scale down the weights of size-1 patterns (i.e., by a factor of 4, which again performs robustly).

Finally, we adjust normalized weights of patterns assigned to each class (lines 32-36) with respect to the class size and pattern support in the class, using a monotonically increasing weight adjustment scheme. We evaluated various monotonically increasing functions for this purpose, and empirically found that the best classification results are achieved when mono(x) = x^p, with $0.05 <= p <= 0.10$. We fix this value to 0.07 (line 34) for all experiments in this paper.

Note that the final form of our pattern weighing scheme was obtained by evaluating many alternatives, including one that used class interestingness instead of global interestingness. We found that these class-specific values are highly unstable, especially on datasets with a high-degree of class imbalance, and are not suitable to be used globally (i.e., to compare significance across classes). Therefore, these values are only used in the more meaningful context of selecting top-*k* patterns for training instances (lines 10-30).

3. Classifying Test Instances

Once the classification model is available, it can be used to classify previously unseen, unlabeled (test) instances by the following three-step process:

Step 1: Given a test instance *t*, and *model*, identify the set of common patterns *CP* (i.e., patterns that exist in both *t* and *model*).

Step 2: Use patterns in *CP* and a scoring function (below) to obtain scores for all classes in *model*.

Step 3: For single-label problems, select the label of the class with the highest score. For multi-label problems, select multiple classes using the "weighted dominant factor-based" scheme in Section V(C-3) of [9], except replacing all uses of confidence with the selected interestingness measure.

The scoring function: Given the set of common patterns *S* (i.e., step-1 above), and a class *c*, our scoring function uses all (i.e., size-1 and size-2) patterns in *S* that also exist in the pattern list of class *c* in *model*, to calculate the score of class *c* with respect to the test instance:

$$Score(S,c)=\sum_{i=1}^{|S|}\begin{cases} TF_{S_i}\times weight(S_i,\mathrm{model}_c) & if\, S_i\in \mathrm{model}_c \\ 0 & if\, S_i\notin \mathrm{model}_c \end{cases}$$

where TF is the term frequency of pattern S_i in the test instance for size-1 patterns, and the average of the TF values of both atomic patterns in S_i for size-2 patterns. The idea of using local pattern frequencies here is similar to the idea of using local pattern significance in our training phase, which aims to capture the notion of a voter's bias towards each candidate.

4. Experimental Results

We conduced an extensive experimental study, and evaluated the performance of our algorithm on 121 text and web datasets, with varying characteristics. For each dataset, we compared the classification results obtained by our algorithm against various state of the art classification algorithms. In order to ensure a fair comparison we obtained data from the same sources and used the same evaluation metrics as used by the existing classifiers. We do not report the details of datasets used in our experiments here and refer the reader to [9, 15, 21, 27].

4.1. Classification performance

We first evaluated the effectiveness of various interestingness measures [12, 13], to determine global and class significance values (i.e., Section 2.2) on a number of datasets, and found that the top measures reported in [19], in the context of hierarchical document clustering, also consistently performed well in our context (in a slightly different order). We observe that *Added Value* generally outperformed other measures, while *Mutual Information*, *Chi-Square*, and *Yule's Q* achieved very close (i.e., within a few-percent range) classification performance. For the reason of space, we do not compare the relative performance of these measures here and note that all results reported in this section used *Added Value* as the interestingness measure, with *k* fixed to 25.

Additionally, all results reported here used the 10-fold cross validation scheme (with averages of all 10 experiments reported, as usual), except on Reuters-21578 dataset, where we used the ModApte split [21] to ensure an apples-to-apples comparison with results reported by existing studies.

4.1.1. Reuters-21578 (ModApte) text dataset. Reuters-21578 is the most-commonly used benchmark dataset to evaluate the performance of multi-class, multi-label classification algorithms. Existing studies like [9, 22] used the micro-averaged, precision-recall breakeven points on 10-largest categories, as the main performance criteria. We calculated these breakeven points in a way similar to [9], i.e., by changing the dominant factor, and keeping a fixed "score differentia factor" (i.e., 0.8). Note that we already fixed the interestingness measure to Added Value and *k* to 25.

Table 2. Breakeven performance on Reuters-21578

Category	***Harmony***	***Find Sim***	***Naïve Bayes***	***Bayes Nets***	***Trees***	***SVM (linear)***	***ARC-BC***	***Democratic***
acq	**95.3**	64.7	87.8	88.3	89.7	93.6	90.9	95.1
corn	78.2	48.2	65.3	76.4	**91.8**	90.3	69.6	72.5
crude	85.7	70.1	79.5	79.6	85.0	88.9	77.9	**91.6**
earn	**98.1**	92.9	95.9	95.8	97.8	98.0	92.8	96.4
grain	91.8	67.5	78.8	81.4	85.0	**94.6**	68.8	91.9
interest	77.3	63.4	64.9	71.3	67.1	77.7	70.5	**84.2**
money-fx	80.5	46.7	56.6	58.8	66.2	74.5	70.5	**89.0**
ship	**86.9**	49.2	85.4	84.4	74.2	85.6	73.6	85.4
trade	**88.4**	65.1	63.9	69.0	72.5	75.9	68.0	87.2
wheat	62.8	68.9	69.7	82.7	**92.5**	91.8	84.8	73.3
micro-avg	92.0	64.6	81.5	85.0	88.4	92.0	82.1	**92.6**

Table 2 presents the results of this experiment. The results for Find-Sim, Naïve Bayes, Bayes-Nets, Trees (i.e., Decision-Trees), and linear-SVM are obtained from [22], while the results for ARC-BC are obtained from [23]. Note that [9] also used the same results. Finally, the results for Harmony are obtained from Table VIII of [9]. Among the ten-categories, our algorithm achieved the best break-even performance on 3 categories (i.e., crude, interest and trade), and ranked second on another 3 categories (i.e., acq, ship and trade) with ranks 3-5 achieved on the remaining 4 categories. Most importantly, our algorithm outperformed all existing classification algorithms in terms of micro-average performance.

4.1.2. Text datasets. From Table 2, we observe that linear SVM and Harmony outperformed other classification algorithms on the Reuters-21578 dataset. We performed additional experiments on 20 standard text datasets, and evaluated the 10-fold cross validated classification accuracies achieved by our "Democratic Classifier" against these two classification algorithms. Data sets tr11, tr12, tr23, tr31, tr41, tr45, fbis, hitech, la1, la2, la12, and sports are derived from TREC-5, TREC-6, and TREC-7 collections [28]. Data sets re0 and re1 are from Reuters-21578, obtained by removing the relatively easy to classify dominant classes such as learn and acq, and by splitting the remaining classes into two sets. The Classic4 dataset is obtained by combining CACM, CISI, CRAN, and MED abstracts. Datasets k1a, k1b and wap are from the WebACE project, and dataset ohscal was derived from the Ohsumed collection. All of these datasets are available as part of the Cluto clustering toolkit [27].

Table 3. Classification accuracies on text datasets

	Linear SVM (tuned for C)	***Harmony (tuned for support)***	***Democratic (untuned)***
classic4	76.45	**94.17**	91.73
fbis	76.70	77.96	**78.52**
hitech	70.28	67.08	**72.27**
k1a	76.50	76.54	**78.03**
k1b	75.30	**97.79**	94.10
la1	78.15	83.46	**85.39**
la2	80.16	83.53	**87.84**
la12	78.93	85.37	**86.41**
mm	97.30	95.58	**98.93**
ohscal	76.66	**76.91**	73.43
re0	76.20	78.33	**80.92**
re1	75.01	78.02	**82.02**
sports	95.79	94.90	**97.12**
tr11	81.91	84.55	**86.22**
tr12	84.95	81.16	**89.81**
tr23	84.31	87.74	**94.10**
tr31	95.25	96.12	**97.20**
tr41	91.46	92.14	**94.31**
tr45	90.15	**91.04**	90.58
wap	74.55	72.62	**75.51**
average	81.80	84.75	**86.72**

We used the SVM light [29] implementation of linear SVM, and used various values of C (i.e., 0.1, 0.5, 1.0, 1.5, 2.0, 2.5) to tune linear SVM on each dataset. We report the best 10-fold cross validated classification accuracy achieved on each dataset in Table 3. As the authors of Harmony [9] have also noted, we found other SVM kernels (such as RBF) impractical for large text datasets, because of their extensive computational time requirements.

Similarly, we obtained Harmony executables from the first author of [9], and tuned Harmony with various values of minimum support (i.e., 25, 50, 75, and 100) on each dataset. We report the best 10-fold cross validated classification accuracy achieved on each dataset in Table 3. Note that it was not always possible to execute harmony on each dataset for all of these minimum support values (see Section 4.2 for details).

In contrast, our Democratic Classifier used the same fixed parameter values (i.e., *Added Value* as interestingness measure, and $k = 25$) on all 20 datasets. In addition, we only used the top-scoring class for each test instance to calculate the classification accuracies on these single-label datasets.

From Table 3, we observe that the Democratic Classifier, without any parameter tuning, resulted in the highest classification accuracies on 16 out of 20 datasets, and was very competitive on the remaining 4 datasets. Most importantly, the Democratic Classifier achieved the highest average classification accuracy across all 20 datasets. Note that tuning our classifier on each dataset with various values for k and interestingness measure improved the classification accuracies even further (we noticed up to 5% improvement on some datasets). However, we consider extensive parameter tuning to be less meaningful for practical purposes and omit those results.

4.1.3. TechTC-100 web datasets. In recent years, many researchers questioned the usefulness of standard news datasets, such as Reuters-21578 as realistic benchmarks for classification research. Dumais and Chen [24] state that "the Reuters collection is small and very well organized compared with many realistic applications". Scott [25] noted that the Reuters corpus has a very restricted vocabulary, since Reuters in-house style prescribes using uniform unambiguous terminology to facilitate quick comprehension. Considering these issues, a recent study [14] followed a systematic process to produce a new, more realistic collection of 100 benchmark datasets, called TechTC-100 [15]. These datasets are generated using real web-sites, classified by human editors as part of the open directory project [26]. Furthermore, TechTC-100 datasets are very noisy, and high-dimensional (i.e., an average of 18,073 features in each dataset, where the average number of instances is only 149), with categorization difficulties uniformly distributed between 0.6 and 0.92 [15].

Table 4 compares the average classification accuracies achieved by our classifier against SVM, C4.5, K-Nearest-Neighbor (i.e., KNN), and Harmony, on all of the TechTC-100 datasets. We obtained results of SVM, C4.5 and KNN from [16] (and reported at [15]). These results represent the performance of each of the three classifiers at their respective optimal feature selection levels. Note that the un-tuned accuracies of all three classifiers reported at [15] are much lower (i.e., an average of 77% for SVM).

Table 4. Average classification accuracies on all TechTC-100 datasets

SVM (tuned)	*C4.5 (tuned)*	*KNN (tuned)*	*Harmony (tuned)*	*Democratic (untuned)*
85.3	84.3	82.7	85.8	**95.2**

We tuned Harmony on each dataset, using various kRules (i.e., 1, 3, 5, 10), and minimum support (i.e., 10, 13, 15, 20, 25, 30) values. We recorded the best tuned 10-fold cross-validated accuracies for each dataset, and report the average in Table 4. Note that the classification accuracies using the same set of parameters for all datasets peaked at about 83.4%, with kRules = 5, and min support = 25.

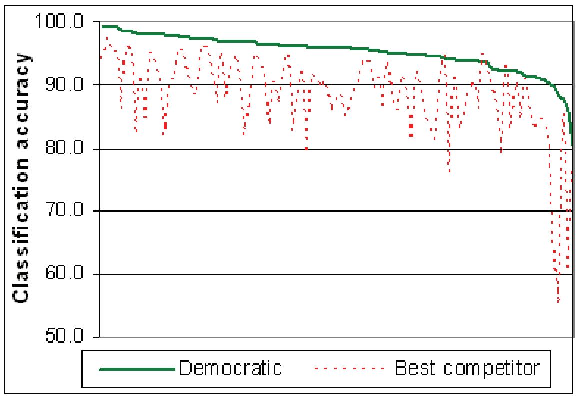

Figure 2. Classification accuracies on TechTC-100

In contrast, our algorithm used the same fixed parameter values on all datasets, and we report the overall average accuracy in Table 4. In addition, we report classification accuracies of our classifier, and the best competitor on each of the TechTC-100 datasets in Figure 2. For the sake of clarity, these accuracies are sorted in the decreasing order of our accuracies across all datasets.

From Figure 2, we observe that our algorithm, without any parameter tuning, outperformed existing classification algorithms with a very significant margin (i.e., an average of 9.4%). Our algorithm was better than all other algorithms on 97 out of 100 datasets, and was ranked second on the remaining 3 datasets. We believe that this happens because of four main reasons. First, the noise-level on these real-life datasets is quite high, and our novel, voting-based pattern selection method is less sensitive to noise, because we select patterns that provide an effective balance between local, class, and global significance. Second, as we noted in Section 2.2, the local pattern significance values can be very important for web datasets, something that is ignored by most of the existing algorithms. Third, a large number of TechTC-100 datasets contain patterns that are shared across classes (especially on datasets with closely-related categories); our unique democracy-inspired pattern assignment scheme allows these patterns to appear in multiple classes, with weights adjusted according to their class significance, whereas most of the existing algorithms make binary decisions on these patterns that may be sub-optimal. Finally, most datasets in this collection are relatively balanced, and democracy is known to work well with balanced constituencies.

4.2. Runtime performance

Since our classifier is most similar to Harmony [9] in that Harmony also builds a classification model directly from patterns mined from the training set, we compare the runtime performance of our classifier against Harmony in this section. For fairness, we note that even though Harmony is shown to run orders of magnitude faster than existing classifiers including linear SVM as implemented in SVM Light [29] (which we also used for experiments reported in this paper), we expect newer linear-time linear SVM implementations such as [33] to run faster than both Harmony and our classifier. Nevertheless, the accuracy gains realized by our classifier may provide a reasonable justification for considering it over SVMs.

To compare our classifier against Harmony, we executed both Harmony (as implemented by the original authors), and our Democratic Classifier on all of the TechTC-100 cross-validation datasets (i.e., a total of 1000 datasets), and summed the total training and testing times. The same dedicated machine (a 64-bit Intel Xeon based server, with Windows XP 64 professional, and 8 GB of memory) was used to execute both algorithms. Furthermore, we repeated this test using various parameter values. For Harmony, we used kRules = 3, 5, and 10, and for each kRules value, we set minimum support to 10, 15, 20, 25 and 30, yielding to a total of 15,000 (i.e., 5 x 3 x 1000) executions. On the other hand, we used k = 20, 25, and 30 for our algorithm. Figure 3 presents the results of this experiment.

For fairness, we note in passing that when minimum support was set as low as 10, many of the executions on dataset 82 caused Harmony to execute for 5+ hours, after which it had to be terminated. Upon manual investigation, we found that some training instances in this dataset are very large (i.e., large number of atomic patterns), and they cause long-pattern-based algorithms to take a very long time. As a result, we do not report the performance of Harmony at this support level.

From Figure 3, we observe that the performance of Harmony significantly depends on the parameter values used, varying by a factor of 10. Setting a low minimum-support causes Harmony to take a long time, whereas it finishes quickly on a high minimum-support value. Unfortunately, high-minimum support values may not always result in good accuracies. Furthermore, higher kRules values also causes Harmony to take more time. Note that these issues are not unique to Harmony and other minimum support based algorithms are likely to exhibit the same behavior.

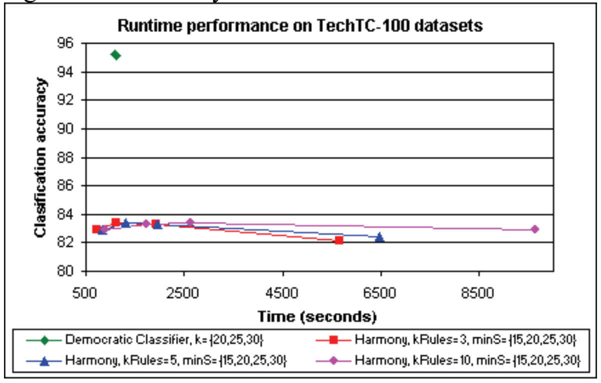

Figure 3. Total runtime and average classification accuracies of Harmony at various support levels, with kRules = {3, 5, 10}, and our classifier, with k = {20, 25, 30} on TechTC-100 datasets

In contrast, the performance of our algorithm did not notably vary with *k* values. This happens because in our algorithm, each training instance first calculates significance scores of all of its size-2 patterns, and then selects top-*k* patterns. As a result, a slightly higher value for *k* only adds a negligible amount of work.

Table 5. Total runtime of Harmony with optimal supports, and our classifier with k = 25 on TechTC-100

Harmony (optimal support)	*Democratic Classifier (k=25)*
3846 seconds	1096 seconds

Table 5 reports the total Harmony execution times using the parameters that resulted in the best classification accuracy values on each dataset (i.e., parameters used for the results reported in Table 4). Table 5 also reports the total execution times of our algorithm using our fixed parameter values. We observe that our algorithm ran about 3.5 times faster. Harmony took an average of 3.9 seconds to train and test a dataset (which is still very fast) whereas our algorithm took an average of 1.1 seconds only. The primary contributor towards this significant difference is the fact that we use very short patterns.

5. Related Work

Our work relates to existing rule and pattern-based classification algorithms, with several important differences. Rule-induction-based classifiers like FOIL [2], RIPPER [3], CPAR [4] and C4.5 [1] use heuristics such as *Gini Index* and *Information Gain* (or *Information Gain* variants), to identify the best literal by which to grow the current rule [9], and many of them follow the sequential covering paradigm. In contrast, association rule-based classifiers such as CBA [5], CAEP [6], CMAR [7], ARC-BC [8], and DeEPs [11] first mine a large set of association rules that satisfy user-defined support and confidence thresholds, and then extract the final set of classification rules by following a database covering technique.

Our algorithm is similar to these algorithms in that we also use patterns. But unlike rule-induction based algorithms, we do not discover one rule at a time, and unlike association-based algorithms, we do not have a global pattern mining step. Instead, we directly find "balanced" patterns from each training instance, for all applicable classes simultaneously.

In these aspects, our approach resembles Harmony [9], which follows an instance-centric approach that mines at least one highest-confidence rule for each training instance. However, our approach differs from Harmony in many ways. First, Harmony uses a minimum support threshold, which is difficult to reuse across datasets (Section 4.1), and may also result in a classification model that do not cover some of the training instances. In contrast, by having each training instance "vote" for top-*k* "balanced" patterns, our approach guarantees that the resulting classification model covers each training instance. We show in Section 4.1 that our fixed parameter *k* performs robustly. Second, Harmony primarily uses global pattern significance, and partially uses the class significance in a local item ranking scheme whereas we balance local, class, and global pattern significance. Third, we replace confidence with a contingency table-based interestingness measure. Fourth, Harmony does not impose any limits on pattern length, whereas we use very short patterns, resulting in significant performance improvements without sacrificing accuracy. Fifth, our score calculation method also considers local significance of patterns in test instances.

We observe that our algorithm shares some similarities with RCBT [10], a pattern-based classification algorithm that achieved high accuracy on gene expression data. For each row in the training set of the gene expression profiles, RCBT finds top-*k* rule groups for the corresponding class, and does not use a minimum confidence threshold. Still, it uses a minimum support threshold, and relies on confidence as a significance measure. Additionally, RCBT imposes no limits on pattern lengths, and training and test phases in RCBT do not utilize local and class significance of patterns.

We finally note that recent work in augmenting training data with discriminative frequent patterns [30, 31], originally applied to low-dimensional numerical UCI datasets, is also related to our research, and we intended to compare our Democratic Classifier with this approach. Unfortunately, the executables that we obtained from the authors of [30, 31] did not work on high-dimensional text datasets used in this paper. Therefore, we leave this comparison for future work.

6. Conclusions and Future Work

We proposed a democracy-inspired, short-pattern-based classification algorithm in this paper. In addition to size-1 patterns, our algorithm selects top-*k* size-2 patterns to represent each training instance, that provide an effective balance between local, class and global significance. Our novel pattern assignment scheme allows patterns to appear in the classification model of multiple classes, with a unique, power law based scheme used to adjust pattern weights. Furthermore, our algorithm replaces hard-to-generalize minimum support and confidence thresholds with *k* and an interestingness measure, parameters that are robust across datasets. With ten-fold cross-validated results of experiments performed on 121 datasets, we show that our algorithm achieves overall classification results that are better than many well known classification algorithms, with most significant gains realized on real-life, noisy, web datasets. In addition, our algorithm ran about 3.5x faster than the fastest existing pattern-based classification algorithm.

In the future, we plan to investigate a more robust scheme to replace z-score standardization, and to investigate better ways of assigning weights to size-1 patterns. We also intend to apply our algorithm in other domains.

7. Acknowledgements

The authors would like to acknowledge that Harmony served as the initial inspiration for this research, and thank Jianyong Wang for providing us the executables. We would also like to thank the anonymous reviewers from ICDM'08 for their useful feedback.

7. References

[1] J. Quinlan, "C4.5: Programs for Machine Learning", *Morgan Kaufman*, ISBN:1-55860-238-0, 1993.
[2] J. Quinlan and R. Cameron-Jones, "FOIL: A Midterm Report", In *Proc. ECML*, 1993.
[3] W. Cohen, "Fast effective rule induction", In *Proc. ICML*, 1995.
[4] X. Yin and J. Han, "CPAR: Classification based on Predictive Association *Rules*", In *Proc. SDM*, 2003.
[5] B. Liu, W. Hsu and Y. Ma. "Integrating Classification and Association Rule Mining", In *Proc. KDD*, 1998.
[6] G. Dong et al., "CAEP: Classification by aggregating emerging patterns", *Discovery Science*, 1999.
[7] W. Li, J. Han and J. Pei, "CMAR: Accurate and Efficient Classification based on multiple class-association rules", In *Proc. ICDM*, 2001.
[8] M. Antonie and O. Zaiane, "Text Document Categorization by Term Association", In *Proc. ICDM*, 2002.
[9] J. Wang and G. Karypis, "On Mining Instance-Centric Classification Rules", *IEEE TKDE*, 2006.
[10] G. Cong et al., "Mining Top-k Covering Rule Groups for Gene Expression Data", In *Proc. SIGMOD*, 2005.
[11] J. Li, G. Dong, K. Ramamohanarao and L. Wong, "DeEPs: A New Instance based Discovery and Classification System", *Machine Learning*, 54(2), 2004.
[12] P. Tan, V. Kumar, and J. Srivastava, "Selecting the right interestingness measure for association patterns", In *Proc. SIGKDD*, 2002.
[13] L. Geng and H. J. Hamilton, "Interestingness Measures for Data Mining: A Survey", *ACM Computing Surveys*, 38(3), September 2006.
[14] D. Davidov, E. Gabrilovich, and S. Markovitch, "Parameterized Generation of Labeled Datasets for Text Categorization Based on a Hierarchical Directory", *In Proc. ACM SIGIR*, 2004.
[15] E. Gabrilovich, "The TechTC-100 Test Collection for Text Categorization", *Technion Inst. of Technology*, available at http://techtc.cs.technion.ac.il/techtc100/techtc100.html.
[16] E. Gabrilovich and S. Markovitch, "Text Categorization with Many Redundant Features: Using Aggressive Feature Selection to Make SVMs Competitive with C4.5", In *Proc. ICML*, 2004.
[17] F. Sebastiani, "Machine learning in automated text categorization", *ACM Computing Surveys*, 34(1), 2002.
[18] Democracy, http://en.wikipedia.org/wiki/Democracy.

[19] H. H. Malik, and J. R. Kender, "High Quality, Efficient Hierarchical Document Clustering Using Closed Interesting Itemsets", In *Proc. ICDM 2006*, pp. 991-996.
[20] Proportional representation, Available at. http://en.wikipedia.org/wiki/Proportional_representation.
[21] S. Bergsma, "The Reuters-21578 (ModApte) dataset", *Dept. of Computer Science, University of Alberta*. Available at http://www.cs.ualberta.ca/~bergsma/HTML/Courses/650/.
[22] S. Dumais, J. Platt, D. Heckerman and M. Sahami, "Inductive Learning Algorithms and Representations for Text Categorization", In *Proc. CIKM*, 1998.
[23] M. Antonie, and O. Zaiane, "Text Document Categorization by Term Association", In *Proc. ICDM*, 2002.
[24] S. Dumais and H. Chen, "Hierarchical classification of web content", In *Proc. SIGIR*, pp. 256–263, 2000.
[25] S. Scott, "Feature engineering for a symbolic approach to text classification", *Master's thesis*, U. Ottawa, 1998.
[26] The Open Directory Proj., http://dmoz.org/.
[27] Cluto, http://glaros.dtc.umn.edu/gkhome/views/cluto.
[28] TREC. Text REtrieval conference. *http://trec.nist.gov*.
[29] Multi-Class Support Vector Machine, http://svmlight.joachims.org/svm_multiclass.html
[30] H. Cheng, X. Yan, J. Han and C.-W. Hsu, "Discriminative Frequent Pattern Analysis for Effective Classification", In *Proc. ICDE*, 2007.
[31] H. Cheng, X. Yan, J. Han and P. S. Yu, "Direct Discriminative Pattern Mining for Effective Classification", In *Proc. ICDE*, 2008.
[32] H. H. Malik, and J. R. Kender, "Classification by Pattern-Based Hierarchical Clustering ", In *From Local Patterns to Global Models Workshop, ECML/PKDD*, 2008.
[33] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification", *Journal of Machine Learning Research* 9(2008),