A Study on Thresholding Strategies for Text Categorization

Yiming Yang
Language Technologies Institute and Computer Science Department
Newell Simon Hall 3612D, Carnegie Mellon University
Pittsburgh, PA 15213-8213, USA
yiming@cs.cmu.edu

ABSTRACT

Thresholding strategies in automated text categorization are an underexplored area of research. This paper presents an examination of the effect of thresholding strategies on the performance of a classifier under various conditions. Using k-Nearest Neighbor (kNN) as the classifier and five evaluation benchmark collections as the testbets, three common thresholding methods were investigated, including rank-based thresholding (RCut), proportion-based assignments (PCut) and score-based local optimization (SCut); in addition, new variants of these methods are proposed to overcome significant problems in the existing approaches. Experimental results show that the choice of thresholding strategy can significantly influence the performance of kNN, and that the "optimal" strategy may vary by application. SCut is potentially better for fine-tuning but risks overfitting. PCut copes better with rare categories and exhibits a smoother trade-off in recall versus precision, but is not suitable for online decision making. RCut is most natural for online response but is too coarse-grained for global or local optimization. RTCut, a new method combining the strength of category ranking and scoring, outperforms both PCut and RCut significantly.

1. INTRODUCTION

Text Categorization (TC), the problem of assigning documents to predefined categories, is an active research area in information retrieval and machine learning. A wide range of supervised learning algorithms have been applied to this problem, using a training set of categorized documents to obtain an empirical mapping from arbitrary documents to relevant categories. This mapping is typically realized by assigning relevance scores to every document-category pair, and then thresholding on those scores to make binary decisions. Both the scoring method and the thresholding method used in a categorization system ("classifier") can influence its results significantly. However, only the scoring algorithms (k-nearest neighbor, naive Bayes, multivariate regression, decision trees, support vector machines, neural networks, boosting, etc.) have been the major focus of research in the TC literature[16, 7, 12], while thresholding strategies were often briefly mentioned as a unimportant post-processing step. The implicit assumption was either that thresholding strategies do not make much difference in the performance of a classifier, or that finding the optimal thresholding strategy for any given classifier is trivial.

Neither of the above assumptions is true. Optimal thresholding is trivial only if the classifier produces accurate probabilities $P(c_j|d_i)$ for all the categories (c_j) and documents (d_j), and if the optimization criterion is to minimize the global number of errors (misses and false alarms) in the decisions made by the system, and possibly some other conditions. Under those conditions, for a 2-category classification problem where a document belongs to one and only one category, for example, the optimal threshold is trivially $P(c_j|d_i) = 0.5$ for all categories and documents. Unfortunately, ideal classifiers producing true probabilities do not exist. Many TC systems do not produce probabilistic scores, including some state-of-the-art methods such as k-nearest neighbor (kNN), support vector machines (SVM) and boosting algorithms. Mapping from the non-probabilistic scores generated by those learning algorithms to probabilities is an open problem for research[11, 5] and would be at least as hard as the TC problem itself. Even in probabilistic frameworks, including naive and non-naive versions of Bayesian classification methods and probabilistic versions of kNN and SVM, only approximations of the true probabilities are obtained. Those approximations could be inaccurate due to independence assumptions among variables, limited and noisy training examples or tractability issues. As recently observed by Paul Bennett and Ghani et al. [1, 4, 19] independently, scores generated by naive Bayesian classifiers tend to converge exponentially to the value of zero or one when the number of features (words from the documents) used by the classifiers increases, yielding unrealistic estimates for the probabilities $P(c_j|d_i)$. Given the limitations of real classifiers, what is the optimal thresholding over non-probabilistic scores or inaccurate likelihood estimates? Is rank-based thresholding more reliable than score-based? Would per-category threshold optimization be a better choice than per-document thresholding? Answering these questions is non-trivial. Although some empirical observations were reported in the TC literature for related issues[8, 16], conclusive answers have not yet emerged and thorough investigations using state-of-the-art classifiers and a broader range of bench-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGIR'01, September 9-12, 2001, New Orleans, Louisiana, USA.
Copyright 2001 ACM 1-58113-331-6/01/0009 ...$5.00.

mark corpora for evaluation and comparison are needed to answer these questions more explicitly. The is a part of the motivation for this paper.

Another important reason for this investigation is the demand for high-precision systems in real-world applications. When searching the Web, for example, the user can only afford to read the top few documents retrieved for a query, and thus a search engine with high precision returns would be preferred to one with a high recall but low precision. Similarly, when a classifier is used to help the user to decide the categories relevant to a document (for computed-aided categorization or concept-based navigation), again only a few candidate categories can be read by the user. A natural and simple approach to building a high-precision classifier is to adjust the thresholding strategy in an existing classifier. How well this approach works is underexplored. Published examinations in the TC literature have mainly focused on optimizing performance in the range where recall and precision are balanced (around the "break-even-point" or where F_1 is optimized), and do not focus on recall-precision tradeoff with respect to alternative thresholding strategies. [1]

The following sections present an examination of the effect of thresholding strategies on the performance of a classifier under various conditions. Using kNN as the classifier and five corpora from various applications as testbeds, I observe the effects of three commonly used thresholding strategies and their new variants for improvement. While the scope of the experimental results presented is necessarily confined to the particular algorithm (kNN) and the datasets, I hope that the analysis will provide useful insights into the problem and suggest a systematic approach to the proper use of thresholding strategies in performance optimization for other classifiers.

2. THRESHOLDING STRATEGIES

Let us first visit the commonly used thresholding methods in the TC literature, then discuss their properties with respect to optimizing performance of a classifier, and introduce some new variants of these methods to address the problems in the original versions.

2.1 Common Strategies

The three commonly used thresholding strategies are RCut, PCut and SCut[8, 16]. Let m be the number of categories in the problem, n be the number of documents in the test (or validation) set, and assume one score is produced by the classifier for each document-category pair. The thresholding algorithms are then defined to be:

> **RCut** – For each document, sort categories by score and assign YES to each of the t top-ranking categories. RCut is parameterized by t whose value (an integer between 1 and m) can be either specified by the user or automatically tuned using a validation set (not a part of the training set or the test set). That is, the value of t optimizing the *global* performance of the classifier on the validation set is fixed when applying the classifier on new documents in a test set. RCut is a common approach to document filtering in TREC if the queries are considered as categories; RCut with $t = 1$ is also commonly used in the machine learning community where TC algorithms have been typically evaluated in a subset of TC problems in which a document has one and only one category[7].
>
> **PCut** – For each category (c_j), sort the test documents by score and assign a YES decision to each of the k_j top-ranking documents, where $k_j = P(c_j) \times x \times m$ is the number of documents assigned to category c_j, and $P(c_j)$ is the prior probability (estimated using a training set) for an arbitrary document to be an member of category c_j. PCut is parameterized by x (real-valued), the average number of documents the system assigns to a category; x can be any real-valued number specified by the user: when $x = n$, the system behaves like "Mr. YES", and when $x = 0$, the system behaves like "Mr. NO". The value of x is automatically tuned in the same fashion as tuning t for RCut, by varying the value of x until the *global* performance of the classifier is optimized on the validation set. PCut was used in several published evaluations of probabilistic classifiers, Naive Bayes, DTree, kNN and the LLSF regression methods[8, 9, 16].
>
> **SCut** – Score a validation set of documents for each category and tune the threshold over the local pool of scores until the optimal performance of the classifier is obtained for that category; fix the per-category thresholds when applying the classifier to new documents in the test set. Differing from RCut and PCut in which a single parameter (t or x) is used to optimize the *global* performance of the classifier on average, SCut optimizes the performance of the classifier on individual categories without guaranteeing a global optimum. SCut was used in the evaluations of many classifiers, including Ripper, FOIL, Winnow, EG, kNN, LLSF and Rocchio[10, 2, 16].

2.2 Property Analysis

Which of the above threshold strategies is optimal? The answer depends on the classifier, the optimization criterion, and the properties of the thresholding algorithms.

- *Comparability Among Scores:* Considering the system-generated scores for all the document-category pairs in a $n \times m$ matrix. RCut compares the category scores by fixing the document, while SCut and PCut compare document scores by fixing the category. Obviously, RCut would work better then SCut or PCut if the within-document scores are more comparable than the the within-category scores; otherwise, it would be worse. However, given a classifier and a data collection, it is often not obvious in which way the system-generated scores are more comparable; a simple way to tell, perhaps, is to test RCut, RCut and SCut on a validation set and compare the results.

[1] Recent research in the domain of Topic Detection and Tracking (TDT) have addressed a relevant issue – the Decision Error Trade-off in classifiers; however, since the TDT domain has special conditions which are not generally true in TC (including the restriction of using 2-way classifiers only, the one-class-per-document nature of the data, the incremental changing of training and test sets, the online-decision requirement, and so forth), those findings are beyond the scope of this paper. Filtering is also a relevant problem but not a focus of this paper; I will report our experiments on filtering in a separate paper.

- *Information Used:* While all three thresholding strategies use scores, PCut is the only one which uses the category distribution observed in the training set to gain a global control of the category assignments to documents in the test set. This gives PCut additional power, but sacrifices the ability to make decisions online for test documents because the scores for the test documents must be accumulated before PCut can be applied. Moreover, PCut assumes that the distribution of categories over documents remains relatively constant – this a good assumption only for certain domains but not for others. In the domain of Topic Detection and Tracking, for example, the distribution of classes (topics or events) in news stories constantly change over time.

- *Suitability for Online Response:* Different from PCut, decisions by RCut on candidate categories for each test document are independent from the decisions on other test documents; this makes RCut suitable for online classification or adaptive filtering. SCut is similar: once the per-category thresholds are optimized (offline) on a validation set, the decisions by the classifier for each test document are independent from the decisions on other test documents.

- *Suitability for Optimizing Specific Performance Measures:* The per-category optimization in SCut makes it particularly effective when macro-averaged performance is the target function to optimize. Since category distributions in real-world applications often exhibit the Zipf's law[18, 4], the macro-averaged performance is likely to be dominated by performance of the system on rare categories. RCut and PCut, on the other hand, with only a single parameter (t or x) to tune, will be generally less effective for optimizing macro-averaged performance.

- *Flexibility in Recall-Precision Trade-off:* PCut is most flexible, allowing the system to assign any number (in total) of "YESes" to the test documents according to user's specification for the value of x, ranging from zero ("Mr. NO") to the maximum of $m \times n$ ("Mr. YES"). RCut has a partial flexibility, allowing the system to assign $n, 2n, \ldots mn$ of "YESes" by setting $t = 1, 2 \ldots m$. SCut is more subtle: it does not directly take any user-specified parameter to control the number of YESes assigned by the system; instead, the trade-off between recall and precision can be indirectly adjusted by altering the optimization criterion (e.g., F_1 or F_β for any value of β; see Section 4 for definitions). For simplicity in this paper, I will only include the results of SCut using F_1 as the optimization criterion.

- *Risk of Overfitting:* While both SCut and PCut use category-specific thresholds, the thresholds in SCut are tuned to optimize the system's performance on the validation set, while the thresholds in PCut only depend on the data (the training-set probabilities of categories). SCut is more likely to overfit than PCut, assuming system performance variance on different subsets is generally larger than the variance in observed category distributions over the subsets. RCut is insensitive to either type of tuning.

2.3 New Variants

I propose the following modifications for RCut and SCut, to address two problems which have not been addressed before:

RTCut – RCut imposes a harsh trade-off between recall and precision. To smooth this trade-off, I have created the *RTCut* scoring method. In RTCut, we assign synthetic scores to the candidate categories computed from both their ranks and confidence scores for the document being classified. These synthetic scores are computed by:

$$f(c|d) = r(c|d) + \frac{s(c|d)}{\max_{c' \in \mathcal{C}}\{s(c'|d)\} + 1}$$

where d is the input document, $r(c|d)$ is the rank of category c with respect the document, $s(c|d)$ is the original system-generated confidence score for assigning category c to d, and $f(c|d)$ is the new score. RTCut preserves the ranked order of categories, but allows us to distinguish between categories with the same rank. By thresholding on $f(c|d)$ instead of $r(c|d)$ (the rank of the category, e.g. RCut), fine-grained trade-offs between recall and precision can be made. As demonstrated in section 4, RTCut performs very well in the high-precision end of recall-precision space and outperforms PCut.

SCutFBR – When the number of training examples for a category is small, SCut runs the risk of overfitting to the training data during cross-validation and can produce a threshold that is too high or too low. A threshold that is too high results in many misses for the category, which lowers the macro-average of F_1. A threshold that is too low, on the other hand, results in many false alarms, which lowers both micro- and macro-average F_1, since there are potentially many more false-alarms than misses for rare categories. To address this problem, a simple and effective heuristic of setting the threshold to infinity (e.g. reject everything) is used, which minimizes the impact on micro-average F_1 at the expense of slightly lowering macro-average F_1. As an alternative that does not hurt macro-average F_1 as much, the threshold is set to the score of the top-ranked document for that category. In the rest of this paper, I refer to the former method (fallback threshold at infinity) as *SCutFBR.0* and the latter (threshold at top-ranked document) as *SCutFBR.1*. Methods which set accurate thresholds using the limited available training data for rare categories form an important and intriguing subject for future research.

3. DATA SETS

Five corpora were chosen for the experiments, allowing observations on the regularities and variations in the effects of thresholding strategies under different conditions.

3.1 Reuters-21578

The Reuters corpus of newswire stories is the most commonly used benchmark corpus in TC evaluations[16]. It consists of over 20,000 Reuters newswire stories in the period between 1987 to 1991. Several versions of this corpus have been derived from the original one; For this paper I use

the ApteMod version of Reuters-21578, which was obtained by eliminating unlabelled documents and selecting the categories which have at least one document in the training set and one in the test set. This process resulted in 90 categories in both the training and test sets. After eliminating documents which do not belong to any of these 90 categories, I obtained a training set of 7769 documents, a test set of 3019 documents, and a vocabulary 24240 unique words after stemming and stop word removal. The number of categories per document is 1.3 on average.

3.2 OHSUMED-233445

The OHSUMED-233445 corpus, a subset of the original corpus prepared by William Hersh and colleagues at the Oregon Health Sciences University[6], is another evaluation benchmark in TC literature[15, 16]. It consists of 233,445 abstracts (with titles) of journal articles in the medical domain during the years from 1987 to 1991. These articles were assigned to 14,321 categories in the Medical Subject Headings (MeSH) taxonomy; there are about about 13 categories per document on average. For the experiments in this paper, I arbitrarily picked the 1987 subset (36,890 documents) for training and the 1988 subset (47,054 documents) for testing; the average number of categories per document in these subsets is 12.

3.3 TREC9-MeSH-Batch

This corpus is the TREC-9 version of OHSUMED with a selected subset of MeSH categories, which is constructed for the TREC-9 Filtering Track. A category was selected if it satisfied the two conditions: 1) it had at least four relevant documents in the 1987 subset (the training set) of OHSUMED, and 2) it had at least one relevant document in each year in the 1988-1991 subset (the test set). The resulting set consists of 4,904 categories (34% of the 14,321 categories in OHSUMED-233445); all the remaining categories (i.e., rare categories, 66% of the total) were collapsed into a giant pseudo-category (namely, the "none" category). Such a modification of the classification scheme, essentially ignoring the fine distinctions among a large portion of the rare categories, made the classification task less challenging on this corpus than on OHSUMED-233445 (see the results in the next section). However, since it is a new benchmark in TREC, empirical results for thresholding strategies on this corpus will provide valuable reference for future research even though our work in TREC-9 Filtering (which will be reported in a separate paper) is not the focus of this paper.

3.4 HV-28 and HV-255

These two corpora, constructed by Ghani et al. at Carnegie Mellon University[3], share the same set of documents which are labelled using two different classification schemes. The document set consists of 4,285 synthetic web pages each of which corresponds to the web site of a company listed by the Hoovers Online Web resource <www.hoovers.com>. The collection was obtained by starting from a list of company names and home-page URLs, visiting the site for each company and retrieving up to the first 50 Web pages (in breadth first order). All the pages crawled for each company are concatenated to obtain a synthetic "page" for that company. The reason for doing so was that the categories (available at Hoover Online) classify a company's web *site*, not its individual pages. There are two sets of categories: a coarse classification scheme of 28 classes (industry sectors such as Oil & Gas, Sporting Goods, Computer Software & Services) and a more fine grained classification scheme consisting of 255 classes. Each web-site is classified into one category only (for each classification scheme). The vocabulary size is 256,715 unique words after removing stop words and stemming.

For the experiments in this paper, each of the two corpora was split into 5 subsets for 5-fold cross validation which consisted of 5 iterations. In each iteration, one of the 5 subsets was used for testing, and the remaining 4 subsets were merged into one as the training set. The performance scores were averaged over the 5 runs for a global measure. Hyperlinks and meta data about the Web sites are also available in these corpora; experiments with using those information are reported in a separate paper[4].

4. EXPERIMENTS

4.1 Performance Measures

To evaluate the categorization performance of kNN with various thresholding strategies, I present the results in both micro-averaged and macro-averaged recall, precision and F_1. Recall (r) is the proportion of correctly predicted YESes by the system among the true YESes for all the document-category pairs given a dataset. Precision (p) is the proportion of correctly predicted YESes among all the system-predicted YESes. The F_1 measure[13] is the harmonic average of recall and precision, defined to be

$$F_1 = \frac{2pr}{p+r}$$

A more general notion for the F-measure is

$$F_\beta = \frac{(\beta^2+1)pr}{\beta^2 p + r}$$

where parameter β is specified to adjust the relative weighting between recall and precision. When scores are *micro-averaged*, the binary decisions are collected in a joint pool and then the recall, precision and F_1 values are computed from that pool. When the scores were *macro-averaged*, the recall, precision and F_1 values for individual categories are computed first and then averaged over categories.

It may be worth mentioning a potential confusion in computing macro-averaged F_1. That is, there are two possible ways: the "correct" way, according to our definition, is to compute the F_1 values for each category and then take the average over the per-category F_1 scores; the "wrong" way is to first compute the macro-averaged recall and macro-averaged precision, and then compute the harmonic average of these averaged recall and precision. The two ways often yield different results: the "correctly" computed F_1 value is often significantly lower than the "wrongly" F_1 value because the F_1 value of a specific category is usually dominated by the smaller value between the recall and precision values for that category when these two values are radically different.

4.2 Experimental Settings

For the experiments in this paper, I used our standard kNN classifier for which detailed descriptions and the parameter setting process were reported in previous papers[14,

16, 18, 17]; stop word removal, stemming and statistical feature selection were applied to documents in a preprocessing step, using the methods described in those papers as well.

All the parameters for the thresholding strategies (RCut, PCut, SCut and RTCut) were tuned using hold-out validation sets (Section 2); the threshold increment unit was manually chosen for each corpus, to obtain sufficient number of recall-precision plots for an interpolated trade-off curve. For SCut, both SCutFBR.0 and SCutFBR.1 settings were examined on each corpus; for the setting with a better performance in the validation condition, its performance on the corresponding test sets was reported for SCut.

4.3 Empirical Observations

Observation 1. The choice of thresholding strategy made significant differences.

Figures 1, 2 and 3 show the micro-averaged recall-precision curves of kNN with RCut, PCut, SCut and RTCut applied to three corpora[2]; Figures 4, 5 and 6 show the corresponding curves using macro-averaged measures. Those curves were obtained by thresholding at $t = 1, 2, 3\ldots$ in RCut, $x = 0.5, 1, 2, 3\ldots$ in PCut, and appropriate incremental thresholds in RTCut to obtain high-precision output; the recall and precision values at each threshold were computed and interpolated. SCut did not produce a curve, but a single point per data set instead. The "break-even line" is drawn in those graphs as a reference line on which recall and precision have equal values, and around which F_1 scores are typically optimized. The performance differences in micro-averaged F_1 on Reuters (.85 for kNN with using SCut vs .80 for kNN with using RCut for example) should be statistically significant, according to a related study on the same corpora[18]. The performance differences yielded by the varying thresholding strategies were even larger for the OHSUMED and HV sets. The observations on the macro-averaged F_1 performance curves are consistent with the observations on micro-averaged F_1 curves: first, the choice of thresholding strategy has a significant impact; second, which strategy is better depends on the data set; third, RTCut consistently outperforms RCut on the high-precision end of the curves; and fourth, RTCut consistently outperforms PCut on four out of the six curves (except Reuters).

Observation 2. SCut tends to overfit.

SCut was the best choice for kNN on Reuters (Figures 1 and 4), and the worst choice for kNN on OHSUMED (Figures 2 and 5); as for kNN on HV-255, it was the worst choice when micro-averaged performance was concerned (Figure 3), and the best choice when macro-averaged performance was concerned instead (Figure 6). These results are rather surprising.

[2]The results on the other two corpora (HV-28 and TREC9-MeSH-Batch) are omitted due to the limited space in this paper.

Figure 1. kNN on Reuters-21578

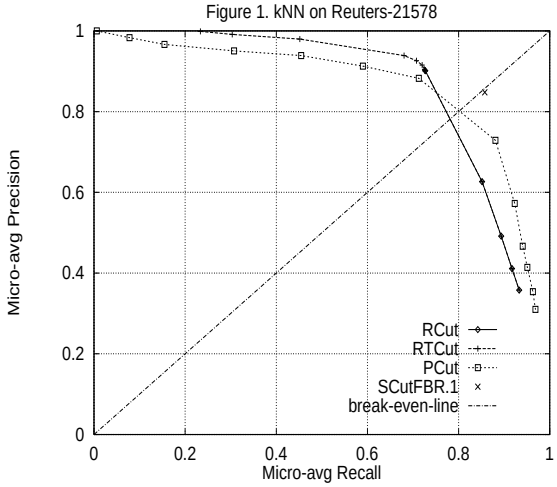


Figure 5. kNN on OHSUMED-233445

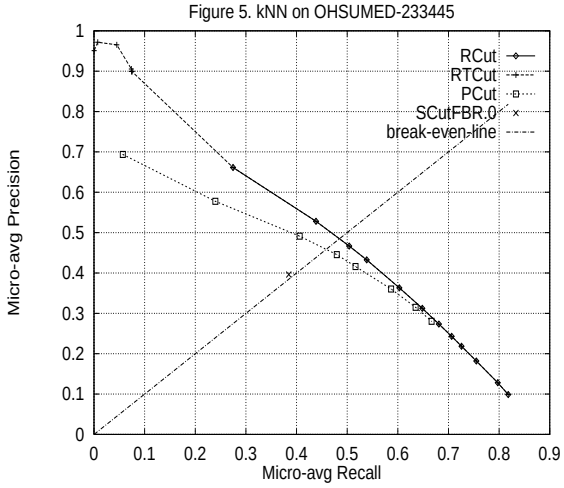


Figure 3: kNN on HV-255

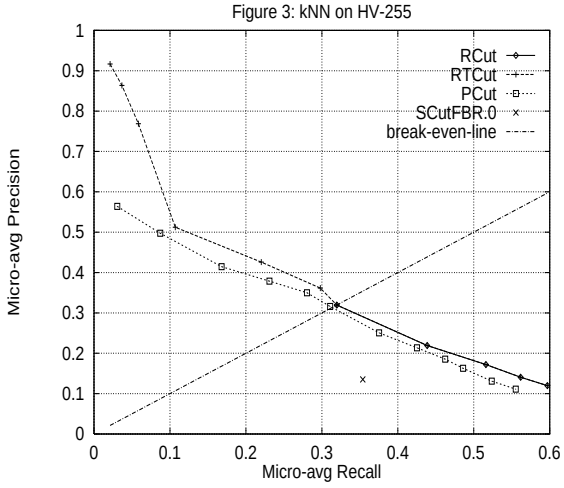


Figure 4. kNN on Reuters-21578

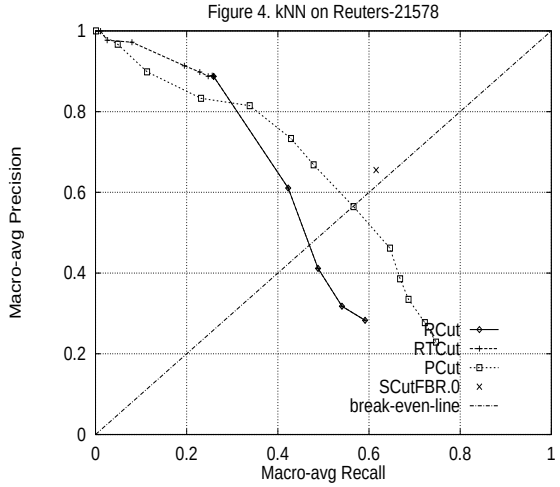


Figure 5. kNN on OHSUMED-233445

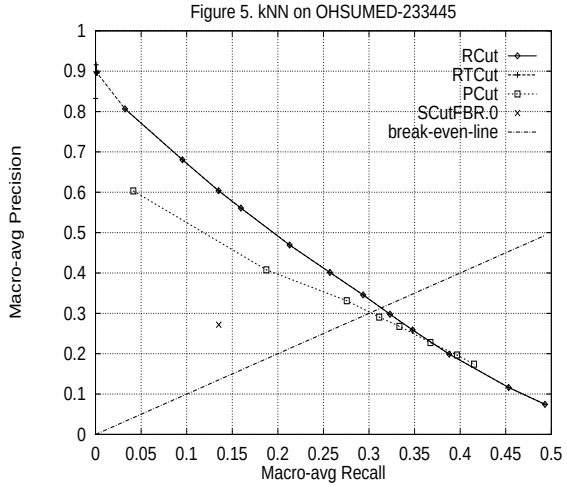
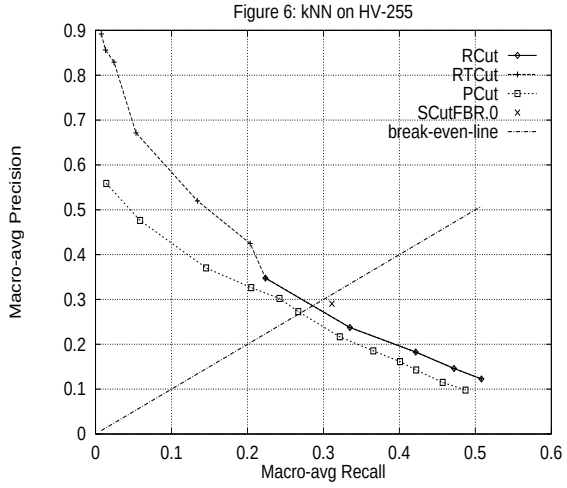


Figure 6: kNN on HV-255



A hypothetical interpretation for the large variation in these results is that the per-category threshold tuning in SCut tends to overfit the validation sets and does not generalize to test sets in a stable manner. Figures 7 and 8 present some evidence for this hypothesis by comparing the F_1 scores (micro-averaged and macro-averaged) pairwise over the validation and test sets in four corpora. The performance degradations of kNN are indeed significant when moving from the validation condition to the test condition. Moreover, when using macro-averaged F_1 instead micro-averaged F_1 as the measure, the performance degradations of kNN was even larger. Recall that the category distributions in those corpora exhibit the Zipf's law[18, 4], and that for such skewed category distributions, the macro-averaged F_1 scores are dominated by the performance of the system on rare categories, the overfitting problem with SCut would be more serious due to the lack of sufficient training examples for rare categories when macro-averaged performance is the primary concern.

Figure 7: Scut in Cross Validation -- micro-avg performance

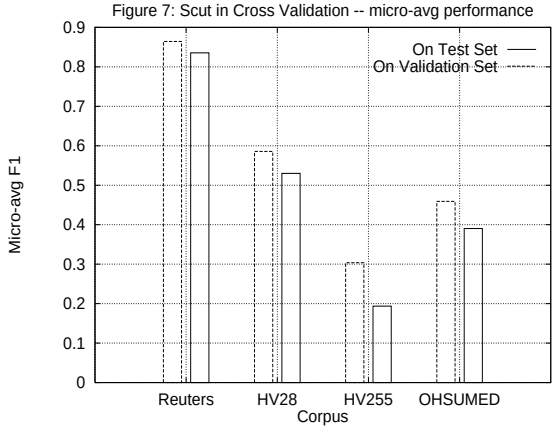
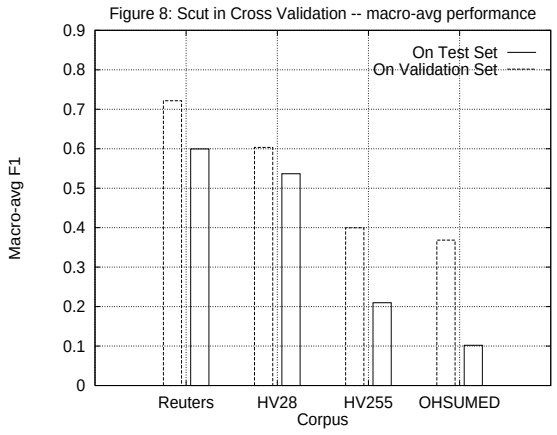


Figure 8: Scut in Cross Validation -- macro-avg performance



Observation 3. PCut performed well on rare categories

Figures 9 to 14 show the interpolated F_1 curves of kNN corresponding to the thresholds in PCut and RCut; the F_1 value produced by SCut is shown in each figure using a dashed line for comparison. RCut is competitive to PCut when measures were micro-averaged (Figures 9, 10 and 11); i.e., when the performance of the system on common categories dominates the global measure. On the other hand, PCut has much better curves than RCut does when measures were macro-averaged (Figures 12, 13 and 14); i.e., when the per-

formance of the system on rare categories dominates the average. This suggests that the use of training-set priors of categories in PCut is an effective scheme for controlling the behavior of the classifier (kNN in our case) when dealing with categories for which few training examples are available.

Observation 4. PCut's peak performance occurs near the threshold equal to the average number of categories per document in the corpus.

Another interesting observation from the curves in Figures 9 to 14 is that PCut's peak performance, for both the micro-averaged and macro-averaged F_1 curves, occurs near the threshold where the number of system-assigned YESes to a document is equal or close to the number of the true categories per document on average. For example, the peak performance of PCut on Reuters is between 1 and 1.5, the corresponding peak on OHSUMED is around 10 to 15, and peak on HV-255 is around 1. In contrast, such a regularity is only observed for RCut when the measures are micro-averaged, not when the measures are macro-averaged.

Observation 5. RTCut is effective for trading recall for precision.

The curves in Figures 15 and 16 show a nearly linear trade-off between recall and precision, as achieved by adjusting the thresholds in RTCut for most data sets. It is very interesting and surprising that kNN can obtain good high-precision performance by the simple mechanism of RTCut. This suggests that both the cross-category ranking and cross-category scores by the standard kNN are quite comparable, which has not been observed before. Further insights from this observation invite future research.

Observation 6. Altering classification schemes significantly changed the difficulty of the TC problems.

The performance curves of kNN in Figures 15 and 16 also suggest some insights into how difficult the classification problems are for those data collections. The tasks in Reuters and HV-28 appear to be much easier than those in the other data sets, at least for kNN. Note that the only difference between the HV-28 and HV-255 data sets is the classification scheme, while the documents are exactly the same. The performance degradation in kNN (with PCut or RCut) when moving from HV-28 to HV-255 is dramatic. Similarly, it is worth noticing that the exclusion of 66% of MeSH categories from OHSUMED in the evaluations of TREC9-MeSH-Batch also eased the problem considerably, as being reflected in the *macro-averaged* recall-precision curves where the performance on rare categories dominated the average.

Figure 9: Micro-avg F1 curves of kNN on Reuters-21578

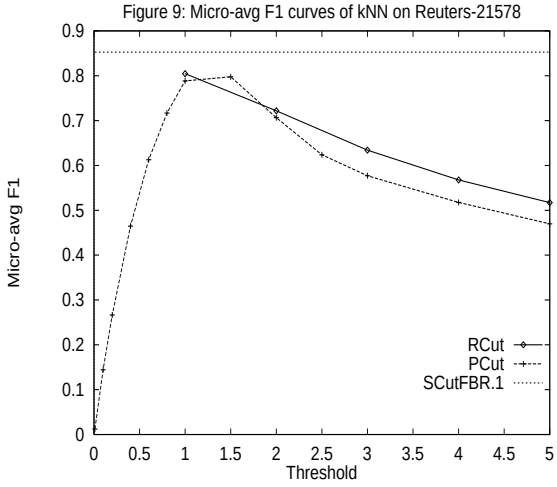


Figure 10: Micro-avg F1 curves of kNN on OHSUMED

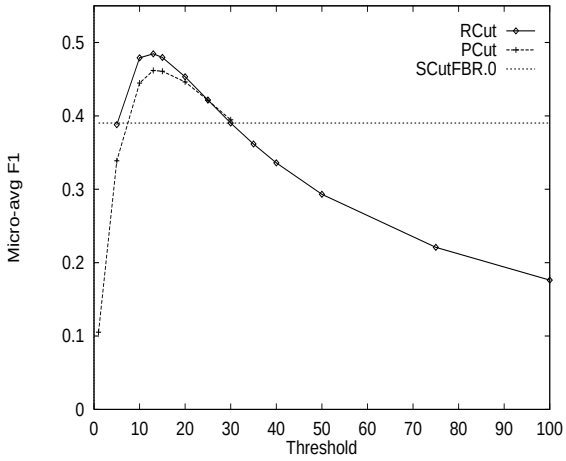


Figure 11: Micro-avg performance of kNN on HV-255

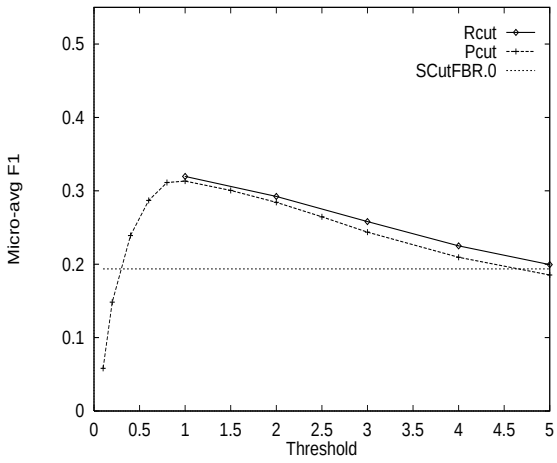


Figure 12: Macro-avg F1 curves of kNN on Reuters-21578

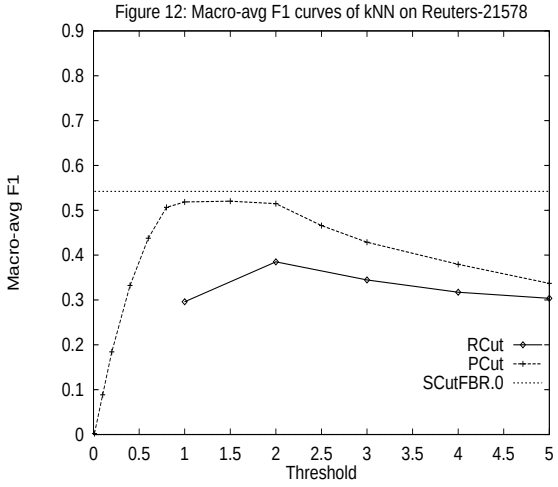


Figure 13: Macro-avg F1 curves of kNN on OHSUMED

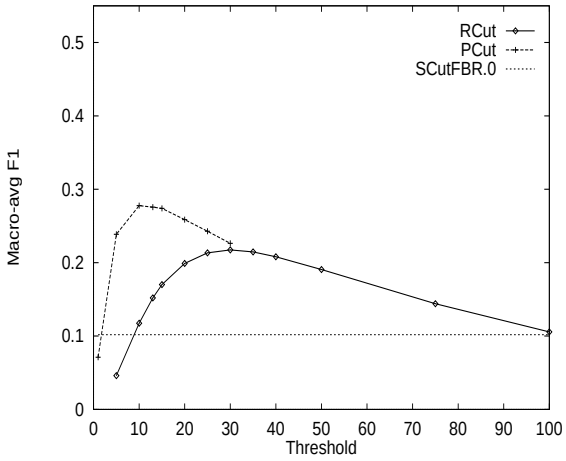


Figure 14: Macro-avg performance of kNN on HV-255

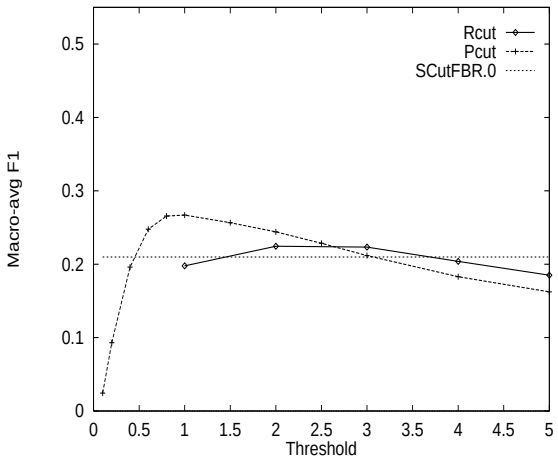


Figure 15: kNN with RTcut on different datasets

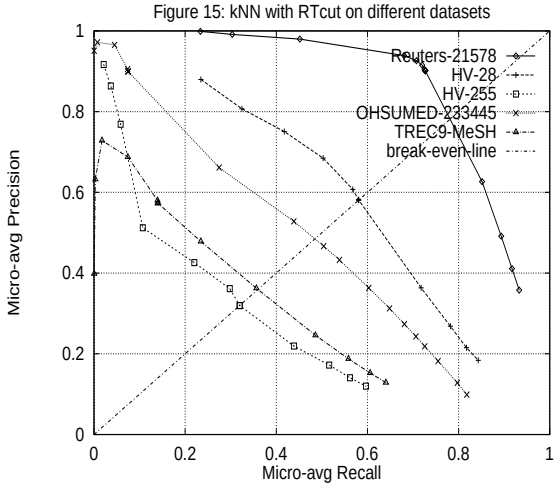
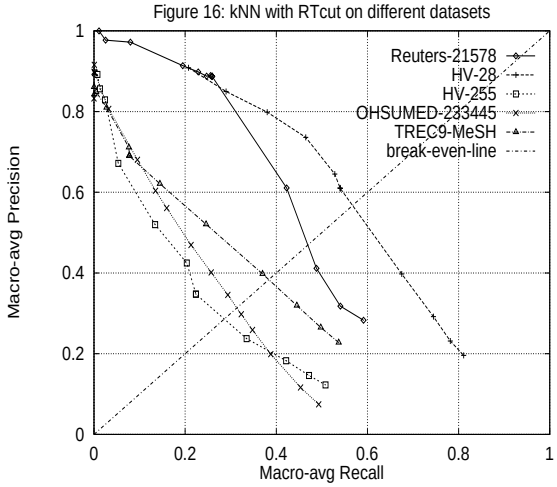


Figure 16: kNN with RTcut on different datasets



5. CONCLUDING REMARKS

This paper presented an examination of the effect of thresholding strategies on the performance of a classifier under various conditions. The experiments with kNN using RCut, PCut, SCut and RTCut (a hybrid method combining category ranking and scoring) on the five benchmark evaluation collections showed that the choice of thresholding strategy could indeed make a significant difference in what we would observe when a classifier is evaluated, and that making the right choice for real-world applications is a non-trivial issue. These experiments also illustrated the strengths and limitations for RCut, PCut, SCut and RTCut. The unstable performance of SCut across applications (data sets) suggests a tendency to overfit and a potential weakness of that method in dealing with rare categories, or when using it in an early stage of adaptive categorization. PCut exhibits more stable performance due to its global control using information about category distribution in the training set, but at the

cost of not be able to make online decisions. RCut also appeared to be less "risky" than SCut because of its rank-based nature (not overly sensitive to absolute scores) and is naturally suitable for online response because its decisions for a document are independent of its decisions over other documents; its weakness is the lack of any global control (which PCut has) and lack of free parameters for local optimization (which SCut has), which implies less room for fine-tuning. RTCut combines the strengths of category ranking in RCut and category scoring in SCut in a simple fashion for an effective and promising solution to the thresholding problem.

To conclude, optimal thresholding in TC remains an important and open issue. How to jointly use the strengths of different strategies in solving TC problems in realistic applications is a challenge. I hope that the analysis presented will provide useful insights and suggest a systematic approach to the understanding and proper use of thresholding strategies in other classifiers and in solving new classification problems. A related area for future research is to combine the strengths of the different thresholding strategies in the context of adaptive document filtering where the category distribution is not given in priori (thus must be learned incrementally over time), delayed categorization decisions are not allowed (so PCut cannot be used directly), and high-precision performance in relevance feedback is crucial for initiating the positive feedback loop.

6. ACKNOWLEDGMENTS

The author would like to thank Jaime Carbonell and Tom Ault for editorial assistances and fruitful discussions, and Rayid Ghani, Sean Slattery and other members of Tom Mitchell's WebKB project for providing the Hoovers data sets. This study was funded in part by National Science Foundation under the grant number KDI-9873009.

7. REFERENCES

[1] P. N. Bennett. Assessing the calibration of Naive Bayes' posterior estimates. In *Technical Report CMU-CS-00-155, computer Science Department, School of Computer Science, Carnegie Mellon University*, 2000.

[2] W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996. 307-315.

[3] R. Ghani, R. Jones, D. Mladenic, K. Nigam, and S. Slattery. Data mining on symbolic knowledge extracted from the web. In *Proceedings of the Workshop on Text Mining at the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000)*, 2000.

[4] R. Ghani, S. Slattery, and Y. Yang. Hypertext categorization using hyperlink patterns and meta data. In *The Eighteenth International Conference on Machine Learning (ICML'01)*, page (submitted), 2001.

[5] N. Govert, M. Lalmas, and N. Fuhr. A probabilistic description-oriented approach for categorising web documents. In *Proceedings of the Eights International Conference on Information and Knowledge Management*, pages 475–482, New York, 1999. ACM.

[6] W. Hersh, C. Buckley, T. Leone, and D. Hickman. Ohsumed: an interactive retrieval evaluation and new large text collection for research. In *Proceedings of ACM SIGIR'94*, pages 192–201, 1994.

[7] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *European Conference on Machine Learning (ECML)*, pages 137–142, Berlin, 1998. Springer.

[8] D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *15th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92)*, pages 37–50, 1992.

[9] D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, Nevada, Las Vegas, 1994. University of Nevada, Las Vegas.

[10] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996. 298-306.

[11] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, 1999.

[12] R. E. Schapire, Y. Singer, and A. Singhal. Boosting and Rocchio applied to text filtering. In *Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 215–223, New York, 1998. The Association for Computing Machinery.

[13] C. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[14] Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *17th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 13–22, 1994.

[15] Y. Yang. An evaluation of statistical approach to text categorization. In *Technical Report CMU-CS-97-127, Computer Science Department, Carnegie Mellon University*, 1997.

[16] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, 1999.

[17] Y. Yang, T. Ault, and T. Pierce. Improving text categorization methods for event tracking. In *Proceedings of ACM SIGIR'2000*, 65-72.

[18] Y. Yang and X. Liu. A re-examination of text categorization methods. In *The 22th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 42–49, 1999.

[19] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. In *Journal of Intelligent Information Systems*. Kluwer Academic Press, (accepted).