Cluster-Based Text Categorization: A Comparison of Category Search Strategies

Makoto IWAYAMA
Advanced Research Laboratory
Hitachi Ltd.
Hatoyama
Saitama 350-03, Japan
iwayama@harl.hitachi.co.jp

Takenobu TOKUNAGA
Department of Computer Science
Tokyo Institute of Technology
2-12-1, Ookayama, Meguro
Tokyo 152, Japan
take@cs.titech.ac.jp

Abstract

Text categorization can be viewed as a process of category search, in which one or more categories for a test document are searched for by using given training documents with known categories. In this paper a cluster-based search with a probabilistic clustering algorithm is proposed and evaluated on two data sets. The efficiency, effectiveness, and noise tolerance of this search strategy were confirmed to be better than those of a full search, a category-based search, and a cluster-based search with nonprobabilistic clustering.

1 Introduction

Text categorization can be viewed as a process of category search: given training documents with known categories, a program searches for one or more categories that a test document is assumed to have. The simplest strategy would be to search the K-nearest training documents to the test document and use the categories assigned to those training documents. This is known as MBR (Memory Based Reasoning) [Stanfill and Waltz, 1986] or K-NN (K-Nearest Neighbor classifiers) [Weiss and Kulikowski, 1990]. Although this *full search* offers promising performance in text categorization [Masand *et al.*, 1992], it requires a large amount of computational power for calculating a measure of the similarity between a test document and every training document and for sorting the similarities.

One alternative strategy is a *cluster-based search* [Salton and McGill, 1983], where training documents are partitioned into several clusters before searching and a test document is compared with each cluster rather than with each document. Cluster-based searches have been used in text retrieval to improve both the *efficiency* and the *effectiveness* of full search [Jardine and Van Rijsbergen, 1971, van Rijsbergen, 1974, Croft, 1980], but their significantly greater advantage of the effectiveness has not been verified. Since the effectiveness of this kind of searching depends on the predictive performance of constructed clusters, selecting a better clustering algorithm is crucial. The most popular algorithm in text retrieval is the single-link method or Ward's method that use the measure of distance between two objects and merge the closer ones [Anderberg, 1973, Cormack, 1971, Griffiths *et al.*, 1984, Willett, 1988]. In text categorization, the simplest version of clustering has been used: all the training documents that are assigned the same category are grouped into a cluster as the representation of the category. We refer this strategy as *category-based search*.

In this paper we propose a probabilistic clustering algorithm called *Hierarchical Bayesian Clustering* (HBC) and use the algorithm to construct a set of clusters for cluster-based search. The searching platform we focus on is the probabilistic model of text categorization that searches the most likely clusters to which an unseen document is classified [Croft, 1981, Fuhr, 1989, Iwayama and Tokunaga, 1994, Kwok, 1990, Lewis, 1992]. Since HBC constructs the most likely set of clusters that contains the the given training documents, HBC gives exactly the same criterion both in constructing and in searching clusters. For this reason, our framework is expected to offer a better performance than does a framework that uses a probabilistic model in searching clusters but uses a nonprobabilistic model in constructing clusters [Croft, 1980].

In the experiments reported here we compared the four category search strategies: full search, category-based search, cluster-based search with nonprobabilistic clustering, and cluster-based search with probabilistic clustering. The two data sets we used are rich in variety: one was Japanese dictionary data (called *Gendai yôgo no kisotisiki*), which is well organized by editors; and the other was a collection of English news stories (from the Wall Street Journal), which is a real-world data set but includes much noise. The results suggest that the most balanced strategy from the standpoints of efficiency, effectiveness, and noise tolerance is the cluster-based search with probabilistic clustering.

2 Category Search Strategies

The category search strategy in probabilistic text categorization can be broken down into the following four steps:

1. Construct clusters $C = \{c_1, c_2, \ldots, c_{N_C}\}$ from the given training documents $D = \{d_1, d_2, \ldots, d_{N_D}\}$.
2. Calculate the posterior probability $P(c_i|d_{test})$ for a test document d_{test} and every cluster c_i.
3. Sort the posterior probabilities and extract the K-nearest training documents.
4. Assign to the test document categories based on the extracted K-nearest documents.

The differences between category search strategies stem from the difference of clustering algorithms used in step 1. For full search (MBR or K-NN), no clustering algorithm is used there. It follows that each training document belongs to a singleton cluster whose

SIGIR'95 Seattle WA USA© 1995 ACM 0-89791-714-6/95/07 $3.50

category search strategy	clustering algorithm	number of clusters N_C	time complexity
full search	---	N_D	---
category-based search	grouping documents according to the assigned categories	number of categories	$O(N_D)$
cluster-based search with nonprobabilistic clustering	single-link method, Ward's method, etc.	$\leq N_D$	$O(N_D^3)$
cluster-based search with probabilistic clustering	Hierarchical Bayesian Clustering etc.	$\leq N_D$	$O(N_D^3)$

Table 1: Clustering algorithms in category search strategies.

only member is the document itself. For category-based search, all the training documents having the same category are grouped into a cluster. The complexity of this grouping is $O(N_D)$ where N_D is the number of training documents. For cluster-based search with nonprobabilistic clustering, a clustering algorithm such as the single-link method or Ward's method is used. These algorithms are based on the Euclid distance between two objects [Anderberg, 1973, Cormack, 1971] and the complexity of constructing clusters is $O(N_D^3)$. For cluster-based search with probabilistic clustering, a clustering algorithm based on probability theory is used. In this paper we propose the use of an algorithm called Hierarchical Bayesian Clustering (HBC) because it uses the same criterion in constructing and searching clusters. The complexity of HBC is also $O(N_D^3)$. Table 1 summarizes the four strategies.

Steps 2 and 3 search the K-nearest training documents to a test document d_{test} using the clusters constructed at step 1. The measure of nearness is the posterior probability $P(c_i|d_{test})$, the probability that the test document d_{test} is classified into a cluster c_i. The training documents in the nearest clusters become the nearest training documents.[1] Although many methods for calculating the posterior probability have been proposed [Robertson and Sparck Jones, 1976, Kwok, 1990, Fuhr, 1989], we use the simple formula proposed in [Iwayama and Tokunaga, 1994]. [2] We first consider an event "$T = t$," meaning that a randomly selected term T from the document d_{test} is equal to t. Conditioning $P(c_i|d_{test})$ for each possible event gives

$$P(c_i|d_{test}) = \sum_t P(c_i|d_{test}, T = t)P(T = t|d_{test}). \quad (1)$$

Assuming conditional independence between c_i and d, given $T = t$,[3] and applying Bayes' theorem, we obtain

$$P(c_i|d_{test}) = P(c_i) \sum_t \frac{P(T = t|c_i)P(T = t|d_{test})}{P(T = t)}. \quad (2)$$

Probabilities on the right-hand side of this equation are estimated as follows:

- $P(T = t|d_{test})$: relative frequency of a term t in a test document d_{test}.
- $P(T = t|c_i)$: relative frequency of a term t in a cluster c_i.
- $P(T = t)$: relative frequency of a term t in the entire set of training documents.
- $P(c_i)$: relative frequency of documents that belong to c_i in the entire set of training documents.

In this paper we limited terms to nouns.

The efficiency of finding the K-nearest training documents depends on the number of clusters N_C constructed in step 1. Since the complexity of this search is $O(N_C \log N_C)$ in the worst case,[4] a smaller set of clusters is preferable from the standpoint of the efficiency.

In step 4, each document d_{train} in the K-nearest training documents votes on each of the categories assigned to d_{train}. The weight of the voting is $\log P(\{d_{train}\}|d_{test})$, that is, the logarithm of the probability that d_{test} is categorized into the cluster $\{d_{train}\}$ whose only member is d_{train}. This voting results in the category ranking for each test document. In a category-based search, it is usual to make this ranking by calculating the posterior probabilities $P(c_i|d_{test})$ for every cluster (i.e., category) c_i directly, not using the voting strategy. According to the category ranking, one or more categories are assigned to each test document using one of the following category assignment strategies.

[k-per-doc]
Assign the top k categories to each test document.

[probability threshold]
Assign all the categories above a user-defined threshold.

[proportional assignment]
Each category is assigned to its top scoring documents in proportion to the number of times the category was assigned in the training documents [Lewis, 1992]. For example, a category assigned to 2% of the training documents would be assigned to the top scoring 0.2% of the test documents if the proportionality constant was 0.1, or to 10% of the test documents if the proportionality constant was 5.0.

Several experiments confirm the advantage of proportional assignment over the other assignment strategies [Lewis, 1992, Iwayama and Tokunaga, 1994]. Note however that proportional assignment requires batch processing of test documents; it has to wait until the sufficient number of test documents are gathered. When real-time response is necessary, a k-per-doc or probability threshold strategy should be used.

3 Hierarchical Bayesian Clustering

In cluster-based search with probabilistic clustering, we propose that Hierarchical Bayesian Clustering (HBC) [Iwayama and Tokunaga, 1995] be used. Given training documents D, HBC constructs the set of clusters C that has the locally maximum value of the posterior probability $P(C|D)$. This is a general form of well-known

[1] This means that the number of the nearest training documents in cluster-based search may not be continuous.

[2] Qualitative and experimental comparisons of probabilistic models are discussed in [Iwayama and Tokunaga, 1994]

[3] More precisely, $P(c_i|d_{test}, T = t) = P(c_i|T = t)$ which assumes that if we know $T = t$, information for c_i is independent of that for d. This assumption seems valid because $T = t$ is a kind of representation of d.

[4] Using inverted file [Salton and McGill, 1983] can reduce the number of comparisons to find the K-nearest documents.

Input:
$D = \{d_1, d_2, \ldots, d_N\}$: a collection of N data;

Initialize:
$C_0 = \{c_1, c_2, \ldots, c_N\}$: a set of clusters;
$c_i = \{d_i\}$ for $1 \leq i \leq N$
calculate $SC(c_i)$ for $1 \leq i \leq N$
calculate $SC(c_i \cup c_j)$ for $1 \leq i < j \leq N$

for $s = 1$ **to** $N - 1$ **do**
$(c_x, c_y) = \arg\max_{c_x, c_y} \frac{SC(C_x \cup C_y)}{SC(C_x)SC(C_y)}$
$C_s = C_{s-1} - \{c_x, c_y\} + \{c_x \cup c_y\}$
calculate $SC(c_x \cup c_z)$ for all $c_z \in C_s$ where $z \neq x$

Function $SC(c)$
return $\prod_{d \in c} P(c|d)$

Figure 1: Hierarchical Bayesian Clustering.

Maximum Likelihood estimation, estimating the most likely model (i.e., set of clusters) from given training data. One advantage of using HBC in the framework of probabilistic text categorization is that the function to be maximized in constructing clusters is exactly the same as the function used in searching for the nearest clusters.

Like most agglomerative clustering algorithms [Cormack, 1971, Anderberg, 1973, Griffiths *et al.*, 1984, Willett, 1988], HBC constructs a cluster hierarchy (also called dendrogram) from bottom to top by merging two clusters at a time. At the beginning (the bottom level in a dendrogram), each document belongs to a cluster whose only member is the document itself. For every pair of clusters, HBC calculates the increase of the posterior probability after merging the pair and selects the pair that results in the maximum increase. To see the details of this merge process, consider a merge step $k + 1$ ($0 \leq k \leq N - 1$). By the step $k + 1$, a data collection $D = \{d_1, d_2, \ldots, d_{N_D}\}$ has been partitioned into a set of clusters $C_k = \{c_1, c_2, \ldots\}$. That is, each datum $d_i \in D$ belongs to a cluster $c_j \in C_k$. The overall posterior probability at this point becomes

$$P(C_k|D) = \prod_{c_j \in C_k} \prod_{d_i \in c_j} P(c_j|d_i). \quad (3)$$

This formula defines the overall probability that every training document is classified into the cluster that includes the training document. Each $P(c_j|d_i)$ is calculated using Eq. (2). When the algorithm would merge two clusters $c_x, c_y \in C_k$, the set of clusters C_k is updated as follows:

$$C_{k+1} = C_k - \{c_x, c_y\} + \{c_x \cup c_y\}. \quad (4)$$

After the merge, the posterior probability is inductively updated as follows:

$$P(C_{k+1}|D) = P(C_K|D) \frac{\prod_{d_i \in c_x \cup c_y} P(c_x \cup c_y|d_i)}{\prod_{d_i \in c_x} P(c_x|d_i) \prod_{d_i \in c_y} P(c_y|d_i)}. \quad (5)$$

Note that this updating is local and can be done efficiently since all we have to recalculate from the previous step is the probability

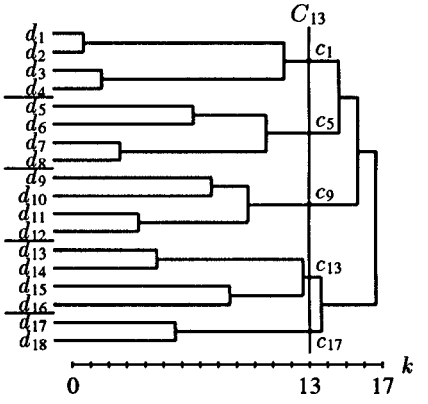

Figure 2: Example of a dendrogram.

for the merged new cluster. For a collection of N documents, merge takes place $N - 1$ times and the last merge produces a single cluster containing the entire document set. Figure 1 shows the HBC algorithm and Figure 2 is an example of a constructed dendrogram. In appendix, we also show a simple program of HBC written in Perl.

For a cluster-based search, we extract a set of clusters by cutting a dendrogram at a certain merge step. Figure 2 shows an example of the set $C_{13} = \{c_1, c_5, c_9, c_{13}, c_{17}\}$ that is extracted just after the merge step 13. There are several possible standards for selecting the best set of clusters. The one used in this paper is to select the set C_k that yields the maximum posterior probability $P(C_k|D)$ over all the merge steps. Another strategy is not to cut a dendrogram but to traverse over the dendrogram on request of each test document to dynamically search for the best clusters for that test document [Jardine and Van Rijsbergen, 1971, van Rijsbergen, 1974, Croft, 1980, Griffiths *et al.*, 1986].

4 Experiments

4.1 Data and Preprocessing

The following two collections of documents were used in our experiments.

Gendai yôgo no Kisotisiki (GK):
A dictionary of contemporary words in Japanese [Jiyû-kokuminsya, 1992]. It contains 18,476 word entries, each of which is classified into one of 149 categories. The length of documents explaining each entry varies from 13 to 1,938 KANJI characters and averages 287 characters. To eliminate the effects of noise, we excluded the entries that include less than 100 terms (nouns) and the categories that include less than 20 entries. The remaining 1,072 entries were classified into 39 categories.

Wall Street Journal (WSJ):
A collection of news stories in English [Liberman, 1991]. We extracted 12,380 articles from '89/7/25 to '89/11/2. Each of the articles is assigned some of 78 categories. After excluding articles that had no category, each of the remaining 8,907 articles had 1.94 categories on the average.

Besides the difference in the languages, a major difference between GK and WSJ is that between the processes used for collecting the data. For GK one editor was responsible for each

category, and this editor not only selected the entries that should be contained in the category but also wrote all of the entries. This contributed to the uniformity of documents within each category. WSJ, on the other hand, not only contained a variety of news stories written by various writers, but categories were later assigned by experts who were not these writers. We used WSJ as a real-world data set containing more noise than did GK, which was used as an well-defined experimental data set.

For preprocessing of documents, we first tagged all the documents by using JUMAN [Matsumoto *et al.*, 1994] for GK, and Xerox Part-of-Speech Tagger [Cutting and Pedersen, 1993] for WSJ. Ispell [Willisson, 1991] was used for the analysis of word stemming in WSJ. GK did not need such a program because there is no inflection of nouns in Japanese. From tagged documents we extracted the root words of nouns as terms and calculated their relative frequencies in order to estimate the probabilities. We did not reduce the number of terms by using stop-word list.

To divide each data set into two sets, one for training and the other for evaluation, different methods were used according to the size of the data set. Since GK contains a relatively small number of entries, a resampling technique of 4-fold cross-validation was used. For WSJ, all stories that appeared from '89/7/25 to '89/9/29 went into a training set of 5,820 documents, and all stories from '89/10/2 to '89/11/2 went into a test set of 3,087 documents.

4.2 Evaluation

We evaluated the performance of the following four category search strategies in the task of probabilistic text categorization.

[full search]

[category-based search]
In case of WSJ, since a document may have several categories, there is overlapping between clusters; that is, the same document might appear within several clusters.

[cluster-based search with nonprobabilistic clustering]
We used Ward's method for nonprobabilistic clustering algorithm because several studies indicate that its performance is better than that of other methods [El-Hamdouchi and Willett, 1986, Griffiths *et al.*, 1986]. The TF-IDF [Salton and Yang, 1973] is used for the term weighting.

[cluster-based search with probabilistic clustering]
We used HBC as the probabilistic clustering algorithm.

The number K of the nearest training documents was set to various values except for category-based search. Category-based search directly ranks all the categories (i.e., clusters) by not using the nearest training documents.

Three category assignment strategies --- k-per-doc, probability threshold, and proportional assignment --- are also compared on WSJ. For GK, since each document has only a single category, only the 1-per-doc strategy was evaluated.

The effectiveness of text categorization was measured as *recall* and *precision* calculated by the following equations:

$$Recall = \frac{\text{the number of categories that are correctly assigned to documents}}{\text{the number of categories that should be assigned to documents}},$$

$$Precision = \frac{\text{the number of categories that are correctly assigned to documents}}{\text{the number of categories that are assigned to documents}}.$$

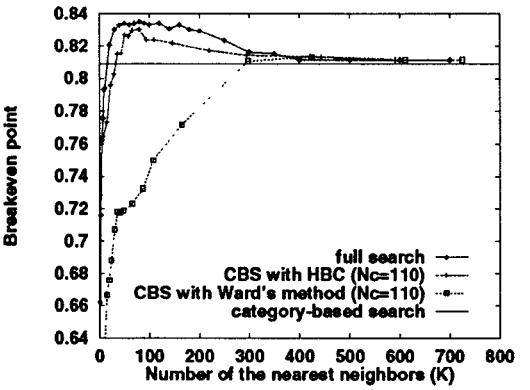

Figure 3: Effectiveness of category search strategies on GK.

Because of the tradeoff between recall and precision, we summarize them into a breakeven point at which they become the same value [Lewis, 1992]. Note here that GK always offers the same value of recall and precision, since only one category was assigned to each document by the editors and 1-per-doc assignment strategy was used by the search programs. For WSJ, we varied the threshold in category assignment strategies and obtained breakeven points.

5 Results and Discussions

Figure 3 compares the effectiveness of the four category search strategies on GK when the number of constructed clusters was set to 110. The number K of the nearest documents varied from 1 to the maximum value (i.e., the number of training documents). Full search offered the best effectiveness unless K was very small. Cluster-Based Search (CBS) with HBC also well approximated the performance curve of full search. This suggests that CBS with HBC would be a balanced model with high efficiency and high effectiveness. CBS with Ward's method, on the contrary, provided poor performance when K was small. One interesting tendency in the performance curves is that all the strategies with complex clustering (i.e., two CBS's) and that with time-consuming cluster search (i.e., full search) converge to the performance of the simple strategy (i.e., category-based search), which does not need complex clustering method and is also efficient in searching clusters. We consider the effectiveness of category-based search as the reference. Other sophisticated search strategies should exceed this reference level because they use more computational resources in clustering or searching. From this viewpoint, CBS with Ward's method is not an appropriate strategy. In later experiments, we only evaluated the performance of CBS with HBC as a representation of CBS's.

Figure 4 shows the results on WSJ. The number of constructed clusters was set to 600, and K varied from 1 to about 1200. The first thing we can see in the figure is the advantage of proportional assignment. Under this assignment strategy, both full search and CBS stay near the category-based search reference level. This result suggests the use of category-based search if the categorization task can be processed in batch where proportional assignment is possible to apply. Under the other two category assignment strategies, both full search and CBS result in a significant improvement over the reference level, confirming the advantage of using these sophisticated search strategies. The difference in the shapes of the performance curves for full search and CBS is also suggestive. For full search, the range of K that offers near the maximum breakeven point is very narrow. That is, soon after the effectiveness reaches at the maximum point it begins to decline steeply toward the reference level. For CBS, on the other hand, the maximum breakeven point

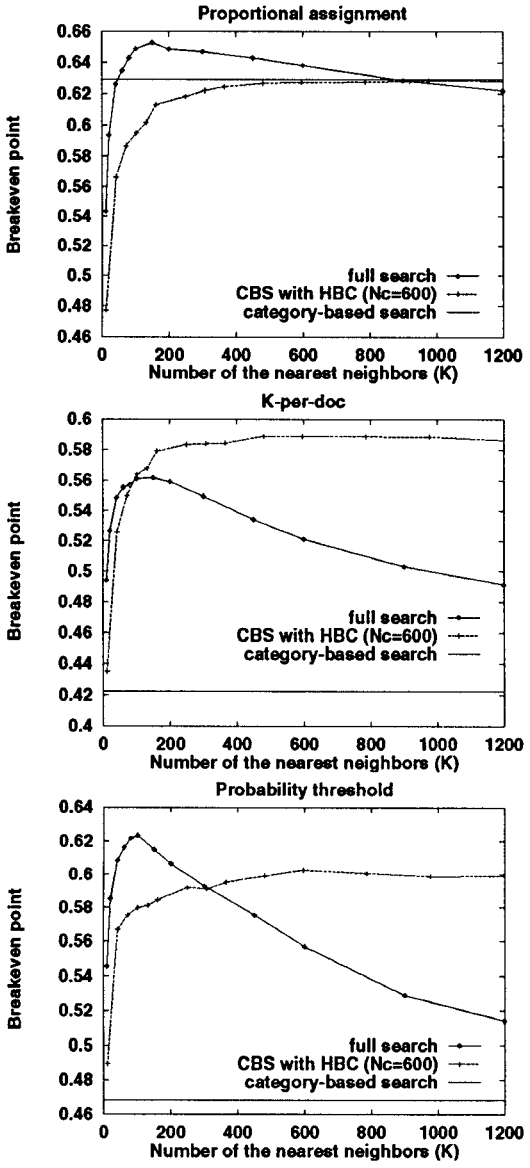

Figure 4: Effectiveness of category search strategies on WSJ.

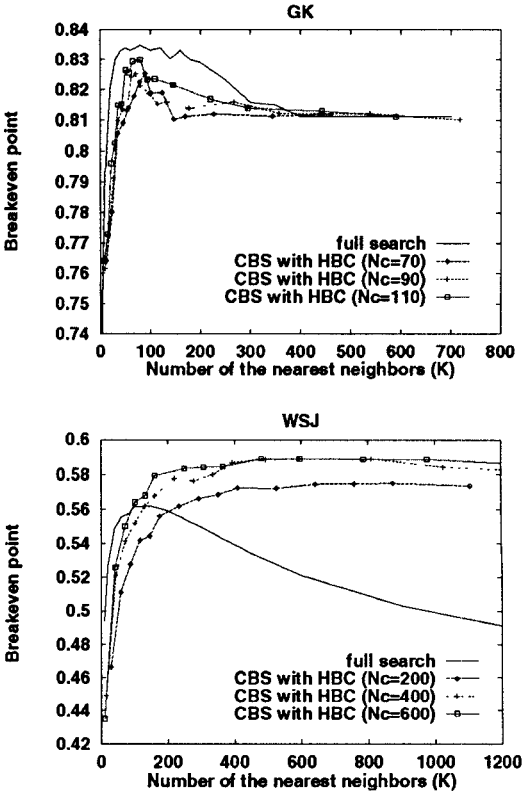

Figure 5: Effect of the number of constructed clusters (N_C).

is maintained over a wide of K and degeneration occurs very slowly. Note also that under the k-per-doc strategy, the maximum breakeven point reached by CBS is higher than that reached by full search. We assume that this is because full search is sensitive to noise in training data; as K grows, more documents that are irrelevant to a test document would thus be easily included into the nearest documents. Since it is difficult to identify the optimum K during the running time of applications, the model is preferred in which the maximum effectiveness is robust against K.

Lastly, we investigated the effect of the number of constructed clusters, N_C, on the performance of CBS. For the GK data set we extracted 110, 90, and 70 clusters from a dendrogram. For the WSJ data set we extracted 600, 400, and 200 clusters. Figure 5 shows the results obtained when k-per-doc was used as the category assignment strategy. We can see that for GK the performance curve of CBS approaches that of full search as N_C increases. Note that the full search corresponds to the extreme case of CBS where N_C is the maximum value (i.e., the number of training data). Since the efficiency in searching clusters also depends on N_C (i.e., $O(N_C \log N_C)$), selecting the appropriate number of clusters would be an important issue. Although we do not discuss this issue more in this paper, one criterion for this selection can be found in statistical model selections like MDL [Rissanen, 1989] and AIC [Akaike, 1974].

6 Summary

This experimental evaluation of category search strategies suggests the following:

- Category-based search and proportional assignment should be used when the categorization task is a batch processing type. This framework would then offer a better performance than others with great efficiency in constructing and searching clusters.
- Otherwise, cluster-based search with probabilistic clustering (for example, HBC) would be a balanced framework with high efficiency and high effectiveness. This framework would also be appropriate when the target data set contains much noise.

Our experiments, like previous experiments [Jardine and Van Rijsbergen, 1971, van Rijsbergen, 1974, Croft, 1980], could not verify that the highest effectiveness of a cluster-based search is necessarily greater than that of a fill search. We should, however, note that the highest performance obtained with the full search was sensitive to the number of the nearest documents, as seen in the experiments on WSJ. With regard to the average performance, the cluster-based search was more effective than the full search owing to the generalization of training documents. Although we should confirm this result on other data sets, the noise tolerance of a cluster-based search is an attractive characteristic of a search strategy used in categorization/retrieval systems.

Acknowledgments

The authors are grateful to Hiroshi Motoda for beneficial discussions, and would like to thank the anonymous reviewers for their useful comments.

References

[Akaike, 1974] H. Akaike. A new look at the statistical model identification. *IEEE Trans. on Automatic Control*, 19(6):716--723, 1974.

[Anderberg, 1973] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[Cormack, 1971] R. M. Cormack. A review of classification. *Journal of the Royal Statistical Society*, 134:321--367, 1971.

[Croft, 1980] W. B. Croft. A model of cluster searching based on classification. *Information Systems*, 5:189--195, 1980.

[Croft, 1981] W. B. Croft. Document representation in probabilistic models of information retrieval. *Journal of the American Society for Information Science*, 32(6):451--457, 1981.

[Cutting and Pedersen, 1993] D. Cutting and J. Pedersen. The xerox part-of-speech tagger version 1.0. `ftp://parcftp.xerox.com`, 1993.

[El-Hamdouchi and Willett, 1986] A. El-Hamdouchi and P. Willett. Hierarchic document clustering using ward's method. In *International Conference on Research and Development in Information Retrieval*, pages 149--156, 1986.

[Fuhr, 1989] N. Fuhr. Models for retrieval with probabilistic indexing. *Information Processing & Retrieval*, 25(1):55--72, 1989.

[Griffiths *et al.*, 1984] A. Griffiths, L. A. Robinson, and P. Willett. Hierarchic agglomerative clustering methods for automatic document classification. *Journal of Documentation*, 40(3):175--205, 1984.

[Griffiths *et al.*, 1986] A. Griffiths, H. C. Luckhurst, and P. Willett. Using interdocument similarity information in document retrieval systems. *Journal of the American Society for Information Science*, 37(1):3--11, 1986.

[Iwayama and Tokunaga, 1994] M. Iwayama and T. Tokunaga. A probabilistic model for text categorization: Based on a single random variable with multiple values. In *Proceedings of 4th Conference on Applied Natural Language Processing*, pages 162--167, 1994.

[Iwayama and Tokunaga, 1995] M. Iwayama and T. Tokunaga. Hierarchical bayesian clustering for automatic text classification. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995. (to appear).

[Jardine and Van Rijsbergen, 1971] N. Jardine and C. J. Van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7:217--240, 1971.

[Jiyûkokuminsya, 1992] Jiyûkokuminsya. Gendai yôgo no kisotisiki. Jiyûkokuminsya, 1992. (in Japanese).

[Kwok, 1990] K. L. Kwok. Experiments with a component theory of probabilistic information retrieval based on single terms as document components. *ACM Transactions on Information Systems*, 8(4):363--386, 1990.

[Lewis, 1992] D. D. Lewis. An evaluation of phrasal and clustered representation on a text categorization task. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37--50, 1992.

[Liberman, 1991] M. Liberman, editor. *ACL/DCI (CD-ROM)*. Association for Computational Linguistics Data Collection Initiative, University of Pennsylvania, September 1991.

[Masand *et al.*, 1992] B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory based reasoning. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59--65, 1992.

[Matsumoto *et al.*, 1994] Y. Matsumoto, S. Kurohashi, T. Utsuro, Y. Myoki, and M. Nagao. JUMAN version 2.0. `ftp://cactus.aist-nara.ac.jp/pub/juman`, 1994.

[Rissanen, 1989] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing, 1989.

[Robertson and Sparck Jones, 1976] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129--146, 1976.

[Salton and McGill, 1983] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Publishing Company, 1983.

[Salton and Yang, 1973] G. Salton and C. S. Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, 29(4):351--372, 1973.

[Stanfill and Waltz, 1986] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213--1228, 1986.

[van Rijsbergen, 1974] C. J. van Rijsbergen. Further experiments with hierarchic clustering in document retrieval. *Information Storage and Retrieval*, 10:1--14, 1974.

[Weiss and Kulikowski, 1990] S. M. Weiss and C. Kulikowski. *Computer Systems That Learn*. Morgan Kaufmann, 1990.

[Willett, 1988] P. Willett. Recent trends in hierarchic document clustering: A critical review. *Information Processing & Management*, 24(5):577--597, 1988.

[Willisson, 1991] P Willisson. Ispell Version 3.0.06. `ftp://ftp.cs.ucla.edu/pub/ispell`, 1991.

A Program

```
#!/usr/local/bin/perl
#
# A Simple Perl Program of HBC
#
# usage: hbc <InputDir>
#
# Object files for clustering should be under
# the directory <InputDir>. The format of an
# object file is like follows.
#
#     <term1> <freq1>\n
#     <term2> <freq2>\n
#       :       :
#
# Each line consists of two fields separated
# by spaces; the first one is the name of the
# term, and the second one is the frequency
# of the term in the object. For example, a
# line "computer 15" means that the term
# "computer" occurs fifteen times in the object
# (i.e. document).

$HUGE = 100000;

if ($#ARGV != 0) {
  die "usage: $0 <InputDir>\n";
}
($Idir) = @ARGV;
$step = 0;

## read documents
##
$NDoc = 0;  # number of documents
$NWord = 0; # number of words
opendir(IDIR, $Idir);
@dlist = grep(!/^\.\.?$/, readdir(IDIR));
closedir(IDIR);
foreach $doc (@dlist) {
  open(DOC, "$Idir/$doc");
  undef(@tmp);
  while(<DOC>) {
    chop;
    ($Word, $Freq) = split;
    push(@tmp, ($Word)); # make word list
    $NWord += $Freq;
    $NWordInD[$NDoc] += $Freq;
    $NWordInC[$NDoc] += $Freq;
    $WFreq{$Word} += $Freq;
    $WFreqInD{$NDoc,$Word} += $Freq;
    $WFreqInC{$NDoc,$Word} += $Freq;
  }
  $NDocs[$NDoc] = 1;
  $DocList[$NDoc] = "$NDoc";
  $WordList[$NDoc] = join($;, @tmp);
  $ID[$NDoc++] = "$doc";
  close(DOC);
}

## initialize
##
for ($c = 0; $c < $NDoc; $c++) {
  # P(c|d)
  $Intra[$c] = &UnitIntra($c);
}
for ($i = 0; $i < $NDoc - 1; $i++) {
  for ($j = $i + 1; $j < $NDoc; $j++) {
    # P(c1 \cup c2|d)
    $Pmatrix{$i, $j} = &MergeIntra($i, $j);
  }
}

## clustering
##
for ($step = 1; $step < $NDoc; $step++) {
  ($c1, $c2, $prob) = &FindClosestPair();
  printf STDOUT ("%d\t%d\t%s\t%s\t%e\n", $c1, $c2,
                 $ID[$c1], $ID[$c2], $prob);
  &MergePair($c1, $c2);
}

## subroutines
##
sub UnitIntra {  # calculate P(c|d)
  local ($c) = @_;
  local ($out, $tmp, $d, $w);

  $out = 0.0;
  foreach $d (split($;, $DocList[$c])) {
    $tmp = 0.0;
    foreach $w (split($;, $WordList[$c])) {
      $tmp += ($WFreqInD{$d,$w}/$NWordInD[$d]) *
          ($WFreqInC{$c,$w}/$NWordInC[$c]) /
          ($WFreq{$w}/$NWord);
    }
    # In clustering we assume that clusters are
```

```
    # equally distributed.
    $out += log($tmp);
  }
  return ($out);
}

sub MergeIntra {  # calculate P(c1 \cup c2|d)
  local ($c1, $c2) = @_;
  local ($out, $tmp, $d, $w);

  $out = 0.0;
  foreach $d (&Union($DocList[$c1],
                     $DocList[$c2])) {
    $tmp = 0.0;
    foreach $w (&Union($WordList[$c1],
                       $WordList[$c2])) {
      $tmp += ($WFreqInD{$d,$w}/$NWordInD[$d]) *
          (($WFreqInC{$c1,$w}+$WFreqInC{$c2,$w})/
          ($NWordInC[$c1]+$NWordInC[$c2])) /
          ($WFreq{$w}/$NWord);
    }
    # In clustering we assume that clusters are
    # equally distributed.
    $out += log($tmp);
  }
  return ($out);
}

sub FindClosestPair {
  local ($i, $j);
  local ($c1, $c2, $maxprob, $prob);

  $maxprob = - $HUGE;
  for ($i = 0; $i < $NDoc - 1; $i++) {
    if ($NDocs[$i] > 0) {
      for ($j = $i + 1; $j < $NDoc; $j++) {
        if ($NDocs[$j] > 0) {
          $prob = $Pmatrix{$i,$j}
                  - $Intra[$i] - $Intra[$j];
          if ($prob > $maxprob) {
            $c1 = $i;
            $c2 = $j;
            $maxprob = $prob;
          }
        }
      }
    }
  }
  return(($c1,$c2,$maxprob));
}

sub MergePair {  # merge c1 and c2 into c1
  local ($c1, $c2) = @_;
  local ($w, $c, $i, $f, @tmp, $c_r, $c_l);

  $NWordInC[$c1] += $NWordInC[$c2];
  foreach $w (split($;, $WordList[$c2])) {
    $WFreqInC{$c1, $w} += $WFreqInC{$c2, $w};
  }
  @tmp = &Union($WordList[$c1], $WordList[$c2]);
  $WordList[$c1] = join($;, @tmp);
  @tmp = split($;, $DocList[$c1]);
  push(@tmp, split($;, $DocList[$c2]));
  $DocList[$c1] = join($;, @tmp);
  $NDocs[$c1] += $NDocs[$c2];
  $NDocs[$c2] = 0;
  $Intra[$c1] = &UnitIntra($c1);
  for ($i = 0; $i < $NDoc; $i++) {
    if (($i != $c1) && ($NDocs[$i] > 0)) {
      if ($i < $c1) {
        $c_l = $i;
        $c_r = $c1;
      } else {
        $c_l = $c1;
        $c_r = $i;
      }
      $Pmatrix{$c_l, $c_r} =
          &MergeIntra($c_l, $c_r);
    }
  }
}

sub Union {
    local ($list1, $list2) = @_;
    local (@tmp, %freq);

    @tmp = split($;, $list1);
    push(@tmp, split($;, $list2));

    foreach $i (@tmp) {
      $freq{$i}++;
    }
    return(keys(%freq));
}
```