Frameworks for Context Recognition in Document Filtering and Classification

Haeng-Kon Kim
Dept. of Computer Information & Communication Engineering, Catholic University of Deagu, Korea
Hae-Sool Yang
Hoseo Graduate School of Venture, Korea

Abstract

Much information has been hierarchically organized to facilitate information browsing, retrieval, and dissemination. In practice, much information may be entered at any time, but only a small subset of the information may be classified into some categories in a hierarchy. Therefore, achieving document filtering (DF) in the course of document classification (DC) is an essential basis to develop an information center, which classifies suitable documents into suitable categories, reducing information overload while facilitating information sharing. In this paper, we present a technique ICenter, which conducts DF and DC by recognizing the context of discussion (COD) of each document and category. Experiments on real-world data show that, through COD recognition, the performance of ICenter is significantly better. The results are of theoretical and practical significance. ICenter may server as an essential basis to develop an information center for a user community, which shares and organizes a hierarchy of textual information.

1. Introduction

Information is often organized into text hierarchies to facilitate information browsing, dissemination, and retrieval. In practice, a text hierarchy is often designed for a specific application, and hence lots of documents in the real world may be entered at any time, but only a small subset of the documents may be classified into some categories in the hierarchy. Therefore, document filtering (DF) and document classification (DC) should be integrated together to classify suitable documents into suitable categories. It aims to reduce information overload while facilitating information sharing. A document is suitable for a hierarchy if the hierarchy contains at least on category that shares enough semantics with the document. Only *suitable documents* are classified into suitable categories. Unsuitable documents should be filtered out of the hierarchy.

This work supported by the MIC, Korea under the ITRC support program supervised by the IITA.

In this paper, we explore how the recognition of the context of discussion (COD) of each document and category may contribute to integrated DF and DC in a text hierarchy. We present a technique ICenter, which employs COD recognition to conduct integrated DF and DC. The basic rationale is: a document could be classified into a category only if its COD matches the category's COD, which depends on the profiles of the category's ancestors. For example, suppose a text hierarchy contains two categories about decision support systems (DSS): (1) "Root → Manufacturing Management → DSS", and (2) "Root → Financial Management → DSS". If a document talks about DSS and its COD is about the usage of DSS in manufacturing (finance), it should be classified into the first (second) category; otherwise it should be filtered out, no matter how and to what extent it talks about DSS, manufacturing, and finance, *individually*. This is the main contribution of the paper.

2. Related work

Table 1 contrasts ICenter with related work. ICenter conducts hierarchical DC, which was a main branch of DC studies. Previous studies developed several hierarchical classifiers [4; 7; 5]. However, DF was not a main functionality of the classifiers, since they were often designed to classify each document into some categories. Moreover, COD recognition was not employed by the classifiers to perform DC either.

ICenter aims to achieve DF and DC by COD recognition. There were studies relying on term contexts to classify documents (e.g. neighboring word strings or linguistic phrases surrounding the terms of interest [3; 8]). These term contexts were quite different from discussion contexts (i.e. COD), which are parts of the semantic contents embedded in documents and categories. There were also studies employing COD recognition to classify queries into most-possible categories [6]. However, the classifiers did not consider DF. When there are many documents not suitable for any category, such a design may classify many unsuitable documents into the hierarchy. In that case, the integration of DF and DC is a must, which calls for significant revisions to the classifier.

Table 1. Contrasting ICenter with related work

Methodology	DC	DF	COD recognition
ICenter	O	O	O
Hierarchical DC	O	X	X
COD-based hierarchical DC	O	X	O
Non-hierarchical DC with thresholding	O	O	X

To conduct integrated DF and DC, previous studies often employed shareholding. Each category was associated with a threshold to autonomously make a yes-no decision for each document entered. A document was "rejected" by a category if its degree of acceptance (DOA) with respect to the category (e.g. similarity with the category or probability of belonging to the category) was lower than the category's threshold; otherwise it was "accepted". When all categories rejected the document, the document was filtered out of the hierarchy.

Obviously, the performance of such classifiers heavily depends on the setting of the thresholds. Previous studies developed two types of thresholds: absolute thresholds and relative thresholds. Absolute thresholds were manually predefined without considering the DOA values of multiple documents (e.g. a Bayesian classifier could accept a document using a probability threshold of 0.5 [2]). On the other hand, relative thresholds were often tuned by performing some statistical analysis on the DOA values of a set of documents (e.g. training documents [2] and validation documents [12]). The thresholds were thus relative to the DOA values of the documents. By analyzing DOA distribution of the documents, the thresholds could be tuned in the hope to optimize the system's performance [2; 9; 10; 12; 14]).

However, both types of thresholds have weaknesses. Absolute thresholds are quite difficult (and even implausible) to predefine properly. On the other hand, as noted above, relative thresholds are derived based on DOA estimations of some documents. The estimations could not be perfect either [1; 14], producing noises to incur improper thresholds.

ICenter employs COD recognition to tackle the weaknesses by employing COD recognition to provide more reliable DOA estimations to make DF and DC decisions. To achieve the mission, previous COD-based classifiers (e.g. [6]) require significant modifications, due to two reasons: (1) previous COD recognition techniques is not suitable for performing DF, since a document may be classified into a category only if its COD matches the category's context at each level of generality (rather than considering the average matching degree at multiple levels, as done by previous COD-based classifiers whose goal was to perform DC by identifying the most possible category for the document), and (2) previous thresholding techniques cannot be suitable for the classifiers to derive COD thresholds, since a category's COD threshold should be tuned by considering both the category and ancestors of the category (rather than considering the category only, as done by previous thresholding techniques).

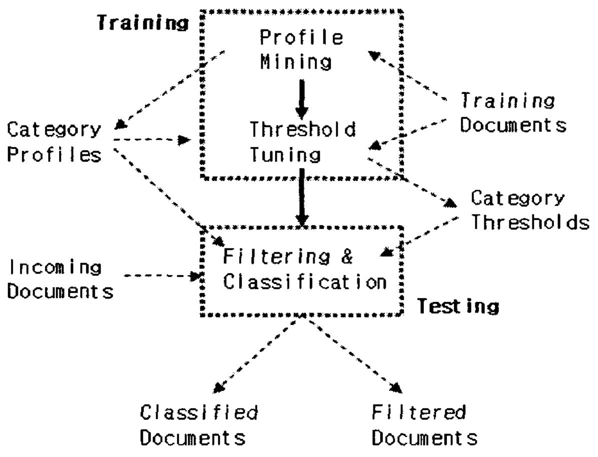


Figure 1. Process flow of ICenter

3. Context recognition for filtering and classification

Figure 1 outlines the process flow of ICenter, which consists of three components: a profile miner, a COD threshold tuner, and a filtering classifier. The former two components are triggered in the training phase, while the third component is triggered once a document is entered. The profile miner identifies content-indicative and generality-indicative features for each category. Based on the profiles mined, the COD threshold tuner estimates the DOA values of all training documents, and accordingly tunes a threshold for each category under the root. Once a document is entered to the system, the filtering classifier consults the profiles and the thresholds to make DF and DC decisions. Next three subsections introduce the three components.

3.1. The profile miner

Table 2 outlines the profile miner. Each category under the root of the hierarchy has a profile. The profile of a category x is a set of 3-tuples <w, $s_{w,x}$, $g_{w,x}$>, where w is a term serving as a profile term (a feature) for x, $s_{w.x}$ is the support of w under x (i.e. including x and its descendents), and $g_{w.x}$ is the strength of w in distinguishing x from siblings of x. That is, $s_{w.x}$ and $g_{w.x}$ consider the distributions of w under x and siblings of x, respectively. A term w is likely to be a good profile term for a category x if (1) it occurs frequently under x (i.e. $s_{w.x}$ is higher), and (2) it occurs relatively infrequently under siblings of x (i.e. $g_{w.x}$ is higher).

Table 2. Mining category profiles

```
Procedure: ProfileMining(c), where c is a category in the
           text hierarchy.
Effect: Build the profile $P_x$ of each descendant category x
        of c.
Begin
 (1) Fore each child category x of c, do
     (1.1)   $Px = \emptyset$
     (1.2)   W={w|w is a word in documents under x,
             and w is not a stop word};
     (1.3)   For each word w in W, do
         (1.3.1) $s_{w.x} = P(w|x)$;
         (1.3.2) $g_{w.x}=P(w|x) \times (B_x/\sum_i P(w|xi))$, where
                 $B_x$ = 1+number of siblings of x, and
                 $x_i$ is the $i^{th}$ child of c, $1 \le i \le B_x$;
         (1.3.3) $P_x=P_x U\{<w, s_{w.x}, g_{w.x}>\}$;
     (1.4)   If x is not a leaf category, recursively
             invoke *ProfileMining*(x) to build the profile
             of each descendant category of x;
End.
```

More specially, $s_{w.x}$ is estimated by P(w|x) (ref Step 1.3.1), which is equal to TF(w,x) / Size(x), where TF(w,x) is the times of occurrences of w in documents under x (i.e. the documents in x and all descendants of x), and Size(x) is the total number of terms occurring in the documents under x. On the other hand, $g_{w.x}$ is estimated by $P(w|x) \times (B_x/\sum_i P(w|x_i))$ (ref. Step 1.3.2), where B_x is one plus the number of siblings of x (i.e. the summation of $P(w|x_i)$ is conducted over x and its siblings). Thus, for example, w may get a higher $g_{w.x}$ if its support under x (i.e. P(w|x)) is higher than its average support under x and siblings of x (i.e. $\sum_i P(w|x_i)/B_x$). In that case, w may be good in distinguishing x from siblings of x. Obviously, $0 \le g_{w.x} \le B_x$, and when w only occurs in x, $g_{w.x} = B_x$ and $g_{w.b} = 0$ for each sibling b of x. If $P(w|x) > \sum_i P(w|x)/B_x$ (i.e. average $P(w|x_i)$), $g_{w.x} > 1$; otherwise $g_{w.x} \le 1$.

Note that a term's g-value may reflect the generality of the term in the hierarchy. A general (specific) term tends to have a higher (lower) g-value in general (specific) categories. Forexample, suppose the text hierarchy contains two categories: (1) "Root → Computer-Based Information System (CBIS) → Decision Support Systems (DSS)" and (2) "Root → CBIS → Management Information Systems (MIS)". The term "computer" may have a higher s-value but a lower g-value in MIS and DSS, since it occurs frequently in both categories. However, both its s-value and g-value may be high in the parent (i.e. CBIS), if siblings of CBIS are not about computer systems. The term "computer" may thus be both representative and general to serve as a good profile term for CBIS (but not for ancestors and descendants of CBIS), and hence may be a good COD indicator for its child categories (i.e. DSS and MIS).

Table 3. Tuning COD thresholds

```
Procedure: CODThresholdTuning(x), where x is a leaf
           category in the text hierarchy.
Effect : (1) For each ancestor a of x, tune a COD
             threshold $h_{a.x}$, and
         (2) Tune a COD threshold $h_{x.x}$ for x.
Begin
  (1) P={P|P is a document belonging to x};
  (2) For each ancestor category a of x, do
      (2.1) UB=Min{$DOA_{p.a}$}, where p $\in$ P, and
            $DOA_{p.a}$ is the DOA value of p with respect
            to a;
      (2.2) $h_{a.x}$=Max{$DOA_{n.a}$}, where n is a document
            not belonging to a, and $DOA_{n.a} \le$ UB;
  (3) Q={q|q is a document not belonging to x};
  (4) For each q in Q, do
      (4.1) For each ancestor a of x
          (4.1.1) IF $DOA_{q.a} \le$ ha.x, Q = Q – {q};
  (5) $h_{x.x}$ = DOAp.x, which maximizes the system's
      performance on P and Q (p$\in$P);
End.
```

3.2. The COD threshold tuner

Table 3 depicts the COD threshold tuner. To derive the thresholds for a leaf category x and its ancestors (for governing the COD of x), ICenter invokes CODThresholdTuning(x). The basic idea is that the thresholds for the ancestors should reflect the *minimum* DOA values of those documents that may be classified into x.

More specially, for an ancestor a of a leaf category x, its threshold is set to make all training documents belonging to x able to pass the test of a (ref. Steps 2.1 and 2.2). Obviously, there might still be documents not belonging to x but able to pass all the tests of the ancestors of x. Let Q be the set of these documents (ref. Steps 3 and 4). To tune a threshold for x, the set Q and the set of those documents really belonging to x (i.e. P in Step 1) are used (ref. Step 5). The threshold is simply the DOA value of a document in P that may maximize the system's performance in a given criterion (e.g. the F-measure).

The contribution of basing thresholding on the set Q deserves discussion. COD recognition helps to produce Q, which excludes those documents that cannot pass the tests of the ancestors. Thresholding may thus be more reliable, since these documents should be noises for thresholding.

Table 4. Estimating DOA values of a document

```
Procedure: DOAEstimation(d), where d is a document.
Effect: For each category c, estimate $DOA_{d.c}$, which is the
        DOA value of d with respect to c.
Begin
  (1) For each category c,
  (2) For each distinct word w in d, do
    (2.1) For each category c, do
      (2.1.1) IF w is a profile term for c and gw.c>1,
        (2.1.1.1) $ng_{w.c} = (g_{w.c} -1)/(\#$ siblings of c);
        (2.1.1.2) rw.c = DFw.c/DFw.root;
        (2.1.1.3) HitScorew.c = $S_{w.c}$ x $ng_{w.c}$ x $r_{w.c}$ x
                  $ts_{w.d}$;
        (2.1.1.4) $DOA_{d.c} \leftarrow DOA_{d.c}$ + $HitScore_{w.c}$;
End.
```

The estimation of the DOA values deserves elaboration as well. Table 4 defines the algorithm for the DOA estimation. The DOA value of a document *d* with respect to a category *c* (i.e. $DOA_{d.c}$) is the summation of the hit scores of the qualified terms (in *d*) on *c* (ref. Step 2.1.1.4). A term *w* in d may be qualified to have a hit score on c (i.e. $HitScore_{w.c}$) if it is a profile term for c and $g_{w.c} > 1$ (ref. Step 2.1.1, also recall that, in this case, w is good in distinguishing c from its siblings).

More specially, the contribution of w to $DOA_{d.c}$ may be more significant if (1) w is content-indicative for c, (2) w is generality-indicative for c, 93) w is strongly correlated with c, and (4) w is content-indicative for d. Therefore, $HitScore_{w.c}$ is equal to $s_{w.c}$ x $ng_{w.c}$ x $r_{w.c}$ x $ts_{w.d}$ (ref. Step 2.1.1.3), where $ng_{w.c}$ is normalized form $g_{w.c}$. The normalization is for maintaining the same scale for DOA estimation across different categories. As noted above, for any term w and any category c, $0 \leq g_{w.c} \leq B_c$, weher B_c is one plus the number of c's siblings. The closer $g_{w.c}$ is to Bc, the more discriminative w would be. Therefore, $g_{w.c}$ is normalized into $ng_{w.c}$ ranging form 0 to 1 (ref. Step 2.1.1.1). On the other hand, $r_{w.c}$ is estimated by $DF_{w.c}/DF_{w.root}$, where $DF_{w.x}$ is the document Frequency of w (i.e. number of documents having w) under category x, and root is the root of the text hierarchy (ref. Step 2.1.1.2). Obviously, $0 \leq r_{w.c} \leq 1$. If all documents containing w are under c, $r_{w.c}$ is equal to 1. Finally, $ts_{w.d}$ is simply [the times w in d/ total number of terms in d].

3.3. The filtering classifier

Based on the profiles mined and the thresholds tuned, context recognition may be conducted for DF and DC. Table 5 defines the algorithm of the filtering classifier. Given a document d, the filtering classifier returns a set of categories to which d may be classified. If the set is empty, d is actually filtered out of the hierarchy. The basic idea is: d may be classified into a category c only if it may pass the tests of c and c' s ancestors under the root. The test of c is for matching the contents of c and d, while the tests of the ancestors are for matching the COD of c and d.

Table 5. Achieving DF and DC by COD recognition

```
Procedure: DOAEstimation (d), where d is a document.
Effect: DF&DC (d), where d is a document.
Return: A set S of categories to which d is classified.
Begin
  (1) Invoke DOAEstimation(d) to estimate DOAd.c,
      for each category c;
  (2) S = ø;
  (3) For each leaf category x, do
    (3.1) IsAccepted = true;
    (3.2) For each ancestor a of x, do
      (3.2.1) IF $DOA_{d.a} \leq h_{a.x}$;
        (3.2.1.1) IsAccepted = false;
        (3.2.1.2) Exit the for-loop;
    (3.3) IF IsAccepted = true,
      (3.3.1) If $DOA_{d.x}$ < hx.x,
        (3.3.1.1) IsAccepted = false;
  (4) IF IsAccepted = true,
    (4.1) S = S U {x};
  (5) Return S;
End.
```

4. Experiment

Experiments on real-world data were conducted to evaluate the contributions of ICenter.

4.1. The Environment

To facilitate objective evaluation and cross-validation, experimental data was extracted from a public database of Yahoo (http://www.yahoo.com). We extracted categories under 5 first-level categories: "science," "computers and Internet," "society and culture," "business and economy," and "Government". The text hierarchy contained 507 categories among which there were 211 leaf categories, which totally contained 3612 documents. Its height was 8.

The amount and distribution of the documents deserve discussion. In the hierarchy, the largest(smallest) leaf categories contained 150(3) documents. Actually the problem of sparse and skewed data was often identified as a practical problem [14; 7]. The text hierarchy may thus help to measure the contributions of COD recognition under such the common practice.

There should be two types of experimental data: in-space data and out-space data. The former was for training the systems and testing DC performances, while the latter was for testing DF performances (since it should be filtered out). Therefore, we randomly and comprehensively removed 20 leaf categories from the text hierarchy (i.e. about 10% of the leaf categories). That is, the documents in these 20 categories served as the out-space data, while the final text hierarchy contained 191 leaf categories (211-20), which served as the in-space data.

To conduct thorough evaluation, we employed 5-fold validation: 20% of the documents in each category were extracted for testing while the other documents were for training, and the process repeated 5 times so that each document was used for testing exactly once. In each fold of the experiment, the out-space data was also used to test the systems' performances in DF.

4.2. Evaluation criteria

DC and DF require different evaluation criteria. For DC, we employed precision (P) and recall (R), which were common evaluation criteria in previous studies. P was estimated by [total number of correct classifications / total number of classifications made], while R was estimated by [total number of correct classifications / total number of correct classifications that should be made]. To integrate P and R into a single measure, the well-known F-measure was employed as well: $F\beta=[(\beta^2+1)PR]/[\beta^2P+R]$, where β is a parameter governing the relative importance of P and R. As in many studies, we set β to 1 (i.e. the F_1 measure), placing the same emphasis on P and R.

To evaluate DF, we employed two criteria: filtering ratio (FR) and average number of misclassifications for misclassified out-space documents (AM). FR was estimated by [number of out-space documents filtered out / number of out-space documents], while AM was estimated by [total number of misclassifications / number of out-space documents misclassified into the text hierarchy]. A better system should reject more out-space documents (i.e. higher FR) and avoid misclassifying out-space documents into many categories (i.e. lower AM).

4.3. Systems evaluated

In addition to ICenter, we implemented a baseline filtering classifier for performance comparison. We focused on those filtering classifiers that do not conduct COD recognition, since previous COD-based classifiers required extensive modifications in order to perform DF and DC by COD recognition (as noted in Section 2). The baseline may thus help to justify the contributions of COD recognition to DF and DC.

The baseline was RO+T, which was implemented by incorporating a thresholding component to a Rocchio classifier (RO). RO is a popular technique routinely applied to similar tasks such as DF (e.g. [10]) and DC (e.g. [11]). As in many previous studies, the thresholding component tuned a relative threshold for each category by analyzing DOA scores of documents, which are estimated by the classifier. As in many studies (e.g. [2]), all training documents were used to tune the thresholds. The thresholds were tuned to optimize the system's performance in F1, which was commonly employed in many previous studies [12].

More specially, RO constructed a vector for each document by employing the popular tfidf technique to estimate the weight of each feature. The profile (i.e. vector) of a category c was set by computing $\eta_1 \times \sum_{Doc \in P} DOC/|P| - \eta_2 \times \sum_{Doc \in N} DOC/|N|$, where Doc was the vector of a document, P was the set of vectors for "positive" documents (i.e. the documents in c), and N was the set of vectors for "negative" documents (i.e. the document not in c). In the experiment, $\eta1$ and $\eta2$ were set to 16 and 4 respectively, since such setting was believed to be promising [11]. Cosine similarity was employed to estimate the DOA score of an input document with respect to each category.

The baseline required a fixed (predefined) feature set, which was built using the training documents. The features are selected according to their weights, which were estimated by the x_2 (chi-square) weighting technique. The technique was shown to be more promising than others [13]. Since there was no perfect way to determine the size of the feature set, we tested 5000, 20000, 40000, and 80000 (almost equal to the total number of different terms in the in-space data).

4.4. Results

Table 6 shows experimental results. Since ICenter did not require feature set tuning, its performances did not vary when the baseline employed different feature set sizes (FS). The results showed that the baseline achieved its best F_1 performance when its feature set size was 40000. Under such a setting, ICenter demonstrated significantly better performance in precision (0.750 vs. 0.643, or 16.6% improvement) and similar performance in recall (0.504 vs. 0.531) and F_1 (0.602 vs. 0.582). This indicated that ICenter was not only more stable (since no system parameter such as feature set size was required), but also more promising in achieving higher-precision DC.

Table 6. Experimental results

Criteria	FS=5000	FS=20000	FS=40000	FS=80000
P(ICenter)	0.750	0.750	0.750	0.750
P(RO+T)	0.518	0.623	0.643	0.645
R(ICenter)	0.504	0.504	0.504	0.504
R(RO+T)	0.471	0.520	0.531	0.528
F1(ICenter)	0.602	0.602	0.602	0.602
F1(RO+T)	0.493	0.566	0.582	0.580
FR(ICenter)	0.739	0.739	0.739	0.739
FR(RO+T)	0.562	0.663	0.696	0.664
AM(ICenter)	1.206	1.206	1.206	1.206
AM(RO+T)	1.458	1.425	1.472	1.484

On the other hand, for DF, ICenter demonstrated better performances on both FR and AM as well. When compared with the baseline using a feature set of size 40000 (as noted above, the baseline performed best in such a setting), ICenter contributed 6.2% improvement on FR (0.739 vs. 0.696) and 18% reduction of AM (1.206 vs. 1.472).

It is interesting to note that, when compared with in-spaced test data, out-space test data was less related to the data for classifier building and threshold tuning. Therefore, the improvements on out-space data further justified the contributions of ICenter: through COD recognition, the system could employ more reliable evidences without over-fitting itself to specific training data. The contributions are also of practical significance, since in practice, there are much more out-space documents than in-space documents.

5. Conclusion

This paper explores *how* and *to what extent* COD recognition may contribute to integrated DF and DC. Given a text hierarchy, ICenter successfully employs COD recognition to reduce the errors of classifying unsuitable documents into unsuitable categories. Moreover, no trial-and-error tuning process is required to build feature sets. The contributions are of theoretical and practical significance. They may be applied to many domains in which information is entered, processed, managed, and shared at any time among a community of users. To further improve ICenter, we are analyzing the errors made by ICenter, and developing effective strategies to tackle them. Analyses indicated that noise filtering and user involvement are two promising extensions.

6. References

[1] Arampatzis A., Beney J., Koster C. H. A., and Weide T. P. van der (2001), KUN on the TREC-9 Filtering Track: Incrementality, Decay, and Threshold Optimization for Adaptive Filtering Systems, Proc. Of the 9th Text Retrieval Conference (Trec-9).

[2] Chai K. M. A., Ng H. T., and Chieu H. L. (2002), Bayesian Online Classifiers for Text Classification and Filtering, Proc. Of ACM SIGIR'02.

[3] Cohen W. W. and Singer Y. (1996), Context-Sensitive Mining Methods for Text Categorization, Proc. Of ACM SIGIR'96

[4] Dhillon I. S., and Kumar R. (2002), Enhanced Word Clustering for Hierarchical Text Classification, Proc. Of ACM SIGKDD 2002.

[5] Koller D. and Sahami M. (1997), Hierarchically Classifying Documents Using Very Few Words, Proc. Of ICML'97.

[6] Liu R.-L. and Lin W.-J. (2003), Mining for Interactive Identification of Users' Information Needs, Information Systems, Vol. 28, No. 7.

[7] McCallum A., Rosenfeld R., Mitchell T., Ng A. Y. (1998), Improving Text Classification by Shrinkage in a Hierarchy of Classes, Proc. Of ICML'98.

[8] Riloff E. and Lehnert W. (1994), Information Extraction as a Basis for High-Precision Text Classification, ACM Transactions on Information Systems, Vol. 12, No. 3.

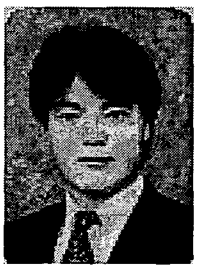
[9] Robertson S. (200), Threshold Setting and Performance Optimization in Adaptive Filtering, Information Retrieval, Vol. 5.

[10] Schapire R. E., Singer Y., and Singhal A. (1998), Boosting and Rocchio Applied to Text Filtering, Proc. Of ACM SIGIR'98

[11] Wu H.., Phang T. H., Liu B., and Li X. (2002), A Refinement Approach to Handling Model Misfit in Text Categorization, Proc. Of ACM SIGKDD'02.

[12] Yang Y. (2001), A Study on Thresholding Strategies for Text Categorization, Proc. Of ACM

SIGIR'01.
[13] Yang Y. and Pedersen J. O. (1997), A Comparative Study on Feature Selection in Text Categorization, Proc. Of ICML'97.
[14] Zhang Y. and Callan J. (2001), Maximum Likelihood Estimation for Filtering Thresholds, Proc. Of ACM SIGIR'01



Dr. Haeng-Kon Kim is currently a professor in the Department of Computer Engineering, and Dean of Engineering College, Catholic University of Daegu in Korea. He received his M.S and Ph.D degree in Computer Engineering from Chung Ang University in 1987 and 1991, respectively. He has been a research staff in Bell Lab. and NASA center in U.S.A. He also has been researched at Central Michigan University in U.S.A. He is a member of IEEE on Software Engineering, KISS and KIPS. Dr. Kim is the Editor of the international Journal of Computer and Information published quarterly by Korea Information Science Society. His research interests are Component Based Development, Component Architecture, & Frameworks Design.

Hae-Sool Yang is a professor at Graduate School of Venture, Hoseo University, Korea. He received his Ph. D. degree in computer science from Osaka University, Japan, in 1990. He was a professor at Kangwon National University, Korea from 1980 to 1995. He was a head of INSQ(Institute of Software Quality) from 1995 to 2002. Currently, he is a vice president of The International Association for Computer & Information Science(ACIS) and Korea Information Processing Society(KIPS). He has been engaged in research in the field of software quality assurance, object-oriented software design & programming, system integration, evaluation of software process & products quality, software project management, component technology. He is a member of KIPS, KISS, IPSJ, ACIS.