A rough set-based case-based reasoner for text categorization

Y. Li [a], S.C.K. Shiu [a,*], S.K. Pal [b], J.N.K. Liu [a]

[a] *Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China*
[b] *Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700 035, India*

Abstract

This paper presents a novel rough set-based case-based reasoner for use in text categorization (TC). The reasoner has four main components: feature term extractor, document representor, case selector, and case retriever. It operates by first reducing the number of feature terms in the documents using the rough set technique. Then, the number of documents is reduced using a new document selection approach based on the case-based reasoning (CBR) concepts of coverage and reachability. As a result, both the number of feature terms and documents are reduced with only minimal loss of information. Finally, this smaller set of documents with fewer feature terms is used in TC. The proposed rough set-based case-based reasoner was tested on the Reuters21578 text datasets. The experimental results demonstrate its effectiveness and efficiency as it significantly reduced feature terms and documents, important for improving the efficiency of TC, while preserving and even improving classification accuracy.

Keywords: Text categorization (TC); Case-based reasoning (CBR); Rough set; Case coverage; Case reachability

1. Introduction

Text categorization, also known as text classification or TC, is the task of using predefined thematic categories to assign labels to natural language texts. In recent years, there has been a tremendous growth in the volume of text documents on the Internet and company-wide intranets, and in digital libraries and news sources. A number of statistically-based text classification algorithms as well as some machine learning techniques have been developed and applied to automatic TC. For example, a sequential algorithm [1] was developed for training text classifiers; a learning algorithm [2] was applied to symbolic knowledge classification in the WWW; a machine learning approach [3] was implemented to find out the interests of users; a technique for sorting electronic mail was developed in [4]; and Yang [5] conducted an evaluation of various statistical approaches to text categorization.

This research, unlike previous work, applies case-based reasoning (CBR) [6], a kind of instance-based learning mechanism, to the task of TC. CBR means reasoning from prior examples. The methodology involves retaining a memory of previous problems and their solutions, and solving new problems by referencing this knowledge [7]. Generally, a CBR reasoner will be presented with a problem and the reasoner then searches its memory of past cases (the case base) and attempts to retrieve a case or cases that most closely match the case under analysis. Retrieved cases are used to provide the solution to the current case. In this research, each stored text document is regarded as a case in the case-based reasoner. An unseen text document which needs to be classified is a problem (also called a query case). Compared with rule-based systems, CBR systems usually require significantly less knowledge acquisition, since they collect a set of past experiences (i.e., cases) without the necessity of extracting a formal domain model from these cases.

In the context of TC, because of the high dimensionality in feature terms and the large number of documents, in this case-based reasoner we incorporate two important processes: feature term reduction and document selection. We also include some TC preprocessing subtasks such as parsing the documents to a set of key words, and the filtering of the common words (also known as stop words). The novel rough set-based case-based reasoner developed in this paper thus consists of four main components: a feature term extractor, a document representor, a case selector, and a case retriever.

To build the feature term extractor, a parser is first used to isolate the words in each document removing the repeated words and retaining distinct ones. Since the stop words which appear in almost every document are considered to contribute little contribution to TC, they are filtered and dropped. The retained set of words is considered to be the original feature terms. A text document often contains often hundreds or even thousands of such feature terms. As leads to low efficiency and low classification accuracy, a feature term reducer is designed for feature term extraction in the proposed rough set-based case-based reasoner.

In this paper, we propose a fast rough set-based feature reduction approach to building the feature term reducer. There is some research work [8,9] that uses rough set theory to discover hidden patterns and dependency relationships among features in information systems. The most important features and patterns can be detected by generating reducts. Unlike other often-used statistical feature reduction methods in TC [10], no additional information about the data such as domain knowledge is required in the process of generating reducts. Some researchers have applied rough set theory to extracting feature terms in text domains. For example, Chouchoulas and Shen [11] proposed a rough set-based approach (RSAR) for TC and tested it using E-mail messages. Based on their work, Bao et al. [12] developed a rough set-based hybrid method using latent semantic indexing (LSI) and rough set theory for TC. Satisfactory text classification accuracies can still be achieved even using very few feature terms. However, due to the large number of feature terms, these methods require a great computational effort during the reduct generation. To make the process of feature reduction less computationally expensive calls for efficient algorithms for generating reducts. In this paper, fast algorithms are developed to generate reducts for building the feature term reducer, computing as the most significant set of features the approximate reduct rather than the exact reduct. The computational complexity of the algorithm is linear with the number of features and cases.

After the reduction of the feature terms, the number of documents (i.e., cases) may still be large, and this will slow down the TC process. Furthermore, text documents may contain redundancy (e.g., duplication of documents) and/or inconsistency (e.g., same feature terms, but different categories), and this will impact classification. Clearly, then, it becomes necessary to build into the case-based reasoner a case selector for case (document) selection.

Various methods on instance selection[1] in the context of machine learning are described and summarized in [13,14]. However, there has not much research work on the issue of document selection for TC. Since most current techniques for instance selection are developed independent of those for feature selection, all the original features are involved in these techniques, and this leads to a high computational load (particularly in TC). Additionally, without feature selection (or feature weight learning), the original large set of features may cause a misleading similarity computation between instances, resulting in an inappropriate instance selection. To tackle these problems, Fragoudis et al. [15] proposed an instance selection strategy for TC (called the FIS algorithm in this paper) which is integrated with the feature selection process. First, an essentially statistics-based feature selection is implemented using a DF-based measure. Those documents which do not contain any selected terms are then removed from the document collection. Experimental results in [15] have shown this document selection strategy to be effective and efficient, however, it can be only preformed on binary classification problems and the document selection is not directly related to the classification quality.

This paper proposes a novel case selection method to be used in the development of a case selector for reducing the size of case base. This method is directly related to the classification performance of a CBR system for TC. The concepts of case coverage and case reachability, which describe the completeness of a CBR system, are used to identify which cases should be removed and which preserved, with the main goal being to reduce the number of cases while maintaining the classification accuracy. Our research group has earlier [16,17] proposed a similar case selection policy for case base maintenance. After the two reduction processes, the most formative feature terms and documents are preserved for the task of categorization. It is guaranteed that in removing features and cases from the case base, minimal useful case base information is lost. This process allows the construction of a reduced case base having many fewer terms and cases. When a query case

[1] In [13], there are two types of instance selections: selection from labeled data or from unlabeled data. In this paper, since the latter type (i.e., instance selection from unlabeled data) is not involved, we assume that "instance selection" is equivalent to "instance selection from labeled data".

is presented to the case-based reasoner, a case retriever computes the similarity between the query case and the cases in the reduced case base, allowing the most similar case(s) to be retrieved to provide the candidate labels for the query case. In this paper, to obtain the final solution, i.e., the predicted label for the unseen document, we use the 1-Nearest Neighbor rule.

The rest of the paper is organized as follows. Section 2 describes the architecture of the case-based reasoner. Section 3 gives the algorithms for building a rough set-based feature term reducer. Section 4 presents the policy for reducing and selecting cases (documents) that are used in building the case selector. This policy is based on the concepts of case coverage and case reachability. In Section 5, reports and analyses experimental results obtained in using the case-based reasoner on the Reuters21578 dataset. Section 6 offers our Conclusion and some directions for possible future work.

2. Architecture of case-based reasoner

This section describes the architecture of the proposed case-based reasoner (Fig. 1). The main components are a feature term extractor, document representor, case selector, and case retriever. Each of these components implements different tasks, including feature term extraction, document representation, case selection, case retrieval, and document label prediction. Since the main differences between the case-based reasoner presented in this paper and a traditional case-based reasoner are to be found in term reduction for the facilitation of feature term extraction and the case selection, these aspects will be dealt with separately, with Section 3 addressing feature term extraction and Section 4 addressing case selection.

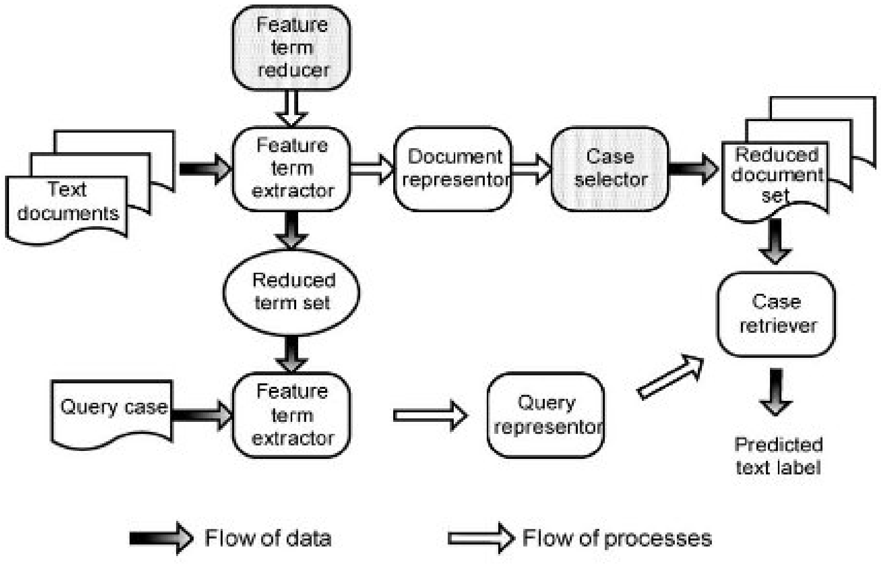

Fig. 1. Architecture of case-based reasoner for TC.

2.1. Feature term extraction

Feature term extraction involves carrying out the following three subtasks.

Step 1: The stop words list is used to isolate and pre-filter words in all of the documents. Stop words are extremely common words which appear in almost every document, such as "a", "the", and "with". In the context of the TC task, these words are often considered to contribute little useful information.

Step 2: Low-frequency words—words which occur just once or twice—are filtered. The words which remain are considered to be the original feature terms.

Step 3: To facilitate the rough set-based feature term reduction, each document is represented using a term vector with respect to the original feature terms acquired in Step 2. Assume there are M terms in the set of original feature terms. A given document D can be described by an M-dimensional term vector $[t_1, t_2, \ldots, t_M]$, where t_k is a Boolean variable which is given by

$$t_k = \begin{cases} 1 & \text{if } D \text{ contains term } k, \\ 0 & \text{if } D \text{ does not contain term } k, \end{cases} \quad k = 1, 2, \ldots, M. \tag{1}$$

Step 4: Reduce the feature dimension using the rough set-based method (to be described in Section 3).

2.2. Document representation

Using the extracted terms in Section 2.1, each document is represented by a term vector. Assume there are m ($m \leqslant M$) feature terms after exaction. Each document is then denoted by an m-dimension vector. An example of a document vector is

$$D = [t_1, t_2, \ldots, t_m], \quad t_k \in [0, 1], \ k = 1, 2, \ldots, m, \tag{2}$$

where t_k is the normalized weight of feature term k in document D. t_k is computed by two steps: weight computation and weight normalization.

Step 1: Weight computation.
Compute the weigh of each feature term in each document using term frequency–inverted document frequency (tf–idf).

$$w_k = -\log(N_k/N)f_k, \quad k = 1, 2, \ldots, m,$$

where w_k is the weight of term k; N_k is the number of documents containing term k; N is the total number of documents; f_k is the frequency of term k.
Note that w_k is the weight of the kth term in the whole set of text documents. Here, in order to reduce the computational load, the term weight for each term in each document is not computed.

Step 2: Weight normalization. Let $w_{\max}$ denote the maximal weight. w_k is normalized to be

$$w_k = w_k/w_{\max}.$$

That is, for each k in Eq. (2), $t_k = w_k$.

2.3. Case selection

This paper proposes a CBR-based case selection approach which can reduce the number of cases as well as preserve the completeness (i.e., the range of problems the system can solve) of the case-based reasoner. To describe the property of completeness, we make use of two CBR concepts, case coverage and case reachability.

In the domain of CBR, it is assumed that the problem space can be well described by the case space. Hence, a case without a corresponding solution can be considered as a problem. Coverage of a case is the range of problems (cases) which can be solved using this case; the reachability of a problem (case) is the set of cases which can solve the case. In various applications, case coverage and reachability have different meanings, depending on the definition of that a case can be "solved" by another case. In this paper, a case q is said to be solved by another case p when (1) p and q fall within the same class; and when (2) the similarity of the two cases is larger than α, where α is the similarity between p and its nearest boundary case. Here a boundary case of p is defined as a case whose class label is different from that of p. The coverage and reachability of a case is then computed-based on the definition of ''solve'' or "be solved''. According to these concepts, the cases with larger coverage sets and smaller reachability sets should make a larger contribution to the completeness to the case-based reasoner. Those case(s) whose coverage set is the largest and whose reachability set is smallest are selected first, and this process of case selection continuous until all the cases in the case base are solved using the selected cases. A threshold can also be given here to determine the size the case base, which would determine when the selection process stops. The case selection algorithm will be given in Section 4.

2.4. Case retrieval and text label prediction

Feature term reduction and case selection generates a smaller case base containing many fewer feature terms and cases. The next step is to build a case retriever which can retrieve the most similar case(s) for the query case in question. Here the query case is also represented by a term vector as mentioned in Section 2.2. By computing the distance between the query and other documents in the vector space, documents with similar semantic content to the query will be retrieved.

To this end, the similarity between cases should be defined and computed. 1-Nearest Neighbour rule is used to determine the retrieved case which will provide the final text label for the query case.

Step 1: Similarity computation.

The similarity of two cases (documents) is defined as the weighted distance

$$\mathrm{sim}(D_i, D_j) = 1 - \sum w_k |t_{ik} - t_{jk}|, \tag{3}$$

where D_i is document vector i, D_j is document vector j; w_k is the weight of term k as mentioned before; t_{ik} is the kth value of document vector i; t_{jk} is the kth value of document vector j.

Step 2: 1-Nearest Neighbor label prediction.

For simplicity, the 1-Nearest Neighbor rule is used to determine the final solution, i.e., the predicted text label for the query case. When a query case (unseen document) occurs, it is first represented using a term vector. After the similarity computation, the most similar case is retrieved to provide a candidate solution. The label of the retrieved case is considered to be the predicted class label of the query case.

3. Rough set-based feature term reduction

To tackle the task of feature term extraction in the case-based reasoner, this paper develops a rough set-based approach. Rough set theory has been successfully applied in a great many domains, such as controlling industrial processes [18,19], diagnosis analysis [20,21], image processing [22], market decision-making [23], environmental problem detection [24], data mining [25], and web and text categorization [26,27]. Rough set theory is used to find reducts in order to extract dependency rules, to discover associate relationship among features, to reduce redundant features, and to select important features [28–32]. Traditional discernibility function-based reduct generation, however, requires a considerable computational effort. For example, if there are N cases and M features, the computational complexity of the regular procedure for computing reducts is $O(M^2 \times N^2)$. Based on the work of Skowron and Rauszer [28], the required computations for reduct generation is $O(M \times N^2)$.

Because of the large number of feature terms and documents, the high computational complexity of the discernibility function-based methods greatly limits the use of rough sets in TC. In this paper, we develop fast algorithms which incorporate the characteristics of TC and reduce the required computation effort to the extent that it is linear with the number of features and the number of cases. Before we describe the feature term reduction algorithms, however, Sections 3.1 and 3.2 describe some related rough set theory concepts.

3.1. Information system and decision table

An *information system* (IS) consists of a triplet $IS = (U, AT, f)$, where U is a finite nonempty set of N objects $\{x_1, x_2, \ldots, x_N\}$; AT is finite nonempty set of M attributes (features) $\{a_1, a_2, \ldots, a_M\}$; $f_a : U \to V_a$ for any $a \in AT$, where V_a is called domain of an attribute a. A *decision table* is an information system $DT = (U, AT \cup \{d\}, f)$, where d is the decision attribute, $d \notin AT$.

In the context of TC, U is the set of documents in the case base; each document is an object $x \in U$. AT is the set of feature terms which describe these documents. Initially, each term that occurs in at least one document is considered as one attribute in AT. d is the class label of the documents, e.g., the topic of each document. Every document can be represented by $f_a(x) \in \{0, 1\}$, $a \in AT$, $x \in U$. When keyword a appears in document x, $f_a(x) = 1$; otherwise, $f_a(x) = 0$.

Each subset of attributes $A \subseteq AT$ determines an *indiscernibility relation* $IND(A)$ on U

$$IND(A) = \{(x, y) \in U \times U \mid \forall a \in A,\ f_a(x) = f_a(y)\}. \tag{4}$$

The relation $IND(A)$ is an equivalent relation and the corresponding equivalent classes construct a partition of the universe U.

A reduct is a key concept in rough set theory and is very useful in knowledge reduction. Reducts are considered to be the essential part in an information system to discern all the objects. A *reduct* is a subset of attributes. These attributes are indispensable in the IS and have the same discriminating power as the original set of attributes. A feature a is *dispensable* in IS if $IND(AT - a) = IND(AT)$; otherwise, a is *indispensable* in IS. In other words, a feature is dispensable in an IS if the discernibility of the IS does not degrade after removing this feature.

3.2. Discernibility matrix and discernibility function

Proposed by Skowron and Rauszer [28], the discernibility matrix and the discernibility function enable the simple computation of reducts. This section presents definitions of these concepts and describes how they are used to generate reducts.

For an IS which has N objects, its *discernibility matrix* (DM) is a $N \times N$ symmetric matrix represented by (dm_{ij}), where

$$dm_{ij} = \{a \in AT : f_a(x_i) \neq f_a(x_j)\} \quad \text{for } i, j = 1, 2, \ldots, N. \tag{5}$$

It is obvious that dm_{ij} is the subset of attributes which can discern object i and object j. The discernibility matrix of an IS completely depicts the ability of the information system to identify the objects in U, and therefore, all reducts of the system are hidden in some discernibility function induced by the discernibility matrix.

It can be easily induced that $B \subseteq A$ is the reduct of A if B is the minimal (with respect to inclusion) subset of A such that

$$B \cap dm_{ij} \neq \emptyset \text{ for any nonempty entry } dm_{ij} (dm_{ij} \neq \emptyset) \text{ in } DM. \tag{6}$$

In other words, a reduct is the minimal subset of attributes that distinguish all objects discernible by the whole set of attributes. The concept of the discernibility function is introduced to describe the reduct computation process.

For any $x_i \in U$, the discernibility function of x_i is defined as

$$f_{DT}(x_i) = \bigwedge_j (\vee dm_{ij} : j \neq i), \quad j \in \{1, 2, \ldots, N\}, \tag{7}$$

where $\vee dm_{ij}$ is the disjunction of all variables s, such that $s \in dm_{ij}$ if $dm_{ij} \neq \emptyset$.

The discernibility function of a given object $x_i \in U$ is the minimal set(s) of attributes which can discern x_i from other objects in U. The reduct can therefore be computed as the conjunction form of the discernibility functions for all the objects.

The discernibility function-based methods for generating reducts are computationally expensive. This is because, first, to calculate the discernibility matrix DM, it is necessary to compare each pair of objects in U and their feature values. If there are N objects and M features, the computational complexity is $O(M \times N^2)$. Next, obtaining the discernibility function for each object $x_i \in U$ requires computing the conjunction of all the elements in the ith row of DM should be computed. This will require $O(M^2 \times N)$ computations. Since there are N objects, the computational load for calculating the discernibility function for all the objects in U is $O(M^2 \times N^2)$. Therefore, the computational complexity of the discernibility function-based methods is $O(M^2 \times N^2)$.

3.3. Reducing dimensions using rough set theory

This section presents three rough set-based feature selection methods, which are built on different reduct computations. In Section 3.3.1, we develop a discernibility matrix-based method incorporating the characteristics of TC. In Section 3.3.2, we describe a relative dependency-based reduct computation method proposed by Han et al. [33]. In Section 3.3.3, based on the work in [33], we develop an approximate reduct-based feature selection method. To demonstrate the reduct computation processes and comparisons of these methods, we will give an illustrative example in Section 3.3.4.

3.3.1. The discernibility matrix-based reduct computation

Here we develop an algorithm for generating reducts that is based on the concept of the discernibility matrix. Unlike the reduct computation in [28], this algorithm incorporates the characteristics of the text domain. The importance of each feature term is taken into account by computing its frequency in the given text dataset. The most significant feature term is the first to be selected in the reduct in the algorithm.

This discernibility matrix-based reduct computation algorithm is described as follows. Step 1 generates the discernibility matrix DM. Step 2 computes the reduct, which is divided into the following substeps. First, the concept of $CORE$ is defined as such a set of dm_{ij} such that each dm_{ij} contains a single feature term which can identify at least two documents, that is, $CORE = \{dm_{ij} \in DM | \text{ card } (dm_{ij}) = 1\}$. The variable $REDU$ is used to store the reduct and is initialized to $CORE$. Then the most frequent feature term is added iteratively to $REDU$ until the intersection of $REDU$ and each dm_{ij} ($1 \leqslant i,j \leqslant N$) is not empty. When the iterations of adding feature terms to $REDU$ stop, the discernibility capability of the set of elements in $REDU$ is the same as that of the original feature terms. From formula 6, it is easy to see that $REDU$ is approximately the minimal set of attributes that preserves the identification capability of the original information system.

Algorithm. Generate reduct

```
Step 1: Create the discernibility matrix $DM = [dm_{ij}]$, $i,j = 1,2,\ldots,N$.
Step 2: Generate the approximate reduct.

        Let $C$ denote the set of original feature terms;
            CORE $= \{dm_{ij} \in DM \mid \text{card } (dm_{ij}) = 1\}$;
            $k$ denotes the number of elements which are not empty in $DM$,
            i.e., $k = |\{dm_{ij} \neq \emptyset\}|$, where $|*|$ is the cardinality of set (*).
        Initialize $REDU$ and $C$:
        REDU = CORE;
        $C = C - REDU$;

    While ($C \neq \emptyset$ and $k \neq 0$) do

        For ($1 \leqslant i,j \leqslant N$)
            {If ($REDU \cap dm_{ij} \neq \emptyset$), $dm_{ij} = \emptyset$;
            Else, break;
            }
        {Sort the term frequency in $C$;
        Select the keyword $t$ having the maximum frequency;
        $REDU = REDU \cup \{t\}$;
        $C = C - \{t\}$;
        }
```

This algorithm is faster than the discernibility function-based method mentioned in Section 3.2. The complexity of this algorithm is $O(MN^2)$, which means its computational complexity has been reduced to be linear with the number of feature terms.

3.3.2. Relative dependency-based reduct computation

The discernibility matrix-based method is still computationally expensive when there are a large number of documents. To further reduce the computational load, Han et al. [33] have developed a reduct computation approach based on the concept of relative attribute dependency. Given a subset of condition attributes B, the relative attribute dependency is a ratio between the number of distinct rows in the sub-decision table corresponding to B only and the number of distinct rows in the sub-decision table corresponding to B together with the decision attributes, i.e., $B \cup \{d\}$. The larger the relative attribute dependency value (i.e., the closer it is to 1), the more useful is the subset of condition attributes B in discriminating the decision attribute values. Some relevant concepts are hereafter defined as:

Definition 1 (*Projection* [33]). Let $P \subseteq A \cup D$, where $D = \{d\}$. The *projection* of U on P, denoted by $\Pi_P(U)$, is a sub-table of U and is constructed as follows:

(1) remove attributes $A \cup D - P$; and
(2) merge all indiscernible rows.

Definition 2 (*relative dependency degree*). Let $B \subseteq A$, A be the set of conditional attributes. D is the set of decision attributes. The *relative dependency degree* of B w.r.t. D is defined as δ_B^D, $\delta_B^D = \frac{|\Pi_B(U)|}{|\Pi_{B \cup D}(U)|}$ where $|\Pi_X(U)|$ is the number of equivalence classes in $U/IND(X)$.

δ_B^D can be computed by counting the number of equivalence classes induced by B and $B \cup D$, i.e., the distinct rows in the projections of U on B and $B \cup D$.

Definition 3 (*consistent decision table*). A decision table DT or U is consistent when $\forall x, y \in U$, if $f_D(x) \neq f_D(y)$ then $\exists a \in A$ such that $f_a(x) \neq f_a(y)$.

Based on these definitions, it can be easily induced that $\delta_A^D = 1$ when U is consistent. A subset of attributes $B \subseteq A$ is found to be a reduct if the sub-decision table is still consistent after removing the attributes in B from A. This is the most important theoretical result and is given as Theorem 1.

Theorem 1. *If U is consistent, $B \subseteq A$ is a reduct of A w.r.t. D, if and only if $\delta_B^D = \delta_A^D = 1$ and for $\forall Q \subset B$, $\delta_Q^D \neq \delta_A^D$ (see [33] for the proof).*

Theorem 1 gives the necessary and sufficient conditions for reduct computation and implies that the reduct can be generated by simply counting the distinct rows in some projections. The computational load is linear to the number of cases N, and the number of attributes M.

3.3.3. Feature selection based on approximate reduct

As noted in Section 3.3.2, although the relative dependency-based reduct computation is fast, in Theorem 1, U is always assumed to be consistent. This assumption is not necessarily true in real life applications. In this section, we relax this condition by finding an approximate reduct rather then an exact reduct. The use of a relative dependency degree in reduct computation is extended to inconsistent information systems. Modifying traditional concepts in rough set theory, we introduce some new concepts, such as the β-dispensable attribute, β-indispensable attribute, β-reduct (i.e., approximate reduct), and β-core. The parameter β is used as the consistency measurement to evaluate the goodness of the subset of attributes currently under consideration. These attributes are explained as follows.

Definition 4 (*β-dispensable attribute and β-indispensable attribute*). If $a \in A$ is an attribute that satisfies $\delta^D_{A-\{a\}} \geqslant \beta \cdot \delta^D_A$, a is called a *β-dispensable* attribute in A. Otherwise, a is called a *β-indispensable* attribute.

The parameter β, $\beta \in [0, 1]$, is called the *consistency measurement*.

Definition 5 (*β-reduct/approximate reduct and β-core*). B is called a *β-reduct* or *approximate reduct* of a conditional attribute set A if B is the minimal subset of A such that $\delta^D_B \geqslant \beta \cdot \delta^D_A$. The *$\beta$-core* of A is the set of β-indispensable attributes.

The consistency measurement β reflects the relationship of the approximate reduct and the exact reduct. The larger the value of β, the more similar is the approximate reduct to the exact reduct computed using the traditional discernibility function-based methods. If $\beta = 1$ (i.e., attains its maximum), the two reducts are equal (according to Theorem 1). The reduct computation is implemented by counting the distinct rows in the sub-decision tables of some sub-attribute sets. β controls the end condition of the algorithm and therefore controls the size of the reduced feature set. It can be determined beforehand by experts or can be learned during the feature selection process. Based on Definitions 4 and 5, the rough set-based feature selection algorithm in our developed approach is given as follows.

```
Feature selection algorithm
Input: U—the entire case base;

        A—the entire condition attribute set;
        D—the decision attribute set.

Output: R—the approximate reduct of A.

Step 1: Initialize R = ∅;
Step 2: Compute the approximate reduct.

        While (A is not empty)
            1. For each attribute a ∈ A
                Compute the significance of a;
```

```
2. Add the most significant one, $q$, to $R$: $R = R \cup \{q\}$;
   $A = A - \{q\}$;
3. Compute the relative dependency degree $\delta_R^D$ for current $R$;
4. If $\delta_R^D > \beta$, return $R$ and stop.
```

Since the computation of the approximate reduct does not increase the computational load of Han et al.'s method, the computation complexities of the feature selection algorithms is also $O(N \times M)$.

3.3.4. An illustrative example

In this section, we use an example to illustrate the three rough set-based feature reduction methods and make some comparisons of them in terms of the generated reducts and the computational complexity.

Let there be an inconsistent decision table $DT1$ (see Table 1), which consists of $N = 9$ cases; $M = 5$ attributes including 4 conditional attributes—a, b, c, d, and 1 decision attribute—e. Since c_1 and c_9 have the same condition attribute values but different decision values, $DT1$ is an inconsistent decision table.

(1) *Discernibility function-based method*

To generate the reduct, firstly, the discernibility matrix (see Definition 1) of $DT1$ needs to be computed as an $N \times N$ matrix. After calculating the discernibility function of each condition attribute, two reducts are found: $\{a, b, c\}$ and $\{a, b, d\}$. In order to compute the discernibility matrix, each pair of cases and their feature values are examined. The required computational load is $O(9^2 \times 5)$.

(2) *Relative dependency-based feature selection method*

This method assumes that the decision table is consistent. If this assumption is not satisfied, the reduct is always found to be the original attribute set. This is demonstrated in the following computation process. Here C is used to denote the set of condition attributes $\{a, b, c, d\}$.

Since $\delta_{C-\{a\}} = \frac{|\Pi_{\{b,c,d\}}(U)|}{|\Pi_{\{b,c,d,e\}}(U)|} = \frac{5}{7} < 1$, $\delta_{C-\{b\}} = \frac{8}{9} < 1$, $\delta_{C-\{c\}} = \frac{6}{7} < 1$, and $\delta_{C-\{b\}} = \frac{3}{5} < 1$, all the condition attributes are considered to be indispensable and should not be removed from C. The reduct that is found is the original set of attributes $\{a, b, c, d\}$.

The relative dependency-based method is implemented by simply counting different rows of the projections on different subset of attributes. Only one pass of the decision table

Table 1
An inconsistent decision table $DT1$

Id	a	b	c	d	e
c_1	1	1	2	1	2
c_2	1	2	1	2	1
c_3	2	2	2	1	2
c_4	3	1	2	2	1
c_5	3	2	2	1	1
c_6	1	2	2	1	2
c_7	3	2	1	2	1
c_8	1	1	1	2	1
c_9	1	1	2	1	1

is required to compute the relative dependency degree for each attribute. Therefore, the computational complexity is O(9×5).

(3) *Approximate reduct-based feature selection method*

To generalize the relative dependency-based method to inconsistent decision tables, we use the procedure given in Section 3.3.3 to generate approximate reducts rather than exact reducts. We set the threshold $\beta = 0.8$ in this example. Here, to evaluate the significance of each attribute, we used the minimum entropy principle. The entropy of each attribute is calculated using the equation

$$E(x) = -\frac{1}{|U|} \sum_{X \in U/\{e\}} \sum_{Y \in U/\{x\}} |X \cap Y| \log_2 \frac{|X \cap Y|}{|Y|}, \quad x \in C.$$

From the entropy computation, we find that $E(a) < E(c) < E(d) < E(b)$. According to the minimum entropy principle, a is first selected in the feature selection algorithm. The current relative dependency degree is $\delta_{\{a\}} = \frac{5}{7} < 0.8$. Attribute c is then added and $\delta_{\{a,c\}} = 4/5 = 0.8$. As this satisfies the end condition, the procedure is complete, ultimately generating the approximate reduct $\{a,c\}$.

Since the computation process is similar to the relative dependency-based method, the computation load is linear with the number of cases and attributes, i.e., the computational complexity is O(9×5).

To evaluate the quality of computed reducts using these methods, we generate a decision table that is similar to $DT1$ (see Table 1) by removing only case 9 to form $DT2$ (see Table 2), a consistent decision table. Using this consistent decision table and the traditional discernibility function-based method, we can find two exact reducts, $\{a,c\}$ and $\{a,d\}$. Since $DT1$ is very similar to $DT2$, the generated reduct that is most similar to $\{a,c\}$ and $\{a,d\}$ is considered to be the most reasonable reduct of $DT1$. It is obvious that the most reasonable reduct is that generated using the approximate reduct-based method (i.e., $\{a,c\}$).

Even after the number of feature terms is reduced, we are faced with the problem that the document collection which will be used as training data in classifying new documents may be still large. This may result in high retrieval costs and low classification accuracy. Therefore, document selection is also an important issue which should be considered in building the case-based reasoner for TC.

Table 2
A consistent decision table $DT2$

Id	a	b	c	d	e
c_1	1	1	2	1	2
c_2	1	2	1	2	1
c_3	2	2	2	1	2
c_4	3	1	2	2	1
c_5	3	2	2	1	1
c_6	1	2	2	1	2
c_7	3	2	1	2	1
c_8	1	1	1	2	1

4. Document selection

In this section, the size of case base is reduced using two CBR concepts: case coverage and case reachability. These concepts describe the problem-solving properties of CBR systems, which are both closely related to the completeness of given CBR systems and affect their performance. The section is organized as follows. Section 4.1 gives the definition of the concepts of case coverage and reachability. Section 4.2 presents our document selection algorithm which is applied in developing the case selector in the case-based reasoner. To compare this document selection algorithm with other methods, Section 4.3 describes another case selection strategy, FIS algorithm.

4.1. Case coverage and case reachability

The CBR methodology assumes that the distribution of cases in the case base provides a good illustration of the distribution of the target problems (i.e., unseen documents). In this view, then, a case in the case base can be considered as a target problem and vice versa.

Coverage of a certain case (see Definition 6, below) is the set of target problems (cases) that the case can be used to solve. The reachability of a target problem (case) (see Definition 7, below) is the set of cases that can solve it. Note that cases with a larger coverage make a greater contribution to the competence of a case base. On the other hand, cases with a larger reachability make a lesser contribution to the competence of a case base. This is because cases with a larger reachability can be solved (reached) by many other existing cases. Clearly, they are significant in selecting/reducing cases for case base construction [2].

Definition 6. Coverage of a case p in a case base CB is defined as

$$\text{Coverage}(p) = \{q | p \text{ can be used to solve } q,\ q \in CB\}.$$

Definition 7. Reachability of a problem (case) p in a case base CB is defined as

$$\text{Reachability}(p) = \{q | p \text{ can be solved by } q,\ q \in CB\}.$$

In various applications, the meanings of the terms case coverage and reachability vary depending on the definition that a case can be "solved" by another case. This paper offers the following more explicit redefinition of the two terms.

Definition 8. *CoverageSet* of a case p is defined as

$$\text{CoverageSet}(p) = \{q | q \in CB,\ \text{sim}(p,q) > \alpha,\ d(p) = d(q)\}, \tag{8}$$

where α is the similarity between case p and its nearest boundary case (the cases which have a different class label of p); d is the decision attribute in D.

Here the coverage set of a case e is the set of cases which fall in the disc centred at p with a radius α. We assume there is only one decision attribute d. It is straightforward to extend this definition to a situation with multiple decision attributes. The definition is illustrated in Fig. 2, where e^* is the nearest boundary case of case e. e_1, e_2, and e_3 are the cases which satisfy

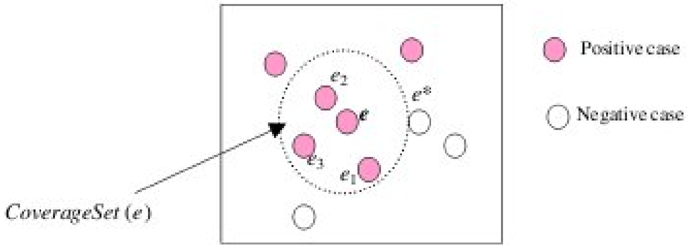

Fig. 2. The CoverageSet of a case e.

$\mathrm{sim}(e, e_i) > \alpha$, where $\alpha = \mathrm{sim}(e, e^*)$,
all of them are positive cases (with the same class label of case e),
$d(e_i) = d(e) =$ positive. $i = 1, 2, 3$.
Therefore, the coverage set of case e is $\{e_1, e_2, e_3\}$.

Definition 9. ReachabilitySet of a case can be derived from Definition 8:

$$\mathrm{ReachabilitySet}(p) = \{q \mid q \in CB,\ p \text{ can be covered by } q\}.$$

For example, in Fig. 2, cases e_1, e_2, and e_3 can be covered by e, e is therefore in their reachability set.

4.2. Document selection algorithm in case-based reasoner

To improve system performance in TC, there is an increasing need to reduce the size of the case base. This suggests that in CBR systems there is always a trade-off between the efficiency and the quality of problem-solving. In making this trade-off, in removing some documents from the case base, it is essential that important information not be lost. In sum, the final constructed case base should be as complete as possible, yet contain the smallest possible number of cases.

In this paper, we propose a reduction policy based on the concepts of case coverage and reachability. The main idea is to preserve such cases that have a larger coverage set but a smaller reachability set. The case selection process is as follows. In Steps 1 and 2, documents are stored and their coverage and reachability are computed, then, in Step 3, the process selects the case with largest coverage. If there are two or more equally large cases, we choose the one with the smallest reachability. If the cases are also identical on this criterion, we opt to select one of the choices at random. The selecting process ends when the selected cases cover all the cases in the case base.

Case selection algorithm (Algorithm 1)

```
Step 1: Initialize document set, $DS = \emptyset$ (empty set). $DS$ will store the set of selected
        documents.
Step 2: Compute the coverage and reachability of every document in $CB$.
Step 3: Select the case which has the maximum coverage.

        If $|\mathrm{MaxCov}| = |\{p\}: \mathrm{Coverage}(p) = \max\{\mathrm{Coverage}\{q\},\ q \in CB\}| = 1$,
            set $D = D \cup \{p\}$;
        Else, select the case that has the smallest reachability.
            If |MinRea| = |{p'}: Reachability(p') = min{Reachability{q}, q ∈ MaxCov}| = 1,
                set $DS = DS \cup \{p'\}$;
            Else, randomly select $p \in$ MaxCov, $DS = DS \cup \{p\}$.
        If (all the cases in $CB$ have been covered by selected cases in $DS$), stop;.

Step 4: Output $DS$.
```

Where $|\cdot|$ is the cardinality of set $(\cdot)$.

Here we should point out that to compute the case coverage of each case, it is necessary to determine a threshold *thr*. In order to avoid excessive information loss, the coverage set of a case p should satisfy not only Eq. (8), but should also satisfy that, if $q \in \text{CoverageSet}(p)$, $\text{sim}(p,q) > thr$, $0 < thr < 1$. The larger the *thr* value, the smaller the number of the selected cases. The classification accuracies are different when different *thr* values are used. *thr* can therefore be determined by the required size of case base or according to a user-predetermined categorization accuracy. In Section 5, our experimental analysis, we shall demonstrate and discuss the relationships between the threshold and the size of case base, and the classification accuracy.

4.3. FIS algorithm

In order to compare our proposed case selection algorithm with other methods, this section presents a document reduction strategy which integrates feature reduction. This strategy is described in the following case reduction algorithm, called FIS (Feature and Instance Selection), which removes documents containing none of the selected feature terms, or in other words, the algorithm preserves documents containing at least one selected feature term. A similar algorithm is proposed in [15], which integrated feature and instance reduction.

FIS (Algorithm 2)
Assume the set of M original feature terms is $\{t_1, t_2, \ldots, t_M\}$; m selected feature terms is $\{t'_1, t'_2, \ldots, t'_m\}$, where $\{t'_1, t'_2, \ldots, t'_m\} \subseteq \{t_1, t_2, \ldots, t_M\}$. Using the document vector representation method, each document D is represented as $[v_1, v_2, \ldots, v_m]$, where

$$v_i = \begin{cases} 1 & \text{if } D \text{ contains } t_i, \\ 0 & \text{if } D \text{ does not contain } t_i, \end{cases} \quad i = 1, 2, \ldots, m.$$

For each document, the corresponding feature values of the selected feature terms $\{t'_1, t'_2, \ldots, t'_m\}$ is denoted by $[v'_1, v'_2, \ldots, v'_m]$.

Initialize document set, $DS = \emptyset$ (empty set). DS will store the set of selected documents.
For every document D in case base CB,

```
    if there exist at least one such $j$ that $v'_j = 1$, for $j = 1, 2, \ldots, m$,
    then add $D$ to $DS$, $DS = DS \cup \{D\}$.
```

(i.e., set $DS = CB$,
for every document D in case base CB,

```
    if for all $j$, $v'_j = 0, j = 1,2,\ldots,m$,
    then remove $D$ from the case base, $DS = DS - \{D\}$.
)
```

The feature reduction and case selection process generates a case base with many fewer feature terms and cases. Using the constructed case base and the methodology of CBR, the document labels are predicted. When a query case (an unseen document) occurs, the most similar case(s) are retrieved-based on a similarity computation. In this paper, the most often used tf–idf is applied in the weight computation of each feature term. A similarity measure is given for the similarity computation. The label of a given unseen document is then predicted using the 1-Nearest Neighbor rule.

5. Experimental analyses

In these experiments, we use text datasets sampled from the Reuters21578 dataset [34]. Table 3 provides details of these datasets. The documents are randomly selected for given topics. Since the main task here is to assign labels to the unseen documents, our main interest is to see what percentage of document class labels is successfully predicted. Here, our main criterion for evaluating the performance of the proposed case-based reasoner for TC is classification accuracy.

For the entire set of distinct words which occur in each text dataset, apply the filtering process using the stoplist and delete the low frequency terms. The distinct words that are consequently preserved are considered as the original feature terms. The initial accuracy is defined as the classification accuracy when using the original text datasets, which have not been reduced either for the number of their feature terms or documents. This accuracy is heavily dependent on the properties of the sampling data, such as the distribution of documents, and the partition of the training data and testing data. While it would also be interesting to try to improve the initial accuracy by analyzing these dataset characteristics, in the focus of this paper is on reducing both the number of feature terms and the number of documents, as well as on maintaining the classification accuracy.

5.1. Feature reduction, document reduction, and classification accuracy

In this section, we present and analyse the experimental results in terms of feature and document reduction and of classification accuracy. Two feature reduction methods are

Table 3
Different text datasets

Text dataset	# Documents	# Distinct words	Initial accuracy (%)
Reut1	110	222	77.8
Reut2	67	151	44.4
Reut3	76	390	41.4
Reut4	46	210	62.5
Reut5	43	256	62.5
Reut6	74	436	53.8
Reut7	1000	653	51.2
Reut8	1578	2018	72.8

used: the discernibility matrix-based method and the approximate reduct-based method. As the discernibility matrix-based method has a relatively high computational load, we make use of only six text datasets, Reut1–6. For the approximate reduct-based feature reduction method, however, we make use of all of the text datasets.

For convenience, we now introduce some notations which will be used throughout this section:

$$\text{Reduced feature} = \frac{|\text{Reduced feature set}|}{|\text{Original feature set}|} \times 100\%,$$

$$\text{Reduced document} = \frac{|\text{Reduced document set}|}{|\text{Original document set}|} \times 100\%,$$

and ΔAccuracy is the difference between the classification accuracy without feature and document reduction and the classification accuracy after feature and document reduction.

5.1.1. Discernibility matrix-based feature reduction and document reduction

Using the discernibility matrix-based feature reduction and document selection in developing the case-based reasoner, greatly reduces the sizes of the original text datasets in terms of both the number of keywords and documents. The classification accuracies change correspondingly.

Table 4 provides results for the accuracy of the feature and document reduction for each of the six datasets. It shows that the features and documents are dramatically reduced. Taking Reut1 as an example, there are 222 different feature words and the initial accuracy is 77.8% (see Table 3). Through generating approximate reduct of the initial feature terms, the number of terms is reduced from 222 to 67, i.e., 69.8% features are removed. Further, based on the concept of case coverage and reachability, the number of cases (documents) in the training dataset is reduced from 41 to 7, i.e., the percentage of reduced documents is about 82.9%. Using the reduced training dataset and feature terms, the problem-solving accuracy is 66.7%, and therefore ΔAccuracy is (77.8%–66.7%) = −11.1%. This experimental result shows that some useful information is lost during the feature and document reduction. In this testing, the parameter *thr* is set to be 0.95. If we set *thr* be larger, the less documents are removed, and the less information is lost. The relationship between *thr* values and accuracy will be analyzed in the following tests.

Using different *thr* values, the amount of reduced documents and the classification accuracy may change correspondingly. Higher accuracy can be achieved by setting larger threshold. Some results are shown in Table 5, where "D-Redu" denotes the percentage

Table 4
Feature and document reduction ($thr = 0.95$)

Dataset	Reduced feature (%)	Reduced document (%)	ΔAccuracy (%)
Reut1	69.8	82.9	−11.1
Reut2	33.1	88.2	+22.3
Reut3	84.1	42.9	+6.9
Reut4	42.4	47.4	−12.5
Reut5	33.6	85.7	−25.0
Reut6	68.8	95.0	−38.5
Average	55.3	73.7	−9.65

Table 5
Threshold *thr* vs. reduced documents and accuracy

thr	Reut1		Reu5	
	D-Redu (%)	Accuracy (%)	D-Redu (%)	Accuracy (%)
0.85	97.6	11.1	–	–
0.90	95.1	33.3	–	–
0.95	82.9	66.7	91.4	37.5
0.98	17.1	77.8	54.3	50.0
0.99	0.0	77.8	22.9	62.5

of reduced documents and "Accuracy" denotes the corresponding classification accuracy using the reduced case base.

Fig. 3 illustrates the relationships between threshold, *thr*, the average of document reduction, and the accuracy of the algorithm when applied to Reut2. When larger thresholds are set, fewer documents are removed but accuracy is improved. The situations are similar for other datasets. In Fig. 3, when $thr = 0.95$, 40% of the documents are removed from the original case base and the accuracy is 60%. When $thr = 0.98$, 20% documents are removed, accuracy is 77.8%. There is always a trade-off between the number of documents removed and the classification accuracy. Different thresholds can be set depending on different requirements of accuracy and case base size.

In this paper, our main goal is to find out the threshold for each text dataset which can produce minimal size of case base and preserve the initial accuracy. For this purpose, we need to determine different thresholds (represented by *thr0*) for different text datasets. Table 6 sets out these thresholds and the degree of document reduction that they produce. Using *thr0* on each dataset removes an average of 36.5% of documents, which is less than using $thr = 0.95$, which removes an average of 73.7% of the documents (see Table 4). As *thr* values do not affect the feature reduction process, using thr0 and other threshold values show no difference in term reduction.

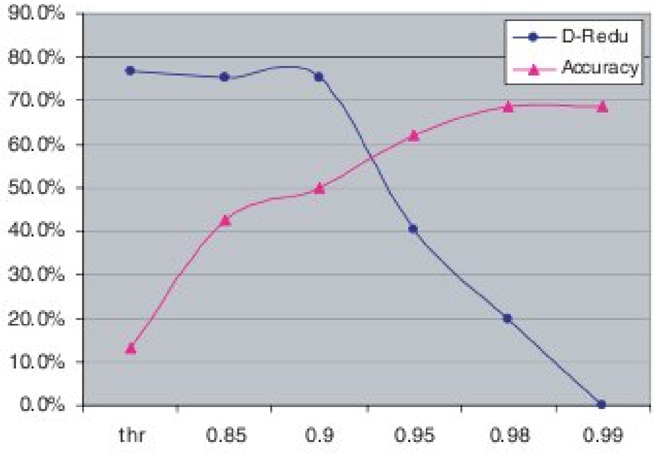

Fig. 3. Threshold vs. average of document reduction and accuracy.

Table 6
Document reduction when using *thr*0

Text dataset	*thr*0	Reduced document (%)
Reut1	0.99	17.1
Reut2	0.95	88.2
Reut3	0.95	42.9
Reut4	0.99	26.3
Reut5	0.99	22.9
Reut6	0.99	21.7
Average	0.98	36.5

5.1.2. Approximate reduct-based feature reduction and document reduction

In this section, we test both the approximate reduct-based feature reduction method and the document reduction together using all eight text datasets. Since this reduct generation algorithm is fast, thousands of feature terms and documents (Reut7 and Reut8) can be handled during feature reduction. In these testing, as the classification accuracy decreases dramatically when $\beta < 1$, we set the parameter β to 1.

Table 7 shows the results of reduced features and documents and the classification accuracy. The approximate reduct-based feature reduction method can be seen to be superior to the discernibility matrix-based method in both feature reduction and classification accuracy. It removes on average 85.9% of features and 37.1% of the documents from the original case base. Further, in most tests it maintained the original classification accuracy and in some text datasets even improved it.

While the results are not provided here, the relationships of the threshold, *thr*, the number of reduced documents and the classification accuracy is the same as can be seen in Table 5 and Fig. 3.

5.2. Problem-solving efficiency: the saved time and cost time

This paper takes the main goals of feature and document reduction as being to reduce the burden of storage and to speed problem-solving. In this section we present some experimental results about the second concern, the relationship between the time taken up in reducing feature terms and cases (documents) and the saved time in classification. The

Table 7
Approximate reduct-based feature and document reduction

Dataset	Reduced feature (%)	Reduced document (%)	ΔAccuracy (%)
Reut1	68.5	85.2	0.0
Reut2	71.5	9.6	0.0
Reut3	89.5	17.1	+3.8
Reut4	91.0	65.1	0.0
Reut5	87.5	12.8	+12.5
Reut6	90.4	5.3	+18.7
Reut7	92.2	46.0	−6.0
Reut8	96.6	56.0	−0.7
Average	85.9	37.1	+3.5

saved time is defined as the difference between the time required to solve a problem using the original, unreduced case base and the time taken to solve a problem using the reduced case base.

In this section, the problem-solving efficiency is described by the ratio between the saved time and the cost time in feature and document reduction. This ratio (i.e., saved time/cost time) is called relative saved time. When the relative saved time is greater than 1 (i.e., the saved time is greater than the cost time), the efficiency of the case-based reasoner is improved; otherwise, the efficiency is degraded.

Figs. 4 and 5 show our experimental results of this relationship between the saved time and the cost time used in feature and document reduction. The x-axis in the two figures is the number of feature terms. The y-axis is the relative saved time, i.e., the ratio between

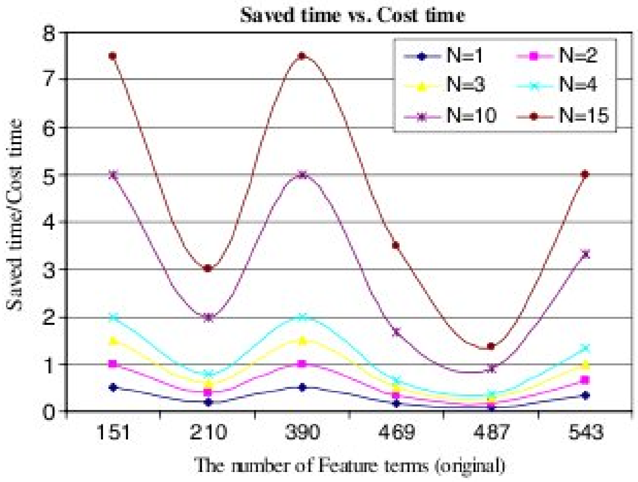

Fig. 4. Saved time and cost time with original number of feature terms.

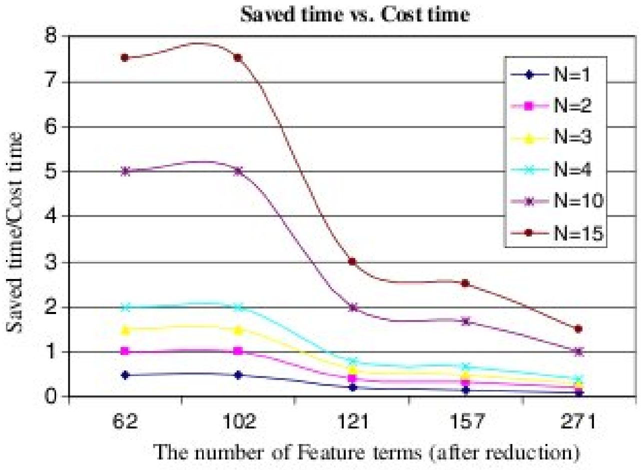

Fig. 5. Saved time and cost time with reduced number of feature terms.

how many seconds are saved using the reduced case base and how many seconds it takes to generate reduct. **N** denotes the number of times that the reduced case base is used in TC. Figs. 4 and 5 show that as **N** increases so does the ratio of saved to cost time proportionally increase. Notice that the cost time is the main time-consumption on generating reducts for the original feature terms. Although this time-consumption is not trivial, this cost can be recovered in the long run with the increasing use of the constructed smaller case base with fewer features and documents. From Fig. 4, it is also observed that there is no necessary relationship between the relative saved time and the number of original feature terms. Nonetheless, in Fig. 5 we note that as the number of selected features rises, the relative saved time falls. In sum, with the increase use of the reduced case base, the efficiency of the case-based reasoner is improved using the reduced case base.

5.3. Comparisons: feature reduction methods and case selection algorithms

This section makes some comparisons to prove the effectiveness of our approximate reduct-based feature reduction method and the developed case selection Algorithm 1. In Section 5.3.1, the approximate reduct-based feature reduction is compared with the DF thresholding method. The main evaluation criteria are reduced features and classification accuracy. In Section 5.3.2, some comparisons are made between the case selection Algorithms 1 and 2, where the reduced documents is the main evaluation index.

5.3.1. Approximate reduct-based feature reduction and DF thresholding method

Document frequency (DF) is a simple and efficient measure for feature term reduction [10]. The feature selection rule in the DF thresholding method is: only those keywords whose DFs are larger than a given threshold are retained. It is one of the most often used methods in TC to limit the number of feature terms. The basic assumption of this method is that rare terms do not contribute useful information to category prediction.

In this section, we make some comparisons between our proposed approximate reduct-based method and the DF thresholding method. The amount of reduced feature terms and the classification accuracy are used as the evaluation indices. Table 8 show the experimental results. In the DF thresholding method, we select the "Threshold" values from [0, 1] during training. For each dataset, the threshold is determined as the smallest value that can preserve the initial classification accuracy. The "Reduced features" is the percentage of the reduced features in the original feature terms. "P(DF)" is the classification accuracy

Table 8
Comparisons between approximate reduct-based method and DF thresholding

Datasets	Threshold	Reduced features (%)	P(DF) (%)	P(AR) (%)
Reu1	0.10	29.7	77.8	77.8
Reu2	0.30	20.5	33.3	33.3
Reu3	0.50	5.1	31.0	41.4
Reu4	0.20	20.0	87.5	62.5
Reu5	0.20	18.8	75.0	75.0
Reu6	0.15	39.0	68.8	75.0
Reu7	0.15	23.1	45.2	50.0
Reu8	0.20	3.7	65.6	69.4
Average	**0.23**	**20.0**	**60.5**	**60.6**

after using the DF thresholding method; while "P(AR)" is the accuracy after using the approximate reduct-based feature reduction. It shows that these two methods can achieve similar classification accuracy (the difference is smaller than 1%), but the DF thresholding has less capability to reduce feature terms. From Table 7, on average, 85.9% of original feature terms can be removed using the approximate reduct-based method. In contrast, there are only 20.0% of feature terms are reduced using the DF thresholding.

5.3.2. *Case selection Algorithms 1 and 2*

This section compares the two case selection algorithms, Algorithms 1 and 2 (Sections 4.2 and 4.3) in terms of their ability to reduce cases. We first use Reut1–8 to test the two algorithms, and then use the modified datasets obtained after adding many duplicated documents in Reut1–8. The experimental results demonstrate that the Algorithm 1 can reduce more documents especially when many of the documents are duplicated.

Table 9 shows the document reduction results on the text datasets Reut1 and Reut2. Here, (1) and (2) denote using Algorithms 1 and 2 to implement the case reduction. It shows that many more documents are reduced using Algorithm 1 than those reduced using Algorithm 2. Since the number of reduced documents is zero in Reut3–8 using Algorithm 2, these results are not included in Table 9. From Table 7, it is obvious that Algorithm 1 removes more documents than Algorithm 2 does in the text datasets Reut3–8. Therefore, using Reut1–8, Algorithm 1 outperforms Algorithm 2 in terms of case reduction ability. Note that the threshold used in Algorithm 1, $thr = thr0$ in different datasets, therefore the classification accuracy is not sacrificed through case reduction.

Next, we give some analyses of Algorithms 1 and 2 when using the modified datasets incorporating duplicated documents. In this situation, the case reduction capability of Algorithm 2 changes with different distributions of duplicated documents. If most of the duplicated documents lack the selected feature terms (which means that most of the duplicated documents will be removed using Algorithm 2), the reduction performance of Algorithm 2 will improve; if the duplicated documents contain at least one of the selected feature terms, the reduction capability will decrease. In contrast, the number of selected cases using Algorithm 1 cannot be affected by the duplicated document. Algorithm 1 can remove all the duplicated cases. Therefore, compared with Algorithm 2, more reasonable and stable performance of case reduction can be achieved by applying Algorithm 1 to the case base.

In the tests of this section, the used modified datasets are the larger versions of Reut1–8 which are achieved by duplicating the documents in Reut1–8 repeatedly. The number of reduced documents using Algorithm 2 is proportional with the number of total documents because the even distribution of duplicated documents. Therefore, the percentage of the selected/reduced documents is the same as that of the original dataset. Using Algorithm 1, the number of selected documents is the same in all the duplicated larger versions of the same dataset. Let r_0 be the original percent of reduced documents using Algorithms 1, r_i be the percent of reduced documents using Algorithm 1 on the ith iteration dataset

Table 9
Case reduction using Algorithms 1 and 2

Dataset	Reduced document (1) (%)	Reduced document (2) (%)
Reut1	17.1	2.4
Reut2	88.2	11.1

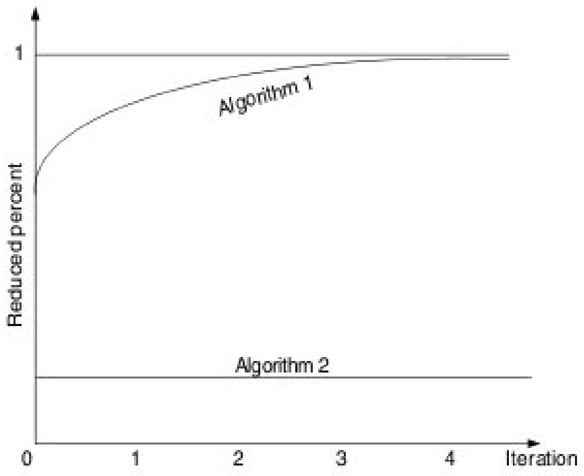

Fig. 6. Comparison of the two algorithms on reducing duplicated documents.

(i.e., the dataset after appending one duplicate of every documents in the $i-1$ iteration). There is a relationship between r_i and r_{i+1} as follows:

$$r_{i+1} = \frac{1}{2} + \frac{1}{2} \times r_i,$$

$$r_{i+1} - r_i = \frac{1}{2} - \frac{1}{2} \times r_i > 0.$$

Fig. 6 shows the document reduction of Algorithms 1 and 2 on the larger version of Reut2. Using Algorithm 1, more documents are reduced when the number of iteration increases. While using Algorithm 2, the percentage of reduced documents does not change with respect to the number of iterations. Thus, Algorithm 1 is shown to be superior to Algorithm 2 in terms of document reduction when there are many duplicated documents.

5.4. Comparisons with 1-NN and SVM classifiers

This section reports on experiments conducted to compare our rough case-based reasoner (or rough CBR classifier) with other classifiers such as the basic 1-Nearest Neighbor (1NN) classifier and the support vector machine (SVM) classifier. SVM is widely regarded as the most successful classification tool, especially in the domain of text categorization. Since most SVM algorithms can handle only binary class problems, we use a text dataset in which the documents are classified as topic ''earn'' and ''nonearn''. In the dataset, there are 100 documents, 280 original feature terms. We use 55 documents as training data and the remaining 45 documents as testing data. The rough CBR classifier uses the approximate reduct-based feature reduction algorithm.

We use three evaluation indices in these comparisons: the number of features, classification accuracy, and the average time for classifying one unseen document. Table 10 shows the comparison results. The rough CBR classifier is shown to be the most promising in terms of all the evaluation indices. It reduces the number of features and documents and consequently reduces the classification time, and also achieves the highest classification accuracy among the three classifiers.

Table 10
Comparisons of 1-NN classifier and SVM classifier

	Number of features	Classification accuracy (%)	Classification time (s)
1-NN classifier	280	68.9	0.07
SVM classifier	280	64.4	0.01
Rough CBR classifier	33	73.3	0.00

6. Conclusions and future work

In this paper, a novel rough set-based case-based reasoner is built to implement the task of TC. Different from the traditional case-based reasoner, we incorporate two components of rough set-based feature term reducer and CBR-based case selector in the reasoner development. The rough set-based feature reduction method is integrated with CBR-based case selection policy to deal with the high dimensionality and large number of documents. To build the feature term reducer, a rough set-based approach based on the concepts of discernibility matrix and approximate reduct is used for reduct computation. On the other hand, to build the case selector, the problem-solving properties of case coverage and case reachability are introduced to address the task of reducing the size of case base. Using the proposed case-based reasoner, we conducted some experiments on the dataset Reuters21578. Different sub-datasets are generated for testing the proposed method. The results show that many fewer feature terms can be selected in each case, and the number of cases can also be reduced depending on the accuracy requirement. Comparisons are made between our proposed case selection algorithm (Algorithm 1) and *FIS* (Algorithm 2). The results show that Algorithm1 demonstrates greater capability to reduce the size of a case base both on the original and the duplicated datasets. Some experiments are also conducted to compare the rough set-based reasoner with the 1-NN classifier and SVM classifier. The rough CBR classifier is shown to be the most promising in terms of classification accuracy and the required time to predict the class label for one unseen document. Future work includes testing the feature and document reduction algorithms in larger text collections. More comparisons should be made between our proposed feature reduction method and other feature reduction/selection or feature weighting techniques. When dealing with the numerical weights, fuzzy sets can be used to better represent the document vectors, instead of considering these numerical weights as Boolean variables.

Acknowledgement

This work is supported by the CERG research grant BQ-496.

References

[1] D.D. Lewis, W.A. Gale, A sequential algorithm for training text classifiers, in: SIGIR'94: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1994, pp. 3–12.
[2] M. Craven, D. Dipasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery. Learning to symbolic knowledge from the World Wide Web, in: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98), 1998, pp. 509–516.
[3] K. Lang. Nesweeder: learning to filter netnews, in: Proceedings of the 12th International on Machine Learning (ICML95), 1995, pp. 331–339.

[4] D.D. Lewis, K.A. Knowles, Threading electronic mail: a preliminary study, Information Processing and Management 3 (2) (1997) 209–217.

[5] Y. Yang, An evaluation of statistical approaches to text categorization, Journal of Information Retrieval 1 (1999) 69–90.

[6] J. Kolodner, Case-Based Reasoning, Morgan Kaufmann, San Francisco, 1993.

[7] S.K. Pal, S.C.K. Shiu, Foundations of Soft Case-Based Reasoning, John Wiley, New York, 2004.

[8] Z. Pawlak, Rough sets, International Journal of Computer and Information Science 11 (1982) 341–356.

[9] Z. Pawlak, Rough sets, Theoretical Aspects of Reasoning about Data, Kluwer Academic, Dordrecht, 1991.

[10] Y. Yang, J.O. Pedersen, A comparative study on feature selection in text categorization, in: Proceedings of ICML-97, 14th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, US, 1997, pp. 412–420.

[11] A. Chouchoulas, Q. Shen, A rough set-based approach to text classification, in: Seventh International Workshop, RSFDGrC'99, Yamaguchi, Japan, 1999, pp. 118–129.

[12] Y. Bao, S. Aoyama, D. Yamada, N. Ishii, X. Du, A rough set-based hybrid method to text categorization, in: Proceedings of the 2nd International Conference on Web Information Systems Engineering (WISE'01), vol. 1, Kyoto, Japan, 2001, pp. 254–261.

[13] L. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artificial Intelligence 97 (1997) 245–271.

[14] D.R. Wilson, T.R. Martinez, Instance pruning techniques, in: Proceedings of the 4th International Conference on Machine Learning (ICML'97), Nashville, TN, 1997, pp. 404–411.

[15] D. Fragoudis, D. Meretakis, S. Likothanassis, Integrating feature and instance selection for text classification, in: Proceedings of SIGKDD'02, Edmonton, Alberta, Canada, 2002, pp. 501–506.

[16] S.C.K. Shiu, C.H. Sun, X.Z. Wang, D.S. Yeung, Transferring case knowledge to adaptation knowledge: an approach for case-base maintenance, in: D. Leake, B. Smyth, D. Wilson, Q. Yang (Special Issue Eds.), Computational Intelligence: An International Journal 17 (2) (2001) 295–314 (Special Issue: Maintaining Case-Based Reasoning Systems).

[17] G. Cao, S.C.K. Shiu, X.Z. Wang, A fuzzy-rough approach for the maintenance of distributed case-based reasoning systems, Soft Computing 7 (8) (2003) 491–499.

[18] A.G. Jackson, Z. Pawlak, S.R. LeClair, Rough sets applied to the discovery of materials knowledge, Journal of Alloys and Compounds 279 (1) (1998) 14–21.

[19] W.C. Chen, N.B. Chang, J.C. Chen, Rough set-based hybrid fuzzy-neural controller design for industrial wastewater treatment, Water Research 37 (1) (2003) 95–107.

[20] P. Mitra, S. Mitra, S.K. Pal, Evolutionary modular MLP with rough sets and ID3 algorithm for staging of cervical cancer, Neural Computing and Applications 10 (2001) 67–76.

[21] F.E.H. Tay, L. Shen, Fault diagnosis based on rough set theory, Engineering Applications of Artificial Intelligence 16 (1) (2003) 39–43.

[22] S.K. Pal, P. Mitra, Multispectral image segmentation using rough set initialized EM algorithm IEEE trans, Geoscience and Remote Sensing 40 (11) (2002).

[23] L. Shen, H.T. Loh, Applying rough sets to market timing decisions, Decision Support Systems 37 (4) (2004) 583–597.

[24] N. Chèvre, F. Gagné, P. Gagnon, C. Blaise, Application of rough sets analysis to identify polluted aquatic sites based on a battery of biomarkers: a comparison with classical methods, Chemosphere 51 (1) (2003) 13–23.

[25] C.C. Chan, A rough set approach to attribute generalization in data mining, Information Sciences 107 (1–4) (1998) 169–176.

[26] R. Jensen, Q. Shen, Fuzzy–rough attribute reduction with application to web categorization, Fuzzy Sets and Systems 141 (3) (2004) 469–485.

[27] A. Chouchoulas, Q. Shen, Rough set-aided keyword reduction for text categorization, Applied Artificial Intelligence 15 (9) (2001) 843–873.

[28] A. Skowron, C. Rauszer, The discernibility matrices and functions in information systems, in: R. Slowinski (Ed.), Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory, Kluwer Academic, Dordrecht, The Netherlands, 1992, pp. 331–362.

[29] R.W. Swiniarski, Rough sets methods in feature reduction and classification, Applied Mathematics and Computer Science 11 (3) (2001) 565–582.

[30] R.W. Swiniarski, A. Skowron, Rough set methods in feature selection and recognition, Pattern Recognition Letters 24 (2003) 833–849.

[31] H.S. Nguyen, A. Skowron, Boolean reasoning for feature extraction problems, in: Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems, 1997, pp. 117–126.
[32] J. Wang, J. Wang, Reduction algorithms based on discernibility matrix: the ordered attributes method, Journal of Computer Science & Technology 16 (6) (2001) 489–504.
[33] J. Han, X. Hu, T.Y. Lin, Feature subset selection based on relative dependency between attributes, in: Proceedings of the 4th International Conference of Rough Sets and Current Trends in Computing (RSCTC04), Springer-Verlag, Berlin, 2004, pp. 176–185.
[34] D.D. Lewis, Reuters-21578 text categorization test collection distribution 1.0. Available from: http://www.research.att.com/~lewis, 1999.