Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

Genetic algorithm rule based categorization method for textual data mining

Mohammed H. Afif[a,b*], **Abdullah Saeed Ghareb**[cd], **Abdulgbar Saif**[cd], **Azuraliza Abu Bakar**[c] **and Omer Bazighifan**[e]

[a]*Department of Management Information systems, Faculty of Business Administration, Prince Sattam Bin Abdulaziz University, Saudi Arabia*
[b]*Department of Management Information Systems, Faculty of Adiminstrative Sciences, Hadramout University, Yemen*
[c]*Department of Computer Information Systems, Faculty of Information Technology and Computer Science. University of Saba Region, Marib, Yemen*
[d]*Center for Artificial Intelligence Technology, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia*
[e]*Department of Mathematics, Faculty of Education, Hadramout University, Seiyun, Yemen*

C H R O N I C L E

Article history:
Received May 7, 2019
Received in revised format: August 25, 2019
Accepted August 25, 2019
Available online August 25, 2019

Keywords:
Rule based categorization
Text categorization
Genetic Algorithm
Rule discovery
Categorization rule
Associative classification

A B S T R A C T

The rule based categorization approaches such as associative classification have the capability to produce classifiers rival to those learned by traditional categorization approaches such as Naïve Bayes and K-nearest Neighbor. However, the lack of useful discovery and usage of categorization rules are the major challenges of rule based approaches and their performance is declined with large set of rules. Genetic Algorithm (GA) is effective to reduce the high dimensionality and improve categorization performance. However, the usage of GA in most researches is limited in the categorization preprocessing stage and its results is used to simplify the categorization process performed by other categorization algorithms. This paper proposed a hybrid GA rule based categorization method, named genetic algorithm rule based categorization (GARC), to enhance the accuracy of categorization rules and to produce accurate classifier for text mining. The GARC consists of three main stages; namely, search space determination, rule discovery with validation (rule generation), and categorization. The experimental results are carried out on three Arabic text datasets with multiple categories to evaluate the efficiency of GARC. The results show that a promising performance was achieved by using GARC for Arabic text categorization. The GARC achieves the best performance with small feature space in most situations.

1. Introduction

Text categorization is a data mining technique that attempts to categorize text documents based on their contents (Harrag & Al-Qawasmah, 2010). The continuous updating of texts in several media needs an efficient mining technique to explore the useful knowledge. The traditional categorization techniques are the most frequently used for text categorization. These include the Support Vector Machine (SVM) (Mesleh, 2011; El-Halees, 2008), Naïve Bayes (NB) (Al-Saleem 2011), Neural Network (NN) and K-Nearest Neighbor (KNN) (Abu Tair & Baraka, 2013; Bawaneh et al., 2008). The rule-based categorization techniques are rarely investigated for Arabic text categorization (Al-diabat, 2012; Khorsheed & Al-Thubaity, 2013). The research effort for this type is very limited when compared with other techniques, but rule-based techniques can achieve competitive results (Ghareb et al., 2016; Al-diabat, 2012; Al-Saleem, 2011). The Associative Classification (AC) (Al-Radaideh et al., 2011; Al-

* Corresponding author. Tel.: +967715339998
E-mail address: m.afiff@psau.edu.sa (M. H. Afif)

doi: 10.5267/j.dsl.2019.8.003

Saleem, 2011), decision tree (DT) (C4.5) and One-Rule (Al-diabat, 2012; Thabtah, 2007; Thabtah et al. 2011, 2012) approaches, which categorize text documents based on categorization rules, have been used to categorize Arabic text in previous works. However, such studies have not specifically studied the impact of feature selection (FS) methods on the rule discovery process. In addition, when adopting rule-based approaches, many challenges should be addressed such as designing a discovery process that includes high-quality categorization rules mining and producing useful rules that cover all training text categories (Al-diabat, 2012; Al-Radaideh et al., 2011; Abbas et al., 2011; Abu Tair & Baraka, 2013; Hattab & Hussein, 2012). Consequently, the lack of useful categorization rules that cover all text categories is a drawback and should be considered when a rule-based classifier is utilized for text categorization. Hence, the use of the GA as a categorization algorithm can create a new text classifier that combines several advantages. The utilization of the advantages of the GA is required as an alternative solution to discover useful categorization rules that have enough predictive power to discriminate different categories of textual datasets through the GA's search capability.

The GA is an evolutionary algorithm (population-based algorithm) that was first proposed by Holland (1975). It emulates the evolution process in nature and it is intended to conduct a search for an optimal solution to a given problem by mimicking natural selection. The GA constructs a population of solutions usually randomly and then applies genetic operators such as crossover and mutation to improve the solutions in order to find the optimal solutions of the given problem. Fig. 1 presents the main steps of the GA (Tan et al., 2008). When applying the GA as an FS method, the search space becomes the feature space. The GA starts by creating the initial population, which is composed of a set of feature subsets (chromosomes), where each subset represents one solution. The GA tries to evolve the best subset by selecting the best subsets (solutions) according to a selection strategy that depends on fitness function. The fitness function identifies the fittest subsets and these subsets survive into the next generation through crossover and mutation reproduction steps. The crossover operation takes two feature subsets (two parents) and reproduces them to create two new subsets (children), whereas the mutation operation modifies a single subset by changing some of the features' values or replacing them randomly in that subset. The three most important aspects to consider when using a GA as a FS approach are the: (i) definition of the fitness function (evaluation), (ii) definition of the selection strategy, and (iii) definition and implementation of the crossover and mutation operators.

```
Population = generate random population;
Repeat the following until stopping criterion is reached
Evaluate the fitness of each solution in the population;
Create a new population by repeating the following steps:
        Select two parents;
        Apply genetic operators (crossover and mutation);
        Update population;
Go to evaluation step;
End
```

Fig. 1. Pseudocode for the traditional GA

Several researchers have demonstrated the advantages of using a GA in a hybrid approach with filter FS methods to solve high dimensionality and FS problems (Gunal, 2012; Lei, 2012; Tsai et al., 2014; Uğuz, 2011; Uysal & Gunal, 2014; Gharib et al., 2009). Many hybrid approaches based on the GA have been proposed for text categorization in some recent works. For instance, Uysal and Gunal (2014) proposed a hybrid approach based on filter methods with GA and Latten Semantic Indexing, while Tsai et al. (2014) employed a biological evolution concept to improve the GA, and Uğuz (2011) proposed a hybrid approach based on information gain (IG) and the GA. Another combination of filter and GA was proposed by Gunal (2012), in which the features are first selected by four filtering methods, Chi Square (CHI), mutual information (MI), document frequency (DF) and IG, and combined together as an input for GA in the second stage. Fang et al. (2012) also investigated the performance of the combination of DF, IG, MI, CHI methods with the GA, while Lei (2012) employed the IG with the GA as an FS method for text categorization. These approaches are effective in reducing text dimensionality and improving the performance of text categorization. However, usage of the GA is limited in the categorization preprocessing stage and its results are used to simplify the categorization process that is performed by other categorization algorithms. Hence, the use of the GA as a categorization algorithm can create a new text classifier that combines several advantages. Particularly, the GA can be considered a rule-based classifier; it can construct categorization rules efficiently for each text category due to its strength to reduce useless knowledge and combine category features that have the highest importance to discriminate the different text categories. Nevertheless, rule-based classifiers in general suffer from a lack of useful categorization rules, which degrades categorization performance especially with high dimensional textual datasets. Another problem is the huge number of categorization rules and their distribution among different categories of datasets, which has a direct impact on the categorization decision. In addition, the automatic discovery process of categorization rules is sensitive to noise in textual datasets. Therefore, this paper handles these challenges by utilizing the GA as a rule-based text categorization algorithm.

In this paper, the GA is employed to discover the categorization rules and build a text classifier directly from the extracted rules. The categorization rules are logical rules in the form: "If condition then category", where the condition (rule body) is a set of features that discriminates the text category in a given dataset and the rule head is a predefined category in that dataset. This type of classifier produces competitive results to those of traditional probabilistic and similarity techniques. However, the major challenge of rule-based techniques is the discovery of useful categorization rules, and as they are sensitive to noise it is also difficult to identify the border between distinct categories within the same text dataset, and so the discovered rules may overlap different categories. In short, the problem is the lack of useful categorization rules that cover all portions of the studied text dataset and the ability of rules to discriminate the different categories of all the text categories. The proposed method addresses these problems by using GA to identify the rule conditions that have the highest fitness and highest rank to form useful categorization rules. The proposed method is called the Genetic Algorithm Rule-based Classifier (GARC).

The rest of this paper is organized as follows: Section 2 describes the proposed categorization method based on the GA. Section 3 presents the experimental results and a discussion and comparisons. Section 4 contains an overall discussion of the best proposed methods and the findings in this research and compares the results of this research with those of related works. Finally, Section 5 summarizes this paper.

2. Proposed GARC Approach

This section describes the use of the GA as a rule-based classifier (named GARC). In this categorization method, the GA is utilized as a learning technique that attempts to improve and generate categorization rules automatically for categorizing Arabic text datasets. The basic elements of the GARC are population generation, fitness function, selection and GA operators (crossover and mutation). The objective of the proposed method is to discover the categorization rules by finding the best combination of rule features that are extracted from a given text dataset by using the GA search capability. The description of the GARC algorithm is presented in Fig. 2. To construct the categorization rules using GARC, training text documents, which contain a set of features and their categories, are desired. The input is a set of features for all categories and the output is a set of categorization rules for all categories. The GARC consists of three main stages: search space determination, rule discovery and validation (rule generation), and categorization. The features for each category are extracted in the first stage by employing a filter FS method; the method works on positive documents for each category in the training dataset and identifies the candidate features that determine the search space and they are used in the population generation process of the GARC. The GARC learning stage (rule discovery and validation) consists of many steps that are repeated **t** times according to the number of generations. At each generation, the best **n** chromosomes are saved, which are comprised of a set of best categorization rules. When the search is completed, the best chromosomes are ordered based on their fitness, the best N chromosomes are selected and a dictionary is constructed that contains the categorization rules for all categories. After the learning process, the GARC is then used to categorize the new text documents which are unseen and not utilized in the learning process. The following subsections explain each stage of the GARC.

```
GARC Algorithm
Input: set of features F ($f_i$, $c_i$) for the training datasets with their known categories; number of generations; number of iteration, population size. Prior information: IG for all features.
Output: set of categorization rules ($f_1$& $f_2$ ... & $f_n \longrightarrow c_i$) // (GARC classifier)
Begin
Stage 1: Determine the search space
-    Select a predefined number of features for each category to form the search space
Stage 2:Rule discovery and validation
While (iteration < max)
    ✓  Create the population from the selected feature space (Old-Pop)
    ✓  Repeat the following until maximum number of generation:
    Step1: Evaluate the fittest of each chromosome in the population;
        New- Pop = empty;
        save the best chromosomes of old Population
    Step2: Create new population (New-Pop)
        select parent1 and parent2 in Old-Pop using roulette wheel selection;
        generate child1, child2 through crossover (parent1, parent2);
        apply mutation;
        add child1, child2 to new Population;
        Old-Pop = New-Pop;
        when reach the population size; go to step 1;
 End-repeat;
End while.
Order the best chromosome based on their fitness values;
Select the best N chromosomes;
For each selected chromosome Ni
    Divide Ni by category
    Construct a dictionary (a set of rules) for each category ($f_1$& $f_2$& ... & $f_n \longrightarrow c_i$);
    Add each category with their rules to global dictionary ($R_i \longrightarrow c_i$);
    Eliminate redundancies from each rule and redundant rules;
End for.
For each extracted rule in global dictionary:
    Get the IG value of rule condition (features) & compute rule support
    Order & Prune rules based on IG & support
    Update the global dictionary (return the best categorization rules that form GARC);
End for.
Stage 3: Categorization using GARC
-    Use the discovered categorization rules to assign the test documents to their favored categories,
-    Evaluate categorization performance
End.
```

Fig. 2. Stages of the proposed GARC

2.1 Search space determination

In this stage, the search space, which comprises a set of features and their categories, is determined based on feature extraction. A set of features is selected after text preprocessing using filter FS methods to reduce the search space and to reduce the randomization effect and computational cost. Three selection methods are employed for this purpose: the class discriminating measure (CDM) (Chen et al., 2009) and odd ratio (OR) (Mladenic & Grobelnik, 1999) and term frequency-inverse document frequency (TF-IDF) (Salton & Buckley, 1988). The mathematical symbolization of these methods which is computed for each feature *f* in a category c_i are as follows:

$$CDM = \left| \frac{\log P(f \mid c_i)}{P(f \mid \hat{c}_i)} \right|, \tag{1}$$

$$OR = \frac{P(f \mid c_i) \times \left(1 - P(f \mid \hat{c}_i)\right)}{P(f \mid \hat{c}_i) \times \left(1 - P(f \mid c_i)\right)}, \tag{2}$$

$$TF - IDF = TF(f, c_i) \times \log\left(\frac{N}{DF(f)}\right), \tag{3}$$

where *P(f|ci)* is the percentage of documents in the category *ci* and feature *f* occurs at least once; $P(\hat{f} \mid c_i)$ is the percentage of documents that belong to category *ci* and does not contain feature *f*; $P(f \mid \hat{c}_i)$ is the percentage of documents that does not belong to category *ci* but contains feature *f*; $P(\hat{f} \mid \hat{c}_i)$ is the percentage of documents that does not belong to category *ci* and does not contain feature *f*; *N* is the total number of training documents in the collection; and *TF (f, ci)* is the frequency of feature *f* in category *ci*.

2.2 Rule discovery and validation

The GARC performs the search process over the search space to generate the categorization rules in four stages; initialization (population generation), rule evaluation and selection, crossover and mutation and finally rule revision. The components needed to construct rules using GARC are explained in the following sub-sections.

2.2.1 Initialization stage

In the initialization stage, a population is generated randomly from the selected feature space that is determined in the initial stage. The population interties (chromosomes) are selected as the starting point of the search process, and each chromosome is comprised of a set of features that are distributed among all categories of the dataset. Thus, each chromosome represents a number of rules (candidate rules) equivalent to the number of categories in the dataset. For example, if there are five categories in a given dataset, then each chromosome will be composed of five rules, one rule for each category.

2.2.2 Rule and population evaluation

For the rule and population evaluation step, the rule accuracy in terms of F-measure is employed and then the fitness function, which determines the fitness of each chromosome in the population, is calculated accordingly. As mentioned above, each chromosome is comprised of a set of rules for all categories, thus each individual chromosome is considered to be a classifier. Each chromosome is divided to *N* parts based on the number of categories in the training dataset, and each part represents one rule for exactly one category. To evaluate each rule, the *F*-measure is used to identify the positive and negative text documents in the training dataset; in other words, it measures the match between a given rule and documents that are associated with a given category. The rule accuracy is computed for each rule *R* that belongs to a category *c* inside a chromosome *I* as defined in Eq. (1). The chromosome accuracy is the sum of the rules' accuracy divided by the number of rules. Thus, the fitness of each chromosome is computed based on chromosome accuracy and size as defined in Eq. (2). The best chromosomes found at each generation are saved and they are also used in the mutation operation to increase algorithm convergence and maintain the categorization rules that have high accuracy.

$$Accuracy\ (R) = \frac{2\ (tp)}{tfp + tp + fn} \tag{4}$$

$$Fitness\ (chrom) = Z\ \times \left(\frac{\sum_{i=1}^{R} Accuracy\ (R)}{R}\right) + (1 - Z) \times (1\ /\ size\ (chrom) \tag{5}$$

where Accuracy (*R*) is the accuracy of a given rule, *tp* is the number of positive documents that match fully or partially the rule for given category, *tfp* is the total number of documents that match the rule, *fn* is the number of negative documents (from other categories) that do not match the rule, *R* is the number of rules in the chromosome, size (chrom) is the number of associated features with the rules that are presented by this chromosome, and *Z* is a control parameter in the interval $[0,1]$ that determines the importance of chromosome accuracy and size in the fitness function.

2.2.3 Selection for reproduction

The selection of chromosomes for reproduction is based on fitness proportionate with the roulette wheel selection method (Uğuz, 2011). In this method, the probability of selecting a chromosome *y* for reproduction is calculated as follows:

$$P\ (y) = \frac{Fitness\ (y)}{\sum_{i=1}^{n} Fitness\ (y)}, \tag{6}$$

where *Fitness (y)* is the fitness value of chromosome *y,* and *n* is the population size.

2.2.4 Crossover and mutation

A crossover operator produces two children chromosomes from two parent chromosomes that are selected in the reproduction stage. The crossover and mutation are applied for the categorization rules of each chromosome based on chromosome partitioning, weight of each part and rule accuracy instead of using fixed probability rate for chromosomes to be crossover or mutated. The chromosome partitioning in the crossover is based on the number of rules in each parent. A one-point crossover for each rule is employed, where each rule is divided into two parts. The weight of each part is calculated as the cumulative weight value of the features in this part based on the feature frequency and document frequency approach. The features of each category (i.e. the rule associated features) in each parent are ordered based on their weight, so the weight is obtained and the cumulative rule weight is computed and then the best two parts of each rule are concatenated together to form a new rule in the new chromosome (first child) and the other two parts form the second rule in the second child. This process is repeated for each rule for each parent and the crossover operation continues until the two parents are combined. The mutation operation performs changes after crossover to new chromosome interties (rules) based on rule accuracy. For each rule *r* that belongs to a category c_i in a chromosome y, if the average accuracy of the original parents is lower than a given threshold, then the rule is mutated by replacing a specific number *n* of its associated features that have low weights by an equivalent number of features from the best found chromosome in the previous generations where the features also belong to category ci and are not associated with the mutated rule. Thus, the subject chromosome is mutated by replacing some of the features that are associated with each rule in this chromosome.

2.2.5 Rule revision: ranking and pruning

Further operations are applied to the best found chromosomes. The chromosomes are ordered based on their fitness values and the best *N* chromosomes (those with the highest ranks) are selected. Consequently, the categorization rules are extracted for each category of the dataset. Each selected chromosome is divided into *x* groups based on the number of rules that are already associated with their given categories in the dataset. Thus, a local dictionary is built for each category comprising a set categorization rules for each category and a global dictionary is also built comprising all extracted categorization rules with their known categories. Furthermore, the categorization rules are ordered and reduced to a reasonable number of rules according to local information gain (IG) and rule support. The initial rule pruning is applied locally for each rule by removing feature redundancies. The redundant rules that belong to the same category are also discarded. Hence, the IG is calculated for each revised rule by computing the IG of each feature with respect to its labeled category. The IG value of each rule is the summation of all IG values of rule features as defined in Eq. (4).

For example, the IG of rule (Ri $\rightarrow$ c1) in the template $(\text{if f1, f2, f3} \rightarrow \text{c1})$ is IG (f1, c1) + IG (f2, c1) + IG (f3, c1).

$$IG\ (R) = \sum_{f=1}^{F} P(f,c)\ \times log\ \frac{P(f,c)}{P(f) \times P(c)} \tag{7}$$

In Eq. (7), *f* is the total number of features in rule *R*, *P (f, c)* is the probability of a feature *f* given category *ci*, *P (f)* is the probability of feature occurrence in the dataset, *P(c)* is the probability of category *ci*, which is computed as the ratio of documents belonging to *ci* with respect to all the documents in the dataset.

The rule support is the percentage of documents for a given category in the training dataset that include the features of a given rule. The rules for each category are ordered according to the following procedure:

For a given two rules, R1 and R2, R1 has a higher rank than R2 if:

- IG (R1) > IG (R2),
- Else if IG (R1) = IG (R2) but support (R1) > support (R1).

In addition, after rule ordering, rule pruning is also applied based on rule support and IG values, where the rules that have the smallest IG values and have support of less than 5% are eliminated from the rules list. Thus, the strong rules contribute to the final categorization in the categorization (testing) stage.

2.3 Categorization using GARC

The rules that are discovered are used for categorizing new documents in the test dataset, which are comprised of test documents for all categories. The purpose here is to assess the precision of the GARC in making categorization decisions based on a list of rules. The testing process is performed based on a set of categorization rules that are ordered in a decision list for all categories; the document to be classified is matched with this list of rules, the category of rules that partially or fully matches the test document is then assigned to the test document. When the returned rules belong to many categories, majority voting is used; this method biased to the highest number of category rules that cover the test document to be categorized, where the document is assigned to the category that has the largest number of matched rules (Ghareb et al., 2014).

3. Experimental Results and Evaluation

The experiment was carried out on three Arabic text datasets: Al-jazeera text dataset (D1), Akhbar Al-Khaleej text dataset (D2), and Al-waten text dataset (D3) (Abbas et al., 2011; Chanter & Corne, 2011; Ghareb et al., 2016), Table 1 shows the distribution of documents into the different categories for each dataset. The datasets were partitioned into training and test datasets without overlap. In the case of D1, 80% was used for GARC learning and 20% for testing and in the cases of D2 and D3, 70% was used for GARC learning and 30% for testing. The experiment tested the capability of the GARC method to discover useful categorization rules and produced a highly accurate categorization. Table 2 presents the experimental setup for assessing the performance of the GARC algorithm. Three FS methods (TF-IDF, MCDM and OR2) were used to determine the search space; the size of the search space that was used in the first stage ranges from 1000 to 5000 features. In the second stage, the population size was set to 32 chromosomes and when the search was completed the best 32 chromosomes were returned. After that, a set of categorization rules was extracted and utilized for categorization. The maximum number of rules was restricted by the number of chromosomes in the population; each category had 32 rules and the discovered rules were also revised according to IG and rule support. For the fitness function, the control parameter (Z) was set to 85%, which means chromosome accuracy was given higher importance than chromosome size. The proposed method was implemented in a C# programming environment, and all experiments were performed on a PC with an Intel(R) Core™ i3 processor, 2.27 GHz, 4 GB RAM, and Windows 7 operating system. The performance evaluation measures included the macro average of precision, recall and F-measure (Mesleh, 2011). The classifier size was also considered in terms of the number of categorization rules for each experimental dataset.

Table 1
Distribution of the experimental text datasets

D1		D2		D3	
category	# documents	category	# documents	category	# documents
Economy	300	Economy	273	Religion	3860
Politics	300	Sports	429	Economy	3468
Sport	300	Local news	720	Sports	4550
Science	300	International news	286	Culture	2782
Art	300	Total	1708	International news	2035
Total	1500	# words	746,307	Total	16695
# words	389,766	-	-	# words	8,351,615

Table 2
Experimental setup for GARC method

Search space	1000 to 5000 features	Number of iterations	40
Population size	32	Number of generations	5
Fitness function	Combine chromosome accuracy and size	Rule ordering & pruning	Based on IG and rule support
Selection method	Roulette wheel	Final *N* best chromosomes	32
Crossover	Based on parent division and rules weight	Maximum number of rules	32 × number of text categories
Mutation	Replace 20 features based on parents' accuracy		

3.1 Results of GARC for Multi Search Spaces

This section presents the computational results of the GARC. Specifically, it determines the effectiveness of the GARC in improving the categorization process based on a set of useful categorization rules. Table 3 shows the performance of the GARC for each dataset (D1, D2 and D3) when the search space (feature space) was identified based on the top ranked features using CDM. As can be seen from Table 3, the GARC achieved the highest average for all measures with D1, the highest macro average of recall and F-measure for D1 were obtained when 3000 of the top ranked features were used, while the highest macro average of precision was obtained with 4000 features. In the case of D2, the highest macro average of recall and F-measure were achieved when 4000 of the top ranked features were used; however, for D3, the GARC achieved the best performance with 1000 features. In addition, the GARC achieved the highest precision for both D1 and D2 when 1000 of the top ranked features were used. On average, the GARC performed best for D1 followed by D3 and the lowest average was obtained for D2.

Table 3
Performance of GARC (macro average %) for each dataset based on the top ranked features using CDM

	D1			D2			D3		
Search space	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
1000	90.768	90.166	90.466	85.933	74.322	79.707	93.142	85.343	89.072
2000	90.391	90.403	90.397	85.209	76.322	80.522	86.228	73.067	79.104
3000	90.862	90.779	90.820	80.136	75.831	77.924	86.651	76.557	81.292
4000	91.117	89.905	90.079	84.315	79.729	81.958	86.426	80.079	83.132
5000	89.305	88.975	89.139	79.897	76.312	78.064	84.087	73.187	78.259
Average	90.4886	90.0456	90.1802	83.098	76.5032	79.635	87.3068	77.6466	82.1718

The GARC performance with two other FS methods (OR and TF-IDF) was also examined. Table 4 and Table 5 show the performance of the GARC for each dataset (D1, D2 and D3) when the search space was identified based on the top ranked features that were selected by using OR and TF-IDF, respectively. From the above tables it can be seen that the GARC achieved the best results on D1 with both FS methods. In addition, the GARC performed well with a small feature space in most situations. For example, the GARC obtained the best macro average of precision (91.394%), recall (93.73%) and F-measure (92.547%) with TF-IDF when 1000 of the top ranked features were used.

Table 4
Performance of GARC (macro average %) for each datasets based on the top ranked features using OR

	D1			D2			D3		
Search space	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
1000	83.223	82.796	83.009	76.717	77.405	77.059	83.707	71.649	77.209
2000	82.520	81.709	82.113	76.078	74.489	75.275	80.694	69.373	74.607
3000	84.447	83.363	83.902	72.802	73.091	72.946	79.449	66.205	72.225
4000	82.431	81.92	82.175	73.255	74.692	73.967	81.230	67.586	73.782
5000	84.274	83.770	84.021	74.637	74.876	74.757	76.014	63.847	69.401
Average	83.379	82.7116	83.044	74.6978	74.9106	74.8008	80.2188	67.732	73.444

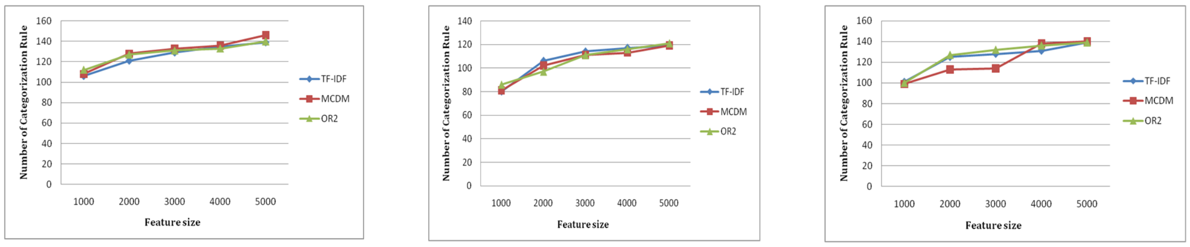
Table 5
Performance of GARC (macro average %) for each dataset based on the top ranked features using TF-IDF

	D1			D2			D3		
Search space	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
1000	91.394	93.730	92.547	70.633	72.802	71.701	85.287	80.605	82.879
2000	87.976	91.315	89.614	67.181	66.872	67.026	83.048	76.408	79.589
3000	84.699	72.045	77.862	81.025	68.737	74.377	82.458	78.658	80.513
4000	86.837	76.182	81.161	81.108	69.519	74.868	84.477	74.644	79.257
5000	84.290	74.677	79.193	75.828	68.538	71.998	81.112	59.994	68.973
Average	87.0392	81.5898	84.0754	75.155	69.2936	71.994	83.2764	74.0618	78.2422

Comparing the results of the GARC in Table 4 and Table 5 with those in Table 3, it can be seen that the GARC always exhibited the best performance with CDM (Table 3) for all text datasets. It achieved the best macro average of precision for both D2 and D3 (85.933%, 93.142%) with a small feature subset (1000 features) and the best macro average of recall and macro-average F-measure when the number of features was 1000 in the case of D3 and 4000 features in the case of D2. The results indicate that satisfactory performance was achieved by using the GARC for Arabic text categorization. The GARC was able to work better with a small feature space (i.e. 1000 features) and achieve the best performance in most situations. A possible explanation for this result is that the number of categorization rules for this level of feature size was smaller than that of other feature spaces. Also, the number of features in each rule was smaller than that of other feature spaces. This indicates that when the number of categorization rules is increased the performance decreases because a large number of rules participate in the categorization decision, which results in categorization mistakes; however, in general the categorization mistakes were reduced by GRAC because it restrict the number of rules in the categorization decision. The next section discusses the classifier size in terms of the number of categorization rules.

3.2 GARC Size

The number of categorization rules for each of the FS methods for each text dataset is plotted in Fig. 3. As shown in this figure, the number of categorization rules increases as the number of features increases. The number of rules at each level of feature size is slightly different between FS methods, but it should be noted that the length of rules is different where each rule has its own associated features. With respect to categorization performance, the best performance was achieved with the smallest number of rules in most cases with all text datasets. This means that the proposed GARC was able produce useful rules with a small feature subset that were able to make an accurate categorization. The GARC generated the smallest number of rules with TF-IDF on D1 and D2 and with MCDM on D3.



(a) Classifier size for D1 (b) Classifier size for D2 (c) Classifier size for D3

Fig. 3. GARC size with three FS methods for each dataset

In summary, the GARC was able to produce a limited number of categorization rules that have good predictive power to differentiate the categories of text and achieve promising results for Arabic text categorization. Regarding the problem of lack of useful categorization rules, the proposed method was able to discover categorization rules that have good predictive power to discriminate text categories. As the results indicate, the proposed method was able to reduce feature dimensionality and produce revised and useful categorization rules at the same time. In the literature, some of the rule inductions such as AC and One-Rule classifier fail to discover rules for all text categories. In addition, in some cases, the AC generates a large number of rules that are unbalanced for all categories, thus categorization performance in those cases declines. However, the proposed GARC ensured that all the training text categories were covered by the discovered rules. Another advantage of the GARC is its capability to reduce the number of categorization rules and at the same time reduce the number of features associated with each rule. To sum up, the experimental results show that the GARC method has the capability to produce efficient categorization rules with promising results in terms of categorization performance and classifier size.

3.3 Comparison of the GARC with the Associative Classification Approach

In this section, the computational results of the GARC are compared with Associative Classification (AC); both methods are rule-based categorization algorithms. A set of categorization rules was discovered by each method, then revised and used for categorization. The same settings of the GARC parameters (Table 2) were used, while the parameters of the AC were as follows: 10% for minimum support and 50% for minimum confidence. A detailed description of AC can be found in (Thabtah, 2007; Al-Radaideh et al., 2011; Ghareb et al., 2014). The majority voting method was utilized by both the GARC and AC in the categorization phase to determine the valid categories for new test documents. The comparison was based on categorization performance, classifier size and modeling time in terms of the total time needed for categorization with each method. The GARC was also compared with another version of the AC when the CDM and GA were employed as a preprocess of AC to reduce feature dimensionality.

Table 6
Comparison of GARC with ACs based on the categorization performance (macro average F-measure %)

	GARC			AC1			AC2		
	Performance (F_{macro})			Performance (F_{macro})			Performance (F_{macro})		
	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.
D1	89.14	90.82	90.18	59.11	88.31	77.16	57.12	90.28	73.77
D2	77.92	81.96	79.64	59.54	68.10	64.33	61.67	78.29	71.58
D3	78.26	89.07	82.17	73.63	82.19	77.21	64.74	84.73	74. 81

Table 6 shows the comparison of GARC performance with two variations of AC: AC with CDM (AC1) and AC with CDM-GA (AC2). In addition, Table 7 shows the size of the GARC compared with the two variations of AC. The comparison between the GARC and AC was carried out based on their results in terms of minimum (Min.), maximum (Max.) and average (Avg.) of categorization performance and classifier size when the CDM was used to determine the search space for the GARC and AC. The categorization performance was measured in terms of macro-average F-measure (F-macro), while the classifier size was measured in terms of number of categorization rules. The categorization performance results in Table 6 show that the GARC outperformed the AC methods and achieved the best min, max and average of macro average F-measure with all datasets. The AC1 method achieved the second highest average in the case of D3. The performance of both variations of AC declined with large feature subsets due to the lack of useful categorization rules and because a large number of rules were used for categorization. The proposed GARC avoided this limitation by revising the rules more times and restricting the number of rules that could participate in the categorization decision. Overall, from the comparison results, it is clearly demonstrated that the performance of the proposed GARC was better than that of the AC approaches. In addition, the proposed GARC was more efficient in terms of classifier size. As shown in Table 7, the size of the GARC is smaller than the ACs and it is reasonable for all datasets. The average number of rules with the GARC did not exceed 130; whereas in the case of the ACs a large number of rules was generated, which had an adversely impact on categorization performance.

Table 7
Comparison of GARC and ACs based on the classifier size

	GARC			AC1			AC2		
	Classifier size | #Rules |			Classifier size | #Rules |			Classifier size | #Rules |		
	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.
D1	108	146	130	92	647	275	201	1770	930
D2	81	119	105	118	263	154	122	1266	483
D3	99	140	121	112	438	204	95	3219	1352

The comparison between the GARC and AC approaches in terms of time consumption is demonstrated in Fig. 4.

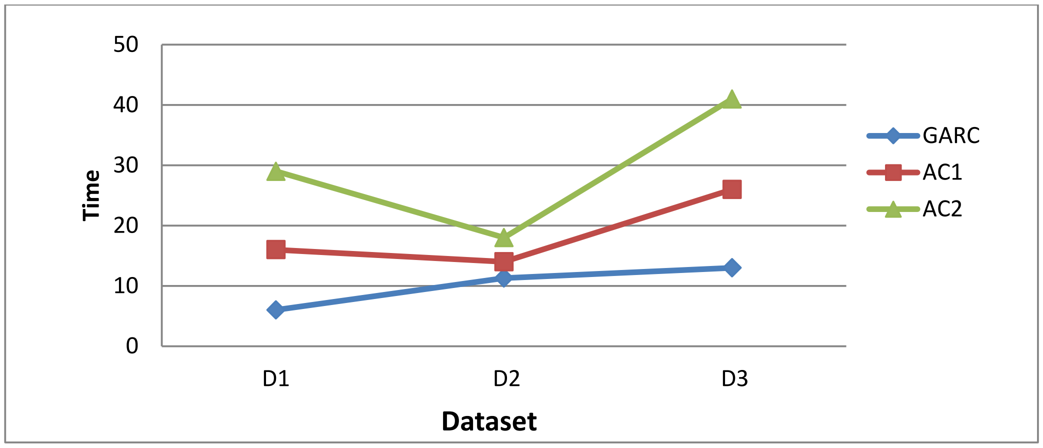


Fig. 4. Comparison of time consumption (minute) for GARC and AC approaches for each dataset

The results show that the GARC outperformed both AC1 and AC2. This is because the GARC generates the categorization rules and performs categorization in a reasonable time with all datasets compared to AC1 and AC2. The long period of time required when using AC1 and AC2 is because a large set of frequent features has to be generated, especially with large feature subsets, through an iterative search for each dataset, which requires a lot of time for rule discovery, so they produce a large set of categorization rules and this adversely affects the performance of both ACs, as discussed earlier. However, the GARC performs a search for rules that have higher accuracy through its capability to isolates rules that have low predictive power. Thus the GARC was able to reduce the search space and discover the categorization rules rapidly. On average, it is clear from the results that the proposed GARC was able to perform better than the AC approach. This is because the categorization rules that are generated by the GARC are revised more than those generated by AC1 and AC2 . In addition, in the case of the GARC, the number of rules that contribute to the categorization decision is limited, which reduces the errors during final categorization. Also, the categorization rules are strongly related to text categories through the objective function of the GARC, which determines the accuracy of the rules and also reduces rule sizes. Moreover, the GARC reduces the time needed for rule generation and categorization. In conclusion, the GARC was able to efficiently address the problem of the lack of useful categorization rules as it achieved the best performance combined with a reasonable size of classifier.

3.4 Comparison with Other related work

The experimental results proved that the GARC is better than ACs in terms of categorization performance and classifier size. To further evaluate the performance of the GARC, it was further assessed by comparing it with the other available results and approaches in the literature. A performance comparison is conducted between GARC and two other approaches from related works. The comparator approaches are based on Particle Swarm Optimization (PSO), where the PSO was used with KNN for FS and the efficiency of the selected features was examined by NB and decision tree (DT) (J48) (Chantar & Corne 2011). The same text datasets that have been used throughout this paper (i.e. Al-Jazeera text dataset (D1) and Akhbar Al-khaleej text dataset (D2) and a small subset of the Al-waten text dataset (D3)) were used. The performance comparison was based on categorization performance in terms of macro-average precision, macro-average recall and macro-average F-measure. In addition, the number of features and rules that controls the categorization process were highlighted.

Table 8
Comparison of results of the GARC and results of other methods

Datasets	Measure	GARC	PSO-KNN+J48	PSO-KNN+NB
D1	*# Feature /(Rule)*	*1000/ (108)*	*2967*	*2967*
	Precision	91.39	74.7	85.8
	Recall	93.73	72.3	84.3
	F-measure	92.55	72.9	84.6
D2	*# Feature /(Rule)*	*4000/ (113)*	*4562*	*4562*
	Precision	84.315	79.0	83.8
	Recall	79.729	78.7	82.8
	F-measure	81.958	78.5	83.1
D3	*# Feature /(Rule)*	*1000/ (99)*	*6578*	*6578*
	Precision	93.14	77.3	89.0
	Recall	85.34	76.7	88.6
	F-measure	89.07	76.9	88.7

Notes: with respect to the number of features (# Feature), the number in parentheses in the GARC case refers to the number of rules discovered with those feature subsets, categorization performance calculated in terms of macro-average precision, macro-average recall and macro-average F-measure.

The comparison is shown in Table 8. From these results, it can be seen that the results of proposed GARC methods are better than the most results of the other compared methods. In addition, the proposed method was able to achieve better categorization performance for all the measures with a smaller classifier size than that used by the compared methods. The GARC is a rule-based categorization method. This type of method categorizes text based on a set of categorization rules. As the comparison result indicates, the GARC was able to achieve competitive categorization performance with a limited number of categorization rules. With respect to the number of features, and the number of categorization rules that participate in categorization, the GARC produced better performance with a small set of categorization rules. The GARC results are superior to those of the DT with PSO (PSO-KNN+J48) for all datasets and it also superior PSO-KNN+NB in all situations except for D2. To summarize, the proposed methods performed best in terms of categorization when they were applied to D1 followed by D3. The GARC is a good categorization method that is able to produce a reasonable number of useful categorization rules with competitive categorization performance. In general, the results of the GARC are satisfactory and their superiority is proven for Arabic text categorization.

4. Conclusion

This paper has presented a new rule based method for Arabic text categorization based on GA named GARC. The GARC method was developed to enhance the predictive power of categorization rules and overcome the lack of useful categorization rules. The GARC involved the use of an GA to discover the categorization rules, where a set of categorization rules were extracted based on the improved GA search capability to find the best features that compose the categorization rules, then the extracted rules are revised and used for categorization. The experimental results demonstrated that improved categorization performance could be achieved when the GARC is utilized for Arabic text categorization. The GARC achieved promising results compared to other techniques such as AC, and DT and NB with PSO. The results also revealed that the GARC could produce higher predictive categorization rules than the AC method, which is reflected in the high performance of the GARC. Additionally, because the GARC is developed based on a GA, this method is effective for producing a limited set of useful categorization rules that address the problem of lack of categorization rules. The results have also indicated that significant improvements could be achieved in terms of not only producing strong categorization rules, but also categorization precision by utilizing the GARC for Arabic text categorization.

References

Abbas, M., Smaili, K. & Berkani, D. (2011). Evaluation of topic identification methods on Arabic corpora. *Journal of Digital Information Management, 9*(5), 185-192.

Abu Tair, M. M. & Baraka, R. S. (2013). Design and evaluation of a parallel classifier for large-scale Arabic text. *International Journal of Computer Applications, 75*(3), 13-20.

Al-diabat, M. (2012). Arabic text categorization using classification rule mining. *Applied Mathematical Sciences, 6*(81), 4033-4046.

Al-Radaideh, Q. A., Al-Shawakfa, E. M., Ghareb, A. S. & Abu-Salem, H. (2011). An approach for Arabic text categorization using association rule mining. *International Journal of Computer Processing Of Languages, 23*(1), 81-106.

Al-Saleem, S. (2011). Automated Arabic text categorization using SVM and NB. *International Arab Journal of e-Technology, 2*(2), 124-128.

Al-Zaghoul, F. & Al-Dhaheri, S. (2013). Arabic text classification based on features reduction using artificial neural networks. *Proceedings of 15th International Conference on Computer Modelling and Simulation (UKSim),* pp. 485-490.

Azam, N. & Yao, J. (2012). Comparison of term frequency and document frequency based feature selection metrics in text categorization. *Expert Systems with Applications, 39*, 4760–4768.

Bawaneh, M. J., Alkoffash, M. S. & Al Rabea, A. (2008). Arabic text classification using K-NN and Naive Bayes. *Journal of Computer Science, 4*(7), 600-605.

Chantar, H. K. & Corne, D. W. (2011). Feature subset selection for Arabic document categorization using BPSO-KNN. *Proceedings of the 3rd World Congress on Nature and Biologically Inspired Computing (NaBIC)*, 546-551.

Chen, J., Huang, H., Tian, S. & Qu, Y. (2009). Feature selection for text classification with Naïve Bayes. *Expert Systems with Applications, 36*(3), 5432-5435.

El-Halees, A. (2008). A comparative study on Arabic text classification. *Egyptian Computer Science Journal, 20*(2).

Fang, Y., Chen, K. & Luo, C. (2012). The algorithm research of genetic algorithm combining with text feature selection method. *Journal of Computational Science and Engineering, 1*(1), 9-13.

Ghareb, S. A. & Hamdan, A.R. & Bakar, A.A. (2016). Hybrid Feature Selection based on Enhanced

Genetic Algorithm for Text Categorization. *Expert System with Applications*, *49*, 31-47.

Gharib, T. F., Habib, M. B. & Fayed, Z. T. (2009). Arabic text classification using support vector machines. *International Journal of Computers and Their Applications, 16*(4), 192-199.

Günal, S. (2012). Hybrid feature selection for text classification. *Turkish Journal of Electrical Engineering & Computer Sciences*, *20*(Sup. 2), 1296-1311.

Harrag, F. & Al-Qawasmah, E. (2010). Improving Arabic text categorization using Neural Network with SVD. *Journal of Digital Information Management (JDIM), 8*(2), 125-135.

Hattab, A. M., & Hussein, A. K. (2012). Arabic content classification system using statistical Bayes classifier with words detection and correction. *World of Computer Science and Information Technology Journal, 2*(6), 193-196.

Holland, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan press.

Khorsheed, M. S. & Al-Thubaity, A. O. (2013). Comparative evaluation of text classification techniques using a large diverse Arabic dataset. *Language Resources and Evaluation, 47*(2), 513-538.

Mesleh, A. (2011). Feature sub-set selection metrics for Arabic text classification. *Pattern Recognition Letters, 32*(14), 1922-1929.

Mladenic, D. & Grobelnik, M. (1999). Feature selection for unbalanced class distribution and naive bayes. *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pp. 258-267.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management, 24*(5), 513-523.

Tan, F., Fu, X., Zhang, Y. & Bourgeois, A. G. (2008). A genetic algorithm-based method for feature subset selection. *Soft Computing, 12*(2), 111-120.

Thabtah, F. (2007). A review of associative classification mining. *The Knowledge Engineering Review, 22*(01), 37-65.

Thabtah, F., Gharaibeh, O. & Abdeljaber, H. (2011). Comparison of rule based classification techniques for the Arabic textual data. *Proceedings of the 4th International Symposium on Innovation in Information and Communication Technology (ISIICT), IEEE,* pp. 105-111.

Thabtah, F., Gharaibeh, O. & Al-Zubaidy, R. (2012). Arabic text mining using rule based classification. *Journal of Information and Knowledge Management, 11*(1), 1250006-1-1250006-10.

Tsai, C-F., Chen, Z-Y. & Ke, S-W. (2014). Evolutionary instance selection for text classification. *Journal of Systems and Software, 90*, 104-113.

Uğuz, H. (2011). A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. *Knowledge-Based Systems, 24*(7), 1024-1032.

Uysal, A. K. & Gunal, S. (2014). Text classification using genetic algorithm oriented latent semantic features. *Expert Systems with Applications, 41*(13), 5938-5947.