An Evolutionary Algorithm for Feature Selective Double Clustering of Text Documents

S. N. Nourashrafeddin
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada B3H 4R2
nourashr@cs.dal.ca

Evangelos Milios
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada B3H 4R2
eem@cs.dal.ca

Dirk V. Arnold
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada B3H 4R2
dirk@cs.dal.ca

***Abstract*—We propose *FSDC*, an evolutionary algorithm for Feature Selective Double Clustering of text documents. We first cluster the terms existing in the document corpus. The term clusters are then fed into multiobjective genetic algorithms to prune non-informative terms and form sets of keyterms representing topics. Based on the topic keyterms found, representative documents for each topic are extracted. These documents are then used as seeds to cluster all documents in the dataset. *FSDC* is compared to some well-known co-clusterers on real text datasets. The experimental results show that our algorithm can outperform the competitors.**

***Keywords*—Genetic algorithm, co-clustering, multiobjective optimization, text clustering.**

I. Introduction

Clustering plays an important role for browsing in text document collections. Grouping similar documents provides invaluable information about the text topics. The problem of clustering is widely studied in the data mining literature and numerous clusterers are proposed for different types of data [10], [14]. However, naive clustering techniques such as k-means do not typically work well for text [2]. This is mainly because text data has challenging characteristics.

A text dataset can be represented as a document-term matrix. Each entry of this matrix indicates the importance of a term in the respective document. The dimensionality of this representation is high but the underlying matrix is typically sparse. This is because each document contains only a small fraction of all terms existing in a dataset. Besides, many terms are too general to discriminate among topics. These non-informative terms act as noisy attributes in document clustering. Terms are also correlated in the document collections. These difficulties suggest the usefulness of developing a technique that reduces the dimensionality of the representation and extracts discriminative terms.

Since the number of terms in a text dataset is often multiple times larger than the number of documents, it has been proposed to first focus on term clusters. Double clustering refers to algorithms that perform term clustering before document clustering [13], [17]. Given the term clusters, the document-term matrix can be represented in a more compact way based on the presence of the term clusters in the documents [17].

Simultaneous clustering of terms and documents is referred to *co-clustering* [4], [7], [9]. A *co-clusterer* maps documents to document clusters and terms to term clusters iteratively. Term clustering is performed in the space of document clusters. Document clustering is similarly performed in the space of term clusters rather than terms. However, term clusters are typically noisy due to non-informative terms. They might also include only general terms that do not represent any specific topic. Using the term clusters without any prior analysis causes the quality of document clusters to deteriorate.

We do not cluster documents in the space of term clusters in our work. Instead, we find keyterms associated with document clusters. For this purpose, we use a multiobjective genetic algorithm (*MOGA*) to find discriminative terms that exist in each term cluster. Our experiments show that using topic keyterms results in better document clusters than those obtained by using term clusters without any prior analysis.

The number of document clusters is usually fixed to the known number of classes in experimental evaluations of double and co-clustering algorithms [4], [7], [9], [17]. However, finding an optimal number of term clusters is not a trivial task [9]. For this reason, the quality of document clusters is analyzed for different numbers of term clusters using the ground truth [4], [7], [9], [17]. This method of finding the number of term clusters raises the following questions:

1) Since the document-term matrices are often large, is it practical to run algorithms multiple times on each dataset to find the best value for this parameter?
2) What range of numbers should be tested in order to find an optimal number of term clusters?
3) How should one find a good value for this parameter when there is no ground truth about documents available?

We present a new way of double clustering that does not attempt to find the optimal number of term clusters. The proposed algorithm outperformed the competitive co-clusterers in our experiments.

We use the search capability of genetic algorithms to extract characteristic keyterms of topics. We show how a term cluster can be pruned by removing non-discriminative terms. We then propose a new way of document clustering using the spaces spanned by these keyterms. In our double clustering there is no need to evaluate the quality of clusters multiple times in order to find the optimal number of term clusters. Instead, we designed our algorithm in a way that the

number of term clusters is equal to the number of document clusters. Experimental results on some real datasets show that our approach results in better document clusters.

The remainder of this paper is organized as follows. Section 2 reviews some co-clusterers and evolutionary algorithms used for text data. Section 3 explains the proposed algorithm in detail. Experimental results on some real text datasets are shown in Section 4. Section 5 presents conclusions and future work.

II. Related Work

A double clusterer for text data is proposed in [17]. The algorithm first finds the term clusters that preserve most of the mutual information about documents. Hence, the co-occurrence matrix of documents and terms is replaced by the co-occurrence matrix of documents and term clusters. The algorithm then finds document clusters that preserve most of the information about term clusters. The main advantage of the double clusterer is that the matrix of documents versus term clusters is denser and reveals the structure of document clusters better [17].

A co-clusterer resembling the algorithm of [17] is proposed in [9]. Information-Theoretic Co-clustering views a co-occurrence matrix as an empirical joint distribution of two random variables, the set of rows and the set of columns. The goal is to preserve the mutual information between these two variables as much as possible. Given a co-clustering, the joint distribution is represented as a joint distribution of the co-clusters. The loss in mutual information due to the co-clustering is then computed. The optimal co-clustering that minimizes this loss is desired.

Like the algorithms mentioned above, we use term clusters to cluster documents in this study. However, rather than representing documents by term clusters, we use a multiobjective algorithm to find keyterms needed to represent the topic of each term cluster. In other words, we use a multiobjective algorithm as a feature selection step so as to distill the term clusters by removing non-discriminative terms. The keyterms are found based on the similarity of the documents characterized by the keyterms.

Two co-clustering algorithms for gene expression data are presented in [7]. Given a co-clustering, two squared *residue* measures are defined to compute the homogeneity of each co-cluster. The first measure is the sum of squared distances between each entry of a co-cluster and the mean of the co-cluster. The second measure is the sum of squared distances between each entry of a co-cluster and the corresponding row mean and column mean that the entry belongs in. Using these two measures, two co-clustering algorithms similar to k-means are proposed.

A general framework for matrix approximation is presented in [4]. The co-clustering problem is viewed as a special case of matrix approximation of the original data matrix. The quality of a co-clustering is obtained by the approximation error. The best co-clustering generates the best approximation of the original matrix where the error is measured using *Bregman* divergences. Six different co-clustering bases are introduced in [4]. Algorithms including two of the bases are used as competitors in our work.

A semi-supervised algorithm for document clustering is proposed in [13]. The algorithm is based on the assumption that an expert can provide a few good labeled documents for each topic. Fuzzy c-means [5] is initialized by using the keyterms of the labeled documents so as to cluster terms. After term clustering each document is assigned to the term cluster in which it has the largest contribution:

$$d_i \in c_j \quad \textit{if} \quad j = \arg\max_p \sum_{l=1}^{w} f(t_l, d_i) u_{pl} \qquad (1)$$

where w is the number of terms in the dataset, u_{pl} is the membership value of the term t_l in term cluster c_p, and $f(t_l, d_i)$ is the document-term matrix value of term l in document i.

This algorithm is similar to our algorithm in that fuzzy c-means is used to generate term clusters. The documents are then clustered by using the contribution of their terms in term clusters. However, we use term clusters to find some representative documents. The representative documents are used as seeds to cluster all documents. Our experiments show that *FSDC* outperforms this algorithm.

Topic keyterm clusters are also used for document clustering in [6]. Each document is first pre-processed to identify meaningful keyterms. These keyterms are then used to form a weighted graph. Each node of this graph corresponds to a keyterm. A co-occurrence-based correlation of keyterms is then defined to compute the weights of edges. The assumption of this study is that highly co-occurring terms in documents characterize topics. A heuristic graph partitioning method is then used to generate topic keyterm clusters. The cosine similarity between keyterms of clusters and documents are finally used to assign documents to the topics.

Genetic and Differential Evolution algorithms are used to find centroids of document clusters in [15] and [1]. Each individual encodes k documents as the centroids of k clusters. The remaining documents are assigned to the clusters with the closest centroids. The drawback of these algorithms is that a document does not contain enough data to represent a document cluster [2]. This is due to the nature of text where each document includes only a small fraction of all terms that exist in a corpus.

A co-evolutionary algorithm for co-clustering document-term matrices is proposed in [3]. A binary document-term matrix indicates whether terms appear in documents. Each individual contains two binary strings. Documents marked with '1' in a document string form a document cluster. A term string similarly indicates the terms associated with each document cluster. The fitness of an individual is defined as the decrease of overall entropy due to the clustering. The main drawback of the algorithm is that co-clusters are formed based on the presence of terms in documents regardless of their relevance degrees in documents. Moreover, due to the sampling approach used in the algorithm, even general terms can form co-clusters.

III. Methodology

We represent a text dataset as a document-term matrix in the bag of words model [2]. Considering the rows of this matrix, each document is a vector of terms existing in the

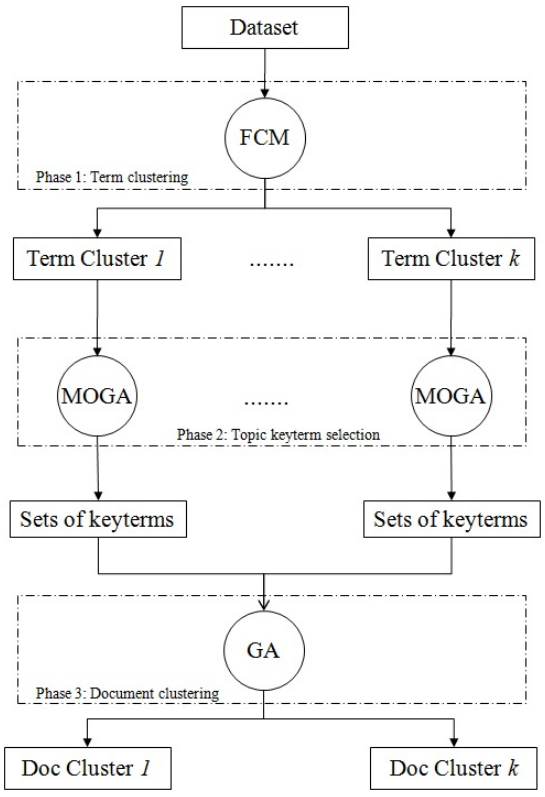

Fig. 1. The structure of *FSDC*. Fuzzy c-means (*FCM*) is used to cluster terms. Topic keyterm selection is performed by the *MOGA*. A genetic algorithm (*GA*) is finally used for document clustering.

dataset. Considering the columns of this matrix, each term is a vector of documents. Each entry of the matrix is term frequency–inverse document frequency (*tf-idf*), that indicates the importance of a term in the respective document.

The proposed algorithm is based on the idea that before finding document clusters, it is better to focus on term clustering and the keyterms that represent topics. *FSDC* consists of three phases:

1) Term clustering
2) Topic keyterm selection
3) Document clustering

The structure of *FSDC* and its main steps are depicted in Fig. 1 and Fig. 2 respectively. In the first phase, fuzzy c-means is used to cluster terms (columns of the document-term matrix) into k groups. We assume that each term cluster includes some keyterms representing a topic. They also include some general terms which are not useful in the clustering process. The goal of the second phase is to extract the topic keyterms that exist in a term cluster by removing the general terms.

In Phase 2, we select topic keyterms of each term cluster. For this purpose, a multiobjective genetic algorithm is applied on each term cluster independently. From topic keyterms, we extract some documents called representative in this work.

A single-objective genetic algorithm (*GA*) in Phase 3 clusters documents using the centroids of the representative documents obtained in Phase 2.

Input: a document-term matrix, k.
Output: k document clusters
1: Use fuzzy c-means to generate k term clusters
2: for each term cluster:
  2.1: Apply the $MOGA$ to prune non-discriminative terms
  2.2: for each non-dominated solution
    2.2.1: Extract the representative documents
    2.2.2: Compute a centroid for the representative documents
3: Apply the GA to identify the best document centroids
  3.1: Cluster document using the best individual of the GA

Fig. 2. The main steps of *FSDC*.

A. Term Clustering

Having similar topics is a common case in text clustering. Similar topics share common terms and each term usually belongs to multiple topics with different degrees of relevance. This is a key issue in term clustering.

The integration of fuzzy paradigm with the simplicity and efficiency of k-means, make fuzzy c-means a good candidate for term clustering. Given a document-term matrix, we apply fuzzy c-means on the term vectors to generate k term clusters. We use the maximum method to defuzzify the membership matrix [5].

B. Topic Keyterm Selection

The input of this phase are k term clusters each including a set of terms. Some terms in each term cluster are too general to represent a topic. Given a term cluster, we use a multiobjective genetic algorithm to find the keyterms and eliminate general terms as shown in Fig. 1. The method of this phase is inspired by the genetic algorithms proposed in [16], [18] in searching for candidate subspaces.

Each individual of our MOGA includes a set of terms in the form of:

$$individual_i = (t_1, t_2, ..., t_{T_i}) \tag{2}$$

where t_j is the column number of a term in the document-term matrix, and T_i is the number of terms used in the i^{th} individual.

Since it is not clear how many keyterms should be used to represent a topic [6], a variable-length integer representation is used for individuals.

The goodness of an individual depends on the goodness of the subspace spanned by its terms. Finding a good subspace can be defined as a multiobjective task including the following criteria:

1) The number of similar documents in the subspace.
2) The degree of similarity between the documents and the size of subspace in which similarities are measured.

The best subspace has the minimum number of terms while characterizing the maximum number of similar documents.

Fig. 3. A sample of four term-centroids generated from four classes of the 20-Newsgroups dataset. The block structure of the representative documents (white areas) is clear in this image.

For the fitness assignment step, we need to measure the similarity of the documents characterized by the terms of each individual. These representative documents are found by using the following steps:

1) We first compute a term-centroid for each individual. A term-centroid is the average of the column vectors corresponding to the terms included in an individual.
2) We then apply k-means with $k = 2$ on the term-centroid (in a one dimensional space) to partition its elements into two clusters. One cluster includes elements with near-zero values and the other cluster includes elements with values larger than zero. Elements with large values correspond to the representative documents with larger *tf-idf* values. The other cluster consists of the non-representative documents with near-zero *tf-idf* values.

A sample of four term-centroids for four classes of the 20-Newsgroups dataset is shown in Fig. 3. Each column of this gray scale image corresponds to a term-centroid and each row is a document. Black areas contain near-zero *tf-idf* values while white areas contain larger *tf-idf*. The representative documents of each term-centroid have larger values (white color) compared to the non-representative documents (black color) as shown in Fig. 3.

After finding the representative documents of each individual, two objective function values are computed for fitness assignment:

1) *Number of representative documents*: the number of representative documents of an individual. This objective should be maximized. We use the representative documents as seeds to cluster all documents. Having a larger number of these documents results in better document clusters.
2) *Distortion*: The representative documents of an individual should be similar to each other. Distortion or intra-cluster variation is a common criterion to evaluate quality of clusters [13], [18]. The distortion of the i^{th} individual is computed as:

$$\text{Distortion}(ind_i) = \frac{1}{T_i} \sum_{d_j \in rep_i} \parallel d_j - dc_i \parallel^2_{SP_i} \quad (3)$$

where rep_i is the set of representative documents associated with the i^{th} individual, dc_i is the document-centroid of the documents of rep_i, and SP_i is the feature space spanned by the keyterms of the i^{th} individual. The distances are normalized by the size of subspace T_i. This objective function should be minimized.

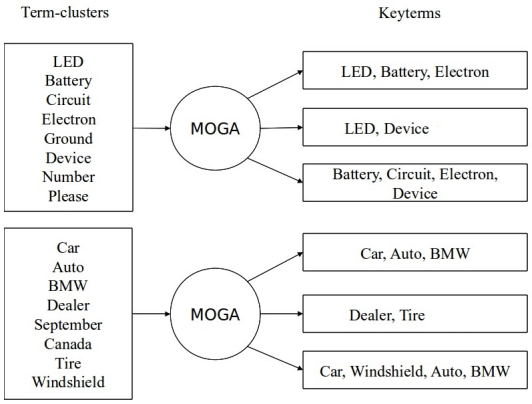

Fig. 4. A sample output of Phase 2 (Topic keyterm selection). Two term clusters are fed into the MOGA so as to prune non-discriminative terms. Each output is a non-dominated solution representing a set of topic keyterms.

Individuals are initialized by using the high-variance terms. For each term, we measure the variance of its *tf-idf* values over all documents using the following formula:

$$\text{Var}(t_j) = \frac{1}{N-1} \sum_{i=1}^{N} ([dt]_{ij} - m_j)^2 \quad (4)$$

where t_j is the j^{th} term, $[dt]_{ij}$ is the ij^{th} entry of the document-term matrix, m_j is the average of the entries of the j^{th} column of the matrix, and N is the number of documents in the dataset.

Parent selection is performed by tournament selection. Uniform crossover and the following mutation operators are considered in our MOGA:

1) Add-term: this operator increases the length of an individual by 1 and adds a new randomly selected term.
2) Remove-term: this operator decreases the length of an individual by 1 and removes a randomly selected term.
3) Replace-term: this operator randomly replaces a term by a new term.

The output of multiobjective optimization is a set of non-dominated solutions. A solution is non-dominated if and only if there is no other solution that has better values for all objective functions. Each solution of our MOGA is a set of topic keyterms. The best solution of each MOGA will be identified in Phase 3, Document clustering.

A sample output of this phase is shown in Fig. 4. In this figure, terms are already clustered into two term clusters. The topic of the first term cluster is *Electrical* and the topic of the second one is *Car/Auto*. Some terms in this example are too general to discriminate a topic, like Canada, September, Please, and Number. We applied the MOGA on each term cluster individually to remove these general terms.

FSDC is independent of the multiobjective optimization algorithm and any kind of multiobjective method can be used.

In this study, we used a Matlab implementation[1] of *NSGA-II* [8].

It is also noteworthy to mention that this phase of *FSDC* can be run independently for each term cluster in parallel as shown in Fig. 1.

C. Document Clustering

Multiple non-dominated solutions are obtained for each term cluster in Phase 2. In this phase, we want to identify the best solution of each term cluster.

Each solution indicates a set of keyterms and their representative documents. For each solution, we compute a document centroid (dc). A document centroid is the row average of document vectors corresponding to the representative documents.

Since *MOGA* produces multiple non-dominated solutions, there are multiple document centroids for each term cluster. We need to choose only k document centroids, one from each term cluster, to cluster all documents.

Suppose that the number of document centroids of the i^{th} term cluster is tk_i. To find the best document centroids, we need to examine $tk_1 \times tk_2 \times ... \times tk_k$ cases. This is very time-consuming where k or the number of non-dominated solutions is large.

To search for the best document centroids, a single-objective *GA* is used in this phase. In this *GA*, the length of individuals is fixed to k. Each entry of an individual is dedicated for document centroids of one term cluster. The value of an entry indicate a document centroid. For fitness assignment, all documents are assigned to the nearest document centroids indicated in an individual to generate k clusters $\{C_i\}_{i=1}^{k}$. We then compute the Separation (inter-cluster distances) of the clusters as the fitness value of the individual:

$$\text{Separation} = \sum_{\mu_i, \mu_j} \parallel \mu_i - \mu_j \parallel^2 \tag{5}$$

where μ_i is the centroid of documents in document cluster C_i. After a predefined number of iterations, the best individual in the population generates k document clusters.

Parent selection is performed by tournament selection and no crossover is considered. The only genetic operator is a replacement mutator which randomly replaces one document centroid. An example of this phase is shown in Fig. 5. As shown in this figure, from all non-dominated solutions, the third solution of the first term cluster and the third solution of the second term cluster are chosen for document clustering.

IV. Experimental Results

In this section we show the benefits of *FSDC* as compared to seven co-clustering algorithms including: Information-Theoretic Co-clustering (*ITCC*) [9], Euclidean Co-clustering (*ECC*) and Minimum Sum-squared Residue Co-clustering (*MSRCC*) [7], Square Euclidean co-clustering and I-divergence co-clustering with bases $C2$ and $C5$ (*SECC2*, *SECC5*, *IdivCC2*, *IdivCC5*) [4]. We did not include the double clusterer of [17] in our experiments since it was demonstrated that *ITCC* outperforms the clusterer on some subsets of the *20-Newsgroups* dataset [9]. The unsupervised version of [13] (*UVFCM*) is also implemented in this study in order to compare with *FSDC*.

[1] http://www.mathworks.com/matlabcentral/fileexchange/10429-nsga-ii-a-multi-objective

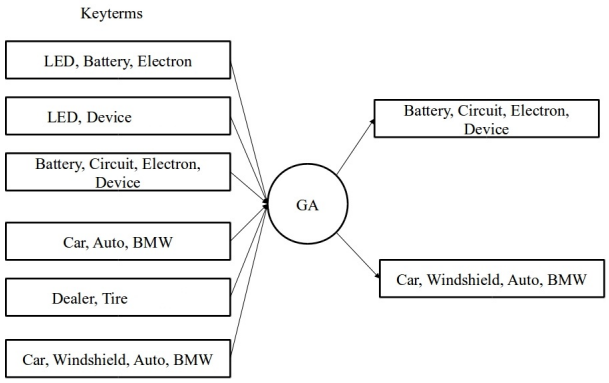

Fig. 5. A sample output of Phase 3 (Document clustering). The *GA* identified the best solutions that should be used for document clustering.

A. Evaluation Measures

We used the true labels of documents to evaluate clusters. For this purpose, a confusion matrix is formed after each clustering process. Each element of this matrix shows the number of common documents between the corresponding cluster and class. The dimensionality of this matrix is k by k in this study. This confusion matrix is then used to compute two evaluation measures:

1) *Fmeasure* is a commonly used measure in information retrieval. The harmonic mean of precision and recall is used in our experiments [19]:

$$Fmeasure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{6}$$

2) *Normalized Mutual Information* (*NMI*) measures the amount of information we get about classes if we have clusters [14]. It has a maximum value of one when the clustering process recreates classes perfectly and it has a minimum of zero:

$$NMI = \frac{I(W,C)}{[H(W) + H(C)]/2} \tag{7}$$

$$I(W,C) = \sum_k \sum_j P(w_k \cap c_j) \log \frac{P(w_k \cap c_j)}{P(w_k)P(c_j)} \tag{8}$$

$$H(W) = -\sum_k P(w_k) \log(P(w_k)) \tag{9}$$

where $W = \{w_1, w_2, .., w_k\}$ and $C = \{c_1, c_2, ..., c_k\}$ denote clusters and classes respectively and N is the number of documents.

TABLE I. SUMMARY OF THE TEXT DATASETS USED IN OUR EXPERIMENTS

Dataset Name	No. of Documents	No. of Terms	Reduced No. of Terms	No. of Classes	Sparsity Percentage
Reviews	4069	36746	7724	5	97.9%
Classic4	7094	41681	5099	4	99.5%
Cacmcisi	4663	14409	2528	2	99.4%
News-sim3	2924	20753	4697	3	99.7%
News-multi7	6632	33469	7006	7	99.8%

B. Datasets and Implementation

In our experiments, we used five text datasets whose characteristics are summarized in Table I. The last column of the table shows the percentage of zero values that exists in the document-term matrices of the datasets. *Reviews* includes articles about movies, food, restaurants, music, and radio gathered from *San Jose Mercury News*[2]. *Classic4* is one of the well-known benchmark datasets used in text clustering. It is created from the *SMART* data repository[3] containing abstracts of papers about medical, information retrieval, aerodynamics, and computing algorithms. *Cacmcisi* is also made from the *SMART* repository but includes only abstracts about computing algorithms and information retrieval. The last two datasets are derived from the *20-Newsgroups* dataset [11]. This dataset consists of approximately 20000 news articles grouped into 20 different topics[4]. *News-sim3* includes articles about three similar topics including *comp.graphics*, *comp.os.ms-windows.misc*, and *comp.windows.x*. There are articles about seven topics in *News-multi7* including *alt.atheism*, *comp.sys.mac.hardware*, *misc.forsale*, *rec.sport.hockey*, *sci.crypt*, *alt.politics.guns*, and *soc.religion.Christian*.

Stop-word removal, stemming, and removing low-variance terms are applied on the datasets in a pre-processing step. Each dataset is then represented as a document-term matrix in the bag of words model. The effect of document length is reduced by using the $L2$ norm to normalize the length of document vectors to one. The importance of terms in documents are measured using *tf-idf* values.

We use Euclidean distance to cluster document vectors since their length is normalized to one and cosine distance to cluster term vectors.

C. Removing Low-variance Terms

Not all terms in the bag of words model are required in clustering. We assume that only high-variance terms represent topics and the other terms are non-discriminative [12]. For this reason, we measured the variance of terms over all documents. We used only terms whose variances are larger than the average of all variances. In this way, *Reviews* has 7724 terms, *Classic4* has 5099 terms, *Cacmcisi* has 2528 terms, *News-sim3* has 4697, and *News-multi7* has 7006 terms.

To show the effectiveness of this pre-processing step, we performed the following experiment. We used *Naive Bayes* classifier implemented in *WEKA*[5] on the datasets of Table I before and after this step. The outputs of the classifier using 10-fold cross validation method are shown in Table II.

TABLE II. NMI VALUES OBTAINED BEFORE AND AFTER REMOVING LOW-VARIANCE TERMS USING NAIVE BAYES CLASSIFIER

Dataset Name	Before Removing Low-variance Terms	After Removing Low-variance Terms
Reviews	0.651	0.651
Classic4	0.918	0.958
Cacmcisi	0.903	0.944
News-sim3	0.292	0.539
News-multi7	0.668	0.833

TABLE III. CHARACTERISTICS OF THE COMPETITORS USED IN OUR EXPERIMENTS

Clustering Algorithm	Term Clustering	Number of Document Cluster	Number of Term Cluster
FSDC	✓	k	k
UVFCM	✓	k	k
ITCC	✓	k	*variable*
ECC	✓	k	*variable*
MSRCC	✓	k	*variable*
SECC2	✓	k	*variable*
SECC5	✓	k	*variable*
IdivCC2	✓	k	*variable*
IdivCC5	✓	k	*variable*

The quality of classes is improved except for the dataset *Reviews*. This experiment shows that not only this step did not cause the performance of the classifier deteriorate but also improved it in most cases.

D. Results and Discussion

We evaluated the performance of nine clustering algorithms on the five datasets using two evaluation measures. The number of term clusters for *FSDC* and *UVFCM* is the same as the number of document clusters as shown in Table III. For the co-clustering algorithms, we used different numbers of term clusters. For *ITCC*, *ECC*, and *MSRCC* the number of term clusters are $\{1, 2, 4, 8, ..., 128\}$ as suggested in [9]. These numbers are $\{5, 10, 15, ..., 50\}$ for *SECC2*, *SECC*, *IdivCC2*, and *IdivCC5* as in the experiments performed in [4].

For each dataset, we ran the experiments in the following way. Each co-clusterer is run 20 times for each number of term clusters and the average F-measure and NMI of these 20 runs are computed. We only reported the best F-measure and NMI corresponding to the best number of term clusters. The other algorithms are also run 20 times and the average F-measure and NMI are shown in Fig. 6 to Fig. 10.

Our experimental results reveal that *FSDC* outperformed the competitors on all the datasets. This is more evident on Classic4, and News-sim3. It is noteworthy to mention that News-sim3 is the most difficult dataset to cluster in our experiments since it includes three similar topics.

An important observation in our experiments is that the outputs of the co-clusterers are sometimes sensitive to the number of term clusters. It is quite noticeable for *Reviews* and *Classic4*. Even though it is suggested in [9], [17] to increase the number of term clusters to get better results, this did not happen on some datasets in our experiments.

[2]http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

[3]ftp://ftp.cs.cornell.edu/pub/smart/

[4]http://qwone.com/~jason/20Newsgroups/

[5]http://www.cs.waikato.ac.nz/ml/weka/

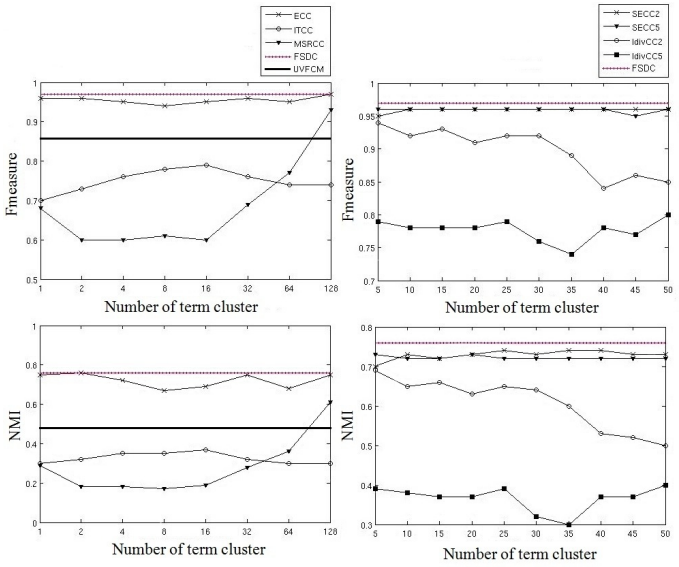

Fig. 6. The quality of clusters for *FSDC* and the co-clusterers on Cacmcisi. *FSDC* outperformed the competitors. The algorithms *ECC*, *SECC2* and *SECC5* generated comparable results. The quality of clusters of *IdivCC2* decreases as the number of term clusters increases. The algorithms *SECC2* and *SECC5* are the least sensitive co-clusterers to the number of term clusters.

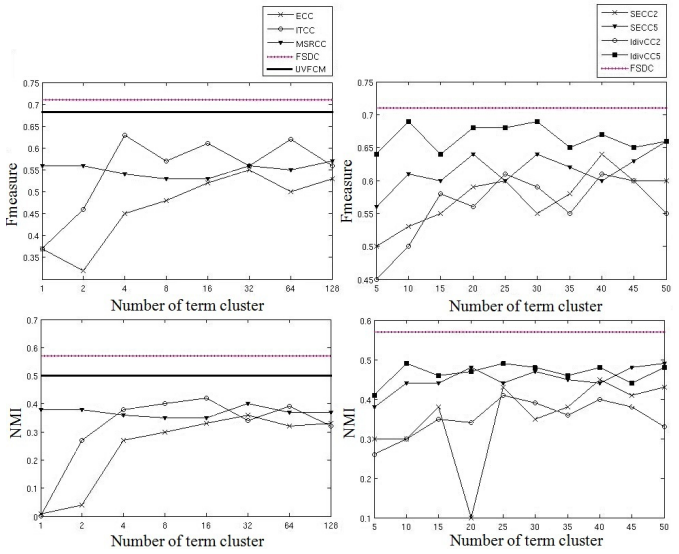

Fig. 7. The quality of clusters for *FSDC* and the co-clusterers on Reviews. *FSDC* is the best clustering algorithm. The outputs of the co-clusterers (*SECC2, SECC5, IdivCC2, IdivCC5, ITCC*) are sensitive to the number of term clusters. The best co-clusterer is *IdivCC5.*

Overall, the comparison between the proposed algorithm and the other double or co-clustering algorithms demonstrates that our algorithm always outperformed the competitors in our experiments. This observation confirms that the number of term clusters being selected as equal to the number of document cluster is a rational assumption and no further attempts for finding this number is needed for *FSDC.*

V. Conclusion and Future Work

We proposed a new approach for topic keyterm selection of text documents using evolutionary algorithms.

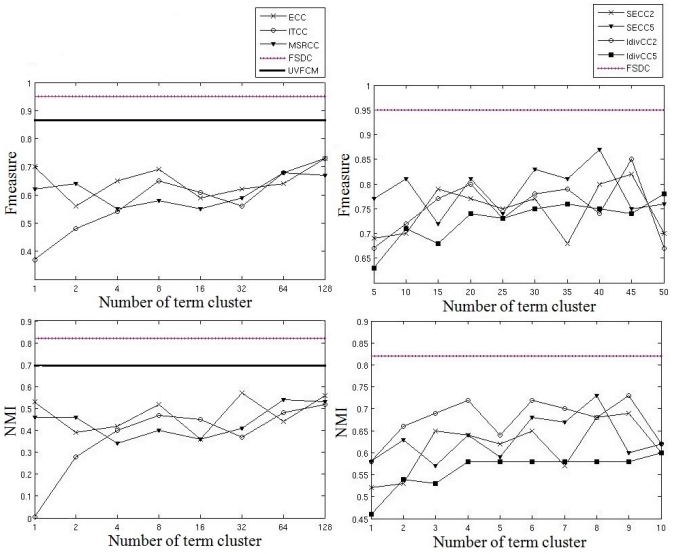

Fig. 8. The quality of clusters for *FSDC* and the co-clusterers on Classic4. *FSDC* is the best clustering algorithm and the quality of its clusters are much better than those of the competitors. The outputs of the co-clusterers (*SECC2, SECC5, IdivCC2*) are sensitive to the number of term clusters.

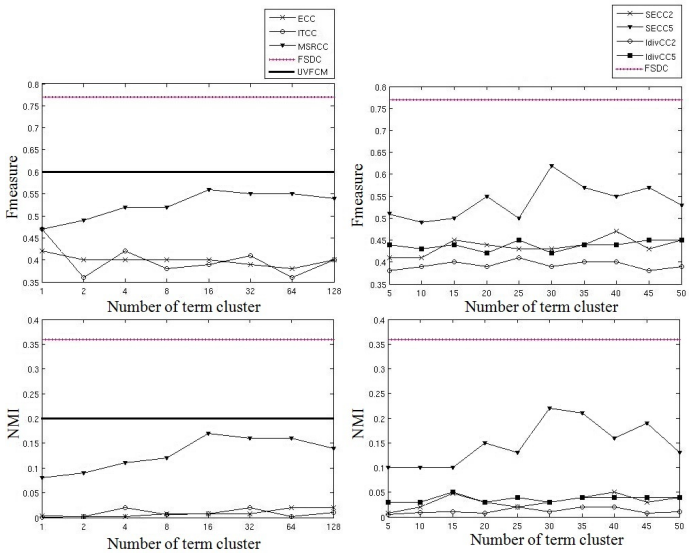

Fig. 9. The quality of clusters for *FSDC* and the co-clusterers on Newssim3. *FSDC* is the best clustering algorithm and the quality of its clusters are much better than those of the competitors. The algorithm *SECC5* is the best co-clusterer.

A multiobjective genetic algorithm is designed so as to prune non-discriminative terms from term clusters. Each non-dominated solution of our *MOGA* includes a set of topic keyterms.

We then proposed a heuristic method to extract the representative documents associated with each solution. These documents are used as seeds to cluster all documents. We applied k-means on the term centroid of each solution for this purpose.

Rather than selecting one non-dominated solution from each *MOGA*, we examined combinations of all non-dominated solutions using a genetic algorithm. The genetic algorithm

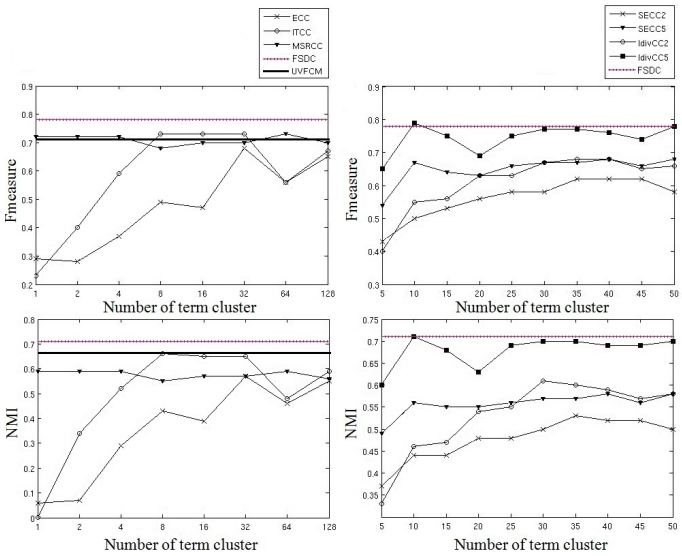

Fig. 10. The quality of clusters for *FSDC* and the co-clusterers on News-multi7. *FSDC* is the best clustering algorithm. Only *IdivCC5* could generate clusters with similar quality.

identifies the best solutions based on the quality of clusterings that they can generate. The centroids of the representative documents associated with the solutions are used for this purpose.

Our work also demonstrated that distilling term clusters can result in better document clusters compared to the competitors in which term clusters are used without any prior analysis.

ACKNOWLEDGMENT

This research was supported by the NSERC (Natural Sciences and Engineering Research Council of Canada) Business Intelligence Network.

REFERENCES

[1] A. Abraham, S. Das, and A. Konar, "Document clustering using differential evolution," in *IEEE Congress on Evolutionary Computation (CEC)*, 2006, pp. 1784–1791.

[2] C. C. Aggarwal and C. Zhai, *Mining Text Data*. Springer US, 2012, ch. A Survey of Text Clustering Algorithms, pp. 77–128.

[3] A. Aizawa, "A co-evolutionary framework for clustering in information retrieval systems," *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1787–1792, 2002.

[4] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to Bregman co-clustering and matrix approximation," *Journal of Machine Learning Research*, vol. 8, pp. 1919–1986, 2007.

[5] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[6] H. C. Chang and C. C. Hsu, "Using topic keyword clusters for automatic document clustering," *Third International Conference on Information Technology and Applications (ICITA)*, vol. 1, pp. 419–424, July 2005.

[7] H. Cho, I. Dhillon, Y. Guan, and S. Sra, "Minimum sum-squared residue co-clustering of gene expression data," in *Proceedings of the fourth SIAM International Conference on Data Mining*, vol. 114, 2004.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[9] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 89–98.

[10] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and P. Leon, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 39, no. 2, pp. 133–155, 2009.

[11] Y. Hu, E. E. Milios, and J. Blustein, "Enhancing semi-supervised document clustering with feature supervision," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 929–936.

[12] J. Kogan, C. Nicholas, and V. Volkovich, "Text mining with information-theoretic clustering," *Computing in Science and Engineering*, vol. 5, no. 6, pp. 52–59, Nov. 2003.

[13] H. Mahmoodi and E. Mansoori, "Document clustering based on semi-supervised term clustering," *International Journal of Artificial Intelligence & Applications (IJAIA)*, vol. 3, no. 3, pp. 69–82, 2012.

[14] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[15] Y. K. Meena, Shashank, and V. P. Singh, "Text documents clustering using genetic algorithm and discrete differential evolution," *International Journal of Computer Applications*, vol. 43, no. 1, 2012.

[16] S. Nourashrafeddin, D. Arnold, and E. Milios, "An evolutionary subspace clustering algorithm for high-dimensional data," in *Proceedings of the fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion*, ser. GECCO Companion '12. New York, NY, USA: ACM, 2012, pp. 1497–1498.

[17] N. Slonim and N. Tishby, "Document clustering using word clusters via the information bottleneck method," in *Proceedings of the 23rd annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '00. New York, NY, USA: ACM, 2000, pp. 208–215.

[18] A. Vahdat, M. Heywood, and N. Zincir-Heywood, "Bottom-up evolutionary subspace clustering," in *IEEE Congress on Evolutionary Computation (CEC)*, 2010, pp. 1–8.

[19] Y. Zhao and G. Karypis, "Comparison of agglomerative and partitional document clustering algorithms," Univ. of Minnesota, Tech. Rep. #02-14, 2002.