Computers and Mathematics with Applications 57 (2009) 1901–1907

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Genetic algorithm for text clustering based on latent semantic indexing☆

Wei Song*, Soon Cheol Park

Division of Electronics and Information Engineering, Chonbuk National University, Jeonju, 561756, Republic of Korea

ARTICLE INFO

Keywords:
Document representation model
Genetic algorithm
Latent semantic indexing
Text clustering

ABSTRACT

In this paper, we develop a genetic algorithm method based on a latent semantic model (GAL) for text clustering. The main difficulty in the application of genetic algorithms (GAs) for document clustering is thousands or even tens of thousands of dimensions in feature space which is typical for textual data. Because the most straightforward and popular approach represents texts with the vector space model (VSM), that is, each unique term in the vocabulary represents one dimension. Latent semantic indexing (LSI) is a successful technology in information retrieval which attempts to explore the latent semantics implied by a query or a document through representing them in a dimension-reduced space. Meanwhile, LSI takes into account the effects of synonymy and polysemy, which constructs a semantic structure in textual data. GA belongs to search techniques that can efficiently evolve the optimal solution in the reduced space. We propose a variable string length genetic algorithm which has been exploited for automatically evolving the proper number of clusters as well as providing near optimal data set clustering. GA can be used in conjunction with the reduced latent semantic structure and improve clustering efficiency and accuracy. The superiority of GAL approach over conventional GA applied in VSM model is demonstrated by providing good Reuter document clustering results.

1. Introduction

Clustering is an unsupervised pattern classification technique which is defined as group n objects into m clusters without any prior knowledge. The number of partitions/clusters may or may not be known a priori. The task of document clustering is both difficult and intensively studied [1,2]. Several algorithms for clustering data when the number of clusters is known a priori are available in the literature. K-means algorithm [3], one of the most widely used, attempts to solve the clustering problem into a fixed number of clusters K known in advance. It is an iterative hill-climbing algorithm and solution suffering from the limitation of the sub optimal which is known to depend on the choice of initial clustering distribution [4]. In [5], a branch and bound algorithm uses a tree search technique to search the entire solution space. It employs a criterion of eliminating sub trees which do not contain the optimal result. In this scheme, the number of nodes to be searched becomes huge as the size of the data set becomes large. Several types of biologically inspired algorithms have been proposed in the literature. Ant clustering algorithm [6] is to project the original data into bidimensional output grid and position that are similar to each other in their original space of attributes. By doing this, the algorithm is capable of grouping together items that are similar to each other. Genetic algorithm (GA) [7,8] belongs to search techniques that mimic the principle of natural selection. GA performs a search in complex, large and multimode landscapes, and provides near-optimal solutions for objective or fitness function of an optimization problem. However, the cost of computational time is high because its long string representation evolves in high dimensional space typical for textual data [9].

☆ This work was partially supported by the Korea Research Foundation (Grant Nos. KRF-2006-321-A00012) and partially supported by Brain Korea 21.
* Corresponding author.
E-mail addresses: songwei9988@yahoo.com.cn (W. Song), scpark@chonbuk.ac.kr (S.C. Park).

0898-1221/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2008.10.010

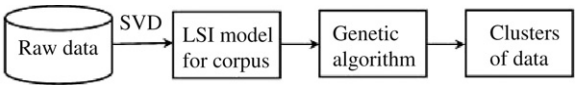

Fig. 1. The process of GA for text clustering based on LSI.

The most general and straightforward approach to represent text is the vector space model (VSM), it means each unique term in the vocabulary represents one dimension in feature space. Unfortunately, it needs a large number of features to represent high dimensions, and it is not suitable for GA since the scalability will be poor and the cost of computational time will be high. Meanwhile, if we represent all texts in this way many documents that are related to each other semantically might not share any words and thus appear very distant, and occasionally documents that are not related to each other might share common words and thus appear to be closer. This is due to the nature of text, where the same concept can be represented by many different words, and words can have ambiguous meanings. Latent semantic indexing (LSI) [10] is an automatic method that reduces this large space to one that hopefully captures the true relationships between documents [11]. LSI uses the singular value decomposition (SVD) technique to decompose the large term-by-document matrix into a set of k orthogonal factors. In this semantic structure, even two documents do not have any common words, we also can find the associative relationships, because similar contexts in the documents will have similar vectors in semantic space. The process of GA for text clustering based on LSI is shown in Fig. 1.

In this paper, we propose a variable string length GA using a gene index to encode chromosomes in the semantic space. The gene index indicates the location of each gene in the chromosome, which has a greater chance of obtaining the appropriate center combination and to find the proper number of clusters.

In the next section, we give a brief review of LSI, and describe how we use it for text clustering. Details of genetic algorithms for text clustering based on the LSI model are described in Section 3. Experiment results are given in Section 4. Conclusions and future works are given in Section 5.

2. Latent semantic indexing model for documents representation

The purpose of LSI is to extract a smaller number of dimensions that are more robust indicators of meaning than individual terms. Once a term-by-document matrix is constructed, LSI requires the singular value decomposition of this matrix to construct a semantic vector space. Due to the word-choice variability, the less important dimensions corresponding to "noise" are ignored. A reduced rank approximation to the original matrix is constructed by dropping these noisy dimensions.

2.1. Singular value decomposition

Our corpus can be firstly represented as a term-by-document matrix $X(m \times n)$, assuming there are m distinct terms in an n documents collection. The singular value decomposition of X is given by

$$X = U\Sigma V^{\mathrm{T}} \tag{2.1}$$

where U and V are the matrices of the left and right singular vectors. Σ is the diagonal matrix of singular values. LSI approximates X with a rank k matrix.

$$X_k = U_k \Sigma_k V_k^{\mathrm{T}} \tag{2.2}$$

where U_k is comprised of the first k columns of the matrix U and V_k^{T} is comprised of the first k rows of matrix V^{T}. $\Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$ is the first k factors. That is, the documents are represented in the k dimensional LSI space spanned by the basis vectors.

2.2. General LSI model for information retrieval

When LSI is used for the purpose of information retrieval [12,13] query q is a $m \times 1$ matrix, where m is the number of terms in the documents collection. The query vector $\hat{q}$ is constructed by

$$\hat{q} = q^{\mathrm{T}} U_k \Sigma_k^{-1}. \tag{2.3}$$

2.3. Our approach of LSI model for document representation

When LSI is used for the purposes of document representation, a document d is firstly initialized as a $m \times 1$ matrix, where m is the number of terms. Because matrix U in (2.1) represents the matrix of terms vectors in all documents and the proper number of rank U_k spans the basis vectors of U. In our approach we remove Σ_k^{-1} matrix and use the multiplying of matrices d^{T} and U_k to represent the document vector. The results of our experiment also show that the representation of multiplying of two matrix U_k and Σ_k^{-1} is not as good as that of multiplying the single U_k matrix. So each document vector $\hat{d}$ is represented by $1 \times k$ matrix.

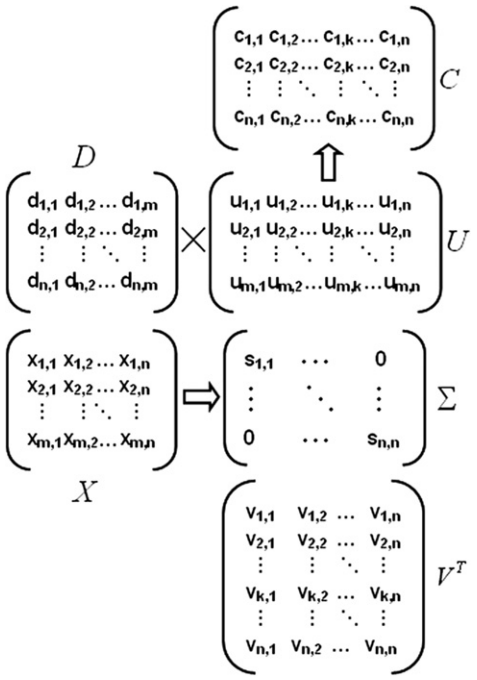

Fig. 2. LSI model for document representation.

$$\hat{d} = d^{\mathrm{T}} U_k. \tag{2.4}$$

The texts corpus can be organized by another representation of document-by-term matrix $D(n \times m)$, and the corpus matrix can be newly organized by

$$C = DU_k. \tag{2.5}$$

Fig. 2 shows our LSI model for document representation. Once the texts corpus is defined by this way, the relevance value between the documents can be computed using the cosine similarity coefficient. The corpus with these new text vectors is performed by the genetic algorithms for clustering in next section.

In Fig. 2 the rank k of the matrix X is equal to the number of nonzero diagonal elements of Σ. Just as the first k columns of U are a basis for the column space of X. In addition, the first k rows of V^{T} are a basis for row space of X.

3. Genetic algorithm for text clustering

Genetic algorithms belong to search techniques that mimic the principle of natural selection. Clustering is a popular unsupervised pattern classification technique which partitions the input space into K regions based on some similarity/dissimilarity metric. The number of partitions/clusters may or may not be known a priori. We apply the capability of GA to evolving the proper number of clusters as well as providing appropriate data clustering. The parameters in the search space are represented in the form of chromosomes. A collection of such chromosomes is called a population. An objective and fitness function is associated with each chromosome that represents the degree of fitness. Biologically inspired operators such as selection, crossover and mutation are applied to yield new child chromosomes. These operators continue several generations until the termination criterion is satisfied. The fittest chromosome seen up to the last generation provides the best solution to the clustering problem.

3.1. Encoding chromosomes

In GAL clustering, the chromosome is encoded by a string of real numbers. Each chromosome Chro_i in the population is initially encoded by a number of K centers, where K is assumed to lie in the range $[K_{\min}, K_{\max}]$.

$$\mathrm{Chro}_i = \{\mathrm{Center}_{i,1}, \mathrm{Center}_{i,2}, \ldots, \mathrm{Center}_{i,k}\}. \tag{3.1}$$

For initializing Center_i, a row of elements are chosen randomly from the corpus matrix C in (2.5).

$$\mathrm{Center}_i = \{c_{i,1}, c_{i,2}, c_{i,3}, \ldots, c_{i,n}\}. \tag{3.2}$$

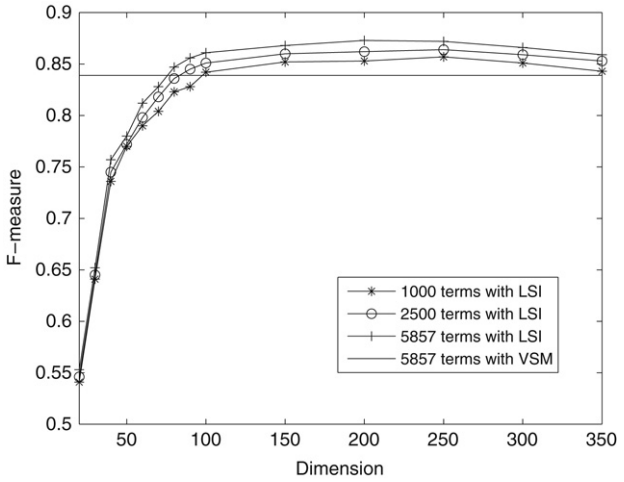

Fig. 3. Cluster performance against the number of dimension on data set 1.

From the view of Fig. 2, n is the number of total texts and the dimensions can be reduced from n to k ($k < n$).

$$\text{Center}_i = \{c_{i,1}, c_{i,2}, c_{i,3}, \ldots, c_{i,k}\}. \tag{3.3}$$

For encoding a variable string length GA, each chromosome in the population is encoded by K_i centers, where K_i is assumed to lie in the range $[K_{\min}, K_{\max}]$. In order to make a compact encoding, a string of random gene indices are chosen to denote the relative positions of genes in the chromosome. These indices are very suitable for the crossover. If the crossover point is c_p we only need to exchange the genes whose gene indices are greater than c_p. The proposed chromosomes with the variable string length have a greater chance to obtain the proper combinations of centers.

3.2. Population initialization

The diversity of the population is an important factor that affects the success of GA. Low diversity in the original population may lead GA to a premature convergence or take a long evolution time to find the global optimal solution. Hence, we should ensure a high diversity in the population initialization. In our algorithm, we initialize 1000 chromosomes in the mating pool and choose the best 500 chromosomes from the pool to construct the first generation.

3.3. Evolution principles

The fittest concept selection, classical single-point crossover and Gaussian mutation [14] are adopted in this paper. Fitness function is defined to be $1/DB$, where DB is Davies–Bouldin index [15,16]. The proportion of selection, crossover and mutation are s, c and m, respectively. The termination criterion is the iteration of the best fitness value without improvement exceeding consecutive $N_{\max}$ iterations.

The next section provides the experiment results of GAL for text clustering, along with its comparison with the performance of the original genetic algorithm using simple vector space model.

4. Experiments

We present experiments in this section to demonstrate the effectiveness of the GAL algorithm. The experiments are performed on the benchmark data set Reuters-21578. Documents without labels or with multiple labels are discarded. Data set 1 including 600 texts from three topics (acq, crude, trade) and data set 2 including 1000 texts from five topics (acq, crude, grain, money-fx, and trade) are tested. After being processed by word extraction, stop word removal, and stemming, there are 5857 and 7274 terms, respectively. In our algorithm the weight of a term is measured by term frequency. However, the whole number of terms is not suitable for GA, so we choose the terms with highest weight in the vocabulary. We vary the number of terms from 5857, 2500 to 1000 and 7274, 3500 to 1500 for data set 1 and data set 2 respectively to construct corpus matrix C in (2.5). GA is implemented with the following parameters: $s = 0.6, c = 0.2, m = 0.2, N_{\max} = 10$ and the population size is taken to be 500. The F-measure [9] is used for clustering evaluation. The cluster results are shown in Figs. 3 and 5. The horizontal lines represent the cluster results in VSM with 5857 and 7274 features. Comparisons of the computational time are shown in Figs. 4 and 6.

From Fig. 3 we can see that from about 100 dimensions the performance of GALs outperform that of GA using VSM. For 5857 terms with LSI, GAL obtains its best performance on 200 dimensions. For 1000 and 2500 terms with LSI, GAL methods

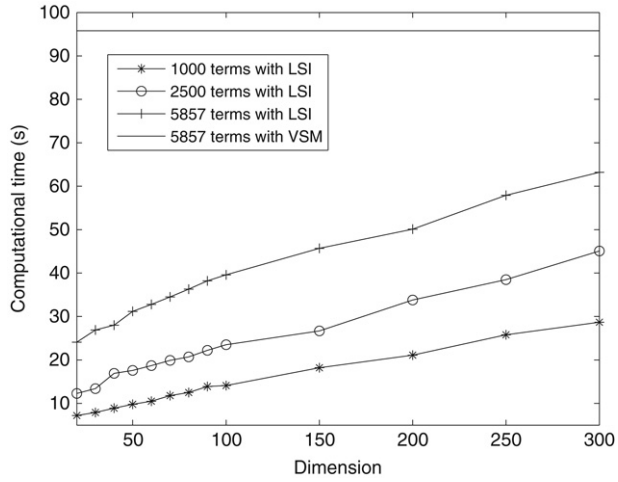

Fig. 4. Computational time against the number of dimension on data set 1.

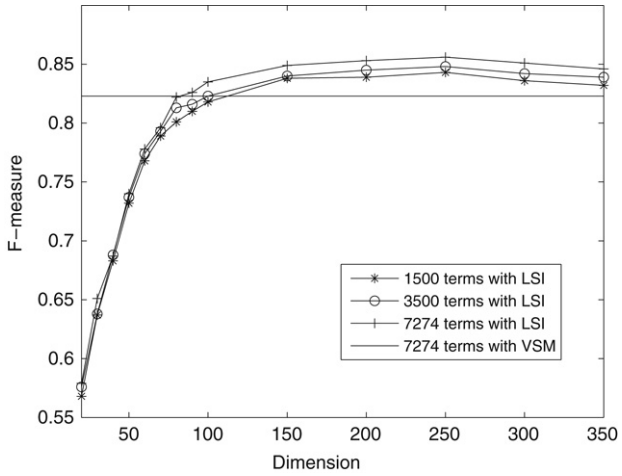

Fig. 5. Cluster performance against the number of dimension on data set 2.

obtain their best performance on 250 dimensions. Furthermore, on the dimension of 250, the result of the GAL method with 1000 terms is very close to that with 2500 terms.

We can see from Fig. 4 that the computational time of GAL is increased with a higher dimensionality. For all numbers of dimensions, GALs are faster than genetic algorithms using the vector space model. Furthermore, with 250 dimensions it takes 26.8 s for the GAL method with 1000 terms to obtain its best performance, which is much faster than that on VSM model.

From Fig. 5 we can see that from about 110 dimensions the performance of GALs outperform that of genetic algorithms using the vector space model. When the dimension is 250, GAL methods with 1500, 3500 and 7274 terms obtain their best performance. Also, with 250 dimensions, the performance of 1500 terms with LSI is close to that of 3500 terms with LSI.

We can see from Fig. 6 that for all numbers of dimensions, GALs are faster than genetic algorithms using VSM. It takes 30.8 s for GAL method with 1500 terms to obtain its best performance on 250 dimensions.

In our approach OpenCV software package [17] is used for computing the singular value decomposition on term-by-document matrix. It provides decomposition results with high quality and reliability. We count the computational time of the best performance of GALs with 1000 terms on data set 1 and 1500 terms on data set 2, respectively, for real time comparisons. The computational time comparisons are shown in Table 1.

5. Conclusions and future works

In this paper, we propose a method of genetic algorithm based on the latent semantic indexing model (GAL) for text clustering. GAL automatically evolves the proper number of clusters as well as providing near optimal text clustering. Analysis shows that LSI not only provides an underlying semantic structure for text models, but also reduces the dimensions

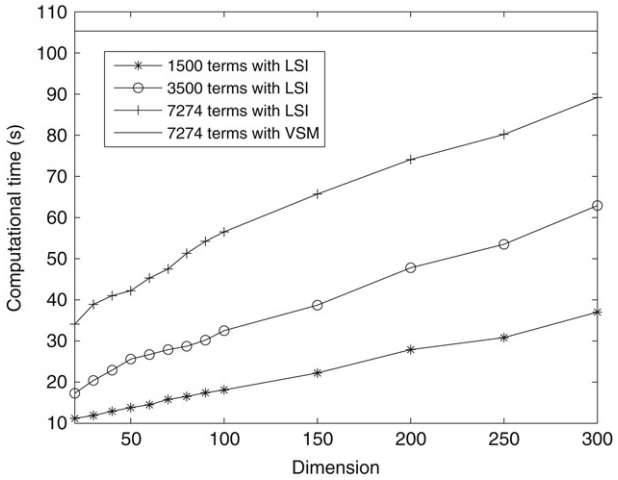

Fig. 6. Computational time against the number of dimension on data set 2.

Table 1
The average computational time comparison.

Methods	Time (s)	
	Data set 1	Data set 2
GA (VSM)	95.8	105.3
SVD	3.6	14.3
GAL	26.8	30.8
SVD + GAL	30.4	45.1

drastically which is very suitable for GA for evolving the optimal text clusters. We apply GAL to Reuter-21578 text collection and demonstrate the effect of our clustering algorithm which is superior to that of GA using VSM. Data set 1 including 600 texts from 3 topics and data set 2 including 1000 texts from 5 topics are tested. When the dimensions are reduced to 250, GALs with 1000 terms for data set 1 and 1500 terms for data set 2 obtain their best performance. In comparison with dimensions of 5857 and 7274 in VSM, GAL has a much lower cost of computational time due to the reduction of dimensions. The experimental results also verify that we have succeeded in reducing the number of terms with highest weight from 5857 to 1000 and 7274 to 1500 before there is a sharp drop in F-measure. These represent reductions of 82.9% and 79.4%. Even fewer numbers of terms are also tested, but the F-measure drops dramatically. In the future, we will refine our program by reducing the evolving time.

Acknowledgements

The authors would like to thank the anonymous referees for their constructive suggestions, which led to the improvement of the paper.

References

[1] W.L.G. Koontz, P.M. Narendra, K. Fucunaga, A graph theoretic approach to nonparametric cluster analysis, IEEE Trans. Comput. C-25 (1975) 936–944.
[2] H. Frigui, R. Krishnapuram, A robust competitive clustering algorithm with application in computer vision, IEEE Trans. Pattern Anal. Mach. Intell. 21 (1999) 450–465.
[3] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley Publishing Co., 1974.
[4] S.Z. Selim, M.A. Ismail, K-means-type algorithm: Generalized convergence theorem and characterization of local optimality, IEEE Trans. Pattern Anal. 6 (1984) 81–87.
[5] W.L.G. Koontz, P.M. Narendra, K. Fukunaga, A branch and bound clustering algorithm, IEEE Trans. Comput. C-24 (1975) 908–915.
[6] A.L. Vizine, Leandro N. de Castro, Eduardo R. Hruschka, Ricardo R. Gudwin, Towards improving clustering ants: An adaptive ant clustering algorithm, Informatica 29 (2005) 143–154.
[7] U. Maulik, S. Bandyopadhyay, Genetic algorithm-based clustering technique, Pattern Recognit. 33 (2000) 1455–1465.
[8] U. Mauilk, S. Bandyopadhyay, Performance evaluation of some clustering algorithms and validity indices, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 1650–1654.
[9] W. Song, S.C. Park, Genetic algorithm-based text clustering technique, LNCS 4221 (2006) 779–782.
[10] S. Zelikovitz, H. Hirsh, Using LSI for text classification in the presence of background text, in: 10th International Conference on Information and Knowledge Management, Atlanta, 2001, pp. 113–118.
[11] S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, R.A. Harshman, Indexing by latent semantic analysis, J. Amer. Soc. Inform. Sci. 41 (1990) 391–407.
[12] J.-T. Sun, Z. Chen, H.-J. Zeng, Y. Lu, C.-Y. Shi, W.-Y. Ma, Supervised latent semantic indexing for document categorization, in: 4th IEEE International Conference on Data Mining, Beijing, 2004, pp. 535–538.

[13] M.W. Berry, S.T. Dumais, G.W. OBrien, Using linear algebra for intelligent information retrieval, in: SIAM Conf., 1995, pp. 573–595.
[14] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.
[15] S. Bandyopadhyay, U. Mauilk, Nonparametric genetic clustering: Comparison of validity indices, IEEE Trans. System Man Cybern.-Part C Applications and Reviews 31 (2001) 120–125.
[16] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Intell. 1 (1979) 224–227.
[17] Program Open Computer Vision Library. Available: http://sourceforge.net/projects/opencvlibrary/.