A Fuzzy Self-Constructing Feature Clustering Algorithm for Text Classification

Takbhate T.K.

Abstract- **Feature clustering is a powerful method to reduce the dimensionality of feature vectors for text classification. In this paper, we propose a fuzzy similarity-based self-constructing algorithm for feature clustering. The words in the feature vector of a document set are grouped into clusters, based on similarity test. Words that are similar to each other are grouped into the same cluster. Each cluster is characterized by a membership function with statistical mean and deviation. When all the words have been fed in, a desired number of clusters are formed automatically. We then have one extracted feature for each cluster. The extracted feature, corresponding to a cluster, is a weighted combination of the words contained in the cluster. By this algorithm, the derived membership functions match closely with and describe properly the real distribution of the training data. Besides, the user need not specify the number of extracted features in advance, and trial-and-error for determining the appropriate number of extracted features can then be avoided. Experimental results show that our method can run faster and obtain better extracted features than other methods.**

Keywords —**Fuzzy similarity, feature clustering, feature extraction, feature reduction, text classification.**

I. Introduction

In text classification, the dimensionality of the feature vector is usually huge. For example, 20 Newsgroups and Reuters21578 top 10, which are two real-world data sets, both have more than 15,000 features. Such high dimensionality can be a severe obstacle for classification algorithms. To alleviate this difficulty, feature reduction approaches are applied before document classification tasks are performed. Two major approaches, feature selection and feature extraction have been proposed for feature reduction. In general, feature extraction approaches are more effective than feature selection techniques, but are more computationally expensive. Therefore, developing scalable and efficient feature extraction algorithms is highly demanded for dealing with high-dimensional document data sets.

College of Engineering Osmanabad
tushar.takbhate@gmail.com

Classical feature extraction methods aim to convert the representation of the original high-dimensional data set into a lower-dimensional data set by a projecting process through algebraic transformations. For example, Principal Component Analysis, Linear, Discriminant Analysis Maximum Margin Criterion and Orthogonal Centroid algorithm perform the projection by linear transformations, while Locally Linear Embedding , ISOMAP , and Laplacian Eigenmaps do feature extraction by nonlinear transformations. In practice, linear algorithms are in wider use due to their efficiency. Several scalable online linear feature extraction algorithms have been proposed to improve the computational complexity. However, the complexity of these approaches is still high. Feature clustering , is one of effective techniques for feature reduction in text classification. The idea of feature clustering is to group the original features into clusters with a high degree of pair wise semantic relatedness. Each cluster is treated as a single new feature, and, thus, feature dimensionality can be drastically reduced.

The first feature extraction method based on feature clustering was proposed by Baker and McCallum which was derived from the “distributional clustering” idea of Pereira et al. Al-Mubaid and Umair used distributional clustering to generate an efficient representation of documents and applied a learning logic approach for training text classifiers. The Agglomerative Information Bottleneck approach was proposed by Tishby et al. The divisive information-theoretic feature clustering algorithm was proposed by Dhillon et al. which is an information-theoretic feature clustering approach, and is more effective than other feature clustering methods. In these feature clustering methods, each new feature I generated by combining a subset of the original words. However, difficulties are associated with these methods. A word is exactly assigned to a subset, i.e., hard-clustering, based on the similarity magnitudes between the word and the existing subsets, even if the differences among these magnitudes are small. Also, the mean and the variance of a cluster are not considered when similarity with respect to the cluster is computed. Furthermore, these methods require the number of new features be specified in advance by the user.

II. Proposed System

In this paper we propose a fuzzy similarity-based self-constructing feature clustering algorithm, which is an incremental feature clustering approach to reduce the number of features for the text classification task. The words in the feature vector of a document set are represented as distributions, and processed one after another. Words that are similar to each other are grouped into the same cluster. Each cluster is characterized by a membership function with statistical mean and deviation. If a word is not similar to any existing cluster, a new cluster is created for this word. Similarity between a word and a cluster is defined by considering both the mean and the variance of the cluster. When all the words have been fed in, a desired number of clusters are formed automatically. We then have one extracted feature for each cluster. The extracted feature corresponding to a cluster is a weighted combination of the words contained in the cluster. Three ways of weighting, hard, soft, and mixed, are introduced. By this algorithm, the derived membership functions match closely with and describe properly the real distribution of the training data.

III. Feature Clustering

We propose a fuzzy similarity-based self-constructing feature clustering algorithm, which is an incremental feature clustering approach to reduce the number of features for the text classification task. The words in the feature vector of a document set are represented as distributions, and processed one after another. Words that are similar to each other are grouped into the same cluster. Each cluster is characterized by a membership function with statistical mean and deviation. If a word is not similar to any existing cluster, a new cluster is created for this word. Similarity between a word and a cluster is defined by considering both the mean and the variance of the cluster. When all the words have been fed in, a desired number of clusters are formed automatically. We then have one extracted feature for each cluster. The extracted feature corresponding to a cluster is a weighted combination of the words contained in the cluster.

Three ways of weighting, hard, soft, and mixed, are introduced. By this algorithm, the derived membership functions match closely with and describe properly the real distribution of the training data. Besides, the user need not specify the number of extracted features in advance, and trial-and-error for determining the appropriate number of extracted features can then be avoided. Experiments on real world data sets show that our method can run faster and obtain better extracted features than other methods

Feature Reduction:

In general, there are two ways of doing feature reduction, feature selection, and feature extraction. By feature selection approaches, a new feature set $\mathbf{W}' = \{w'_1, w'_2, \ldots, w'_k\}$ is obtained, which is a subset of the original feature set W. Then W' is used as inputs for classification tasks. Information Gain (IG) is frequently employed in the feature selection approach. It measures the reduced uncertainty by an information-theoretic measure and gives each word a weight. The weight of a word w_j is calculated as follows:

$$IG(w_j) = -\sum_{l=1}^{p} P(c_l) log P(c_l) + P(w_j) \sum_{l=1}^{p} P(c_l|w_j) log P(c_l|w_j) + P(\overline{w}_j) \sum_{l=1}^{p} P(c_l|\overline{w}_j) log P(c_l|\overline{w}_j),$$

where P(cl) denotes the prior probability for class cl, P(wj) denotes the prior probability for feature wj, P(wj) is identical to 1 _ P(wj), and P(cl|wj) and P(cl|wj) denote the probability for class cl with the presence and absence, respectively, of wj. The words of top k weights in W are selected as the features in W'.

In feature extraction approaches, extracted features are obtained by a projecting process through algebraic transformations. An incremental orthogonal centroid (IOC)
algorithm. Let a corpus of documents be represented as an m*n matrix $\mathbf{X} \in \bar{\mathbf{R}}^{m \times n}$, where m is the number of features in the feature set and n is the number of documents in the document set. IOC tries to find an optimal transformation matrix $\mathbf{F}^* \in \mathbf{R}^{m \times k}$, where k is the desired number of extracted features, according to the following criterion:

$$\mathbf{F}^* = \arg\max trace(\mathbf{F}^T \mathbf{S}_b \mathbf{F}),$$

where $\mathbf{F} \in \mathbf{R}^{m \times k}$ and $\mathbf{F}^T\mathbf{F} = \mathbf{I}$, and

$$\mathbf{S}_b = \sum_{q=1}^{p} P(c_q)(\mathbf{M}_q - \mathbf{M}_{all})(\mathbf{M}_q - \mathbf{M}_{all})^T$$

with $P(c_q)$ being the prior probability for a pattern belonging to class c_q, M_q being the mean vector of class c_q, and Mall being the mean vector of all patterns.

Feature Extraction

Formally, feature extraction can be expressed in the following form:

$$\mathbf{D}' = \mathbf{DT},$$

where

$$\mathbf{D} = [\mathbf{d}_1\ \mathbf{d}_2\ \cdots\ \mathbf{d}_n]^T,$$

$$\mathbf{D}' = [\mathbf{d}'_1\ \mathbf{d}'_2\ \cdots\ \mathbf{d}'_n]^T,$$

$$\mathbf{T} = \begin{bmatrix} t_{11} & \cdots & t_{1k} \\ t_{21} & \cdots & t_{2k} \\ \vdots & \ddots & \vdots \\ t_{m1} & \cdots & t_{mk} \end{bmatrix},$$

with

$$\mathbf{d}_i = [d_{i1}\ d_{i2}\ \cdots\ d_{im}],$$
$$\mathbf{d}'_i = [d'_{i1}\ d'_{i2}\ \cdots\ d'_{ik}],$$

For $1 \le i \le n$ . . Clearly, T is a weighting matrix. The goal of feature reduction is achieved by finding an appropriate T such that k is smaller than m. In the divisive information theoretic feature clustering algorithm described , the elements of T in are binary and can be defined as follows:

$$t_{ij} = \begin{cases} 1, & \text{if } w_i \in \mathbf{W}_j, \\ 0, & \text{otherwise,} \end{cases}$$

Where $1 \le i \le m$ and $1 \le j \le k.$ That is, if a word w_i belongs to cluster $\mathbf{W}_j$, t_{ij} is 1; otherwise t_{ij} is 0.

By applying our clustering algorithm, word patterns have been grouped into clusters, and words in the feature vector $\mathbf{W}$ are also clustered accordingly. For one cluster, we have one extracted feature. Since we have k clusters, we have k extracted features The elements of $\mathbf{T}$ are derived based on the obtained clusters, and feature extraction will be done. We propose three weighting approaches: hard, soft, and mixed. In the hard-weighting approach, each word is only allowed to belong to a cluster, and so it only contributes to a new extracted feature. In this case, the elements of $\mathbf{T}$ in are defined as follows:

$$t_{ij} = \begin{cases} 1, & \text{if } j = arg\ \max_{1 \le \alpha \le k}(\mu_{G_\alpha}(\mathbf{x}_i)), \\ 0, & \text{otherwise.} \end{cases}$$

Note that if j is not unique in , one of them is chosen randomly. In the soft-weighting approach, each word is allowed to contribute to all new extracted features, with the degrees depending on the values of the membership functions. The elements of $\mathbf{T}$ in are defined as follows:

$$t_{ij} = \mu_{G_j}(\mathbf{x}_i).$$

The mixed-weighting approach is a combination of the hard-weighting approach and the soft-weighting approach. For this case, the elements of $\mathbf{T}$ in are defined as follows:

$$t_{ij} = (\gamma) \times t_{ij}^H + (1 - \gamma) \times t_{ij,}^S$$

t_{ij}^H is obtained by and t_{ij}^S is obtained , and γ is a user-defined constant lying between 0 and 1. Note that γ is not related to the clustering. It concerns the merge of component features in a cluster into a resulting feature. The merge can be “hard” or “soft” by setting γ to 1 or 0. By selecting the value of γ , we provide flexibility to the user. When the similarity threshold is small, the number of clusters is small, and each cluster covers more training patterns. In this case, a smaller γ will favor soft-weighting and get a higher accuracy. On the contrary, when the similarity threshold is large, the number of clusters is large, and each cluster covers fewer training patterns. In

this case, a larger γ will favor hard-weighting and get a higher accuracy.

III. Methodology

1. Preprocessing:

In this module we construct the word pattern of training document set. Read the document set and remove the stop words and perform stemming process. Get the feature vector from the training document .Next we construct the word pattern. Suppose, we are given a document set D of n documents d1, d2, . . ., d_n, together with the feature vector W of m words w1, w2, . . ., w_m and p classes c1, c2, . . . , c_p, We construct one word pattern for each word in W. For word w_i, its word pattern x_i is defined, similarly by

$$\begin{aligned}\mathbf{x}_i &= <x_{i1}, x_{i2}, \ldots, x_{ip}> \\ &= <P(c_1|w_i), P(c_2|w_i), \ldots, P(c_p|w_i)>,\end{aligned}$$

where

$$P(c_j|w_i) = \frac{\sum_{q=1}^{n} d_{qi} \times \delta_{qj}}{\sum_{q=1}^{n} d_{qi}},$$

for 1 < j< p. Note that d_{qi} indicates the number of occurrences of w_i in document d_q

Also, S_{qj}

$$\delta_{qj} = \begin{cases} 1, & \text{if document } \mathbf{d}_q \text{ belongs to class } c_j, \\ 0, & \text{otherwise.} \end{cases}$$

2. Self-constructing clustering:

In this module we cluster the features using self constructing clustering algorithm. For each word pattern, the similarity of this word pattern to each existing cluster is calculated to decide whether it is combined into an existing cluster or a new cluster is created. Once a new cluster is created, the corresponding membership function should be initialized. On the contrary, when the word pattern is combined into an existing cluster, the membership function of that cluster should be updated accordingly.

3. Feature Extraction:

Word patterns have been grouped into clusters, and words in the feature vector W are also clustered accordingly. For one cluster, we have one extracted feature. Since we have k clusters, we have k extracted features. The elements of T are derived based on the obtained clusters, and feature extraction will be done. We propose three weighting approaches: hard, soft, and mixed. In the hard-weighting approach, each word is only allowed to belong to a cluster, and so it only contributes to a new extracted feature. In the soft-weighting approach, each word is allowed to contribute to all new extracted features, with the degrees depending on the values of the membership functions. The mixed-weighting approach is a combination of the hard-weighting approach and the soft-weighting approach.

4. Text Classification:

Given a set D of training documents, text classification can be done as follows: Get the training document set and specify the similarity threshold ρ. Assume that k clusters are obtained for the words in the feature vector W. Then find the weighting matrix T and convert D to D`. Using D` as training data, a classifier based on weka (Waikato Environment for Knowledge Analysis)) is built. **Weka** (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. Weka is a collection of machine learning algorithms for data mining tasks. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Using weka we classify the text.

IV Results

Similarity-based clustering is one of the techniques we have developed in our machine learning research. In this paper, we apply this clustering technique to text categorization problems. We are also applying it to other problems, such as image segmentation, data sampling, fuzzy modeling, web mining, etc. The work of this paper was motivated by distributional word clustering proposed in We found that when a document set is transformed to a collection of word patterns, as by , the relevance among

word patterns can be measured, and the word patterns can be grouped by applying our similarity based clustering algorithm. Our method is good for text categorization problems due to the suitability of the distributional word clustering concept.

V Conclusion

We have presented a fuzzy self-constructing feature clustering (FFC) algorithm, which is an incremental clustering approach to reduce the dimensionality of the features in text classification. Features that are similar to each other are grouped into the same cluster. Each cluster is characterized by a membership function with statistical mean and deviation. If a word is not similar to any existing cluster, a new cluster is created for this word. Similarity between a word and a cluster is defined by considering both the mean and the variance of the cluster. When all the words have been fed in, a desired number of clusters are formed automatically. We then have one extracted feature for each cluster. The extracted feature corresponding to a cluster is a weighted combination of the words contained in the cluster. By this algorithm, the derived membership functions match closely with and describe properly the real distribution of the training data. Besides, the user need not specify the number of extracted features in advance, and trial-and-error for determining the appropriate number of extracted features can then be avoided. Experiments on three real-world data sets have demonstrated that our method can run faster and obtain better extracted features than other methods

References

[1] H. Kim, P. Howland, and H. Park, "Dimension Reduction in Text Classification with Support Vector Machines," J. Machine Learning Research, vol. 6, pp. 37-53, 2005.

[2] K. Daphne and M. Sahami, "Toward Optimal Feature Selection," Proc. 13th Int'l Conf. Machine Learning, pp. 284-292, 1996.

[3] R. Kohavi and G. John, "Wrappers for Feature Subset Selection," Aritficial Intelligence, vol. 97, no. 1-2, pp. 273-324, 1997.

[4] Y. Yang and J.O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," Proc. 14th Int'l Conf. Machine Learning, pp. 412-420, 1997.

[5] D.D. Lewis, "Feature Selection and Feature Extraction for Text Categorization," Proc. Workshop Speech and Natural Language, pp. 212-217, 1992.

[6] H. Li, T. Jiang, and K. Zang, "Efficient and Robust Feature Extraction by Maximum Margin Criterion," T. Sebastian, S. Lawrence, and S. Bernhard eds. Advances in Neural Information
Processing System, pp. 97-104, Springer, 2004.

[7] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," Advances in Neural Information Processing Systems, vol. 14, pp. 585-591, The MIT Press 2002.

[8] J. Weng, Y. Zhang, and W.S. Hwang, "Candid Covariance-Free Incremental Principal Component Analysis," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 25, no. 8, pp. 1034-1040, Aug. 2003.

[9] L.D. Baker and A. McCallum, "Distributional Clustering of Words for Text Classification," Proc. ACM SIGIR, pp. 96-103, 1998.

[10] I.S. Dhillon, S. Mallela, and R. Kumar, "A Divisive Infomation-Theoretic Feature Clustering Algorithm for Text Classification," J. Machine Learning Research, vol. 3, pp. 1265-1287, 2003.

[11] D. Ienco and R. Meo, "Exploration and Reduction of the Feature Space by Hierarchical Clustering," Proc. SIAM Conf. Data Mining, pp. 577-587, 2008. [29] N. Slonim and N. Tishby, "The Power of Word Clusters for Text Classification," Proc. 23rd European Colloquium on Information Retrieval Research (ECIR), 2001.