A Novel Feature Selection for Text Data Using Fuzzy C-MEANS Clustering

[Group 10] *Achyudh Ram (2013B3A7439G), George Joseph (2013B5A7112G), Shobhik Bhadraray (2013B5A7665G), Shantanu Jain (2013B3A7547G), Krutarth Nakade (2013B3A7574G)* under the guidance of *Prof. Rajendra Roul* under partial fulfillment of the course CS F469 Information Retrieval

***Abstract* — We make an attempt on developing a novel feature selection technique for text data using Term-Term correlation, based on clusters generated using the Fuzzy C-Means (FCM) Algorithm and we test it on WebKB and Reuters-R8 datasets. Top 'k' features are selected from these datasets using cosine similarity scores on the semantic centroids calculated from the normalized correlation factors. We attempt to show that the features selected through this mechanism shall result in comparable F-measures for classification tasks in comparison to more traditional feature selection techniques like Chi-Squared, Mutual Information and Variance Thresholding. We also intend to show that this feature selection technique is more robust with a lower reduction in F-measure with a given reduction in the number of top features chosen vis-a-vis the other approaches and thus, the resulting lower classification time, to an extent, makes up for the increased feature selection time.**

***Index Terms*—Feature selection, Fuzzy C-means, Semantic centroids, WebKB, Reuters-R8,**

I. Introduction

Clustering is the involves grouping data into different sets of objects in a way that objects belonging to the same cluster are more 'similar', the similarity being measured using a metric (like Euclidean distance), to each other than to objects in another cluster. Many popular algorithms exist for clustering data into such sets, and in this approach we use the Fuzzy C-Means (FCM) algorithm. This approach is also compatible with the more popular K-Means clustering, though that offers less flexibility in hyper parameter tuning. The algorithm distributes a finite corpus of *n* documents into *c* clusters, by returning a list of *c* cluster centroids that represent each of these clusters. A document is not said to belong to any particular cluster, but is instead assumed to have a degree of membership in each of the *c* clusters. The FCM algorithm returns a matrix that details this degree of membership of each document within every cluster.

The FCM Algorithm aims to minimize the following function:

$$T_m = \sum_{j=1}^{n} \sum_{i=1}^{c} v_{ji} ||d_{ji}||^2$$

where, distance $d_{ij} = x_i - c_j$, *m* is the fuzzy coefficient generally set to 2, c_j is the centroid (vector) of cluster *j*, x_i is the i^{th} document, $v_{ij} \in [0,1]$ is the degree of membership of x_i with respect to c_j and $\sum v_{ji} = 1$, *i*=1,2,..,*n*. One can iteratively find the values of c_j and v_{ij} updated with each iteration by using the following equations:

$$c_i = \frac{\sum_{j=1}^{n} v_{ji}^m - x_j}{\sum_{j=1}^{c} v_{ji}^m}$$

$$v_{ji} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ji}}{x_j - c_k} \right)^{\frac{2}{m-1}}}$$

II. Methodology

We used the following approach to select the top features:

1. Pre-process the Corpus *P* and convert the document into vectors. All the documents *d* of this corpus are represented as Vectors using the TF-IDF weights for each term in the term space. Dimension of *P* is *b* x *l*, where *b* and *l* are the total number of documents and the terms in the corpus respectively.

2. The FCM algorithm is now run on the corpus *P* and this generates *k* clusters. Now the aim remains to select important terms from each of these *k* clusters.

3. Now to select important keywords from these clusters based on Term-Term correlation, we calculate the Frequency based correlation factor CF. For every pair of terms *i* and *j* of the cluster h:

$$CF_{ji} = \sum_{m \epsilon P} f_{jm} * f_{im}$$

where f_{im} and f_{jm} represent the frequency of ith and jth terms in the mth document of the cluster h.

4. An association matrix is now constructed where each row consists of the correlation values between terms W_i and W_j, which generates a semantic component of (W_i). $CF_{ij}=CF_{ji}$ i.e $CF_{ij}=CF^T_{ij}$ and each W_i represents a term vector.

$$\begin{array}{c|cccccc}
 & W_1 & W_2 & W_3 & \dots & W_n \\
\hline
W_1 & CF_{11} & CF_{12} & CF_{13} & \dots & CF_{1n} \\
W_2 & CF_{21} & CF_{22} & CF_{23} & \dots & CF_{2n} \\
W_3 & CF_{31} & CF_{32} & CF_{33} & \dots & CF_{3n} \\
. & . & . & . & . & . \\
. & . & . & . & . & . \\
. & . & . & . & . & . \\
W_n & CF_{n1} & CF_{n2} & CF_{n3} & \dots & CF_{nn}
\end{array}$$

5. The association scores of CF_{ij} are normalized using the following equation, which floats these values between 0 and 1. Normalized score is 1 if two keywords have the same frequency in all the documents. All diagonal entries of the NCF are clearly 1 when *i* = *j*.

$$NCF_{ji} = \frac{CF_{ji}}{CF_{jj} + CF_{ii} - CF_{ji}}$$

$$\begin{array}{c|cccccc}
 & W_1 & W_2 & W_3 & \dots & W_n \\
\hline
W_1 & NCF_{11} & NCF_{12} & NCF_{13} & \dots & NCF_{1n} \\
W_2 & NCF_{21} & NCF_{22} & NCF_{23} & \dots & NCF_{2n} \\
W_3 & NCF_{31} & NCF_{32} & NCF_{33} & \dots & NCF_{3n} \\
. & . & . & . & . & . \\
. & . & . & . & . & . \\
. & . & . & . & . & . \\
W_n & NCF_{n1} & NCF_{n2} & NCF_{n3} & \dots & NCF_{nn}
\end{array}$$

6. For each term vector W_i the mean of all correlation values belonging to NCF are calculated and this gives a *n*-dimensional vector named as the semantic centroid vector *sc*.

$$sc_i = \frac{\sum_{j=1}^{n} NCF_{ij}}{n}$$

$$\begin{bmatrix}
sc_1 = \frac{(NCF_{11} + NCF_{12} + NCF_{13} + \dots + NCF_{1n})}{n} \\
sc_2 = \frac{(NCF_{21} + NCF_{22} + NCF_{23} + \dots + NCF_{2n})}{n} \\
sc_3 = \frac{(NCF_{31} + NCF_{32} + NCF_{33} + \dots + NCF_{1n})}{n} \\
. \\
. \\
. \\
sc_n = \frac{(NCF_{n1} + NCF_{n2} + NCF_{n3} + \dots + NCF_{nn})}{n}
\end{bmatrix}$$

7. Now the cosine-similarity score is calculated between the semantic centroid vector *sc* and the vectors W_i using the following equation:

$$cosine\text{-}similarity(\vec{sc}, \vec{W_i}) = \frac{\vec{sc}.\vec{W_i}}{|\vec{sc}| * |\vec{W_i}|}$$

All terms are ranked and the top '*m%*' terms, with *m* decided empirically, are selected on the basis of their cosine-similarity scores and the irrelevant terms are discarded from the given cluster.

8. The above procedure is repeated for all the k clusters, and finally top m% terms are obtained from each of these.

III. EXPERIMENTAL SETUP

The implementation made use of the classifiers and feature selection algorithms implemented in the Scikit-learn library for Python, and Scikit-fuzzy package was used for the FCM algorithm. Stopword removal, Snowball Stemmer and WordNet Lemmatizer from the NLTK library were used to preprocess the corpora. The computation of the CF matrix is highly time consuming and hence to perform the computations and store the intermediate results, the popular Numpy package was used. The Pandas package provided the necessary utilities for reading and storing the pre-processed text data. Further, to ensure that fair comparisons are made across classifiers, the clustering generated by the FCM algorithm is saved for every dataset and then reused.

Feature selection using FCM algorithm was initially tested over the multi-class classification Iris dataset. Preprocessed Iris dataset was used along with the FCM algorithm with a fuzzy coefficient of 2. The resulting clustering had a Rand score of 0.78. However, this served as just a proof of concept and the IRIS dataset already had a limited number of features and is very easy to classify. The takeaway from this part of the experiment was this feature selection technique can accurately identify the high correlation between two of the four features in the dataset and eliminate one of them. If we were to restrict the classification to just one feature from the dataset, the classifier still did well by correctly identifying the *Petal.Width* feature as the key attribute that determines the classes of the samples. This resulted in an accuracy of around 95% with just one attribute.

The traditional feature selection techniques used for comparing with the proposed method are:

- *Variance Thresholding:* Feature selection method that ranks the features in the dataset according to the variance that it shows across the various data points in the dataset, and then according to the mentioned threshold selects only the top k features. It has an advantage over other feature selection methods in being unsupervised, i.e., it doesn't need to look at the labels. Given a TF-IDF matrix where each row is a TF-IDF vector representing the document in the term space. Thus if the number of total documents is b and the number of terms is l, we find ourselves with a b x l matrix. The variance of the TF-IDF values is calculated for each of the columns. This results in a 1 x l matrix of column wise variance values. We can now set a threshold such that all the features with variance less than the threshold are dropped. The basic rationale behind the method is that features that exhibit lesser variance across documents, contribute lesser towards discriminating between documents. This is equivalent to filtering on TF-IDF values. For our experiments, for each dataset that we worked with, we chose a threshold value such that we select only the top 10% of the features exhibiting highest variance across documents.

- *Chi-Squared:* Feature Selection method that uses the χ^2 test. Unlike variance thresholding, this method requires is supervised. The χ^2 test is used to determine the independence between two events. In a classification scenario, the two events are the feature and the class label. Thus we apply the χ^2 test between every feature of the dataset and the class labels. Based on the score of the test, we rank the features. A feature belonging to the (feature, label) pair showing the highest dependency is ranked the highest, while the feature belonging to the (feature, label) pair showing the highest independence is ranked the least. Once we have a ranking, we can take the top k features. For our experiments, for each dataset that we worked with, we chose the top 10% of the features exhibiting highest dependence with the class labels.

- *Mutual Information Gain:* This supervised feature selection technique is also a probabilistic measure, that for two random variables A and B, quantifies how much information can we get about one random variable from another random variable. It measures how similar p(AB) is to p(A)p(B).

The following popular classifiers were used in our experiment:

- *Weighted K-Nearest Neighbors*: Each of the neighbors is weighted according to the inverse of the distance between the data sample and the neighbors. Thus providing greater influence to the neighbors that are closer than the ones that are further. This gives better results than uniform weighting. We used both $k = 3$ and $k = 5$ in this experiment.

- *Random Forest Classifiers:* An ensemble classification technique, wherein we have an ensemble of decision trees, each decision tree operating on a subset of the dataset. The approach uses averaging to improve predictive accuracy and reduce overfitting. During our study, we tested using random forests with 50 and 100 decision trees, with splitting based on entropy gain.

- *Support Vector Machine:* A type of classifier that seeks to maximize the margin between the decision boundary and the samples of the different classes by constructing a hyperplane in an n-dimensional space. A technique called the kernel trick, which is a transformation of the data samples to a higher dimension enables them to classify non-linear datasets. In our experiments, we made use of Support Vector Machines with linear as well as RBF kernels, with an all-vs-all decision boundary shape.

- *Naive Bayes:* A probabilistic classification method that has two major underlying assumptions. The features of the dataset are mutually independent and the dataset is drawn from an underlying distribution. This underlying distribution is assumed according to the variant of Naive Bayes being used. In our experiments we performed classification using Gaussian, Binomial and Multinomial Naive Bayes.

We tested all the above combinations feature selection techniques and classifiers on some popular text corpora like WebKB and Reuters-R8. Five-fold cross validation was performed on the train set of both the datasets. The algorithm to generate the CF matrix was made to scale to that many cores as the number of classes (in this case 4 and 8 respectively) in the dataset to make it feasible to generate the matrix. Since the generation of CF matrix is the bottleneck step, it sped up the feature selection and classification task considerably. The overall computational cost (in time) of the generation of the matrix is O(kn^2), where n is the total number of terms in all the documents and k is the total number of classes in the corpus. Upon SIMD parallelization, this cost reduces to O(n^2) with k threads operating in parallel.

Attempts to implement the given algorithm on the 20-Newsgroups dataset proved to be futile due to the large size of the dataset and the limitation on the computing power available for feature selection. The relative success of the implementation on the other datasets provides a fair chance that the same would be successful on the 20-Newsgroups, if given sufficient computing power and time.

As the evaluation metric, we used the weighted F-measure. For multiclass classification scenarios where the number of data samples in each class is skewed, simply averaging the F-measure over all the class labels is not a true representation of the classification performance. Thus, in order to find a more representative metric, we calculate a weighted average with the weight as the number of true data samples in the class.

IV. Results

The experiment was setup into two stages. In the initial stage, we take the top 10% of the features across all the three feature selection techniques and compare the final classification accuracies resulting from these features. In the second stage, only top 5% of the features from the proposed technique was selected and compared with the performance of the traditional feature selection algorithms which choose the default top 10% of the features. This is done to show that the proposed technique is robust and that it can actually recover the longer feature selection time by attaining comparable accuracies with a lesser number of features and thereby saving on the time taken for classification.

Feature Selection Technique	F-Measure on WebKB*	F-Measure on Reuters-R8*
Variance Thres.	0.87	0.94
Chi-Squared	0.88	0.94
Mutual Info.	0.87	0.93
C-Means	0.89	0.94

Table 1 - F-MEASURE FROM USING TOP 10% FEATURES ON A RANDOM FOREST CLASSIFIER WITH 100 TREES

Feature Selection Technique	F-Measure on WebKB*	F-Measure on Reuters-R8*
Variance Thres.	0.87	0.94
Chi-Squared	0.88	0.94
Mutual Info.	0.87	0.93
C-Means	0.87	0.94

Table 1 - F-MEASURE FROM USING TOP 5% FEATURES ON A RANDOM FOREST CLASSIFIER WITH 100 TREES

Feature Selection Technique	F-Measure on WebKB*	F-Measure on Reuters-R8*
Variance Thres.	0.85	0.95
Chi-Squared	0.90	0.95
Mutual Info.	0.88	0.95
C-Means	0.87	0.95

Table 1 - F-MEASURE FROM USING TOP 10% FEATURES ON A LINEAR SUPPORT VECTOR CLASSIFIER

Feature Selection Technique	F-Measure on WebKB*	F-Measure on Reuters-R8*
Variance Thres.	0.85	0.95
Chi-Squared	0.90	0.95
Mutual Info.	0.88	0.95
C-Means	0.86	0.94

Table 1 - F-MEASURE FROM USING TOP 5% FEATURES ON A LINEAR SUPPORT VECTOR CLASSIFIER

From the above tables, we can summarize that there was less than 3% decrease in accuracy over traditional feature selection techniques with half the number of features. The results from all the classifiers tested are shown in the appendix.

The Achilles heel of the proposed algorithm is its reliance of being able to perform a moderately well-distributed clustering of the documents in the corpus. Since the proposed feature selection method basically works by selecting the top *k%* percent features from each cluster, inherently the performance of the proposed technique depends heavily on the performance of the clustering on the data. In the course of the study, we experimented with several datasets using 2 popular clustering algorithms (K-Means and Fuzzy C-Means) and measured the feature selection performance with respect to the Rand score of the clustering. We found out empirically that a minimum Rand score of about 0.3 is essential for the proposed algorithm to give competitive results when compared to that of the traditional feature selection methods. When working on the WebKB Dataset, we found that Fuzzy C-Means gave a consistent rand score of about 0.35 and that enabled the proposed algorithm to perform efficient and robust feature selection. On the other hand, with the Reuters-R8 Dataset, Fuzzy C-Means gave varied Rand scores depending on the random initialization of cluster centroids. Though optimization using the fuzzy coefficient and initial cluster centroids did result in a Rand score of 0.3, often clustering with Fuzzy C-Means resulted in skewed clusters with zero documents, in which case the proposed algorithm didn’t run at all, as it requires the number of members in each cluster to be a non-zero positive integer.

We also investigated the clustering performance as given by K-means clustering algorithm, and found it to give a consistent Rand score of 0.25 on the Reuters-R8 dataset, and consequently the proposed algorithm also consistently gave good performance in the final classification task.

V. Conclusion

From the experimental results, we can see that the proposed algorithm is more robust as it leads to a slower reduction in F-measure with decreasing percentage of features selected. Further, on classifiers like Linear-SVM and RBF, it usually matches or outperforms the traditional feature selection techniques. The method can also be extended to work with K-Means clustering as we have shown for the Reuters-R8 dataset. This provides equally good F-measures on the final classification task when compared with the Fuzzy C-Means clustering, but results in faster performance than the latter.

However, there is still considerable scope for future improvement. It takes considerable time and computational resources for feature selection, with upwards of 3 hours and 24 GBs of RAM for generating the CF matrix for the Reuters-R8 dataset despite penalization. Further the possibility of an uneven clustering adds to the time required for feature selection as a hyperparameter search would have to be made to improve the Rand score of the clustering. The same dataset can also result in a varied clustering performance due to random cluster centroid initializations.

We also note that the proposed feature selection algorithm, when coupled with Naive Bayes classifiers, consistently gives lower F-measures as compared to traditional feature selection methods. This is because, the basis for feature selection in the proposed algorithm is the correlation between features, thus it ends up selecting features that are highly correlated between documents that belong to different classes. This is a contradiction to the assumptions made by the Naive Bayes classifier, which has a prerequisite that the features being used for classification should be independent.

Finally, we observe that the SVM with an RBF kernel, gives bad F-measure scores across all feature selection techniques (traditional as well as the proposed method). This is due to the already high dimensionality of the features of the datasets under consideration. Thus when SVM tries to project the data in an even higher dimensionality space, it ends up grossly overfitting the data consequently giving bad classification performance.

References

A. Kraskov, H. Stogbauer and P. Grassberger, “Estimating mutual information”. Phys. Rev. E 69, 2004.

B. C. Ross “Mutual Information between Discrete and Continuous Data Sets”. PLoS ONE 9(2), 2014.

C. S. Tan, “Neighbor-weighted K-nearest neighbor for unbalanced text corpus,” Expert Systems with Applications, vol. 28, no. 4, pp. 667–671, 2005.

Appendix

Results for five-fold cross validation of WebKB

Classifier	*Feature Selection*	*F-measure*
K-NN, k=3	Variance Tresholding	0.64
K-NN, k=5	Variance Tresholding	0.63
SVM, Linear	Variance Tresholding	0.89
SVM, RBF	Variance Tresholding	0.22
RFC, 50 trees	Variance Tresholding	0.87
RFC, 100 trees	Variance Tresholding	0.88
Gaussian Naive Bayes	Variance Tresholding	0.71
Bernoulli Naive Bayes	Variance Tresholding	0.81
K-NN, k=3	Mutual Information	0.68
K-NN, k=5	Mutual Information	0.70
SVM, Linear	Mutual Information	0.86
SVM, RBF	Mutual Information	0.22
RFC, 50 trees	Mutual Information	0.88
RFC, 100 trees	Mutual Information	0.88
Gaussian Naive Bayes	Mutual Information	0.55
Bernoulli Naive Bayes	Mutual Information	0.80
K-NN, k=3	Chi-Squared	0.66
K-NN, k=5	Chi-Squared	0.65
SVM, Linear	Chi-Squared	0.90
SVM, RBF	Chi-Squared	0.22
RFC, 50 trees	Chi-Squared	0.88
RFC, 100 trees	Chi-Squared	0.88
Gaussian Naive Bayes	Chi-Squared	0.77
Bernoulli Naive Bayes	Chi-Squared	0.84
K-NN, k=3	C-Means	0.60
K-NN, k=5	C-Means	0.60
SVM, Linear	C-Means	0.86
SVM, RBF	C-Means	0.22
RFC, 50 trees	C-Means	0.87
RFC, 100 trees	C-Means	0.87
Gaussian Naive Bayes	C-Means	0.37
Multinom. Naive Bayes	C-Means	0.70
Bernoulli Naive Bayes	C-Means	0.72

Results for five-fold cross validation of Reuters-R8

Classifier	*Feature Selection*	*F-measure*
K-NN, k=3	Variance Tresholding	0.87
K-NN, k=5	Variance Tresholding	0.86
SVM, Linear	Variance Tresholding	0.96
SVM, RBF	Variance Tresholding	0.35
RFC, 50 trees	Variance Tresholding	0.93
RFC, 100 trees	Variance Tresholding	0.94
Gaussian Naive Bayes	Variance Tresholding	0.77
Bernoulli Naive Bayes	Variance Tresholding	0.86
K-NN, k=3	Mutual Information	0.90
K-NN, k=5	Mutual Information	0.90
SVM, Linear	Mutual Information	0.95
SVM, RBF	Mutual Information	0.35
RFC, 50 trees	Mutual Information	0.93
RFC, 100 trees	Mutual Information	0.93
Gaussian Naive Bayes	Mutual Information	0.73
Bernoulli Naive Bayes	Mutual Information	0.86
K-NN, k=3	Chi-Squared	0.86
K-NN, k=5	Chi-Squared	0.84
SVM, Linear	Chi-Squared	0.95
SVM, RBF	Chi-Squared	0.35
RFC, 50 trees	Chi-Squared	0.94
RFC, 100 trees	Chi-Squared	0.94
Gaussian Naive Bayes	Chi-Squared	0.82
Bernoulli Naive Bayes	Chi-Squared	0.88
K-NN, k=3	C-Means	0.67
K-NN, k=5	C-Means	0.66
SVM, Linear	C-Means	0.94
SVM, RBF	C-Means	0.29
RFC, 50 trees	C-Means	0.94
RFC, 100 trees	C-Means	0.94
Gaussian Naive Bayes	C-Means	0.46
Multinom. Naive Bayes	C-Means	0.89
Bernoulli Naive Bayes	C-Means	0.83