A New Method For Attribute Extraction with Application on Text Classification

Göksel Biricik[1], Banu Diri[1], Ahmet Coşkun Sönmez[1]

[1]Computer Engineering Department, Yildiz Technical University, İstanbul, Turkey.

[1] goksel@ce.yildiz.edu.tr , [2]banu@ce.yildiz.edu.tr , [3]acsonmez@ce.yildiz.edu.tr

Abstract

We introduce a new method for dimensionality reduction by attribute extraction and evaluate its impact on text classification. The textual contents in body sections of the news in Reuters-21758 are the selected attributes for classification. Using the offered method, high dimension of attributes- words extracted from the news bodies- are projected onto a new hyper plane having dimensions equal to the number of classes. Results show that processing times of classification algorithms dramatically decrease with the attribute extraction method we offer. This is achieved by the fall of the number of attributes given to classifiers. Accuracies of the classification algorithms also increase compared to tests run without using the proposed method.

1. Introduction

Text classification is an information retrieval task in which documents are grouped into different classes or categories. The grouping task classifies documents into a fixed number of predefined categories [1]. One of the models widely used in text classification is the vector space model in which the documents are represented as vectors described by a set of identifiers, for example, words as terms. This model is also known as bag-of-words model. According to this model, every document acts as a bin containing its words. Thinking in the vector space, each term is a dimension for the document vectors. The nature of this representation causes a very high-dimensional and sparse feature space, which is a common problem to deal with when using bag-of-words model. There are two effective ways to overcome this dimensionality problem: Attribute selection and attribute extraction.

Attribute selection algorithms output a subset of the input attributes, results in a lower dimensional space. Instead of using all words as attributes, attribute selection algorithms evaluate attributes on a specific classifier to find the best subset of terms [2]. This results in reduced cost for classification and better classification accuracy. The most popular attribute selection algorithms include document frequency, chi statistic, information gain, term strength and mutual information [3]. Chi-square and correlation coefficient methods have been shown to produce better results than document frequency [4]. The lack of attribute selection algorithms is that the selection procedure is evaluated on a certain classifier. Hence, the produced subset may not be suitable for another classifier to improve its performance.

Attribute extraction algorithms simplify the amount of resource required to describe a large set of data. The high-dimensional and sparse structure of vector space model requires large amount of memory and computation power. The aim of attribute extraction is to combine terms to form a new description for the data with sufficient accuracy. Attribute extraction works by projecting the high-dimensional data into a new, lower-dimensional hyperspace. Mostly used techniques are Principal Components Analysis (PCA), Isomap and Latent Semantic Analysis (LSA). Latent Semantic Indexing is based on LSA and it is the most commonly used algorithm in text mining tasks nowadays. Our method also extracts attributes from the original vector space and projects them into a new hyperspace with dimensions equal to number of classes.

In section 2, we introduce our evaluation dataset and the preprocessing steps. Section 3 gives brief description about related attribute extraction algorithms and introduces the proposed method. In section 4 we discuss our experimental results. Section 5 addresses conclusions and future work.

2. Evaluation Dataset and Preprocessing

2.1. Evaluation Dataset

We prepare our dataset from news feeds of Reuters 21758 dataset, which is a standard test collection for text categorization tasks in information retrieval and machine learning. The dataset contains 21758 documents collected from Reuters newswire in 1987. There are 135 topics to label the categories of the news. While some documents may have one or more topic labels, there are ones that do not contain even a single entry.

We discard the documents with multiple topic entries as well as the documents without topics. Some documents in the dataset contain short description for the news and do not have news body section. We also filter these ones. After this elimination step, we have 12297 documents in 81 topics, each having exactly one topic label and news body section.

2.2. Pre-processing

We use Porter's stemmer [5] to stem the terms of the documents in the dataset. We remove stopwords, numbers and all punctuation marks after stemming. We have 9554 unique documents in hand when we remove the duplicate documents grew out of this process. Distribution of documents among 81 classes is given in Figure 1. Note that Figure 1 is drawn in logarithmic scale. We can see that documents are unevenly distributed among classes. This situation is inconvenient for classification tasks because heterogeneous distributions over classes generally decrease classification accuracies.

A filter similar to Box-Plot is used to find the outlying classes in this distribution. The mean and standard deviation of the y-axis are calculated and a box is drawn on the distribution with the center ($mean_y$) and boundaries $\tilde{(}0{,}2 \times \sigma_y)$, as shown in Fig.1. The classes that fall into the area within the

9781-4244-3428-2/09/$25.00 ©2009 IEEE

boundaries are used as the dataset classes; the ones outside these are considered as outliers and cleared. This filter gives us 21 classes out of the dataset, each containing approximately equal number of instances. We use 1623 documents that belong to the filtered classes as our input dataset. Our filtered input dataset contains 8120 words.

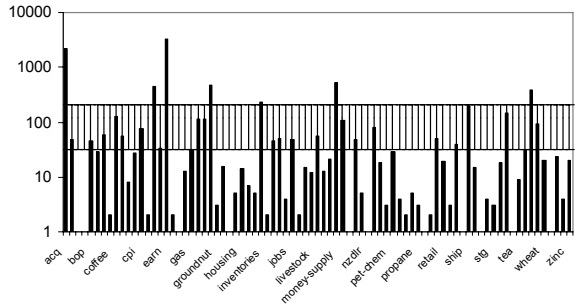

Figure 1: Distribution of documents among classes, in logarithmic scale.

Looking from the bag-of-words view, we have a term-document matrix with 8120 rows and 1623 columns. Because any of the documents will contain only a tiny subset of our terms, our matrix is very sparse. The cells of the matrix contain tf-idf values of terms calculated by (1), (2) and (3), where n_i is the number of occurrences of the considered term in document d_j, $|D|$ is the total number of documents, $|\{d_j:t_i \in d_j\}|$ is the number of documents where term t_i appears.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (1)$$

$$idf_i = \log \frac{|D|}{\left|\{d_j : t_i \in d_j\}\right|} \quad (2)$$

$$tfidf_{i,j} = tf_{i,j} \times idf_i \quad (3)$$

Our matrix leads us to an 8120 dimensional vector space, which is hard to deal with because running classification algorithms on such a multidimensional space is very time and memory consuming. Number of documents per class after pre-processing step is given in Fig.2.

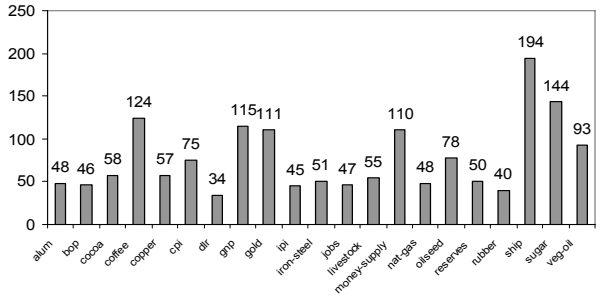

Figure 2: Distribution of documents among filtered classes after pre-processing step.

3. Attribute Extraction

Attribute extraction algorithms simplify the amount of resource required to describe a large set of data. The aim of attribute extraction is to combine terms to form a new description for the data with sufficient accuracy. In this section, commonly used extraction algorithms -principal component analysis, PCA and latent semantic analysis, LSA- and the proposed method are introduced. Our method projects attributes into a new hyperspace with dimensions equal to number of classes.

3.1. Related Research

3.1.1. Principal Component Analysis

PCA transforms correlate variables into a smaller number of correlated variables- principal components. Invented by Pearson in 1901, it is generally used for exploratory data analysis [6].

PCA is used for attribute extraction by retaining the characteristics of the dataset that contribute most to its variance, by keeping lower order principals, which tend to have the most important aspects of data. This is accomplished by a projection into a new hyper plane using Eigen values and Eigen vectors.

PCA is a popular technique in pattern recognition, but its applications are not very common because it is not optimized for class separability [7]. It is widely used in image processing.

3.1.2. Latent Semantic Analysis

Patented in 1988, LSA is a technique that analyzes relationships between a document set and the terms that they contain by producing a set of concepts related to them [8]. As the name suggests, singular value decomposition breaks our matrix down into a set of smaller components.

LSA uses singular value decomposition (SVD) method to find the relationships between documents. Singular value decomposition breaks term-document matrix down into a set of smaller components. The algorithm alters one of these components (reduces the number of dimensions), and then recombines them into a matrix of the same shape as the original, so we can again use it as a lookup grid. The matrix we get back is an approximation of the term-document matrix we provided as input, and looks much different from the original.

LSI, based on LSA, is mostly used for web page retrieval and document clustering purposes. It is also used for document classification via voting, or information filtering. Many algorithms utilize LSI in order to improve performance by working in a less complex hyperspace.

3.2. Proposed Attribute Extraction Method

Our attribute extraction method neither uses Eigen values, Eigen vectors, nor singular value decomposition. In this method, weights of terms and their probabilistic distribution over the classes are taken into account. We project term probabilities to classes, and sum up those probabilities to get the impact of each term to each class [9].

Assume we have I terms, J documents and K classes. Let $n_{i,j}$ be the number of occurrences of term t_i in document d_j and

N_i be the total number of documents that contain t_i. We calculate $nc_{i,k}$, the total number of occurrences of a term t_i in class c_k with (4). We calculate $w_{i,k}$, the weight of term t_i (in a document d_j) that affects class ck with (5).

$$nc_{i,k} = \sum_j n_{i,j}, \qquad d_j \in c_k \tag{4}$$

$$w_{i,k} = \log\left(nc_{i,k} + 1\right) \times \log\left(\frac{N}{N_i}\right) \tag{5}$$

We calculate the total effect of terms in a document over a class c_k with (6). At the end, I terms are projected onto number of classes; we have K new terms in hand for a document. Repeating this procedure for all documents gives us a reduced matrix with J rows (one row per document) and K columns (number of extracted features equals the number of classes). Finally, we normalize new terms by using (7).

$$Y_{j,k} = \sum_i w_{i,k}, \qquad w_i \in d_j \tag{6}$$

$$newTerm_k = \frac{Y_k}{\sum_k Y_k} \tag{7}$$

We give a brief example to explain the method comprehensibly. Assume we have 8 terms, 6 documents and 2 classes given in Fig.3. The corresponding term-document matrix is in Table 1. The cells of Table 1 correspond to term weights calculated with (4).

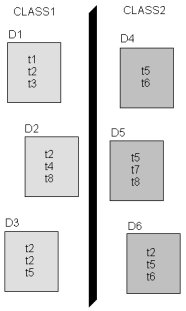

Figure 3: Sample dataset to explain the proposed extraction method comprehensibly.

Table 1: Term-Document matrix of the sample dataset. We have 8 terms in 6 documents.

	D1	D2	D3	D4	D5	D6
t1	1	0	0	0	0	0
t2	1	1	2	0	0	1
t3	1	0	0	0	0	0
t4	0	1	0	0	0	0
t5	0	0	1	1	1	1
t6	0	0	0	1	0	1
t7	0	0	0	0	1	0
t8	0	1	0	0	1	0

Table 2: Term weights over classes for the sample dataset.

	C1	C2
t1	0.234	0
t2	0.123	0.053
t3	0.234	0
t4	0.234	0
t5	0.053	0.106
t6	0	0.228
t7	0	0.234
t8	0.144	0.144

We calculate term weights over classes with (5). This produces I by K matrix, given in Table 2. After applying (6) and (7), we have the reduced J by K matrix with the extracted features of the processed documents. Result matrix is given in Table 3. This matrix is visualized in Fig.4. We can also read the extracted values as the membership probabilities of documents to classes, as seen on Fig.4. For example, the content of D4 tells us that it is 84% Class2 and 16% Class1. Thus, the extraction method we propose may also be used as a stand-alone classifier. When we feed a new document, the system decides which class this document belongs to by calculating (6) and (7) with the new document's content.

Table 3: Extracted attributes of documents over classes for the sample dataset. Each document has attributes as much as classes.

	C1	C2
D1	0.918	0.082
D2	0.718	0.282
D3	0.585	0.415
D4	0.137	0.863
D5	0.289	0.711
D6	0.313	0.687

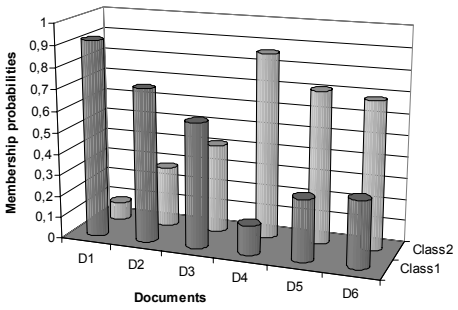

Figure 4: Visualization of Table 3. Extracted values can be taken as the membership probabilities of documents to the classes.

4. Experimental Results

We prepare our evaluation dataset and test our attribute extraction method's performance using 5 different classifiers. We use 10-fold cross-validation for testing.

We use Naïve Bayes as a simple probabilistic classifier, which is based on applying Bayes' theorem with strong independence assumptions [10]. We choose J48 Tree, an

implementation of Quinlan's [11] C4.5 decision tree algorithm for a basic tree based classifier, and a random forest with 10 trees to construct a collection of decision trees with controlled variations [12]. We choose 1-nearest neighbour and 10-nearest neighbour algorithms to test instance-based classifiers.

We evaluate the classifiers' performance with their classification accuracy which is calculated by (8) and F-Measure by using (9) .The classification performance results with and without utilizing the proposed method are given in Table 4. We can see that both classification accuracies and F-measures increase. We can see that our attribute method increases the classification accuracy by 22.3% when using Naïve Bayes, 7.6% using J48, 34.6% using 1-nearest neighbour, 35% using 10-nearest neighbour, and 38.4% using a random forest. In Fig.5 we visualize the classification accuracies for easy comparing. Another outcome of our method is that it decreases classification times on both classifiers dramatically.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{8}$$

$$F = \frac{2 \times TP}{\left(2 \times TP + FP + FN\right)} \tag{9}$$

Table 4: Results for the classifiers evaluated on filtered Reuters dataset with and without using the proposed attribute extraction method.

Classification Algorithm	Without Attribute Extraction		With Attribute Extraction	
	Accuracy	F-M.	Accuracy	F-M.
Naïve Bayes	70.8%	0.715	93.1%	0.931
J48 Tree (C4.5)	83.5%	0.834	91.3%	0.912
1-Nearest Neigh.	61.9%	0.633	96.5%	0.965
10-Nearest Neigh.	61.9%	0.633	96.9%	0.969
Random Forest (with 10 trees)	54.5%	0.536	92.9%	0.929

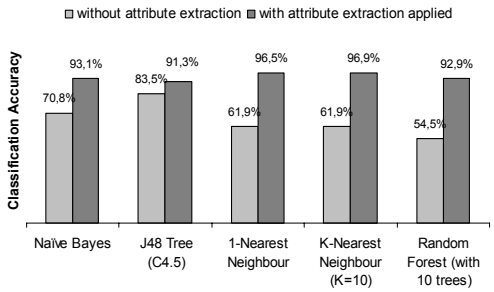

Figure 5: Comparison of classification accuracies with and without using the proposed attribute extraction method.

5. Conclusions and Future Work

We introduce a new method for attribute extraction and its application on text classification. We filter classes from Reuters-21758 dataset using a box-plot with boundaries $mean_y \tilde{+}(0,2 \text{ x. } \sigma_y)$ and use the filtered news bodies as the documents of our dataset. Selected attributes for classification are the textual content of the news bodies, in other words, preprocessed terms according to vector-space model. Using the offered method, high dimension of attributes - terms extracted from the documents - are projected onto a new hyper plane having dimensions equal to the number of classes. Results show that processing times of classification algorithms dramatically decrease with our attribute extraction method. This is achieved by the fall of the number of attributes given to classifiers. The most important outcome is the improvement on the accuracies of classification algorithms. Without using our method, classification algorithms score 66.52% average accuracy. Our method boosts average accuracy to 94.14%, in other words, improves the classification accuracy by 27.62% on average.

We scheduled to compare our attribute extraction method with other extraction algorithms like LSA and PCA or with attribute selection methods such as chi-square or information gain. As we see that our method can also act as a classifier, we program to test our method as a classifier with other classification algorithms. Furthermore, we have in view to run time-based tests to quantify the improvement on process running times.

6. References

[1] Joachims, T., "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization", *In Proc. 14th Int.Conf on Machine Learning,*1997, pp.143-151.

[2] Yiming, Y., Pedersen, J.O., "A comparative study on feature selection in text categorization", *In Proc. 14th Int.Conf on Machine Learning,*1997, pp.412-420.

[3] Zhu, J., Wang, H. and Zhang, X., "Discrimination-based feature selection for multinomial naïve bayes text classification", *In Y.Matsumoto, et.al., ed. LNAI 4285, Springer-Verlag Berlin, Heidelberg*, pp.194-156, 2006.

[4] Jensen, R., Shen, Q., "Computational intelligence and feature selection rough and fuzzy approaches." *Wiley-IEEE Press*, 2008.

[5] Porter, M.F., "An algorithm for suffix stripping", *Program*, 14(3), pp.130-137., 1980.

[6] Pearson, K., "On lines and planes of closest fit systems of points in space", *Philosophical Magazine*, 2(6), pp.559-572, 1901.

[7] Fukunaga, K., "Introduction to statistical pattern recognition", *Academic Press*, 1990.

[8] Landauer, T.K., Dumais, S.T., "Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge." *Psychological Review*, vol:104, no:2, pp. 211-240, 1997.

[9] Kaban, Z., Diri, B., "Recognizing type and author in Turkish documents using artificial immune systems*", In Proc. 16th Signal Processing and its Applications Congress*, April 2008.

[10] McCallum, A., Nigam, K., "A comparison of event models for naïve bayes text classification." *In Proc. AAAI/ICML-98 Workshop on Learning for Text Categorization*, July 1998.

[11] Quinlan, J.R., "Programs for machine learning", *Morgan Kaufmann Publishers*, 1993.

[12] Breiman, L., "Random Forests", *Machine Learning*, 45(1), pp.5-32, 2001.