A Methodology Combining Cosine Similarity with Classifier for Text Classification

Kwangil Park, June Seok Hong & Wooju Kim

To cite this article: Kwangil Park, June Seok Hong & Wooju Kim (2020) A Methodology Combining Cosine Similarity with Classifier for Text Classification, Applied Artificial Intelligence, 34:5, 396-411, DOI: 10.1080/08839514.2020.1723868

To link to this article: https://doi.org/10.1080/08839514.2020.1723868

Published online: 08 Feb 2020.

APPLIED ARTIFICIAL INTELLIGENCE
2020, VOL. 34, NO. 5, 396–411
https://doi.org/10.1080/08839514.2020.1723868

A Methodology Combining Cosine Similarity with Classifier for Text Classification

Kwangil Park[a], June Seok Hong[b], and Wooju Kim[a]

[a]Department of Industrial Engineering, Yonsei University, Seoul, Republic of Korea; [b]Department of Management Information Systems, Kyonggi University, Gyeonggi-do, Republic of Korea

ABSTRACT
Text Classification has received significant attention in recent years because of the proliferation of digital documents and is widely used in various applications such as filtering and recommendation. Consequently, many approaches, including those based on statistical theory, machine learning, and classifier performance improvement, have been proposed for improving text classification performance. Among these approaches, centroid-based classifier, multinomial naïve bayesian (MNB), support vector machines (SVM), convolutional neural network (CNN) are commonly used. In this paper, we introduce a cosine similarity-based methodology for improving performance. The methodology combines cosine similarity (between a test document and fixed categories) with conventional classifiers such as MNB, SVM, and CNN to improve the accuracy of the classifiers, and then we call the conventional classifiers with cosine similarity as enhanced classifiers. We applied the enhanced classifiers to famous datasets – 20NG, R8, R52, Cade12, and WebKB – and evaluated the performance of the enhanced classifiers in terms of the confusion matrix's accuracy; we obtained outstanding results in that the enhanced classifiers show significant increases in accuracy. Moreover, through experiments, we identified which of two considered knowledge representation techniques (word count and term frequency-inverse document frequency (TFIDF)) is more suitable in terms of classifier performance.

Introduction

Text classification involves assigning text documents to predefined categories, and in recent years, it has received significant attention because of the proliferation of digital documents. Such proliferation has led to increased demand for improved classification performance in applications such as filtering and recommendation, which in turn has led to much research on improving text classification performance (Chen et al. 2009). In particular, approaches to improve performance in text classification based on statistical theory or machine learning have become commonplace (Shang et al. 2007), and such approaches utilized data mining algorithms such as C4.5, K-means,

CONTACT Wooju Kim wkim@yonsei.ac.kr Department of Industrial Engineering, Yonsei University, Seoul 120749, Republic of Korea

SVM, Apriori, EM, PageRank, AdaBoost, KNN, Naïve Bayes, and CART (Oliverio and Poli-Neto 2017). Among these algorithms, Naïve Bayes is popular because it shows good computational efficiency and relatively good predictive performance (Chen et al. 2009); moreover, it is known to be superior to KNN and C4.5 according to their research (Ardhapure et al. 2016). Equally, SVM has been shown to perform consistently well across various datasets (Zhang et al. 2017), and centroid-based classifier is one of the most popular classifiers in the text classification domain (Liu et al. 2017). Centroid-based classifier utilizes cosine similarity to classification and has been shown to perform better than other algorithms such as Decision Tree, KNN, and Naïve Bayes with respect to text classification according to their research (Nguyen, Chang, and Hui 2013).

Approaches such as centroid-based classifier (CBC), multinomial naïve bayesian (MNB), support vector machines (SVM), and convolutional neural network (CNN) are widely used for text classification, and research into methodologies for improving text classification performance is in full swing in that domain. In this paper, a cosine similarity-based methodology is introduced. The proposed methodology combines cosine similarity (between a test document and predefined categories) with conventional classifiers such as MNB, SVM, and CNN to produce improved versions of the classifiers, which can then be used for text classification performance improvement, and conventional classifiers (MNB, SVM, and CNN) provide estimated values for text classification, cosine similarity is combined with the estimated values, and then we call the conventional classifiers with cosine similarity as enhanced classifiers.

The enhanced classifiers and conventional classifiers are evaluated on five datasets (20NG, R8, R52, Cade12, and WebKB) by using confusion (or misclassification) matrix's accuracy (Zeng 2019), which is representative performance evaluation method. Consequently, we obtain improved results in that the enhanced classifiers demonstrate statistically significant increases in accuracy. Average accuracy per dataset – [MNB: 20NG (2.28%), R8 (0.96%), R52 (1.01%), Cade12 (3.28%), WebKB (2.74%), SVM: 20NG (0.94%), R8 (0.39%), R52 (1.29%), Cade12 (2.73%), WebKB(0.18%), CNN: 20NG (3.79%), R8 (1.05%), R52 (2.71%), WebKB (1.16%)]. Moreover, through experiments, which of two considered knowledge representation techniques (word count and TFIDF) is the more suitable in terms of classifier performance is identified. Because dataset representation can affect experimental outcome, all datasets in this paper are represented in two different ways: by word count and by TFIDF. Two conventional classifiers are used in the experiments: MNB and SVM. Since each conventional classifier can be represented in two different ways, four different types of conventional classifier are considered: word count-based MNB, word count-based SVM, TFIDF-based MNB, and TFIDF-based SVM. The proposed methodology is applied to each of the four classifier types, and the enhanced classifiers are then applied to each of the considered datasets.

In the world, there are many architectures of deep learning technique. Our CNN models are actually constructed with very little modification of Kim's architecture (Kim 2014). We also use pre-trained word embedding vectors Google News, which is available on (https://code.google.com/p/word2vec).

Related Work

Efforts to improve the performance of conventional classifiers such as MNB and SVM are currently ongoing. Diab and El Hindi (2017) designed a fine-tuning methodology for improving performance for MNB. The methodology utilizes three metaheuristic approaches – genetic algorithms, simulated annealing, and differential evolution – to transform a probability estimation problem into an optimization problem with the aim of finding a more accurate probability value for each word of a text document. Jain and Mishra (2016) proposed a methodology combining the results of two classifiers for text classification, for example, MNB and a modified maximum entropy classifier which was modified version of conventional maximum entropy classifier proposed by the authors, and three measures were used for the combination of the results of two classifiers – average, max, and harmonic. Isa et al. (2008) proposed a hybrid model integrating two classifiers (MNB and SVM), but they did not utilize MNB as a classifier; rather, they utilized MNB as a vectorizer in the pre-processing process to improve the performance of SVM, and then, they named MNB as Bayes Vectorizer because it generates vectors of text documents for text classification based on Bayes' theorem. Each training document was vectorized by a trained MNB through the calculation of posterior probability by Bayes' theorem. SVM then used for training phase the vectors supplied by the Bayes Vectorizer. Lee et al. (2012) proposed a new text classification framework called Euclidean-SVM, which uses SVM in the training phase and Euclidean distance in the classification or test phase. In the test phase, the average of Euclidean distance (between the vectors of a test document and each set of support vectors of different categories) was used, after SVs have been identified in the training phase.

Cosine similarity is widely used in the text classification domain because of its computational efficiency and good performance. In particular, classifiers utilizing cosine similarity already exist, namely, centroid-based classifier (Liu et al. 2017; Nguyen, Chang, and Hui 2011). Representative centroid-based classifier includes Arithmetical Average Centroid (AAC), Cumuli Geometric Centroid (CGC), and Class Feature Centroid (CFC), where the method for generating a prototype vector of a class for CBC (i.e. the initialization methods) is known as centroid. Regarding centroid, AAC uses the arithmetical average of all words of each class; CGC uses the summation of all words of each class; CFC uses the inner-class term index and the inter-class term index (Guan, Zhou, and Guo 2009). (Liu et al. 2017) designed a new CBC model named gravitation model, which concentrates on the adjustment of a classification hyperplane.

Methodology

Cosine similarity is a measure of the degree of similarity between two vectors and is the most popular in the inner product family (Kocher and Savoy 2017). Within the text classification domain, it can be used to indicate the degree of similarity between two documents. It takes values between 0 and 1, where a value of 0 indicates that there is no similarity between the documents and a value of 1 indicates that the documents are identical. Thus, for two documents, say “*doc1*” and “*doc2,*” the degree of similarity between them can be expressed as follows:

$$Sim(doc1, doc2) = \frac{doc1 \cdot doc2}{||doc1||\ ||doc2||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

where A_i and B_i represent the components of vectors *doc1* and *doc2*, respectively.

Many classifiers based on MNB, SVM, and CNN provide the estimated values of categories for text classification. In other words, text classification is essentially the selection of the category having the largest estimated value among all estimated values derived from a classifier. During the final step (that is, before classifying particular documents), cosine similarity is combined with estimated values provided by conventional classifiers such as MNB, SVM, and CNN. And in doing so, it improves the performance of the classifiers. Classifier performance is improved by combining the similarity between a test document and a category with the estimated value pertaining to the category. To obtain cosine similarity between a test document and each category, we use the centroid technique of CGC, but the proposed methodology has a problem in the case of cosine similarity equals zero, since the logarithm of zero is infinity. To solve this problem, a small positive value (1×10^{-20}) is assigned to cosine similarity in such a case. In Figure 1, the module calculating cosine similarity represents cosine similarity calculator.

The methodology of enhanced classifier:

$$C_{predicted}(d) = argmax_{c_j}\left[ln\left(EstimatedValue_{c_j}\right) + ln\left(Sim\left(d, c_j\right)\right)\right] \tag{2}$$

where d is a test document, c_j is the jth category among all possible categories, $EstimatedValue_{c_j}$ is the estimated value of c_j derived from a conventional classifier, and $Sim\left(d, c_j\right)$ is cosine similarity between d and all documents belonging to c_j.

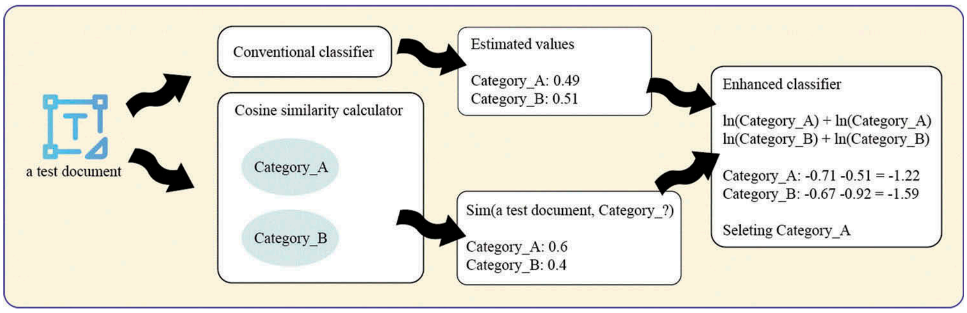
a test document
Conventional classifier
Cosine similarity calculator
Category_A
Category_B
Estimated values
Category_A: 0.49
Category_B: 0.51
Sim(a test document, Category_?)
Category_A: 0.6
Category_B: 0.4
Enhanced classifier
ln(Category_A) + ln(Category_A)
ln(Category_B) + ln(Category_B)
Category_A: -0.71 -0.51 = -1.22
Category_B: -0.67 -0.92 = -1.59
Seleting Category_A

Figure 1. Our process of text classification.

From now on, we discuss the application of the proposed methodology to the conventional classifiers MNB and SVM. Regarding MNB, equations (3~5) are the algorithms of conventional MNB (Diab and El Hindi 2017), and equation (6) is the algorithms of the proposed methodology.

$$C_{predicted}(d) = argmax_{c_j}[ln(P(c_j)) + \sum_{k=1}^{n} f_k \; ln(P(w_k|c_j))] \tag{3}$$

where d is a test document, n is the number of words in d, c_j is the jth category among all possible categories, w_k is the kth word in d, and f_k is the frequency count of w_k.

$$P(w_k|c_j) = \frac{N_{c_{jk}} + 1}{N_{c_j} + N_{all}} \tag{4}$$

where $N_{c_{jk}}$ is the number of w_k in c_j, N_{all} is the number of all unique words in training documents, and N_{c_j} is the number of all words in c_j.

$$P(c_j) = \frac{N_k}{N} \tag{5}$$

where N_k is the number of all documents in c_j, N is the number of all documents in training documents.

The equation of enhanced MNB:

$$C_{predicted}(d) = argmax_{c_j}[ln(P(c_j)) + \sum_{k=1}^{n} f_k \; ln(P(w_k|c_j)) + ln(Sim(d, \; c_j))] \tag{6}$$

Regarding SVM, it is a quadratic programming problem that can be solved by Lagrange multipliers (Isa et al. 2008). It contains kernels such as Linear, Sigmoid, Polynomial, RBF, and Exponential RBF, which allow SVM to perform a separation, even with very complex boundaries. Additionally, slack variable C adjusts the balance between the size of slack variables and the margin; namely, the C defines the number of permissible errors (i.e. penalty). In the literature, Lee et al. (2012) state that "if the value of C is small, the number of training errors will increase due to underfitting. On the other hand, the large value of C will lead to overfitting where a high penalty for non-separable points occurs." Finally, LIBSVM is generally used in the text classification domain and provides an estimated value for each category. In Chang and Lin (2011), an estimated value is referred to as a probability estimate.

The equation of enhanced SVM:

$$C_{predicted}(d) = argmax_{c_j}\left[ln(ProbabilityEstimate_{c_j}) + ln\left(Sim\left(d,\ c_j\right)\right)\right] \tag{7}$$

Regarding CNN, we use word2vec vectors from Google News that has 300-dimentionality, and details in (Kim 2014). Our CNN architecture is very similar to him for all datasets, with starting static channels, next convolutional layer with multiple filters, and then max-pooling layer, finally fully connected layer, but parameter details are a little bit different. In convolutional layer, we set up relu activation, sequence padding, 100 filters, 5 filter windows, and 1 stride, and then max-pooling is conducted. In fully connected layer, relu activation, l2 regularizer, dropout rate of 0.5, categorical cross-entropy loss, stochastic gradient descent optimizer, softmax output are set. Finally, we use the early stopping function in all experiments.

The equation of enhanced CNN:

$$C_{predicted}(d) = argmax_{c_j}\left[ln(Softmax_{c_j}) + ln\left(Sim\left(d,\ c_j\right)\right)\right] \tag{8}$$

Setting of Experiments

To perform experiments, both the conventional classifiers and the enhanced classifiers were developed in JAVA using an integrated development environment known as IntelliJ IDEA except for CNN. In the case of SVM, LIBSVM was used (cf. https://www.csie.ntu.edu.tw/~cjlin/libsvm): a linear kernel and probability estimates were used among the functions provided by LIBSVM. All classifiers were constructed from datasets represented by word count or TFIDF because these knowledge representation techniques are known to affect classifier performance. In the case of CNN, it was developed in KERAS on python. In total, 15 classifiers were constructed:

Conventional classifiers constructed according to datasets represented by word count or TFIDF:

- MNBWC (word count-based MNB)
- MNBTFIDF (TFIDF-based MNB)
- SVMWC (word count-based SVM)
- SVMTFIDF (TFIDF-based SVM)
- CNN (word2vec)

To apply the proposed methodology to conventional classifiers, cosine similarity between a test document and categories is obtained according to datasets represented by word count or TFIDF:

- CSWC (word count-based cosine similarity)
- CSTFIDF (TFIDF-based cosine similarity)

The enhanced classifiers constructed by combining cosine similarity to five conventional classifiers:

- MNBWC_CSWC (MNBWC combined with CSWC)
- MNBWC_CSTFIDF (MNBWC combined with CSTFIDF)
- MNBTFIDF_CSWC (MNBTFIDF combined with CSWC)
- MNBTFIDF_CSTFIDF (MNBTFIDF combined with CSTFIDF)
- SVMWC_CSWC (SVMWC combined with CSWC)
- SVMWC_CSTFIDF (SVMWC combined with CSTFIDF)
- SVMTFIDF_CSWC (SVMTFIDF combined with CSWC)
- SVMTFIDF_CSTFIDF (SVMTFIDF combined with CSTFIDF)
- CNN_CSWC (CNN combined with CSWC)
- CNN_CSTFIDF (CNN combined with CSTFIDF)

For evaluating our experimental results, we note the problem that all classifiers may not classify a test document correctly (Asim et al. 2018). For solving this problem, we evaluate the performance of all classifiers using accuracy in the confusion matrix. Accuracy is expressed as a value between 0 and 1 or the percentage of a value: a higher value means a more accurate result (Kocher and Savoy 2017).

Datasets and Document Representation

Vectorization is the problem of how to convert words in documents such as training documents and test documents into numbers. Typically, word count and TFIDF – knowledge representation techniques – are widely used for vectorization, each of which can affect classifier performance. Therefore, all documents in the datasets are separately vectorized by word count and by TFIDF for evaluating the performance of the constructed classifiers.

Word count means the number of occurrences of each word (i.e. absolute frequency), and TFIDF is generally composed of multiplication of TF and IDF, and it is a term weighting approach. In this paper, TF follows (Cardoso-Cachopo 2007) and IDF follows (Yang et al. 2000), and TFIDF is defined as follows:

$$f_{(w,d)} = \frac{f_{(t,d)}}{max\left(f_{(t',d)}\right)} \times log_{10}\frac{N}{n} \tag{8}$$

where d is a document, t and t' are words in d, $f_{(t,d)}$ is the word count of t, $max\left(f_{(t',d)}\right)$ is the most frequently occurring word in d, N is the total number of documents in training documents, n is the number of documents existing in t in training documents, and $f_{(w,d)}$ is a term weighting value of t.

Datasets were downloaded from the homepage (cf. http://ana.cachopo.org/datasets-for-single-label-text-categorization), which provides users interested in classification with suitable resources. Among the resources, the stemmed versions of datasets were downloaded and used. The downloaded datasets, a few datasets contain blank documents: such documents were removed for research purposes. A summary of the datasets used in the research is provided in Table 1, and detailed descriptions of the datasets are in (Cardoso-Cachopo 2007).

Performance Comparisons

MNBWC

In Table 2, the accuracies of the conventional classifier MNBWC (i.e. word count-based MNB) and the enhanced classifiers MNBWC_CSWC and MNBWC_CSTFIDF are presented. MNBWC_CSWC and MNBWC_CSTFIDF show a slightly better performance than MNBWC. However, their *improvement rate of accuracy* (for convenience, we will refer to it as IRA) does not exceed 0.5%; thus, they appear similar to MNBWC. Their IRAs are distributed between 0.0724% and 0.4539%, and the highest accuracy of each dataset is bolded in tables of this section.

Table 1. Datasets used in this article.

Datasets	Categories	# Training Docs	# Test Docs	# Categories
20NG	alt.atheism, comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.ibm.mac.hardware, comp.windows.x, misc.forsale, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, soc.religion.christian, talk.politics.guns, talk.politics.mideast, talk.politics.misc, talk.religion.misc	11,293	7,527	20
R8	acq, crude, earn, grain, interest, money-fx, ship, trade	5,485	2,189	8
R52	acq, alum, bop, carcass, cocoa, coffee, copper, cotton, cpi, cpu, crude, dlr, earn, fuel, gas, gnp, gold, grain, heat, housing, income, install-debt, interest, ipi, iron-steel, jet, jobs, lead, lei, livestock, lumber, meal-feed, money-fx, money-supply, nat-gas, nickel, orange, pet-chem, platinum, potato, reserves, retail, rubber, ship, strategic-metal, sugar, tea, tin, trade, veg-oil, wpi, zinc	6,532	2,568	52
Cade12	servicos, sociedade, lazer, informatica, saude, educacao, internet, cultura, esportes, noticias, ciencias, compras-online	27,322	13,661	12
WebKB	Course, faculty, project, student	2,785	1,383	4

Table 2. Accuracy comparison and the values in parentheses are the percentage of accuracy difference of the two classifiers. For example, (0.811479–0.810416) x100 = 0.1063%, this rule is retained in this section (i.e. IRA).

Dataset	MNBWC	MNBWC_CSWC	MNBWC_CSTFIDF
20NG	0.810416	0.811479 (0.1063%)	**0.813206 (0.2790%)**
R8	0.960713	**0.962997 (0.2284%)**	0.962540 (0.1827%)
R52	0.869159	0.872664 (0.3505%)	**0.873053 (0.3894%)**
Cade12	0.572652	0.573530 (0.0878%)	**0.577191 (0.4539%)**
WebKB	0.840202	0.840926 (0.0724%)	**0.841649 (0.1447%)**

SVMWC

Experiments involving the conventional classifier SVMWC (i.e. word count-based SVM) were conducted in accordance with changes to the slack variable *C*. Six values were selected and compared: performance was optimal in the cases of $C = 0.05$ or $C = 0.1$. In Table 3, the accuracies of the conventional classifier SVMWC constructed from datasets represented by word count are presented.

In Tables 4 and 5, the accuracies of the enhanced classifiers SVMWC_CSWC and SVMWC_CSTFIDF, respectively, are presented. The accuracy of SVMWC_CSWC is higher than that of SVMWC, and SVMWC_CSWC's IRAs are distributed between −0.1196% and 5.7902% across all datasets. In the case of SVMWC_CSTFIDF, its IRAs are distributed between 0.1447% and 6.3685% across all datasets. In particular, SVMWC_CSWC exhibits a noticeable

Table 3. Accuracy comparison of SVMWC according to C-values.

Dataset	SVMWC					
C	0.005	0.05	0.1	0.5	1	10
20NG	0.748771	**0.755414**	0.748771	0.734423	0.732563	0.729241
R8	0.953860	**0.958885**	**0.958885**	0.956601	0.956601	0.956601
R52	0.899533	0.902259	**0.903037**	0.900701	0.899533	0.900701
Cade12	0.483859	0.520606	**0.529390**	0.528073	0.509553	0.471561
WebKB	0.895155	**0.895155**	0.888648	0.870571	0.862617	0.861171

Table 4. Accuracy comparison of SVMWC_CSWC according to C-values, and the values in parentheses are the percentage of accuracy difference of the two classifiers. For example, (0.747575–0.748771) x100 = −0.1196%, this rule is retained in this section.

Dataset	SVMWC_CSWC					
C	0.005	0.05	0.1	0.5	1	10
20NG	0.747575 (−0.1196%)	**0.757672 (0.2258%)**	0.751295 (0.2524%)	0.738010 (0.3587%)	0.735884 (0.3321%)	0.733493 (0.4252%)
R8	0.958429 (0.4569%)	0.961626 (0.2741%)	**0.964824 (0.5939%)**	0.963454 (0.6853%)	0.962997 (0.6396%)	0.962997 (0.6396%)
R52	0.908489 (0.8956%)	0.913162 (1.0903%)	0.915109 (1.2072%)	**0.917835 (1.7134%)**	0.915109 (1.5576%)	0.916277 (1.5576%)
Cade12	0.523827 (3.9968%)	0.550179 (2.9573%)	**0.555743 (2.6353%)**	0.554498 (2.6425%)	0.548715 (3.9162%)	0.529463 (5.7902%)
WebKB	0.894432 (−0.0723%)	**0.896602 (0.1447%)**	0.890094 (0.1446%)	0.872740 (0.2169%)	0.867679 (0.5062%)	0.865510 (0.4339%)

Table 5. Accuracy comparison of SVMWC_CSTFIDF according to C-values.

Dataset	SVMWC_CSTFIDF					
C	0.005	0.05	0.1	0.5	1	10
20NG	0.770559 (2.1788%)	0.778531 (2.3117%)	0.772818 (2.4047%)	0.761392 (2.6969%)	**0.783181 (5.0618%)**	0.760197 (3.0956%)
R8	0.961626 (0.7766%)	0.966195 (0.7310%)	**0.967108 (0.8223%)**	0.963454 (0.6853%)	0.964367 (0.7766%)	0.964367 (0.7766%)
R52	0.918614 (1.9081%)	0.924844 (2.2585%)	0.924065 (2.1028%)	0.926402 (2.5701%)	0.923676 (2.4143%)	**0.926791 (2.6090%)**
Cade12	0.532538 (4.8679%)	0.557353 (3.6747%)	**0.563941 (3.4551%)**	0.561452 (3.3379%)	0.555596 (4.6043%)	0.535246 (6.3685%)
WebKB	**0.898771 (0.3616%)**	0.896602 (0.1447%)	0.895879 (0.7231%)	0.877802 (0.7231%)	0.870571 (0.7954%)	0.870571 (0.9400%)

increase in accuracy in only Cade12, whereas SVMWC_CSTFIDF exhibits a noticeable increase in accuracy in 20NG, R52, and Cade12.

MNBTFIDF

In Table 6, the accuracies of the conventional classifier MNBTFIDF (i.e. TFIDF-based MNB) and the enhanced classifiers MNBTFIDF_CSWC and MNBTFIDF_CSTFIDF are presented. There are definite differences between MNBTFIDF and MNBTFIDF_CSWC, or MNBTFIDF and MNBTFIDF_CSTFIDF. MNBTFIDF_CSWC and MNBTFIDF_CSTFIDF show better performance than MNBTFIDF. The enhanced MNB's IRAs are distributed between 1.3629% and 7.8814%. However, the accuracy of MNBTFIDF is much lower than that of MNBWC across all datasets.

SVMTFIDF

Like SVMWC (i.e. word count-based SVM), when $C = 0.05$ or $C = 0.1$, SVMTFIDF (i.e. TFIDF-based SVM) performs optimally; however the accuracy of SVMTFIDF is similar to that of SVMWC or higher. In the case of 20NG, R8, R52, and Cade12, there are large differences in accuracies between SVMTFIDF and SVMWC, especially the difference in bolded accuracies with reference to Cade12 is that SVMTFIDF is 6.37% higher than SVMWC. In respect to WebKB, accuracies between SVMTFIDF and SVMWC are similar, and details on SVMTFIDF are in Table 7.

In Tables 8 and 9, the accuracies of the enhanced classifiers SVMTFIDF_CSWC and SVMTFIDF_CSTFIDF are presented. The accuracy of SVMTFIDF_

Table 6. Accuracy comparison.

Dataset	MNBTFIDF	MNBTFIDF_CSWC	MNBTFIDF_CSTFIDF
20NG	0.744121	0.771356 (2.7235%)	**0.804305 (6.0184%)**
R8	0.898127	0.912746 (1.4619%)	**0.917771 (1.9644%)**
R52	0.786994	0.800623 (1.3629%)	**0.806464 (1.9470%)**
Cade12	0.439538	0.499817 (6.0279%)	**0.505087 (6.5549%)**
WebKB	0.704266	0.733189 (2.8923%)	**0.783080 (7.8814%)**

Table 7. Accuracy comparison of SVMTFIDF according to C-values.

Dataset	SVMTFIDF					
C	0.005	0.05	0.1	0.5	1	10
20NG	0.784111	**0.826226**	0.823303	0.815863	0.815066	0.813206
R8	0.964824	**0.973961**	0.973504	0.970306	0.968936	0.969849
R52	0.905763	0.938084	**0.938863**	0.936916	0.934579	0.935748
Cade12	0.573238	**0.593075**	0.586341	0.558305	0.545860	0.501793
WebKB	0.845987	**0.890817**	0.888648	0.885033	0.884309	0.875633

Table 8. Accuracy comparison of SVMTFIDF_CSWC according to C-values.

Dataset	SVMTFIDF_CSWC					
C	0.005	0.05	0.1	0.5	1	10
20NG	0.784908	**0.826226**	0.825827	0.818520	0.816129	0.814534
	(0.0797%)	**(0.0000%)**	(0.2524%)	(0.2657%)	(0.1063%)	(0.1328%)
R8	0.966195	**0.974418**	**0.974418**	0.972590	0.970763	0.970763
	(0.1371%)	**(0.0457%)**	**(0.0914%)**	(0.2284%)	(0.1827%)	(0.0914%)
R52	0.920950	0.941199	**0.943536**	0.940031	0.937305	0.940031
	(1.5187%)	(0.3115%)	**(0.4673%)**	(0.3115%)	(0.2726%)	(0.4283%)
Cade12	0.573384	**0.598126**	0.593514	0.573677	0.562404	0.536857
	(0.0146%)	**(0.5051%)**	(0.7173%)	(1.5372%)	(1.6544%)	(3.5064%)
WebKB	0.844541	0.887202	**0.890817**	0.884309	0.879971	0.876356
	(−0.1446%)	(−0.3615%)	**(0.2169%)**	(−0.0724%)	(−0.4338%)	(0.0723%)

Table 9. Accuracy comparison of SVMTFIDF_CSTFIDF according to C-values.

Dataset	SVMTFIDF_CSTFIDF					
C	0.005	0.05	0.1	0.5	1	10
20NG	0.790488	0.826226	**0.826491**	0.820646	0.819981	0.819583
	(0.6377%)	(0.0000%)	**(0.3188%)**	(0.4783%)	(0.4915%)	(0.6377%)
R8	0.965738	0.972590	**0.974874**	0.972590	0.970763	0.971677
	(0.0914%)	(−0.1371%)	**(0.1370%)**	(0.2284%)	(0.1827%)	(0.1828%)
R52	0.928349	0.943536	**0.945093**	0.945093	0.941978	0.942757
	(2.2586%)	(0.5452%)	**(0.6230%)**	(0.8177%)	(0.7399%)	(0.7009%)
Cade12	0.577630	**0.598199**	0.594686	0.576459	0.565186	0.539492
	(0.4392%)	**(0.5124%)**	(0.8345%)	(1.8154%)	(1.9326%)	(3.7699%)
WebKB	0.844541	0.888648	**0.889371**	0.885033	0.884309	0.878525
	(−0.1446%)	(−0.2169%)	**(0.0723%)**	(0.0000%)	(0.0000%)	(0.2892%)

CSWC is higher than that of SVMTFIDF, and SVMTFIDF_CSWC's IRAs are distributed between −0.4338% and 3.5064% across all datasets. In the case of SVMTFIDF_CSTFIDF, its IRAs are distributed between −0.2169% and 3.7699% across all data sets. Across all datasets, SVMTFIDF_CSWC and SVMTFIDF_CSTFIDF generally perform better than SVMTFIDF.

Word Embedding

CNN experiments are conducted four datasets except for Cade12. In Table 10, the accuracies of CNN, CNN_CSWC, and CNN_CSTFIDF are presented. CNN_CSWC shows higher performance than that of CNN, and its IRAs are distributed between 0.6852% and 1.9860% across all datasets. In the case of

Table 10. Accuracy comparison of CNN.

Dataset	CNN	CNN_CSWC	CNN_CSTFIDF
20NG	0.656038	0.668261 (1.2223%)	**0.719543 (6.3505%)**
R8	0.946551	0.953403 (0.6852%)	**0.960713 (1.4162%)**
R52	0.871885	0.891745 (1.9860%)	**0.906153 (3.4268%)**
WebKB	0.864064	0.871294 (0.7230%)	**0.879971 (1.5907%)**

CNN_CSTFIDF, its IRAs are distributed between 1.4162% and 6.3505% across all data sets. Across all datasets, while CNN_CSWC and CNN_CSTFIDF generally perform better than CNN, CNN_CSTFIDF shows the best performance.

Ranking Analysis

To analyze the ranking, we use the accuracies of all classifiers to ranking each datum for 17 classifiers in total (two classifiers, CGCWC and CGCTFIDF were additionally experimented for this section), and for the ranking, all the accuracies derived from the experiments are utilized for MNB and CNN, and the highest accuracies (i.e. only the bolded) are utilized for SVM, and then the ranking of classifiers is sorted in ascending order of mean rank which is the mean of the ranks. In the ranking, No.1 means the highest performance and No.17 means the worst performance between the classifiers. Details are in Table 11.

In general, SVMTFIDF_CSTFIDF shows the highest performance and MNBTFIDF shows the worst performance. Especially, the TFIDF-based SVMs are in top 3, except for WebKB, regardless of whether it is constructed according to cosine similarity combined with word count or TFIDF. On the other hand, in the case of WebKB, the word count-based SVMs are in top 3, regardless of the combined type of cosine similarity.

For showing the performance difference of the conventional classifiers, a direct comparison analysis based on the knowledge representation technique is presented. Regarding the comparison between MNBWC and MNBTFIDF, the mean rank of MNBWC is 9, whereas that of MNBTFIDF is 15.4. In addition, MNBTFIDF does not perform better than MNBWC across all datasets; that is, MNB shows excellent performance when constructed from datasets represented by word count. Regarding the comparison between SVMWC and SVMTFIDF, the mean rank of SVMWC is 8.4, whereas that of SVMTFIDF is 3. In addition, SVMTFIDF performs overwhelmingly better than SVMWC across 20NG, R8, R52, and Cade12. In particular, the difference between them is only 1 in the case of WebKB; therefore, SVM shows better performance when constructed from datasets represented by TFIDF.

Table 11. Ranking analysis of all classifiers.

Classifier	Type	20NG	Rank	R8	Rank	R52	Rank	Cade12	Rank	WebKB	Rank	Mean rank
SVMTFIDF	CSTFIDF	0.826491	1	0.974874	1	0.945093	1	0.598199	1	0.889371	6	2
SVMTFIDF	CSWC	0.826226	2	0.974418	2	0.943536	2	0.598126	2	0.890817	4	2.4
SVMTFIDF	-	0.826226	2	0.973961	3	0.938863	3	0.593075	3	0.890817	4	3
SVMWC	CSTFIDF	0.783181	9	0.967108	4	0.926791	4	0.563941	7	0.898771	1	5
SVMWC	CSWC	0.757672	11	0.964824	5	0.917835	5	0.555743	8	0.896602	2	6.2
MNBWC	CSTFIDF	0.813206	4	0.96254	7	0.873053	10	0.577191	4	0.841649	10	7
CNN	CSTFIDF	0.719543	14	0.960713	8	0.906153	6	-	-	0.879971	7	7
MNBWC	CSWC	0.811479	5	0.962997	6	0.872664	11	0.57353	5	0.840926	11	7.6
SVMWC	-	0.755414	12	0.958885	10	0.903037	7	0.52939	10	0.895155	3	8.4
CNN	CSWC	0.668261	15	0.953403	11	0.891745	8	-	-	0.871294	8	8.4
MNBWC	-	0.810416	6	0.960713	8	0.869159	13	0.572652	6	0.840202	12	9
CNN	-	0.656038	16	0.946551	13	0.871885	12	-	-	0.864064	9	10
CGCTFIDF	-	0.791019	8	0.949292	12	0.881231	9	0.531659	9	0.82791	13	10.2
MNBTFIDF	CSTFIDF	0.804305	7	0.917771	15	0.806464	15	0.505087	11	0.78308	14	12.4
MNBTFIDF	CSWC	0.771356	10	0.912746	16	0.800623	16	0.499817	12	0.733189	16	14
CGCWC	-	0.541252	17	0.922339	14	0.843069	14	0.424347	14	0.746927	15	14.8
MNBTFIDF	-	0.744121	13	0.898127	17	0.786994	17	0.439538	13	0.704266	17	15.4

Discussion

In this paper, we have showed that the enhanced classifiers are superior to the conventional classifiers. Reasons why we are able to derive the results can be found in the sense of the proposed methodology. Cosine similarity in accordance with the Equation (1) indicates how similar two documents are, is always exists between the documents. In addition, as with the centroid techniques used by CBC mentioned in the related work section, the document (i.e. *doc1* or *doc2*) in Equation (1) can mean a particular category in the text classification domain. We applied these characteristics of cosine similarity and the centroid technique of CGC to the conventional classifiers.

Consequently, we obtained outstanding results in terms of accuracy, and then conducted the ranking analysis using them, have finally showed that the enhanced classifiers are superior to the conventional classifiers. One possible reason is that each estimated value of the conventional classifiers can be compensated by cosine similarity between the test document and each category; namely, MNB and SVM classify test documents with their own algorithms, but there are possibilities of misclassification, and the possibilities can be reduced by cosine similarity.

Moreover, through experiments, we also have identified which knowledge representation technique is more suitable in terms of classifier performance for the conventional classifiers. Regarding MNB, MNBWC is superior to MNBTFIDF. One possible reason is that MNB based on the conditional probability may be to regard the existence of the word itself as more important; namely, MNB can perform better when the dataset is expressed in word count without additional meaning (i.e. how important is the word) such as TFIDF. Regarding SVM, SVMTFIDF is superior to SVMWC. One possible reason is that SVM considers the margins between categories; namely, when the datasets are expressed in TFIDF, the meaning of TFIDF can be further emphasized by the margins of SVM.

Conclusions

This paper introduces the methodology for text classification that combines cosine similarity to improve the accuracy of conventional classifiers. The enhanced classifiers show outstanding performance in terms of accuracy compared to their conventional counterparts. In addition, this paper determines which of two knowledge representation techniques is most suitable for particular classifiers by means of various experiments. All datasets are represented by word count and by TFIDF. The following conclusions can be drawn from the experimental results:

- In case of cosine similarity, the use of TFIDF is superior to that of word count.
- In case of MNB, the use of word count is superior to that of TFIDF.
- In case of SVM, the use of TFIDF is superior to that of word count.
- In case of the use of the proposed methodology, the use of TFIDF is superior to that of word count.

Finally, applying the proposed methodology to a conventional classifier improves the performance of the classifier.

ORCID

Wooju Kim http://orcid.org/0000-0001-5828-178X

References

Ardhapure, O., G. Patil, D. Udani, and K. Jetha. 2016. Comparative study of classification algorithm for text based categorization. *International Journal of Research in Engineering and Technology* 5:217-220.

Asim, Y., A. R. Shahid, A. K. Malik, and B. Raza. 2018. Significance of machine learning algorithms in professional blogger’s classification. *Computers and Electrical Engineering* 65:461–73. doi:10.1016/j.compeleceng.2017.08.001.

Cardoso-Cachopo, A. 2007. Improving methods for single-label text categorization. PhD diss., IST – Instituto Superior Tecnico, Lisbon, Portugal.

Chang, C. C., and C. J. Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2:1–27. doi:10.1145/1961189.1961199.

Chen, J., H. Huang, S. Tian, and Y. Qu. 2009. Feature selection for text classification with Naïve Bayes. *Expert Systems with Applications* 36:5432–35. doi:10.1016/j.eswa.2008.06.054.

Diab, D. M., and K. M. El Hindi. 2017. Using differential evolution for fine tuning naïve Bayesian classifiers and its application for text classification. *Applied Soft Computing* 54:183–99. doi:10.1016/j.asoc.2016.12.043.

Guan, H., J. Zhou, and M. Guo. 2009. A class-feature-centroid classifier for text categorization. proceedings of the 18th ACM International Conference on World Wide Web, Madrid, Spain.

Isa, D., L. H. Lee, V. P. Kallimani, and R. RajKumar. 2008. Text document preprocessing with the Bayes formula for classification using the support vector machine. *IEEE Transactions on Knowledge and Data Engineering* 20:1264–72. doi:10.1109/TKDE.2008.76.

Jain, A., and R. D. Mishra. 2016. An effective approach for text classification. *International Journal of Research in Engineering and Technology* 5:24-30.

Kim, Y. 2014. Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar.

Kocher, M., and J. Savoy. 2017. Distance measures in author profiling. *Information Processing and Management* 53:1103–19. doi:10.1016/j.ipm.2017.04.004.

Lee, L. H., C. H. Wan, R. Rajkumar, and D. Isa. 2012. An enhanced support vector machine classification framework by using euclidean distance function for text document categorization. *Applied Intelligence* 37:80–99. doi:10.1007/s10489-011-0314-z.

Liu, C., W. Wang, G. Tu, Y. Xiang, S. Wang, and F. Lv. 2017. A new centroid-based classification model for text categorization. *Knowledge-Based Systems* 136:15–26. doi:10.1016/j.knosys.2017.08.020.

Nguyen, T. T., K. Chang, and S. C. Hui. 2011. Word cloud model for text categorization. Proceedings of the 11th IEEE International Conference on Data Mining, Vancouver, Canada.

Nguyen, T. T., K. Chang, and S. C. Hui. 2013. Supervised term weighting centroid-based classifiers for text categorization. *Knowledge and Information Systems* 35:61–85. doi:10.1007/s10115-012-0559-9.

Oliverio, V., and O. B. Poli-Neto. 2017. Case study: Classification algorithms comparison for the multi-label problem of chronic pelvic pain diagnosing. Proceedings of the 33th IEEE International Conference on Data Engineering (ICDE), San Diego, USA

Shang, W., H. Huang, H. Zhu, Y. Lin, Y. Qu, and Z. Wang. 2007. A novel feature selection algorithm for text categorization. *Expert Systems with Applications* 33:1–5. doi:10.1016/j.eswa.2006.04.001.

Yang, Y., T. Ault, T. Pierce, and C. W. Lattimer. 2000. Improving text categorization methods for event tracking. Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, Athens, Greece.

Zeng, G. 2019. On the confusion matrix in credit scoring and its analytical properties. *Communications in Statistics – Theory and Methods*. doi:10.1080/03610926.2019.1568485.

Zhang, C., C. Liu, X. Zhang, and G. Almpanidis. 2017. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications* 82:128–50. doi:10.1016/j.eswa.2017.04.003.