Improving K Nearest Neighbor into String Vector Version for Text Categorization

Taeho Jo

School of Game, Hongik University, 2639 Sejongro Sejong South Korea 30016
tjo018@hongik.ac.kr

Abstract— This research is concerned with the string vector based version of the KNN which is the approach to the text categorization. Traditionally, texts have been encoded into numerical vectors for using the traditional version of KNN, and encoding so leads to the three main problems: huge dimensionality, sparse distribution, and poor transparency. In order to solve the problems, this research propose that texts should be encoded into string vectors the similarity measure between string vectors is defined, and the KNN is modified into the version where string vector is given its input. The proposed KNN version is validated empirically by comparing it with the traditional KNN version on the three collections: NewsPage.com , Opiniopsis, and 20NewsGroups. The goal of this research is to improve the text categorization performance by solving them.

Keyword— String Vector, K Nearest Neighbor, Text Categorization

I. INTRODUCTION

THE text categorization is defined as the process of classifying a text into its category or categories among the predefined ones. Its preliminary task is to predefine a list of categories and allocate sample texts each of them. As the learning process, using the sample labeled texts, the classification capacity which is given as equations, parameters, or symbolic rules is constructed. As the generalization process, subsequent texts which are given separately from sample labeled ones are classified by the constructed classification capacity. In this research, we assume that the supervised learning algorithms will be used as the approaches, even if other kinds of approaches are available.

Let us consider the facts which provide the motivations for doing this research. Encoding texts into numerical vectors cause problems such as the huge dimensionality and the sparse distribution [1][2][3][5][11]. Previously, we proposed the table based classification algorithm which was called the table matching algorithm, its performance was unstable by impact of noisy examples [2][3]. The computation of the similarity between two tables as the essential task in the approach was very expensive [2][3]. Therefore, in this research, we try to find the solution to the problems by encoding texts into alternatives to both numerical vectors and tables.

What we propose in this research is to encode texts into string vectors and to modify the KNN as solutions to the above problems. Texts are encoded into string vectors as their structured forms, instead of numerical vectors. The semantic similarity measure between two string vectors is defined as the operation which corresponds to the cosine similarity between two numerical vectors. Using the similarity measure, we modify the KNN (K Nearest Neighbor) into the version where a string vector is given as an input vector. The modified version is applied as the approach to the task of classifying news articles and opinions automatically.

In this research, we will validate empirically the proposed approach to the text summarization as the better version than the traditional KNN version. We extract paragraphs from the collections of news articles: NewsPage.com, Opinopsis, and 20NewsGroups. The traditional KNN version and the proposed version are compared with each other. We observe the better results of the proposed KNN version in classifying news articles into their own topics. It potentially possible to require less dimension in encoding texts into string vectors, in addition.

This article is organized into the four sections. In Section II, we survey the relevant previous works. In Section III, we describe in detail what we propose in this research. In Section IV, we validate empirically what is proposed in this research. In Section V, we mention the remaining tasks for doing the further research.

II. PREVIOUS WORKS

Let us survey the previous cases of encoding texts into structured forms for using the machine learning algorithms to text mining tasks. The three main problems, huge dimensionality, sparse distribution, and poor transparency, have existed inherently in encoding them into numerical vectors. In previous works, various schemes of preprocessing texts have been proposed, in order to solve the problems. In this survey, we focus on the process of encoding texts into alternative structured forms to numerical vectors. In other words, this section is intended to explore previous works on solutions to the problems.

Let us mention the popularity of encoding texts into numerical vectors, and the proposal and the application of string kernels as the solution to the above problems. In 2002, Sebastiani presented the numerical vectors are the standard

Manuscript received December 27, 2017. This work is sponsored by 2017 Hongik University Research Fund, and a follow-up of the invited journal to the accepted & presented paper of the 19th International Conference on Advanced Communication Technology (ICACT2017).

Taeho Jo is with School of Game, Hongik University, Sejong, Republic of Korea (phone: +82-44-860-2125; e-mail: tjo018@hongik.ac.kr).

representations of texts in applying the machine learning algorithms to the text classifications [6]. In 2002, Lodhi et al. proposed the string kernel as a kernel function of raw texts in using the SVM (Support Vector Machine) to the text classification [7]. In 2004, Lesile et al. used the version of SVM which proposed by Lodhi et al. to the protein classification [8]. In 2004, Kate and Mooney used also the SVM version for classifying sentences by their meanings [9].

It was proposed that texts are encoded into tables instead of numerical vectors, as the solutions to the above problems. In 2008, Jo and Cho proposed the table matching algorithm as the approach to text classification [2]. In 2008, Jo applied also his proposed approach to the text clustering, as well as the text categorization [13]. In 2011, Jo described as the technique of automatic text classification in his patent document [11]. In 2015, Jo improved the table matching algorithm into its more stable version [12].

Previously, it was proposed that texts should be encoded into string vectors as other structured forms. In 2008, Jo modified the k means algorithm into the version which processes string vectors as the approach to the text clustering [13]. In 2010, Jo modified the two supervised learning algorithms, the KNN and the SVM, into the version as the improved approaches to the text classification [14]. In 2010, Jo proposed the unsupervised neural networks, called Neural Text Self Organizer, which receives the string vector as its input data [15]. In 2010, Jo applied the supervised neural networks, called Neural Text Categorizer, which gets a string vector as its input, as the approach to the text classification [16].

The above previous works proposed the string kernel as the kernel function of raw texts in the SVM, and tables and string vectors as representations of texts, in order to solve the problems. Because the string kernel takes very much computation time for computing their values, it was used for processing short strings or sentences rather than texts. In the previous works on encoding texts into tables, only table matching algorithm was proposed; there is no attempt to modify the machine algorithms into their table based version. In the previous works on encoding texts into string vectors, only frequency was considered for defining features of string vectors. In this research, based on [14], we consider the grammatical and posting relations between words and texts as well as the frequencies for defining the features of string vectors, and encode texts into string vectors in this research.

III. Proposed Approach

This section is concerned with encoding words into string vectors, modifying the KNN (K Nearest Neighbor) into the string vector based version and applying it to the text categorization, and consists of the three sections. In Section III-A, we deal with the process of encoding texts into string vectors. In Section III-B, we describe formally the similarity matrix and the semantic operation on string vectors. In Section III-C, we do the string vector based KNN version as the approach to the text categorization. Therefore, this section is intended to describe the proposed KNN version as the text categorization tool.

A. Text Encoding

This section is concerned with the process of encoding texts into string vectors. As shown in Figure 1, the three steps are involved in encoding texts. A single is given as the input and a string vector which consists of words is generated as the output. The features in each string vector are posting, statistic, and grammatical relationships between a text and a word. Therefore, this section is intended to describe in detail each step involved in encoding texts.

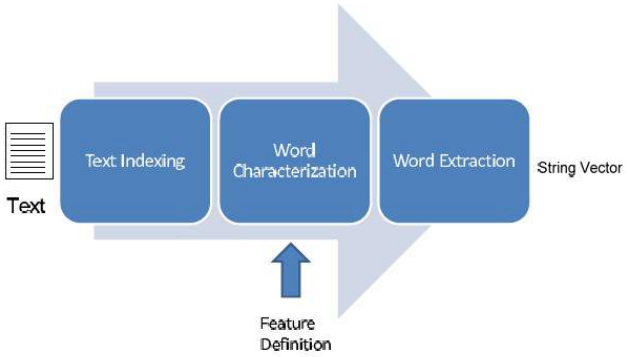

Fig. 1. The Process of Text Encoding.

The first step of encoding texts into string vectors is to index the corpus into a list of words. The texts in the corpus are concatenated into a single long string and it is tokenized into a list of tokens. Each token is transformed into its root form, using stemming rules. Among them, the stop words which are grammatical words such as propositions, conjunctions, and pronouns, irrelevant to text contents are removed for more efficiency. From the step, verbs, nouns, and adjectives are usually generated as the output.

We need to define the relationships between a word and a text as the features of string vectors, and mention the three types of them. The first type is statistical properties of words in a text such as the highest frequent word and the highest TF-IDF (Term Frequency-Inverse Term Frequency) weighted one. The grammatical properties of a word such as subjective noun, objective noun, and verb may be considered as another feature type. Posting properties of a word which indicates its position in the given text, such as the first word in the text, the last of in the text, and the first word in the last paragraph, may be regarded as a feature type. In this research, we define the ten features of string vectors as follows:

- Highest Frequent Word in the given Text
- Second Highest Frequent Word in the given Text
- Third Highest Frequent Word in the given Text
- Highest TF-IDF Weighted Word
- Second Highest TF-IDF Weighted Word
- Third Highest TF-IDF Weighted Word
- The Last Word in the Text
- The First Word in the last Paragraph
- The Last Word in the First Paragraph

Let us explain the process of encoding a text into a string, once the above features are defined. A text is indexed into a list of words, their frequencies, and their TF-IDF weights, and it is partitioned into a list of paragraphs. Corresponding to the above features, words are extracted as elements of the string vector. As the given text representation, the ten dimensional string vector which consists of the above feature values is

constructed. The similarity matrix is required for performing the operation on string vectors, and is described in Section III-B1.

Let us consider the differences between the word encoding and the text encoding. Elements of each string vector which represents a word are text identifiers, whereas those of one which represents a text are word. The process of encoding texts involves the link of each text to a list of words, where as that of doing words does the link of each word to a list of texts. For performing semantic similarity between string vectors, in text processing, the word similarity matrix is used as the basis, while in word processing, the text similarity matrix is used. The relations between words and texts are defined as features of strings in encoding texts and words.

B. String Vector

This section is concerned with the operation on string vectors and the basis for carrying out it. It consists of two subsections and assumes that a corpus is required for performing the operation. In Section III-B1, we describe the process of constructing the similarity matrix from a corpus. In Section III-B2, we define the string vector formally and characterize the operation mathematically. Therefore, this section is intended to describe the similarity matrix and the operation on string vectors.

Similarity Matrix

This subsection is concerned with the similarity matrix as the basis for performing the semantic operation on string vectors. Each row and column of the similarity matrix corresponds to a word in the corpus. The similarities of all possible pairs of words are given as normalized values between zero and one. The similarity matrix which we construct from the corpus is the N X N square matrix with symmetry elements and 1's diagonal elements. In this subsection, we will describe formally the definition and characterization of the similarity matrix.

Each entry of the similarity matrix indicates a similarity between two corresponding words. The two words, t_i, and t_j are viewed into two sets of texts which include them, T_i, and T_j. The similarity between the two words is computed by equation (1),

$$sim(t_i, t_j) = \frac{2|T_i \cap T_j|}{|T_i| + |T_j|} \tag{1}$$

where $|T_i|$ is the cardinality of the set, T_i. The similarity is always given as a normalized value between zero and one; if two words are exactly same to each other, the similarity becomes 1.0 as equation (2):

$$sim(t_i, t_i) = \frac{2|T_i \cap T_i|}{|T_i| + |T_j|} = 1.0 \tag{2}$$

and if two words have no shared texts, $T_i \cap T_j = \varnothing$, the similarity becomes 0.0 as equation (3):

$$sim(t_i, t_i) = \frac{2|T_i \cap T_i|}{|T_i| + |T_j|} = \frac{2 \times 0}{|T_i| + |T_j|} = 0.0 \tag{3}$$

The more advanced schemes of computing the similarity will be considered in next research.

From the text collection, we build N X N square matrix as follows:

$$\begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1N} \\ s_{21} & s_{22} & \cdots & s_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ s_{N1} & s_{N2} & \cdots & s_{NN} \end{bmatrix}$$

N individual words which are contained in the collection correspond to the rows and columns of the matrix. The entry, s_{ij} is computed by equation (1) as equation (4):

$$s_{ij} = sim(t_i, t_j) \tag{4}$$

The overestimation or underestimation by text lengths are prevented by the denominator in equation (1). To the number of words, N, it costs quadratic complexity, $O(N^2)$, to build the above matrix

Let us characterize the above similarity matrix, mathematically. Because each column and row corresponds to its same text in the diagonal positions of the matrix, the diagonal elements are always given 1.0 by equation (2). In the off-diagonal positions of the matrix, the values are always given as normalized ones between zero and one, because of $0 \le 2|T_i \cap T_i| \le |T_i| + |T_j|$ from equation (1). It is proved that the similarity matrix is symmetry, as equation (5):

$$\begin{aligned} s_{ij} &= sim(t_i, t_j) = \frac{2|T_i \cap T_j|}{|T_j| + |T_j|} = \frac{2|T_j \cap T_i|}{|T_j| + |T_i|} \\ &= sim(t_j, t_i) = s_{ji} \end{aligned} \tag{5}$$

Therefore, the matrix is characterized as the symmetry matrix which consists of the normalized values between zero and one.

The similarity matrix may be constructed automatically from a corpus. The N texts which are contained in the corpus are given as the input and each of them is indexed into a list of words. All possible pairs of words are generated and the similarities among them are computed by equation (1). By computing them, we construct the square matrix which consists of the similarities. Once making the similarity matrix, it will be used continually as the basis for performing the operation on string vectors.

String Vector and Semantic Similarity

This section is concerned with the string vectors and the operation on them. A string vector consists of strings as its elements, instead of numerical values. The operation on string vectors which we define in this subsection corresponds to the cosine similarity between numerical vectors. Afterward, we characterize the operation mathematically. Therefore, in this section, we define formally the semantic similarity as the semantic operation on string vectors.

The string vector is defined as a finite ordered set of strings as equation (6):

$$\mathbf{str} = [str_1, str_2, ..., str_d] \tag{6}$$

An element in the vector, str_i indicates a word which corresponds to its attribute. The number of elements of the string vector, **str**, is called its dimension. In order to perform

the operation on string vectors, we need to define the similarity matrix which was described in section 2.1, in advance. Therefore, a string vector consists of strings, while a numerical vector does of numerical values.

We need to define the semantic operation which is called 'semantic similarity' in this research, on string vectors; it corresponds to the cosine similarity on numerical vectors. We note the two string vectors as equation:

$$\mathbf{str}_1 = [t_{11}, t_{12}, ..., t_{1d}] \text{ and } \mathbf{str}_2 = [t_{21}, t_{22}, ..., t_{2d}]$$

where each element, t_{1i} or t_{12}, indicates a word. The operation is defined as equation (7) as follows:

$$sim(\mathbf{str}_1, \mathbf{str}_2) = \frac{1}{d}\sum_{i=1}^{d} sim(t_{1i}, t_{2i}) \quad (7)$$

The similarity matrix was constructed by the scheme which is described in section 2.1, and the $sim(t_{1i}, t_{2i})$ is computed by looking up it in the similarity matrix. Instead of building the similarity matrix, we may compute the similarity, interactively.

The semantic similarity measure between string vectors may be characterized mathematically. The commutative law applies as equation (8):

$$sim(\mathrm{str}_1, \mathrm{str}_2) = \frac{1}{d}\sum_{i=1}^{d} sim(t_{1i}, t_{2i}) = \frac{1}{d}\sum_{i=1}^{d} sim(t_{2i}, t_{1i}) = sim(\mathrm{str}_2, \mathrm{str}_1) \quad (8)$$

If the two string vectors are exactly same, its similarity becomes 1.0 as follows:

If $\mathbf{str}_1 = \mathbf{str}_2$ with $\forall_i, sim(t_{1i}, t_{2i}) = 1.0$,

$$sim(\mathbf{str}_1, \mathbf{str}_2) = \frac{1}{d}\sum_{i=1}^{d} sim(t_{1i}, t_{1i}) = \frac{d}{d} = 1.0$$

However, note that the transitive rule does not apply as follows:

If $sim(\mathbf{str}_1, \mathbf{str}_2) = 0.0$ and $sim(\mathbf{str}_2, \mathbf{str}_3) = 0.0$, not always $sim(\mathbf{str}_1, \mathbf{str}_3) = 0.0$

We need to define the more advanced semantic operations on string vectors for modifying other machine learning algorithms. We define the update rules of weights vectors which are given as string vectors for modifying the neural networks into their string vector based versions. We develop the operations which correspond to computing mean vectors over numerical vectors, for modifying the k means algorithms. We consider the scheme of selecting representative vector among string vectors for modifying the k medoid algorithms so. We will cover the modification of other machine learning algorithms in subsequent researches.

C. The Proposed Version of KNN

This section is concerned with the proposed KNN version as the approach to the text categorization. Raw texts are encoded into string vectors by the process which was described in Section III-A. In this section, we attempt to the traditional KNN into the version where a string vector is given as the input data. The version is intended to improve the classification performance by avoiding problems from encoding texts into numerical vectors. Therefore, in this section, we describe the proposed KNN version in detail, together with the traditional version.

The traditional KNN version is illustrated in Figure 2. The sample words which are labeled with the positive class or the negative class are encoded into numerical vectors. The similarities of the numerical vector which represents a novice word with those representing sample words are computed using the Euclidean distance or the cosine similarity. The k most similar sample words are selected as the k nearest neighbors and the label of the novice entity is decided by voting their labels. However, note that the traditional KNN version is very fragile in computing the similarity between very sparse numerical vectors.

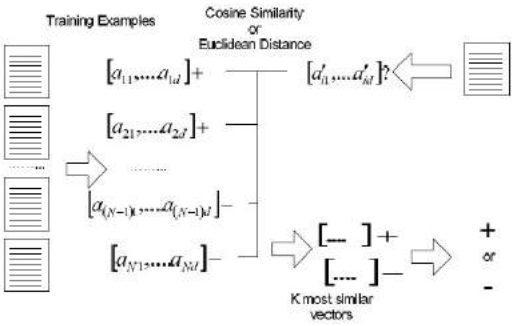

Fig. 2. The Traditional Version of KNN.

Separately from the traditional one, we illustrate the classification process by the proposed version in Figure 3. The sample texts labeled with the positive or negative class are encoded into string vectors by the process described in Section III-A. The similarity between two string vectors is computed by the scheme which was described in Section III-B2. Identically to the traditional version, in the proposed version, the k most similarity samples are selected, and the label of the novice one is decided by voting ones of sample entities. Because the sparse distribution in each string vector is never available inherently, the poor discriminations by sparse distribution are certainly overcome in this research.

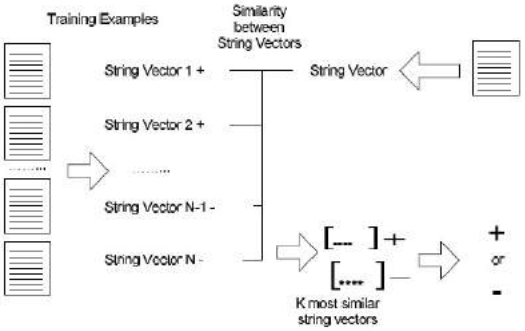

Fig. 3. The Proposed Version of KNN.

We may derive some variants from the proposed KNN version. We may assign different weights to selected neighbors instead of identical ones: the highest weights to the first nearest neighbor and the lowest weight to the last one. Instead of a fixed number of nearest neighbors, we select any number of training examples within a hyper-sphere whose center is the given novice example as neighbors. The categorical scores are computed proportionally to similarities with training examples, instead of selecting nearest neighbors. We may also consider the variants where more than two variants are combined with each other.

Because string vectors are characterized more symbolically

than numerical vectors, it is easy to trace results from classifying items in the proposed version. It is assumed that a novice item is classified by voting the labels of its nearest neighbors. The similarity between string vectors is computed by the scheme which is described in Section III-B2. We may extract the similarities of individual elements of the novice string vector with those of nearest neighbors labeled with the classified category. Therefore, the semantic similarities play role of the evidence for presenting the reasons of classifying the novice one so.

IV. Experimental Results

This section is concerned with the empirical experiments for validating the proposed version of KNN, and consists of the four sections. In Section I, we present the results from applying the proposed version of KNN to the text categorization on the collection, NewsPage.com. In Section II, we show the results from applying it for categorizing texts from the collection, Opinosis. In Section III, we mention the results from comparing the two versions of KNN with each other in categorizing texts from 20NewsGroups. In Section IV, we make the general discussions which is concerned with results from validating the proposed version of KNN, finally.

A. NewsPage.com

This section is concerned with the experiments for validating the better performance of the proposed version on the collection: NewsPage.com. The four categories are predefined in this collection, and texts are gathered from the collection category by category as labeled ones. Each text is classified exclusively into one of the four categories. In this set of experiments, we apply the traditional and proposed version of KNN to the classification task, without decomposing it into the binary classifications, and use the accuracy as the evaluation measure. Therefore, in this section, we observe the performance of the both versions of KNN by changing the input size.

In Table I, we specify the text collection, NewsPage.com, which is used in this set of experiments. This text collection was used for evaluating approaches to text categorization in previous works [1][3][13]. In the collection, the four categories are predefined: Business, Health, Internet, and Sports, and 375 texts are selected at random in each category. In each category, the set of 375 texts is partitioned into the 300 texts as training ones and the 75 texts as test ones. The text collection was built by copying and pasting individual news articles from the web site, newspage.com, in 2005, as plain text files whose extension is 'txt'.

TABLE I
The Number of Texts in NewsPage.com

Category	#Texts	#Training Texts	#Test Texts
Business	500	300	75
Health	500	300	75
Internet	500	300	75
Sports	500	300	75
Total	2000	1200	300

Let us mention the experimental process for validating empirically the proposed approach to the task of text categorization. In this collection, the texts are labeled with one of the four categories which are presented in Table I, and they are encoded into numerical and string vectors. For each test example, the KNN computes its similarities with the 1200 training examples and selects the three most similarity training examples as its nearest neighbors. Each of the 300 test examples is classified into one of the four categories: Business, Sports, Internet, and Health, by voting the labels of its nearest neighbors. We compute the classification accuracy by dividing the number of correctly classified test examples by the number of test examples, for evaluating the both versions of KNN algorithm.

In Figure 4, we illustrate the experimental results from categorizing texts, using the both versions of KNN algorithm. The y-axis indicates the accuracy which is the rate of the correctly classified examples in the test set. In the x-axis, each group indicates the input size which is the dimension of numerical vectors which represent texts. In each group, the gray bar and the black bar indicate the achievements of the traditional version and the proposed version of KNN algorithm, respectively. In the x-axis, the most right group indicates the average over the accuracies of the left groups.

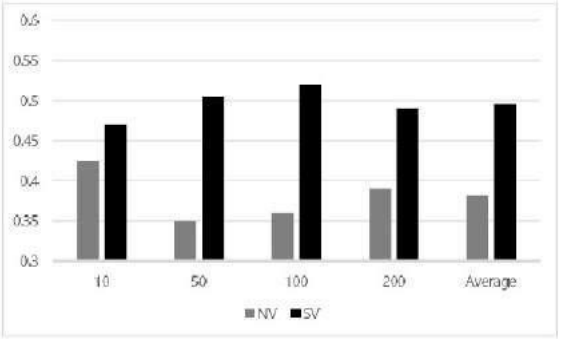

Fig. 4. Results from Classifying Texts in Text Collection: NewsPage.com

Let us make the discussions on the results from doing the text categorization using the both versions of KNN algorithm, as shown in Figure 4. The accuracy which is the performance measure of the classification task is in the range between 0.35 and 0.52. The proposed version of KNN algorithm works strongly better in the all input sizes. The performance difference between the two versions is outstanding in the two input sizes, 50 and 100. From this set of experiments, we conclude that the proposed version works strongly better than the traditional one, in averaging over the four cases.

B. OPINOPSIS

This section is concerned with the set of experiments for validating the better performance of the proposed version on the collection, Opinopsis. The three categories are predefined in the collection, and labeled texts are prepared from it. Each text is classified exclusively into one of the three categories. We do not decompose the given classification into binary classifications and use the accuracy as the evaluation measure. Therefore, in this section, we observe the performances of the both versions of KNN algorithm with the different input sizes.

In Table II, we specify the text collection, Opinosis, which is used in this set of experiments. The collection was used in previous works for evaluating approaches to text categorization. The three categories, 'Car', 'Electronics', and 'Hotel', are predefined, and all texts are used for evaluating the approaches to text categorization, in this set of experiments. We use six texts in each category among all texts as the test set as shown in Table II. We obtained the collection

by downloading it from the web site, http://archive.ics.uci.edu/ml/machine-learningdatabases/ opinion/.

TABLE II
The Number of Texts in Opiniopsis

Category	#Texts	#Training Texts	#Test Texts
Car	23	17	6
Electronic	16	10	6
Hotel	12	6	6
Total	51	33	18

We perform this set of experiments by the process which is described in Section I. We use all of 51 texts which are labeled with one of the three categories and encode them into numerical vectors and string vectors with the input sizes: 10, 50, 100, and 200. For each test example, the both versions of KNN computes its similarities with the 33 training examples and select the three most similar training examples as its nearest neighbors. Each of the 18 test examples is classified into one of the three categories, by voting the labels of its nearest neighbors. The classification accuracy is computed by the number of correctly classified test examples by the number of the test examples for evaluating the both versions of KNN algorithm.

In Figure 5, we illustrate the experimental results from categorizing texts using the both versions of KNN algorithm. Like Figure 4, the y-axis indicates the value of accuracy, and the x-axis indicates the group of both versions by an input size. In each group, the gray bar and the black bar indicate the achievements of the traditional version and the proposed version of KNN algorithm, respectively. In Figure 5, the most right group indicates the averages over results over the left four groups. Therefore, Figure 5 presents the results from classifying each text into one of the three categories by the both versions, on the text collection, Opinosis.

We discuss the results from doing the text categorization using the both versions of KNN algorithm, on Opinosis, shown in Figure 5. The accuracy values of the bother versions range between 0.55 and 1.0. The proposed version works better than the traditional one in the all input sizes. It shows the perfect results in the input size: 200. From this set of experiments, we conclude that the proposed version works outstandingly better than the traditional one, in averaging the four cases.

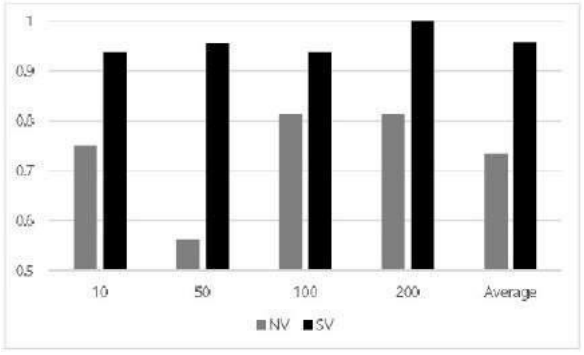

Fig. 5. Results from Classifying Texts in Text Collection: Opiniopsis

C.20NewsGroups

This section is concerned with one more set of experiments where the better performance of the proposed version is validated on 20NewsGroups. In this set of experiments, the four specific categories are predefined in this collection. Each text is exclusively classified into one of the four categories, like the previous sets of experiments. We apply the two versions of KNN algorithm, directly to the classification task, without decomposing it into binary classifications, and use the accuracy as the evaluation metric. Therefore, in this section, we observe the performances of the both versions of KNN algorithm with the different input sizes.

In Table III, we specify the specific version of 20NewsGroups which is used as the test collection, in this set of experiments. Within the general category, sci, we predefine the four categories: 'electro', 'medicine', 'script', and 'space'. In each category, we select 375 texts among approximately 1000 texts, at random. In each category, the set of 375 texts is partitioned into the training set of 300 texts and the test set of 75 texts, like the case in the previous set of experiments.

TABLE III
The Number of Texts in Opiniopsis

Category	#Texts	#Training Texts	#Test Texts
Electro	1000	300	75
Medicine	1000	300	75
Script	1000	300	75
Space	1000	300	75
Total	4000	1200	300

The process of doing this set of experiments is same to that in the previous sets of experiments. We select the balanced number of texts from the collection over categories, and encode them into the representations with the input sizes which are identical to those in the previous set of experiments. We use the two versions of KNN algorithm for their comparisons. Using the two versions of KNN algorithm, we classify each text in the test set into one of the four specific categories within the general category, ' sci': 'electro', 'medicine', 'script', and 'space'. We use the accuracy as the evaluation metric, like the previous set of experiments.

We present the experimental results from classifying the texts using the both versions of KNN algorithm on the specific version of 20NewsGroups. The frame of illustrating the classification results is identical to the previous ones. In each group, the gray bar and the black bar stand for the achievements of the traditional version and the proposed version, respectively. The y-axis in Figure 6, indicates the classification accuracy which is used as the performance metric. The texts are classified directly to one of the four categories like the cases in the previous sets of experiments.

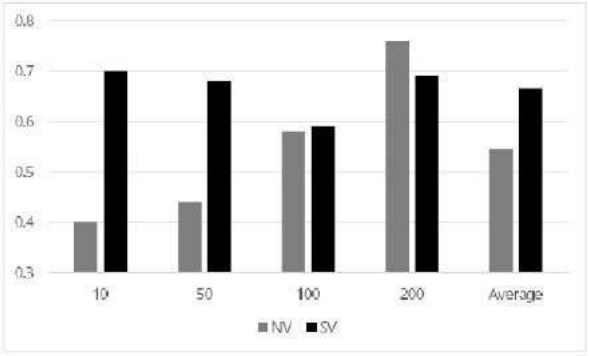

Fig. 6. Results from Classifying Texts in Text Collection: 20NewsGroups

Let us discuss on the results from classifying the texts on the specific version of 20NewsGroups, as shown in Figure 6.

The accuracies of the both versions range between 0.4 and 0.91. The proposed version shows its better performance in the smaller input sizes, but its turning point in the input size, 100. In the traditional version, its performance is proportional to the input size, whereas in the proposed version, its performance is independent of the factor. By the way, from this set of experiments, it is concluded that the proposed version have its outstandingly better performance, by averaging over the accuracies of the four input sizes.

V. Conclusion

Let us discuss the entire results from classifying texts using the two versions of KNN algorithm. The both versions is compared with each other in the task of text categorization, in these sets of experiments. The proposed version show its better results in all of the three collections. The accuracies of the traditional version range between 0.35 and 0.81, while those of the proposed version range between 0.49 and 1.0. From the three sets of experiments, we conclude that the proposed version improves the text categorization performance, as the contribution of this research.

We need to consider the remaining tasks for doing the further research. We will apply and validate the proposed approach in classifying texts in the specific domains such as medicine, engineering, and law, rather than the general domains. In order to improve the performance, we may consider various types of features of string vectors. As another scheme of improving the performance, we define and combine multiple similarity measures between two string vectors with each other. By adopting the proposed approach, we may implement the text categorization system as a module or an independent system.

Acknowledgment

This work was supported by 2017 Hongik University Research Fund.

References

[1] T. Jo, The Implementation of Dynamic Document Organization using Text Categorization and Text Clustering. PhD Dissertation of University of Ottawa, 2006.

[2] T. Jo and D. Cho, "Index Based Approach for Text Categorization", International Journal of Mathematics and Computers in Simulation, vol. 2, pp. 127-132, 2008.

[3] T. Jo, "Table based Matching Algorithm for Soft Categorization of News Articles in Reuter 21578", Journal of Korea Multimedia Society, vol. 11, pp. 875-882, 2008.

[4] T. Jo, "`Topic Spotting to News Articles in 20NewsGroups with NTC", Lecture Notes in Information Technology, pp50-56, vol. 7, 2011.

[5] T. Jo, "Definition of String Vector based Operations for Training NTSO using Inverted Index", Lecture Notes in Information Technology, pp50-56, vol. 7, 2011.

[6] F. Sebastiani, ``Machine Learning in Automated Text Categorization", ACM Computing Survey, vol. 34, pp. 1-47, 2002.

[7] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text Classification with String Kernels", Journal of Machine Learning Research, vol. 2, pp. 419-444, 2002.

[8] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, "Mismatch String Kernels for Discriminative Protein Classification", Bioinformatics, vol. 20, pp. 467-476, 2004.

[9] R. J. Kate and R. J. Mooney, "Using String Kernels for Learning Semantic Parsers", Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pp. 913-920, 2006.

[10] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," IEEE Trans. Neural Networks, vol. 4, pp. 570–578, Jul. 1993.

[11] T. Jo, "Single Pass Algorithm for Text Clustering by Encoding Documents into Tables", Journal of Korea Multimedia Society, vol. 11, pp. 1749-1757, 2008.

[12] T. Jo, "Device and Method for Categorizing Electronic Document Automatically", Patent Document, 10-2009-0041272, 10-1071495, 2011.

[13] T. Jo, "Normalized Table Matching Algorithm as Approach to Text Categorization", Soft Computing, vol. 19, pp. 839-849, 2015.

[14] T. Jo, "Inverted Index based Modified Version of K-Means Algorithm for Text Clustering", Journal of Information Processing Systems, Vol 4, pp. 67-76, 2008.

[15] T. Jo, "Representation of Texts into String Vectors for Text Categorization", Journal of Computing Science and Engineering, vol. 4, pp. 110-127, 2010.

[16] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", Journal of Network Technology, vol. 1, pp. 31-43, 2010.

[17] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", International Journal of Information Studies, vol. 2, pp.83-96, 2010.

Taeho Jo works currently as a faculty member in Hongik University, South Korea. He received his Bachelor degree from Korea University in 1994, his Master degree from Pohang University of Science and Technology in 1997, and his PhD degree from University of Ottawa in 2006. His research area spans mainly over text mining, neural networks, machine learning, and information retrieval. He has the four year experience of working for industrial organizations and ten year experience of working for academic ones. Recently, he is awarded in the world wide biography dictionary, Marquis Who's Who in the World, two times in 2016 and 2018.