Specializing K Nearest Neighbor into String Vector based Version using String Vector Operation in Index Optimization

Taeho Jo
President
Alpha AI Publication
Cheongju, South Korea
tjo018@naver.com

***Abstract*—This article proposes the modified KNN (K Nearest Neighbor) algorithm which receives a string vector as its input data and is applied to the index optimization. The results from applying the string vector based algorithms to the text categorizations were successful in previous works, and the index optimization is able to be viewed into a classification task where each word is classified into expansion, inclusion, and removal. In the proposed system, each word in the given text is classified into one of the three categories by the proposed KNN algorithm, associates words are added to ones which are classified into expansion, and ones which are classified into inclusion are kept by themselves without adding any word. The proposed KNN version is empirically validated as the better approach in deciding the importance level of words in news articles and opinions. We need to define and characterize mathematically more operations on string vectors for modifying more advanced machine learning algorithms.**

***Keywords*-Index Optimization; K Nearest Neighbor; String Vector**

I. Introduction

Index optimization refers to the process of optimizing a list of words which indicates texts for maximizing the information retrieval efficiency and performance. We need to expand the important words by adding their semantically similar words for improving the performance and remove unimportant words for improving the efficiency. In this research, we view the index optimization into the classification task where each word is classified into important words as targets of expansion, neutral words as ones of inclusion, and unimportant words as ones of removal. We prepare the sample words which are labeled with one of the three classes and construct the classification capacity by learning them. In this research, we assume that the supervised learning algorithms are used as the approach to the task, even if other types of approaches are available.

We mention the problems with which this research needs to tackle with. In encoding texts or words into numerical vectors for using the traditional classifiers, many features are required for keeping the robust classifications [29]. Each numerical vector which represents a word or a text has usually zero values dominantly as its elements; the discriminations among numerical vectors get very weak [2][25]. Although we proposed previously that texts or words should be encoded into tables as alternative structured forms to numerical vectors, it is very expensive to carry out the computation on them[2][25]. Therefore, this research challenges against the above problems by encoding words into string vectors.

Let us mention what is proposed in this research, as its ideas. In this research, we encode the words into string vectors each of which consists of text identifiers as its elements. We define the similarity measure between string vectors; it corresponds to the cosine similarity between numerical vectors. We modify the KNN into the version where a string vector is given as the input data, and apply it to the classification task into which we interpret the index optimization. The scope of this research is restricted to the classification of words into one of the three categories; the process of expanding words semantically is set out of this research.

Let us mention the benefits which are expected from this research. From this research, it is expected to represent words with more compactness and efficiency than to do them into numerical vectors. The improve discriminations among string vectors are expected from this research by avoiding almost completely the sparse distributions. The improved performance is also expected by solving the problems from encoding words into numerical vectors. Therefore, the goal of this research is to implement the index optimization module for information retrieval system with the benefits.

This article is organized into the five 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 the proposed approach by comparing it with the traditional one. In Section V, we mention the general discussion on the empirical validations and remaining tasks for doing the further research.

II. Previous Works

This section is concerned with the previous works which are relevant to this research. In Section II-A, we explore the previous cases of applying the KNN algorithm to text mining tasks. In Section II-B, we survey the schemes of

encoding texts or words into structured data. In Section II-C, we describe the previous machine learning algorithms which receive alternative structured data such as tables and string vectors to numerical vectors. Therefore, in this section, we provide the history about this research, by surveying the relevant previous works.

A. Applications to Word Classification Tasks

This section is concerned with the previous works which deal with applying the modified KNN algorithm to the index optimization and its similar tasks. We consider the topic based word categorization which classifies words based on their meaning as the typical word classification, and derive the keyword extraction and the index optimization from it. The KNN algorithm is modified into the modern version which solves the problems in encoding words into numerical vectors. The modern KNN algorithm shown its better results in the mentioned tasks, through the comparison with the traditional algorithm. This section is intended to explore the previous works on the modified KNN algorithm and its applications.

Let us mention the previous cases of applying the modernized KNN to the topic based word categorization, before mentioning its application to the index optimization. The KNN version which considers the similarities among features in order to solve the poor discriminations among sparse distribution in each numerical vector, was used for the word categorization [9]. The KNN was modernized by modifying it into the version which classifies a table directly, as the approach to the word categorization [10]. The KNN was modified into the version which classifies a graph directly for using it for the task [11]. The topic based word categorization to which the modernized KNN was applied becomes the source from which the index optimization is derived.

The keyword extraction may be derived from the word categorization as its special type, and let us survey the previous cases of applying the modernized KNN algorithm to the task. The modified version of KNN algorithm which uses the similarity metric considering the feature similarities was adopted for implementing a keyword extraction system [12]. The modernized version of KNN algorithm which processes tables directly was used for extracting keywords from a text [13]. In implementing a keyword extraction system, the KNN algorithm which was modernized into the version which processes graphs directly, was proposed as the approach to the keyword extraction [14]. The keyword extraction was interpreted into a binary classification in the above literatures.

Let us mention the previous cases of applying the modernized KNN algorithm to the task which is covered in this study. The KNN algorithm which considers the similarities among features in computing the similarity between a training item and a novice item, was used for implementing the index optimization system [15]. One which was modernized into the version which process tables directly was adopted as the approach to the index optimization [16]. The KNN algorithm which is called the graph based version and receives a graph directly was applied to the index optimization [7]. In the above literatures, together with this study, the index optimization is viewed as the classification of words into expansion, inclusion, or removal.

Let us mention some points which distinguish this research from ones which were surveyed above. We explored the previous cases of applying the three KNN versions which were modernized with the different directions to the index optimization and its related tasks. We mentioned the word categorization from which the index optimization is derived and the keyword extraction which is derived, together with the index optimization. The modernized version of the KNN algorithm which is applied to the index optimization in this study, deals with the string vectors, instead of the numerical vectors. In this study, it is applied to the index optimization.

B. Word and Text Encoding

This section is concerned with the previous works on encoding texts or words into replacements of numerical vectors. In previous works, the problems in encoding texts or words into numerical vectors, such as the huge dimensionality and the sparse distribution, were discovered. The previous works tried to solve the problems by encoding them into other types of structured form. In this section, we mention the tables, the string vectors, and the graphs as the numerical vector replacements. This section is intended to survey the previous cases of encoding them into the numerical vector replacements.

Let us survey the previous works on encoding texts or words into tables. Words were encoded into tables in using the AHC algorithm for the word clustering [17]. Texts were encoded into tables in using the KNN algorithm for the text categorization [18]. Texts were encoded so in using the AHC algorithm for the text clustering [21]. In the above literatures, texts and words were encoded into tables in using the AHC algorithm and the KNN algorithm.

Let us survey the previous works on encoding words or texts into string vectors. Words were encoded into string vectors, in using the AHC algorithm for clustering words [19]. Texts were encoded into string vectors in using the KNN algorithm for classifying texts [20]. In using the AHC algorithm for clustering texts, texts were encoded so [22]. In the literatures, we present the previous cases of encoding raw data into string vectors.

Let us explore the previous works on encoding words or texts into graphs. It was suggested that words should be encoded into graphs in applying the AHC algorithm to the semantic word clustering [8]. It was suggested that texts should be encoded into graphs in applying the KNN algorithm to the topic based text classification [23]. It was

suggested that texts should be encoded so in applying the AHC algorithm to the text clustering [24]. In the above literatures, we present the cases of mapping raw data into graphs.

We mentioned the three schemes of encoding words or texts in the previous works. We adopt the second scheme where words were encoded into string vectors, in this research. We define the similarity metric between string vectors, and modify the KNN algorithm into version which processes graphs directly. We use the modified version of KNN for implementing the index optimization system. We empirically validate the modified version in the index optimization of texts, comparing it with the traditional version.

C. Non-Numerical Vector based Machine Learning Algorithms

This section is concerned with the previous works on the non-numerical vector based machine learning algorithms. In the previous section, we presented the cases of encoding words or texts into non-numerical vectors, in using the KNN algorithm and the AHC algorithm. In this section, we mention the string kernel based Support Vector Machine, the table based matching algorithm, and the Neural Text Categorizer, as non-numerical vector based machine learning algorithms which are used for the text categorization. The reason of mentioning the approaches to the task is to map the index optimization into a classification task, in this study. This section is intended to explore the previous works which are involved in the three machine learning algorithms which were mentioned above.

Let us consider the string kernel as the way of avoiding the problems in encoding texts into numerical vectors. The string kernel was initially proposed for modifying the SVM (Support Vector Machine) as the approach to the text categorization by Lodhi et al. in 2002 [28]. It was utilized for modifying the k means algorithm as the approach to the text clustering by Karatzoglou and Feinerer in 2006 [27]. The string kernel based SVM was applied to the sentence classification by Kate and Mooney in 2006 [26]. The string kernel which was mentioned in the above literatures is the similarity metric between two raw texts based on characters.

Let us explore the previous works on the table based matching algorithm as a non-numerical vector based classifier. It was initially proposed as the approach to the text categorization by Jo and Cho, in 2008 [25]. It was applied to the soft text categorization where each text is allowed to be categorized into more than one category [2]. It was upgraded into the more stable version in 2015 [5]. In using the table based matching algorithm which is covered in the above literatures, texts should be encoded into tables, instead of numerical vectors.

Let us mention the Neural Text Categorizer as a non-numerical vector based approach to the text mining tasks. It was initially proposed as the approach to the text categorization by Jo in 2008 [3]. It was empirically validated as the better approach than the Naive Bayes and the SVM in both the soft categorization and the hard categorization in 2010 [4]. It was applied to the classification of Arabian texts by Abainia et al. in 2015 [1]. It was mentioned as an innovative neural network model in the research paper about dynamic neural network models by Vega and Medez-Vasquez [30].

We surveyed the previous works on the non-numerical vector based classification algorithms as the approaches to the text categorization. We mentioned the raw texts in using the string kernel based SVM, the tables in using the table matching algorithm, and the string vectors in using the Neural Text Categorizer. In this research, instead of texts, words are encoded into string vectors, whose elements are text identifiers which include itself. The KNN algorithm is modified into the version which processes string vectors directly as the approach to the index optimization. The task is viewed as the classification of each word into expansion, inclusion, and removal.

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 keyword extraction, and consists of the four sections. In Section III-A, we deal with the process of encoding words 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 keyword extraction. In Section III-D, we explain the architecture of the index optimization system where the proposed KNN is adopted.

A. Word Encoding

This section is concerned with the process of transforming words into string vectors. In Section II-B and II-C, we mentioned the previous cases of encoding raw data into string vectors. The three steps, the feature definition, the feature matching analysis, and the text identifier assignment, in encoding words into string vectors. A string vector which represents a word consists in order of text identifiers which are related with it. This section is intended to describe the three steps which are presented in Figure 1-3.

The features for encoding words into string vectors are presented in Figure 1. In defining the features, for each text, its first paragraph is assumed to be the key part and the dimension of a string vector which represents a word is set as d. The group of d features divided into the four subgroups by combining the relationship between a text and a word, such as the frequency and the weight, with the text scope, such as the entire text and the first paragraph. Texts are ranked by the relationship within each subgroup to $d/4$. The features are defined arbitrary and manually in the current system, and

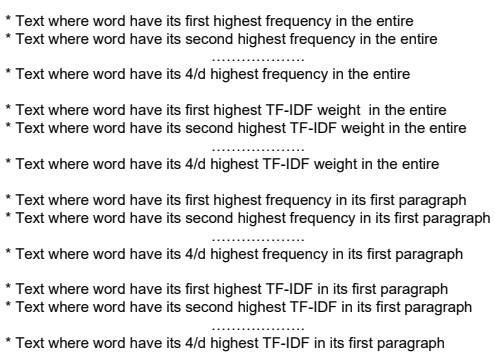

Figure 1. Defined Features

```
searchTextID(List textIDList, Feature featureItem, Word wordItem){
    for each textID in textIDList
        if isMatch(textID, featureItem, wordItem)
            return textID;
```

Figure 2. Feature Matching Analysis

the optimization and the automation of defining features of string vectors are left in the next research.

The process of feature matching analysis is illustrated as a pseudo code in Figure 2. A list of texts in the corpus, one among features which presented in Figure 1, and the word are given as arguments in the pseudo code. The word status is generated from each text and compared with the feature which is given as an argument. If both match with each other, the current text is returned. The status is viewed as the structure with the relationship such as the weight and the frequency, the scope such as the entire text and the first paragraph, and the rank.

The process of assigning text identifiers as feature values in the string vector which represents a word is illustrated in Figure 3. The features which are defined in Figure 1 are notated by $f_1, f_2, \ldots, f_d$ and the process of analyzing the feature matching is viewed as the function, $text\_id_i = F(f_i, word)$. The string vector is expressed by filling text identifiers as feature values as shown in equation (2),

$$\begin{aligned} \mathbf{str} &= [F(f_1, word), F(f_2, word), \ldots, F(f_d, word)] \\ &= [text\_id_1, text\_id_2, \ldots, text\_id_d] \end{aligned} \quad (1)$$

The vector, $\mathbf{str}$ in equation (2), is the d dimensional string vector which is filled with the text identifiers which given as strings. The domain of the function, F is the product of features and words and the range is the texts in the corpus.

In Figure 1-3, we presented the three steps which are involved in encoding a word into a string. A string vector is given as an unordered finite set of strings; its difference from the numerical vector is that elements are given as strings instead of numerical values. In the string vector which represents a word, the features which are shown in Figure 1, are relations between a word and texts and the feature values are given as text identifiers which correspond to the features. If a text is encoded so, the feature values are given as word. We need to define the operations on string vectors, for modifying the machine learning algorithms into the versions which process them directly.

B. Similarity Metric

This section is concerned with the semantic similarity between two string vectors. In the previous section, we studied the process of mapping words into string vectors. We need to define the semantic similarity between two string

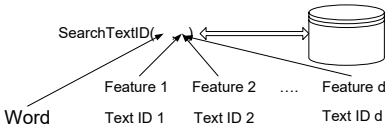

Figure 3. Text Identifier Assignment

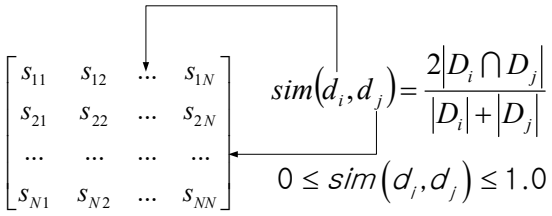

Figure 4. Similarity Matrix

vectors, as an operation on them, for modifying the KNN algorithm which is used for optimizing index. We understand the semantic operation conceptually, and start with defining the semantic operation on strings. This section is intended to describe the computation of semantic similarity between two string vectors.

Let us introduce the semantic operation as the basis for computing the similarity metric between the two string vectors. The semantic operation was initially proposed by Jo in 2015 [6] and it is defined as one on strings based on their meanings. Each string is assumed to have its own meaning and in [6], the semantic similarity between two strings, the semantic similarity average of any number of strings, and the semantic similarity variance were defined as operations. They were characterized mathematically and simulated on text collections with their various domains. The first operation is adopted for defining the similarity metric between string vectors which represent words.

In Figure 4, the semantic similarity matrix between two texts is illustrated. The two texts are notated by d_i and d_j and the similarity between them is notated by $sim(d_i, d_j)$. The two texts, d_i and d_j, are expressed as the two sets of words, D_i and D_j, and $|D_i|$ and $|D_j|$ are cardinalities of the two sets. The similarity between two texts is computed by equation (2),

$$sim(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} \tag{2}$$

and the similarity is always given as a normalized value between zero and one. The rows and the columns of the matrix which is presented in Figure 4, correspond to texts in the corpus, and each element becomes the similarity between corresponding texts.

A string vector is defined as an ordered finite set of strings as shown in equation (3),

$$\mathbf{str} = [str_1, str_2, ..., str_d] \tag{3}$$

The two string vectors are notated by equation (4) and (5),

$$\mathbf{str}_1 = [str_{11}, str_{12}, ..., str_{1d}] \tag{4}$$

$$\mathbf{str}_2 = [str_{21}, str_{22}, ..., str_{2d}] \tag{5}$$

The similarity between the two string vectors is defined as average over semantic similarities of one to one elements,

as shown in equation (6),

$$sim(\mathbf{str}_1, \mathbf{str}_2) = \frac{1}{d}\sum_{i=1}^{d} sim(str_{1i}, str_{2i}) \tag{6}$$

The string vector which represents a word consists of text identifiers and the value of $sim(str_{1i}, str_{2i})$ is looked up from the similarity matrix which is presented in Figure 4. The similarity between the two string vectors, $\mathbf{str}_1$ and $\mathbf{str}_2$ is always given as a normalized value between zero and one.

We mentioned the similarity between two string vectors as a normalized value between zero and one. If the two string vectors are exactly same to each other as shown in equation (7),

$$\mathbf{str}_1 = \mathbf{str}_2 \tag{7}$$

the semantic similarity between them is 1.0 as shown in equation (8),

$$\begin{aligned} sim(\mathbf{str}_1, \mathbf{str}_2) = sim(\mathbf{str}_1, \mathbf{str}_1) = \\ \frac{1}{d}\sum_{i=1}^{d} sim(str_{1i}, str_{1i}) = 1.0 \end{aligned} \tag{8}$$

If the semantic similarities between elements of two string vectors are zeros, the sematic similarity between them is 0.0 as shown in equation (9),

$$sim(\mathbf{str}_1, \mathbf{str}_2) = \frac{1}{d}\sum_{i=1}^{d} sim(str_{1i}, str_{1i}) = \frac{0}{d} = 0.0 \tag{9}$$

Because $0 \leq sim(\mathbf{str}_1, \mathbf{str}_2) \leq 1$ the semantic similarity between them is always given as a normalized value between zero and one by equation (10),

$$\begin{aligned} 0 \leq sim(\mathbf{str}_1, \mathbf{str}_2) \leq 1 \\ 0 \leq \frac{1}{d}\sum_{i=1}^{d} sim(str_{1i}, str_{2i}) \leq 1 \end{aligned} \tag{10}$$

The similarity threshold is set between zero and one in modifying machine learning algorithms using the operation.

C. Proposed Version of KNN

The proposed version of KNN algorithm as the approach to the index optimization is illustrated in Figure 5. We describe the process of encoding words into string vectors in Section III-A and assume that the training examples and a novice one are given as string vectors. The similarity metric between string vectors is used for selecting nearest neighbors from the training examples. The label of a novice example is decided by voting labels of the selected nearest neighbors and more variants may be derived by defining more voting schemes. This section is intended to describe the proposed version of KNN algorithm which deals with string vectors and its variants.

Let us mention the process of selecting the nearest neighbors from the training examples as the references for

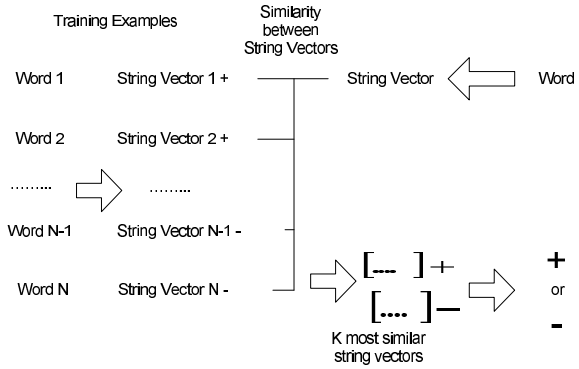

Figure 5. The Proposed Version of KNN

classifying data items. The sample words and a novice word are transformed into string vectors by the process which was described in Section III-A . The similarities of a novice one with the training ones are computed by equation (6). The training examples are ranked by their similarities and most K similar ones are selected as the nearest neighbors. The rank based selection is adopted in selecting nearest neighbors.

Let us mention the process of voting the labels of the nearest neighbors for deciding one of a novice item. We notate the set of nearest neighbors of the novice item, $\mathbf{str}$, whose elements are given as tables and their target labels, by equation (11),

$$\begin{aligned} Ne_k(\mathbf{str}) = \{(\mathbf{str}_1, y_1), (\mathbf{str}_2, y_2), \ldots, (\mathbf{str}_k, y_k)\}, \\ y_i \in \{c_1, c_2, \ldots, c_m\} \end{aligned} \tag{11}$$

where $c_1, c_2, \ldots, c_m$ are the predefined categories and k is the number of nearest neighbors. The number of the nearest neighbors which are labeled with the category,c_i is notated by $Count(Ne_k(\mathbf{str}), c_i)$. The label of the novice item, $\mathbf{str}$, is decided by the majority of categories in the nearest neighbors, as expressed by equation (12),

$$c_{\max} = \underset{i=1}{\overset{m}{\operatorname{argmax}}}\, Count(Ne_k(\mathbf{str}), c_i) \tag{12}$$

The external parameter,k, is usually set as an odd number for avoiding the possibility of largest number of nearest neighbors to more than one category.

Let us mention the weighted voting of labels of nearest neighbors as the alternative scheme to the above. Assuming that the similarity between two tables as a normalized value between zero and one, and we may use the similarities with the nearest neighbors, $sim(\mathbf{str}, \mathbf{str}_1), sim(\mathbf{str}, \mathbf{str}_2), \ldots, sim(\mathbf{str}, \mathbf{str}_k)$ as weights, $w_1, w_2, \ldots, w_k$ by equation (13),

$$w_i = sim(\mathbf{str}, \mathbf{str}_i) \tag{13}$$

indicates the similarity of a novice table with the ith nearest neighbor. The total weight of nearest neighbors which

labeled with the category, c_i by equation (14),

$$Weight(Ne_k(\mathbf{str}), c_i) = \sum_{\mathbf{str}_j \in c_i}^{k} w_j \qquad (14)$$

The label of the novice item, $\mathbf{str}$, is decided by the category which corresponds to the maximum sum of weights as shown in equation (15),

$$c_{\max} = \underset{i=1}{\overset{m}{\arg\max}}\, Weight(Ne_k(\mathbf{str}), c_i) \qquad (15)$$

When the weights of nearest neighbors are set constantly, equation (15) is same to equation (12), as expressed in equation (16),

$$Weight(Ne_k(\mathbf{str}), c_i) = Count(Ne_k(\mathbf{str}), c_i) \qquad (16)$$

We described the proposed version of the KNN algorithm in this section. In using the proposed KNN algorithm, raw data is encoded into string vectors, instead of numerical vectors. The similarities of a novice item with the training examples are computed by the similarity metric which is defined in Section III-B. The rank based selection is adopted as the scheme of selecting nearest neighbors among training examples. Because we are interested in the comparison of the traditional version and the proposed version as the ultimate goal, we use the unweighted voting in the experiments which are covered in Section IV.

D. Index Optimization System

This section is concerned with the index optimization system which adopts the string vector based KNN algorithm. In Section III-C, we described the proposed version of KNN algorithm as the approach to the index optimization. It is viewed into the classification task where each word is classified into removal, expansion, and inclusion. The scope of this system is restricted to indexing a text into words, encoding them into tables, and classifying each table into one of the three categories. This section is intended to describe the index optimization system with respect to its functions and architecture.

The sample words for implementing the index optimization is illustrated in Figure 6. The task is viewed in this study into the classification task where each word is classified into one of expansion, inclusion, and removal. The topic based word classification belongs to the domain independent classification where a word is classified identically with regardless of domain, whereas the index optimization belongs to the domain dependent classification where a word is classified differently depending on the domain. Domain by domain, the sample words which are labeled with one of the three categories are collected as shown in Figure 6. Before executing the index optimization in the system, the domain of the input text should be presented.

The entire architecture of the proposed index optimization system is illustrated in Figure 7. A text is given as the input and the words are extracted from it in the indexing module. The sample words in the three groups, the expansion group, the inclusion one, and the removal one, and ones indexed from the text are mapped into string vectors in the encoding module. The words are classified into one of the three categories, in the similarity computation module and the voting module. The words which are classified into removal will be discarded in the system.

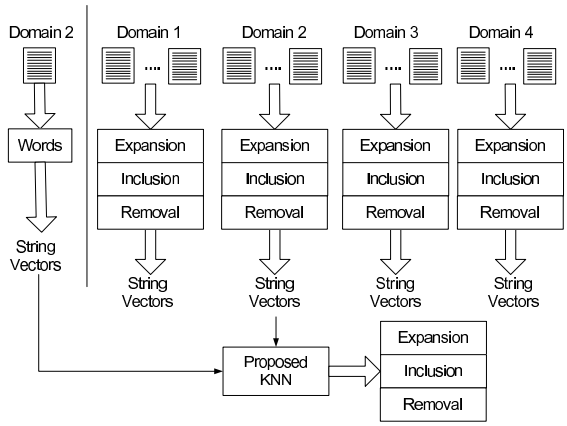

Figure 6. Sample Words

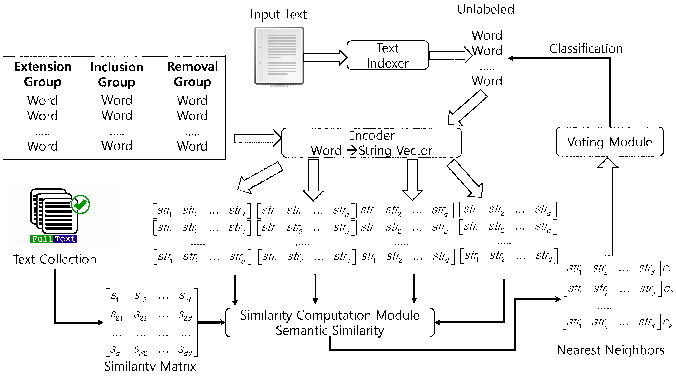

Figure 7. Proposed System Architecture

The execution process of the proposed system is illustrated as a block diagram in Figure 8. The sample words which are labeled with one of the three categories are collected from each domain, and encoded into string vectors. The input text is indexed into a list of words and they are also encoded into string vectors. The nearest neighbors are selected by the similarity computation and the label of decided by ones of its nearest neighbors. The words which are classified into removal are removed from the system.

Let us make some remarks on the proposed system which is illustrated in Figure 7, as the architecture. The index

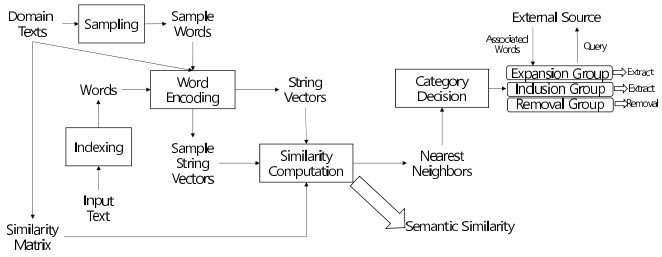

Figure 8. Execution Process of Proposed System

optimization is defined as the classification task where each word is classified into expansion, inclusion, or removal. Each word is encoded into a string vector instead of a numerical vector, and a string vector is classified directly. Ones which are classified into removal are excluded from indexing the input ext. We need to add module which retrieve more words which are relevant to ones which are labeled with expansion for implementing the index expansion.

IV. Experiments

This section is concerned with the empirical experiments for validating the proposed version of KNN, and consists of the four sections. In Section IV-A, we present the results from applying the proposed version of KNN to the index optimization on the collection, NewsPage.com. In Section IV-B and IV-C, we mention the results from comparing the two versions of KNN with each other in the task of index optimization from 20NewsGroups.

A. NewsPage.com

This section is concerned with the experiments for validating the better performance of the proposed version on the collection: NewsPage.com. We interpret the index optimization into the trinary classification where each word is classified into expansion, inclusion, and removal, and gather words which are labeled with one of the three categories, from the collection, topic by topic. Each word is allowed to be classified into one of the three labels, exclusively. We fix the input size as 50 dimensions of numerical vectors and 50 entries of tables, and use the accuracy as the evaluation measure. Therefore, this section is intended to observe the performance of the both versions of KNN in the four different domains.

In Table I, we specify NewsPage.com which is used as the source for extracting the classified words, in this set of experiments. The text collection, NewsPage.com, was used in previous works for evaluating approaches to text categorization [5]. In each topic, 375 words are extracted: 125 words labeled with expansion, 125 words labeled with inclusion, and 125 words labeled with removal. In each category, the set of 375 words is portioned into the 300 words as training examples and the 75 words as the test example, keeping the balanced distributions over the three labels. We decide target labels of words by their frequencies concentrated in the given category, combined with the subjectivity in scanning texts.

Table I
The Number of Texts and Words in NewsPage.com

Category	#Texts	#Training Words	#Test Words
Business	500	300(100+100+100)	75(25+25+25)
Health	500	300(100+100+100)	75(25+25+25)
Internet	500	300(100+100+100)	75(25+25+25)
Sports	500	300(100+100+100)	75(25+25+25)

Let us mention the experimental process of validating empirically the proposed approach to the task of index optimization. We collect sample words which are labeled with expansion, inclusion, or removal, in each of the four domains: Business, Sports, Internet, and Health, depending on subjectivities and concentrated frequencies of words, and encode them into numerical and string vectors. In each domain, for each of the 75 test examples, the KNN computes its similarities with the 300 training examples, and select the three most similar training examples as its nearest neighbors. Independently, we perform the four experiments each of which classifies each word into one of the three labels by the two versions of KNN algorithm. For evaluating the both versions of KNN in the classification which is mapped from the index optimization, we compute the classification accuracy by dividing the number of correctly classified test examples by the number of test examples.

In Figure 9, we illustrate the experimental results from classifying the words into one of the three categories as the process of index optimization, using the both versions of KNN algorithm. The y-axis indicate the accuracy which is the rate of the correctly classified words in the test set. In the x-axis, each group indicates the domain within which the index optimization which is viewed as the classification task is performed, independently. In each group, the gray bar and the black bar indicate the achievements of the traditional version and the proposed version, respectively. In the x-axis, the most right group indicates the average over the accuracies of the left four groups, and the input size which is the dimensional of numerical vectors is fixed to 50.

Let us make the discussions on the results from doing the index optimization, using the both versions of KNN algorithm, as shown in Figure 9. The accuracy which is the performance measure of this classification task is in the range between 0.33 and 0.52. The proposed version of KNN algorithm works better in the two domains: Business and Sports. It loses slightly in the others. From this set of experiments, we conclude that the proposed version works slightly better than the traditional one, in averaging over the

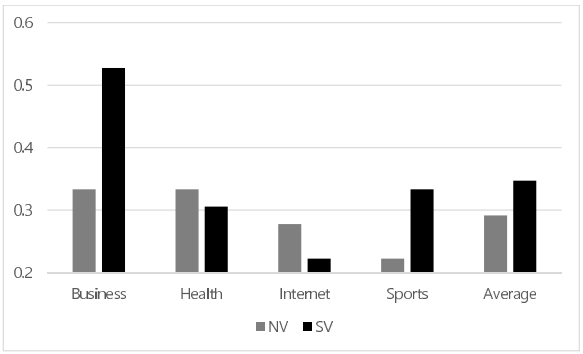

Figure 9. Results from Index Optimization in Text Collection: NewsPage.com

four cases.

B. 20NewsGroups I: General Version

This collection is concerned with one more set of experiments for validating the better performance of the proposed version on text collection: 20NewsGroups I. We gather words which are labeled with 'expansion', 'inclusion' or 'removal' from each broad category of 20NewsGroups, under the view of the index optimization into a binary classification. The task in this set of experiments is to classify each word exclusively into one of the three categories in each topic which is called domain. We fix the input size to 50 in encoding words, and use the accuracy as the evaluation measure. Therefore, in this section, we observe the performances of the both versions in the four different domains.

In Table II, we specify the general version of 20NewsGroups which is used for evaluating the two versions of KNN algorithm. In 20NewsGroup, the hierarchical classification system is defined with the two levels; in the first level, the six categories, alt, comp, rec, sci, talk, misc, and soc, are defined, and among them, the four categories are selected, as shown in Table II. In each category, we select 1000 texts at random and extract 375 words from them. Among the 375 words, one third of them is labeled with 'expansion', the second third is labeled with 'inclusion', and the other third is labeled with 'removal'. As shown in Table II, the 375 words is partitioned into the 300 words in the training set, and the 75 words in the test set, keeping the complete balance over them. In the process of gathering the classified words, each of them is labeled manually into one of the three categories by scanning individual texts.

The experimental process is identical is that in the previous sets of experiments. We collect the words by labeling manually them with 'expansion', 'inclusion', and 'removal', by scanning individual texts in each of the four domains, comp, rec, sci, and talk, and encode them into numerical and string vectors with the input size fixed to 50. For each test example, we compute its similarities with the 300 training

Table II
THE NUMBER OF TEXTS AND WORDS IN 20NEWSGROUPS I

Category	#Texts	#Training Words	#Test Words
Comp	1000	300(100+100+100)	75(25+25+25)
Rec	1000	300(100+100+100)	75(25+25+25)
Sci	1000	300(100+100+100)	75(25+25+25)
Talk	1000	300(100+100+100)	75(25+25+25)

examples, and select the three similar ones as its nearest neighbors. The versions of KNN algorithm classify each of the 75 test examples into one of the three categories by voting the labels of its nearest neighbors. Therefore, we perform the four independent set of experiments as many as domains, in each of which the two versions are compared with each other in the binary classification task.

In Figure 10, we illustrate the experimental results from deciding the importance degree of each word for maximize the information retrieval performance, on the broad version of 20NewsGroups. Figure 10 has the identical frame of presenting the results to those of Figure 9. In each group, the gray bar and the black bar indicates the achievements of the traditional version and the proposed version of KNN algorithm, respectively. Each group in the x axis indicates the domain within which each word is judged as one of the three importance degree. This set of experiments consists of the four binary classifications in each of which each word is classified into one of the three categories as the index optimization.

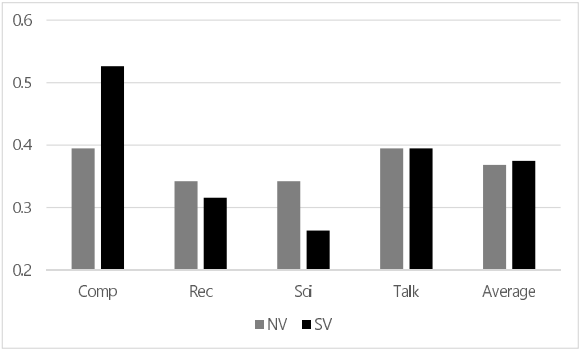

Figure 10. Results from Index Optimization in Text Collection: 20NewsGroup I

Let us discuss the results from doing the index optimization using the both versions of KNN algorithm, on the broad version of 20NewsGroups. The accuracies of the both versions of KNN algorithm range between 0.34 and 0.47. The proposed version shows the better performance in one of the four domains. Tt shows its competitive performances in the two domains, rec and talk, and its less performance in sci. From this set of experiments, the proposed version matches the traditional one, in averaging its four achievements.

C. 20NewsGroups II: Specific Version

This section is concerned with one more set of experiments where the better performance of the proposed version is validated on another version of 20NewsGroups. We gather the words which are labeled with 'expansion', 'inclusion', or 'removal'. We map the index optimization into a binary classification, and carry out the independent four binary classification tasks as many as topics, in this set of experiments. We fix the input size in representing words to 50, and use the accuracy as the evaluation metric. Therefore, in this section, we observe the performances of the both versions of the KNN with the four different domains.

In Table III, we specify the second version of 20NewsGroups which is used in this set of experiments. Within the general category, sci, the four categories, electro, medicine, script, and space, are predefined. In each specific category as a domain, we build the collection of labeled words by extracting 375 important words from approximately 1000 texts. We label manually the words with 'expansion', 'inclusion' or 'removal', maintaining the complete balance. In each domain, the set of 375 words is partitioned with the training set of 300 words and the test set of 75 words, as shown in Table III.

Table III
THE NUMBER OF TEXTS AND WORDS IN 20NEWSGROUPS II

Category	#Texts	#Training Words	#Test Words
Electro	1000	300(100+100+100)	75(25+25+25)
Medicine	1000	300(100+100+100)	75(25+25+25)
Script	1000	300(100+100+100)	75(25+25+25)
Space	1000	300(100+100+100)	75(25+25+25)

The process of doing this set of experiments is same to that in the previous sets of experiments. We collect the sample words which are labeled with 'expansion', 'inclusion', or 'removal', in each of the four domains: 'electro', 'medicine', 'script', and 'space, and encode them, fixing the in input size to 50. We use the two versions of KNN algorithm for their comparisons. Each example is classified into one of the three categories, by the both versions. We use the classification accuracy as the evaluation metric.

We present the experimental results from classifying the words 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 11, indicates the classification accuracy which is used as the performance metric. In this set of experiments, we execute the four independent classification tasks which correspond to their own domains, where each word is classified into 'expansion', 'inclusion', or 'removal'.

Let us discuss on the results from doing the index optimization on the specific version of 20NewsGroups, as shown in Figure 11. The accuracies of both versions of KNN algorithm range between 0.31 and 0.41. The proposed version shows its better results in two of the four domains. However, it is comparable in the others. In spite of that, from this set of experiments, it is concluded that the proposed version wins over the traditional one, according to the average over the four accuracies.

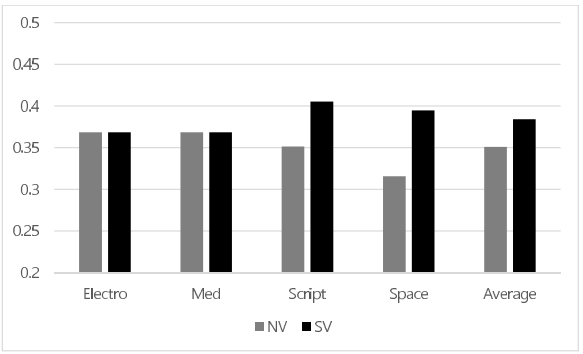

Figure 11. Results from Index Optimization in Text Collection: 20NewsGroup II

V. CONCLUSION

Let us discuss the entire results from performing the index optimization using the two versions of KNN algorithm. The both versions are compared with each other in the task of word classification which is mapped from the index optimization, in these sets of experiments. The proposed version shows its better results in all of the three collections and its matching ones in the others. The accuracies of the traditional version range between 0.21 and 0.39 and those of the proposed version range between 0.21 and 0.51. From the three sets of experiments, we conclude the proposed version improved the index optimization performance as the contribution of this research.

Let us mention the remaining tasks for doing the further research. The proposed approach should be validated and specialized in the specific domains: medicine, engineering and economics. Other features such as grammatical and posting features may be considered for encoding words into string vectors as well as text identifiers. Other machine learning algorithms as well as the KNN may be modified into their string vector based versions. By adopting the proposed version of the KNN, we may implement the index optimization system as a real program.

REFERENCES

[1] K. Abainia, S. Ouamour, and H. Sayoud. "Neural Text Categorizer for topic identification of noisy Arabic Texts", 1-8, Proceedings of 12th IEEE Conference on Computer Systems and Applications, 2015.

[2] T. Jo, "Table based Matching Algorithm for Soft Categorization of News Articles in Reuter 21578", 875-882, Journal of Korea Multimedia Society, Vol 11, No 6, 2008.

[3] T. Jo, “Neural Text Categorizer for Exclusive Text Categorization”, 77-86, Journal of Information Processing Systems, Vol 4, No 2, 2008.

[4] T. Jo, “NTC (Neural Text Categorizer): Neural Network for Text Categorization”, 83-96, International Journal of Information Studies, Vol 2, No 2, 2010.

[5] T. Jo, “Normalized Table Matching Algorithm as Approach to Text Categorization”, 839-849, Soft Computing, Vol 19, No 4, 2015.

[6] T. Jo, “Simulation of Numerical Semantic Operations on String in Text Collection”, 45585-45591, International Journal of Applied Engineering Research, Vol 10, No 24, 2015.

[7] T. Jo, “Graph based KNN for Optimizing Index of News Articles”, 53-62, Journal of Multimedia Information System, Vol 3, No 3, 2016.

[8] T. Jo, “Encoding Words into Graphs for Clustering Word by AHC Algorithm”, 90-95, The Proceedings of 12th International Conference on Multimedia Information Technology and Applications, 2016.

[9] T. Jo, “Semantic Word Categorization using Feature Similarity based K Nearest Neighbor”, 67-78, Journal of Multimedia Information Systems, 2018.

[10] T. Jo, “Table based K Nearest Neighbor for Word Categorization in News Articles”, 1214-1217, The Proceedings of 25th International Conference on Computational Science & Computational Intelligence, 2018.

[11] T. Jo, “K Nearest Neighbor specialized for Word Categorization in Current Affairs by Graph based Version”, 64-65, The Proceedings of 1st International Conference on Advanced Engineering and ICT-Convergence, 2018.

[12] T. Jo, “Extracting Keywords from News Articles using Feature Similarity based K Nearest Neighbor”, 68-71, The Proceedings of International Conference on Information and Knowledge Engineering, 2018.

[13] T. Jo, “Keyword Extraction in News Articles using Table based K Nearest Neighbors”, 1230-1233, The Proceedings of 25th International Conference on Computational Science & Computational Intelligence, 2018.

[14] T. Jo, “Graph based K Nearest Neighbors for Keyword Extraction in Current Affair Domain”, 47-48, The Proceedings of 1st International Conference on Advanced Engineering and ICT-Convergence, 2018.

[15] T. Jo, “Index Optimization in News Articles using Feature Similarity based K Nearest Neighbor”, 106-109, The Proceedings of 17th Int’l Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government, 2018.

[16] T. Jo, “Optimizing Index of News Articles by Table based Version of K Nearest Neighbors”, 1238-1241, The Proceedings of 25th International Conference on Computational Science & Computational Intelligence, 2018.

[17] T. Jo, “Using Table based AHC Algorithm for clustering Words in Domain on Current Affairs”, 1222-1225, The Proceedings of 25th International Conference on Computational Science & Computational Intelligence, 2018.

[18] T. Jo, “Modification into Table based K Nearest Neighbor for News Article Classification”, 49-50, The Proceedings of 1st International Conference on Advanced Engineering and ICT-Convergence, 2018.

[19] T. Jo, “String Vector based AHC Algorithm for Word Clustering from News Articles”, 83-86, The Proceedings of International Conference on Information and Knowledge Engineering, 2018.

[20] T. Jo, “Improving K Nearest Neighbor into String Vector Version for Text Categorization”, 1091-1097, ICACT Transaction on Communication Technology, Vol 7, No 1, 2018.

[21] T. Jo, “Applying Table based AHC Algorithm to News Article Clustering”, 8-11, The Proceedings of International Conference on Green and Human Information Technology, Part I, 2019.

[22] T. Jo, “Introduction of String Vectors to AHC Algorithm for Clustering News Articles”, 150-153, The Proceedings of 21st International Conference on Artificial Intelligence, 2019.

[23] T. Jo, “Graph based Version of K Nearest Neighbor for classifying News Articles”, 4-7, The Proceedings of International Conference on Green and Human Information Technology Part I, 2019.

[24] T. Jo, “Graph based Version for Clustering Texts in Current Affair Domain”, 171-174, The Proceedings of 15st International Conference on Data Science, 2019.

[25] T. Jo and D. Cho, “Index Based Approach for Text Categorization”, 127-132, International Journal of Mathematics and Computers in Simulation, Vol 2, No 1, 2007.

[26] R. J. Kate and R. J. Mooney, “Using String Kernels for Learning Semantic Parsers”, pp913-920, Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, 2006.

[27] A. Karatzoglou and I. Feinerer, “Text Clustering with String Kernels in R”, pp91-98, Advances in Data Analysis, 2006.

[28] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, “Text Classification with String Kernels”, 419-444, Journal of Machine Learning Research, Vol 2, No 2, 2002.

[29] F. Sebastiani, “Machine Learning in Automated Text Categorization”, 1-47, ACM Computing Survey, Vol 34, No 1, 2002.

[30] L. Vega and A. Mendez-Vazquez, “Dynamic Neural Networks for Text Classification”, 6-11, The Proceedings of International Conference on Computational Intelligence and Applications, 2016.