K Nearest Neghbor modified into String Vector based Version for Keyword Extraction

Duke 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 keyword extraction. The results from applying the string vector based algorithms to the text categorizations were successful in previous works and the keyword extraction is able to be mapped into the binary classification where each word is classified into keyword or non-keyword. In the proposed system, a text which is given as the input is indexed into a list of words, each word is classified by the proposed KNN version, and the words which are classified into keyword are extracted ad the output. The proposed KNN version is empirically validated as the better approach in deciding whether each word is a keyword or non-keyword 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*-Keyword Extraction;K Nearest Neighbor; String Vector**

I. Introduction

Keyword extraction refers to the process of extracting important words which are called keywords, from an article. The keywords are important indications for performing the information retrieval tasks, so we are interested very much in developing the schemes of extracting them. In this research, the keyword extraction is viewed into a binary word classification where each word is classified into a keyword or a non-keyword. We prepare the sample words which are labeled with 'keyword' or 'non-keyword', 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 some challenges with which this research attempts to tackle. In encodings texts or words into numerical vectors for using the traditional classification algorithms, many features are required, since each feature has very weak coverage[29]. Each numerical vector which represents a text or a word tends to be very sparse; it includes zero values dominantly[2][25]. Even if we proposed that texts or words should be encoded into tables in previous works, it was very expensive to compute the similarity between tables[2][25]. Therefore, in this research, we challenge against the above problems by encoding words into string vectors.

Let us consider some ideas which are proposed in this research. In this research, words are encoded into string vectors which consist of a finite ordered set of text identifiers as alternative representations to numerical vectors. We define the similarity measure between string vectors which is always given as a normalized value between zero and one; it corresponds to the cosine similarity between numerical vectors. The KNN (K Nearest Neighbor) is modified into the string vector based version, and applied to the special instance of word classification which is mapped from the keyword extraction. Note that in this research, the keyword extraction task is interpreted into the classification task.

Let us consider some benefits which are expected from this research. It is expected that string vectors are more compact representations of words which have much less features than numerical vectors. We expect the much better discriminations among string vectors than those among numerical vectors, since the sparse distributions can be avoided almost completely in each string vector. In this research, we expect also the improved performance by solving the above problems in encoding words into numerical vectors. Therefore, the goal of this research is to implement the keyword extraction systems which have the above 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 cases of modernizing the KNN algorithm and using the version for the keyword extraction and its similar tasks. In addition, we mention the topic based word categorization and the index optimization as the ones which are similar as the keyword extraction. The KNN algorithm which is mentioned in this section is not the traditional version, but the modernized version which solves the problems in encoding words into numerical vectors. The modern KNN versions have better performance than the traditional version in the tasks which are mentioned in this section. This section is intended to explore the previous works on applying the modern KNN version to the three tasks.

Let us review some works on applying the modernized KNN algorithm to the topic based word classification. The modified KNN algorithm which considers the similarities among features for solving the poor discriminations among sparse vectors was used for the word classification [8]. It was proposed that the modernized KNN algorithm which classifies a table directly should be used as a word classification tool [9]. The KNN algorithm was modified into the version which classifies a graph directly in using it for the word classification [10]. The tasks to which the modernized KNN algorithm was applied is to classify each word into one among predefined topics.

Let us present the cases of applying the modernized KNN algorithms to the keyword extraction which is covered in this study. The similarity metric which considers the feature similarities was proposed in using the KNN algorithm to the keyword extraction [11]. The KNN algorithm which was modernized into the version which processes tables directly tables, was considered as the approach to the keyword extraction [12]. The KNN algorithm which was modernized with a different direction for processing graphs directly was proposed as a keyword extraction tool [13]. In the above literatures, the keyword extraction is mapped into the binary classification of words.

Let us consider the text summarization as the task which is similar as the keyword extraction in selecting essential items. The KNN algorithm which considers the feature similarities in computing a similarity between a training example and a novice one, was applied to the text summarization [14]. The KNN algorithm which classifies a table directly, was adopted for implementing a text summarization system [19]. One which processes graphs directly, was proposed as the approach to the text summarization [20]. In the above literatures, the text summarization was mapped into the binary classification of paragraphs.

Let us mention some distinguished points of this study from ones which were surveyed above. We surveyed the cases of applying the three types of modernized versions of KNN algorithm for the keyword extraction and its related tasks. We mentioned the word categorization which classifies a word into one of the predefined categories, and the text summarization which classifies a paragraph into summary or non-summary. The modernized version of the KNN algorithm is proposed as the approach to the keyword extraction and classifies a string vector, instead of a numerical vector. The keyword extraction is mapped into a binary classification of words, following the mapping process in the previous works, and the proposed KNN was applied to the mapped task, in this study.

B. *Word and Text Encoding*

This section is concerned with the previous works on schemes of encoding texts or words. In the previous works, the issues in encoding texts or words into numerical vectors for applying the traditional machine learning algorithms to the text mining tasks were already realized. There were trials of challenging against the issues by encoding them into other types of structured data. In this section, we mention the tables, the string vectors, and the graphs as the structured data which replace the numerical vectors. This section is intended to survey previous cases of encoding texts or words into one of the three types.

Let us mention some cases of encoding texts or words into tables. Words were encoded into tables in using the AHC algorithm for clustering semantically words [15]. Texts were encoded into tables in using the KNN algorithm for categorizing texts [16]. In using the AHC algorithm for clustering texts, texts were encoded so [10]. In the above literatures, texts and words were encoded into tables in using the AHC algorithm and the KNN algorithm.

Let us survey the cases of encoding words or texts into string vectors. Words were encoded into string vectors in using the AHC algorithm for clustering words [17]. Texts were encoded into string vectors in using the KNN algorithm for categorizing texts [18]. In using the AHC algorithm for clustering texts, texts were encoded so [22]. The literatures presented the previous cases of encoding raw data into string vectors.

Let us survey the previous works on encoding words or texts into graphs. It was suggested that words should be encoded into graphs for using the AHC algorithm for clustering words, semantically [7]. It was suggested that texts should be encoded into graphs for using the KNN algorithm for categorizing texts [23]. It was suggested that texts should be encoded so for using the AHC algorithm for clustering texts [24]. From the above literatures, it was suggested that raw data should be encoded into graphs.

We mentioned the three kinds of structured data alternative to numerical vectors in the words which are surveyed above. We adopt the second kind of structured data, called string vectors, as word representations. We define the similarity metric between two string vectors, and modify the KNN algorithm into the version which classifies a string vector directly. We apply the modified KNN algorithm to implementing the keyword extraction system. We empirically validate the version in extracting keywords from a text.

C. Non-Numerical Vector based Machine Learning Algorithms

This section is concerned with the previous works on the supervised learning algorithms which process non-numerical vectors directly. In the previous section, we survey the cases of encoding words or texts into tables, string vectors, or graphs, as non-numerical vectors. In this section, we mention the string kernel Support Vector Machine, the table based matching algorithm, and the Neural Text Categorizer, as the approaches to the text classification, where texts are encoded into non-numerical vectors. Among them, the string kernel based SVM processes raw texts directly, in applying the kernel function. This section is intended to survey the previous works on the above machine learning algorithms.

Let us mention the string kernel as the similarity metric between raw texts. The SVM (Support Vector Machine) with the string kernel was applied to the text classification without encoding texts by Lodhi et al. in 2002 [28]. In 2006, the k means algorithm was modified using the string kernel by Karatzoglou and Feinerer [27]. The string kernel based SVM was applied to the sentence classification by Kate and Mooney in 2006 [26]. The string kernel which was covered in the above literatures, is the similarity between texts based on characters in them rather than meanings.

Let us mention the table based matching algorithm as another type of non-numerical vector based classification algorithm. 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 which allows to classify each text into more than one category [2]. It was improved into the more robust and stable version in 2015 [5]. In using the table based matching algorithm which was mentioned in the above literatures, text should be encoded into tables.

Let us mention the Neural Text Categorizer as a non-numerical vector based neural network model which is specialized for the text categorization. It was initially proposed as the approach to the text categorization by Jo in 2008 [3]. Its better performance than those of the Naive Bayes and the SVM was confirmed through the empirical validations in both the soft text categorization and the hard text categorization [4]. It was used for classifying Arabian texts by Abainia et al. in 2015 [1]. It was mentioned as an innovative neural network model by Vega and Mendez-Vasquez [30].

Let us mention the three non-numerical vector based classification algorithms which classifies raw texts, tables, or string vectors. The text classification is the problem to solve in the above literatures, whereas the keyword extraction is the problem to do in this research. Words are encoded into string vectors, whose elements are text identifiers. The KNN algorithm is modified into the version which deals with string vectors directly as the approach to the keyword extraction. The task is interpreted into the binary classification of words into keyword or non-keyword.

III. Proposed Approach

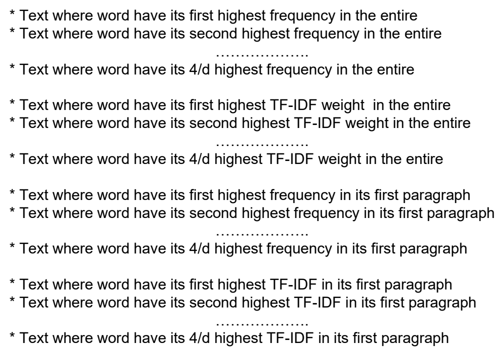
This section is concerned with encoding words into string vectors, modifying the KNN (K Nearest Neighbor) into the string vector based version and applying it to the keyword extraction, and consists of the four sections. In Section III-A, we deal with the process of encoding words into string vectors. 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 system architecture of the keyword extraction where the proposed KNN is adopted.

A. Word Encoding

This section is concerned with the transformation of words into string vectors. We surveyed the previous works on encoding raw data into string vectors in Section II-B and II-C. The three steps, feature definition, feature matching analysis, and text identifier assignment, are involved in encoding words so. A string vector which represents a word is the ordered finite set 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 which are defined as attributes of string vectors are illustrated in Figure 1. In defining the features, it is assumed that the first paragraph is the key part of each text and the dimension of string vector which represents a word is d. The group of features is divided into the four subgroups, and the frequency and the TF-IDF (Term Frequency and Inverse Document Frequency) are regarded as the relationships between a text and a word. Texts are ranked in each subgroup by their frequencies or weights in the entire text or its first paragraph. The issue of encoding words into string vectors is that this step is manual in the proposed system.

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 are presented in Figure 1, and the word are given as arguments. For each text, a relationship of the word is extracted, and the current relationship and the feature which is given as an argument are compared with each other. If they matches with each other, the current text is returned as an object. The feature is viewed as the status


* Text where word have its first highest frequency in the entire
* Text where word have its second highest frequency in the entire
....
* Text where word have its 4/d highest frequency in the entire

* Text where word have its first highest TF-IDF weight in the entire
* Text where word have its second highest TF-IDF weight in the entire
....
* Text where word have its 4/d highest TF-IDF weight in the entire

* Text where word have its first highest frequency in its first paragraph
* Text where word have its second highest frequency in its first paragraph
....
* Text where word have its 4/d highest frequency in its first paragraph

* Text where word have its first highest TF-IDF in its first paragraph
* Text where word have its second highest TF-IDF in its first paragraph
....
* Text where word have its 4/d highest TF-IDF in its first paragraph

Figure 1. Defined Features

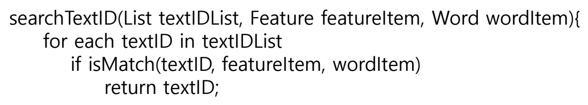

```
searchTextID(List textIDList, Feature featureItem, Word wordItem){
    for each textID in textIDList
        if isMatch(textID, featureItem, wordItem)
            return textID;
```

Figure 2. Feature Matching Analysis

with the three parts: the relation value such as the frequency or the weight, the scope such as the entire text and the first paragraph, and the rank.

The process of assigning text identifiers as feature values of the string vector which represents a word is illustrated in Figure 3. We define d features which are notated by $f_1, f_2, \ldots, f_d$ as shown in Figure 1, and view the process of analyzing the feature matching as the function of a word and a feature:$text\_id_i = F(f_i, word)$. By applying the function, text identifiers are filled in the string vector as shown in equation (2),

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

The d dimensional string vector which is presented above is given as a representation of the word. A corpus is required for encoding a word into a string vector.

In this section, we mentioned the three steps which are involved in encoding a word into a string vector. The difference of the string vector 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 are given as relations between texts and the word, and the feature values are given as text identifiers which correspond to the features. If a text is encoded into a string vector, a word is given as a feature values. 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 mentioned the process of encoding words into string vectors. We need to define the semantic similarity between string vectors as an operation for modifying the KNN algorithm which is used for extracting keywords. We understand the semantic operation conceptually and start with defining the semantic similarity between text identifiers. This section is intended to describe the semantic similarity between string vectors representing words.

The semantic operations are defined as ones on strings based on their meanings. They were initially proposed on strings by Jo in 2015 [6]. The literature defined the three

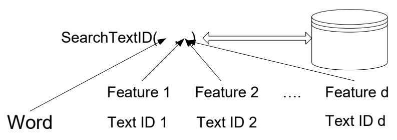


Figure 3. Text Identifier Assignment

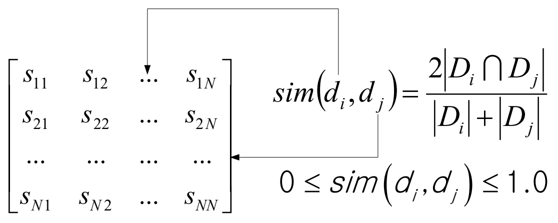

$sim(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|}$
$0 \leq sim(d_i, d_j) \leq 1.0$

Figure 4. Similarity Matrix

semantic operations: the semantic similarity between two strings, the semantic similarity average over strings, and the semantic similarity variance. 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, in this research.

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),

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

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),

$$\mathbf{str} = [str_1, str_2, ..., str_d] \quad (3)$$

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

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

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

The similarity between the two string vectors is defined as average over semantic similarities of one to one elements, as shown in equation (6),

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

The string vector which represents a word consists of text identifiers and the value of $sim(d_{1i}, d_{2i})$ 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

This section is concerned with the proposed version of KNN algorithm which is shown in Figure 5 as the approach to the keyword extraction. We described the process of encoding words into string vectors in Section III-A, and assumed that the training examples and a novice example are given as string vectors. The semantic similarity which was mentioned as the operation on string vectors in Section III-B is used for selecting nearest neighbors from the training examples. In addition, variants may be derived by defining more selection schemes and voting ones. This section is intended to describe the proposed version of the KNN algorithm which classifies string vectors directly and its variants.

Let us mention the process of selecting the nearest neighbors from the training examples as the references for classifying a data item. The training words and a novice word are converted into string vectors by the process which was covered in Section III-A. The similarities of a novice item with the training ones are computed by equation (6). The training examples are ranked by their similarities and the k most similar ones are selected as the nearest neighbors. The rank based scheme is adopted in selecting the nearest neighbors in the KNN algorithm.

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, $\mathbf{str}$

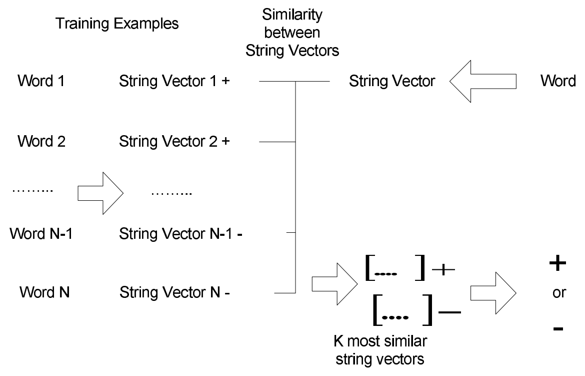


Figure 5. The Proposed Version of KNN

, 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),

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

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),

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

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

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

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. Keyword Extraction System

This section is concerned with the keyword extraction system which adopts the string vector based KNN algorithm. We described the proposed version of KNN algorithm as the approach to the keyword extraction in Section III-C. The keyword extraction is viewed into the binary classification of each word into keyword or non-keyword. In the system, a text is indexed into a list of words and words which are classified into keyword are extracted externally. This section is intended to describe the keyword extraction system with respect to its functions and architecture.

In Figure 6, the sample words which are labeled with keyword or non-keyword for implementing the keyword extraction system domain by domain. In this study, the keyword extraction is viewed into the binary classification where each word is classified into keyword or non-keyword. The topic based word classification belongs to the domain independent classification where the same word is classified absolutely into the same category with regardless of domain, whereas the keyword extraction belongs to the domain dependent classification where the same word is classified differently depending on the domain. The text domains should be predefined as a list and words are manually labeled with keyword or non-keyword in each domain. Presentation of a domain tag of input text is required for selecting keywords from it by the system.

The entire architecture of the proposed keyword extraction system is illustrated in Figure 7. A text is given as the input, and words are extracted from it in the indexing module. The sample words in the keyword group and the non-keyword group and ones which are indexed from the text are mapped into string vectors in the encoding module. The words which indexed from the text are classified into one of the two categories in the similarity computation module and the voting module. The system consists of the four modules: the indexing module, the encoding module, the similarity computation module, and the voting module.

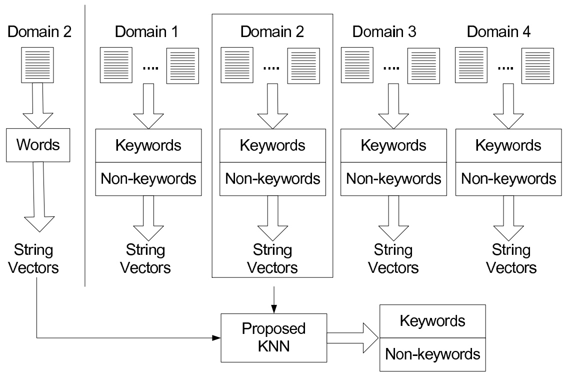


Figure 6. Sample Words

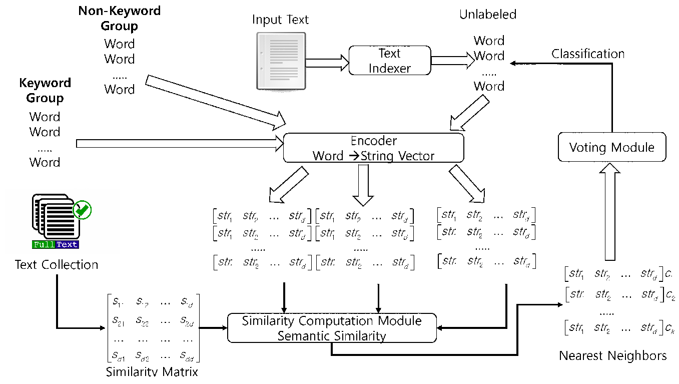


Figure 7. Proposed System Architecture

The execution process of the proposed system is illustrated as the block diagram in Figure 8. The sample words which are labeled with keyword or non-keyword 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 computing its similarities with the sample words and its label is decided by voting ones of its nearest neighbors for each word. The words which are classified into keyword are extracted as the final output.

Let us make some remarks on the proposed system which is illustrated in Figure 7 as the architecture. The keyword extraction is defined as the binary classification where each word is classified into keyword or non-keyword. Each word is encoded into a string vector instead of a numerical vector and a string vector is classified directly. Among words included in the input text, ones which are classified into keyword are selected as keywords of the input text. We need to add the text classification module, in order to predict the domain of the input text.

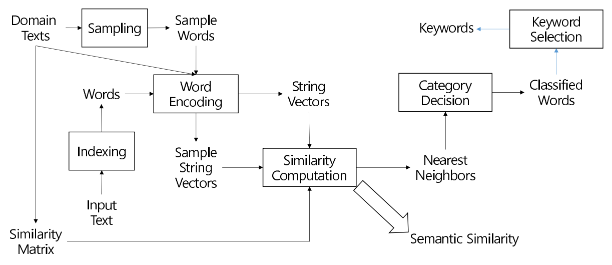


Figure 8. Execution Process of Proposed System

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 keyword extraction 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 keyword extraction 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 keyword extraction into the binary classification where each word is classified into keyword or non-keyword, and gather words which are labeled with one of the two categories, from the collection, topic by topic. Each word is allowed to be classified into one of the two labels, exclusively. We fix the input size as 50 of numerical vectors and string vectors, 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 was used for evaluating approaches to text categorization, in previous works [5]. In each topic, we extracted 125 words labeled with keyword, and 125 words labeled with non-keyword. The set of 250 words in each topic is partitioned into the 200 words as training ones and the 50 words as the test ones, keeping the complete balanced distribution over the two labels, as shown in Table I. In building the test collection of words, we decide whether each word is a keyword or not, depending on its frequency concentrated in the given category combining with the subjectivity, in scanning articles.

Let us mention the experimental process of validating empirically the proposed approach to the task of keyword extraction. We collect sample words which are labeled with keyword or non-keyword 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 50 test examples, the KNN computes its similarities with the 200 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 keyword or non-keyword by the two versions of KNN algorithm. For evaluating the both versions of KNN in the classification which is mapped from the keyword extraction, we compute the classification accuracy by dividing the number of correctly classified test examples by the number of test examples.

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

Category	#Texts	#Training Words	#Test Words
Business	500	200 (100+100)	50 (25+25)
Health	500	200 (100+100)	50 (25+25)
Internet	500	200 (100+100)	50 (25+25)
Sports	500	200 (100+100)	50 (25+25)

In Figure 9, we illustrate the experimental results from decoding whether each word is a keyword, or not, 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. Each group in the x-axis indicates the domain within which the keyword extraction which is viewed into a binary classification is performed, independently. In each group, the gray bar and the black bar indicate the performance of the traditional version and the proposed version of KNN algorithm, respectively. The most right group in Figure 9 indicates the average over accuracies over the left four groups, and set the input size which is the dimension of numerical vectors as 50.

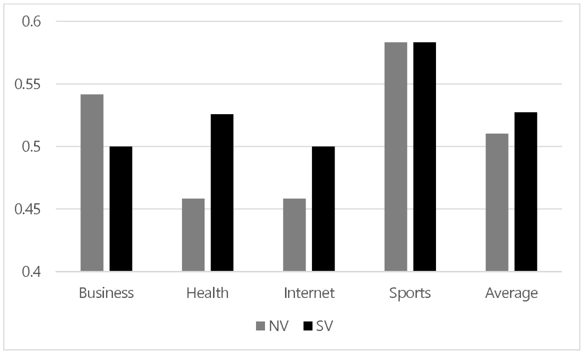


Figure 9. Results from Recognizing Keywords in Text Collection: NewsPage.com

Let us make the discussions on the results from doing the keyword extraction using the both versions of KNN algorithm, as shown in Figure 9. The accuracy which is the performance measure of the classified task is in the range between 0.45 and 0.62. The proposed version of the KNN algorithm works better in the two domains: Health and Internet. It matches with the traditional version in the domain, Sports, but is lost in the domain, Business. However, from this set of experiments, we conclude the proposed version works slightly better than the traditional one, in averaging over the 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 'keyword' or 'non-keyword' from each broad category of 20NewsGroups, under the view of the keyword extraction into a binary classification. The task in this set of experiments is to classify each word exclusively into one of the two 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 250 words from them. Among the 250 words, one half of them is labeled with 'keyword', and the other half is labeled with 'non-keyword'. As shown in Table II, the 250 words is partitioned into the 200 words in the training set, and the 50 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 two categories by scanning individual texts.

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

Category	#Texts	#Training Words	#Test Words
Comp	1000	200 (100+100)	50 (25+25)
Rec	1000	200 (100+100)	50 (25+25)
Sci	1000	200 (100+100)	50 (25+25)
Talk	1000	200 (100+100)	50 (25+25)

The experimental process is identical is that in the previous sets of experiments. We collect the words by labeling manually them with 'keyword' or 'non-keyword' 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 200 training examples, and select the three similar ones as its nearest neighbors. The versions of KNN algorithm classify each of the 50 test examples into one of the two 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 whether each word is a keyword or not 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 a keyword or a non-keyword. This set of experiments consists of the four binary classifications in each of which each word is classified into one of the two categories.

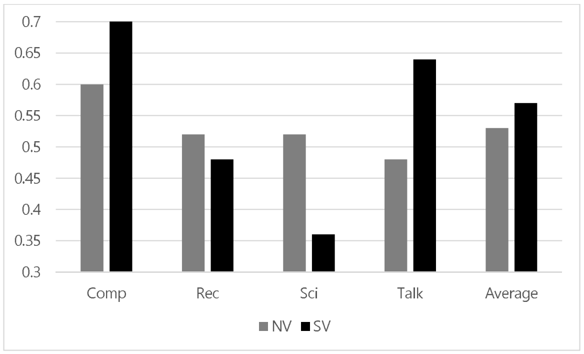


Figure 10. Results from Recognizing Keywords in Text Collection: 20NewsGroup I

Let us discuss the results from doing the keyword extraction using the both versions of KNN algorithm, on the broad version of 20NewsGroups. The accuracies of the both versions of KNN algorithm range between 0.47 and 0.7. The proposed version shows the better performance in the two domains, comp and talk. However, it shows its competitive performances in the domain, rec, and its less performance in the domain, sci. From this set of experiments, the proposed version keeps its better performance, in averaging its four achievements, in spite of its some leaded ones.

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 'keyword' or 'non-keyword'. We map the keyword extraction 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 250 important words from approximately 1000 texts. We label manually the words with 'keyword' or 'non-keyword', maintaining the complete balance. In each domain, the set of 250 words is partitioned with the training set of 200 words and the test set of 50 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	200 (100+100)	50 (25+25)
Medicine	1000	200 (100+100)	50 (25+25)
Script	1000	200 (100+100)	50 (25+25)
Space	1000	200 (100+100)	50 (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 'keyword' or 'non-keyword', 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 two 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 'keyword' or 'non-keyword'.

Let us discuss on the results from doing the keyword extraction on the specific version of 20NewsGroups, as shown in Figure 11. The accuracies of both versions of KNN algorithm range between 0.40 and 0.67. The proposed version shows its better results in three of the four domains. It shows its comparable one in the domain, 'medicine'. From this set of experiments, it is concluded that the proposed version is better by averaging over the accuracies of the four domains.

V. CONCLUSION

Let us discuss the entire results from extracting keywords using the two versions of KNN algorithm. The both versions

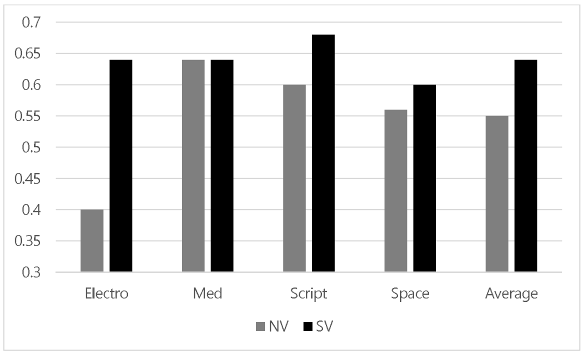


Figure 11. Results from Recognizing Keywords in Text Collection: 20NewsGroup II

are compared with each other in the task of word classification which is mapped from the keyword extraction, in these sets of experiments. The proposed version shows its better results in all of the three collections. The accuracies of the traditional version range between 0.40 and 0.64 and those of the proposed version range between 0.59 and 0.70. From the three sets of experiments, we conclude we conclude the proposed version improved the keyword extraction 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 keyword extraction 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, "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.

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

[9] 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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] T. Jo, "ummarizing News Articles by Feature Similarity based Version of K Nearest Neighbor", 51-52, The Proceedings of 1st International Conference on Advanced Engineering and ICT-Convergence, 2018.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] T. Jo, "Automatic Summarization System in Current Affair Domain by Table based K Nearest Neighbor", 115-121, The Proceedings of 2nd International Conference on Advanced Engineering and ICT-Convergence, 2019.

[20] T. Jo, "Validation of Graph based K Nearest Neighbor for Summarizing News Articles", 5-8, The Proceedings of International Conference on Green and Human Information Technology Part II, 2019.

[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", 913-920, 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", 91-98, 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.