Feature Similarity based K Nearest Neighbor for Optimizing of Text Indexes

Taeho Jo
Alpha AI Publication, 28221, South Korea
tjo018@naver.com

***Abstract*—This article proposes the modified KNN (K Nearest Neighbor) algorithm which considers the feature similarity and is applied to the index optimization. The texts which are given as features for encoding words into numerical vectors are semantic related entities, rather than independent ones, 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. The significance of this research is to improve the classification performance by utilizing the feature similarities.**

***Index Terms*-Feature Similarity, Feature Value Similarity, Index Optimization, K Nearest Neighbor**

I. Introduction

The index optimization refers to the process of adjusting a list of index terms by adding more similar words, reserving some words, and removing irrelevant words, in order to maximize the information retrieval performance. The scope of this research is restricted to the classification task where each word is classified into the three categories: 'expansion', 'reservation', and 'removal'. We prepare the sample words which are labeled with one of the three categories and define the factors which influence on the classification. By learning the sample words, we construct the classification capacity and classify novice words which are given afterward as the input into one of the three categories. In this research, we assume that a supervised learning algorithm is used as the approach to the index optimization which is set as a classification task.

Let us mention some challenges which we try to solve in this research. The dependency among features exist clearly, so the Bayesian networks were previously proposed as a machine learning based approach, but it requires the complicated analysis among features for using it[**?**]. The requirement of many features for keeping the robustness in encoding words into numerical vectors is caused by the assumption of feature independences. Because of very little coverage of each feature, we cannot avoid the sparse distribution where zero values are dominant in each numerical vector with more than 95%[**?**]. Therefore, this research is intended to solve the problems by considering the feature similarity as well as the feature value one.

Let us mention what we propose in this research as its idea. In this research, we consider the both similarity measures, feature similarity and feature value similarity for computing the similarity between numerical vectors. This research interprets the index optimization into a classification task where a machine learning algorithm is applicable. The KNN (K Nearest Neighbor) is modified into the version which accommodates the both similarity measures, and applied to the index optimization task which is mapped into a classification task. Therefore, the goal of this research is to improve the index optimization performance by solving the above problems.

Let us mention the benefits which we expect from this research. Computing the similarity between words using the feature similarity as well as the feature value similarity opens potentially the way of reducing the dimensionality of numerical vectors. The addition of one more similarity measure cuts down the information loss for computing the semantic similarity between words. By considering the both kinds of similarity measures, we expect from this research to improve the discriminations among numerical vectors which tend to be sparse.Therefore, the goal of this research is to implement the index optimization systems with their better performance by obtaining 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 modernized KNN algorithms for the tasks which are relevant to the index optimization. In Section II-B, we survey the previous works on the semantic operations on strings. In Section II-C, we describe the previous works on the schemes of computing the semantic similarity between two words. In Section II-D, we present previous works on the schemes of computing the similarity between texts, as the feature similarity.

A. Relevant Tasks

This section is concerned with the previous works on the semantic operations on strings. The previous research on the semantic operations provide the basis for understanding the index optimization which is an instance of the semantic word classification. The semantic operations which are operations on strings based on their meanings, rather than their spellings, are defined under the assumption that each string has its own meaning. The semantic similarity which is the typical semantic operation on two strings is to compute the similarity between strings based on their meanings. This section is intended to survey the previous works on the semantic operations on strings.

Let us survey on the cases of applying the modernized machine learning algorithm to the word categorization as the first relevant task. In 2015, Jo proposed the idea of modifying the KNN algorithm considering the feature similarity and applying it to the word categorization [3]. In 2018, he presented that its performance is better than that of the traditional KNN version, in his intermediate report [24]. In 2018, in his final report, its better performance was validated in the three test sets: NewsPage.com, Opiniopsis, and 20NewsGroups [25]. In the above literatures, it is effective to modify the KNN algorithm into the version considering the feature similarity in the word categorization.

Let us survey on the effectiveness of the modernized KNN algorithm in using it for the keyword extraction which is a special instance of the word categorization. In 2015, it was proposed that the KNN is modified into the version as the approach to the keyword extraction [4]. In 2016, it was proposed that one more modernized KNN which processes directly graphs instead of numerical vectors as the approach to the keyword extraction [7]. In 2018, the better performance of the KNN version which considers the feature similarity than the traditional version was presented in extracting keywords from a text [26]. In the above literatures, the cases of applying the modernized KNN version to the keyword extraction are presented.

Let us explore the cases of applying the modernized KNN version for the index optimization which is covered in this research. It was initially proposed that the KNN version should be applied to the index optimization as the conceptual idea in 2015 [5]. The KNN version which processes graphs directly was validated as the better approach than the traditional version in the index optimization in 2016 [8]. The better performance of the KNN version was presented in the index optimization in a single test set in 2018 [27]. In the above literatures, the two modernized versions of the KNN which were used for the index optimization are mentioned.

The index optimization is the task which this research tries to solve. The task is viewed into the classification of each word into one of the three categories: expansion, inclusion, and removal. The process of the index optimization is to index a text into a list of words, to classify the words into one of the three categories, and to exclude the words which are labeled with removal. To the words which are labeled with expansion, more words from external sources are added as their associative ones. The word categorization is the classification of each word by its meaning, whereas the index optimization is the classification by its importance in the given text.

B. Semantic Operations

This section is concerned with the previous works on the semantic operations on strings. The previous research on the semantic operations provide the basis for understanding the index optimization which is an instance of the semantic word classification. The semantic operations which are operations on strings based on their meanings, rather than their spellings, are defined under the assumption that each string has its own meaning. The semantic similarity which is the typical semantic operation on two strings is to compute the similarity between strings based on their meanings. This section is intended to survey the previous works on the semantic operations on strings.

Let us explore the previous works on the semantic similarity for presenting its history. The semantic similarity was initially proposed for defining the string vector kernel in modifying the SVM as the approach to the text classification in 2008 [1]. The semantic similarity between string vectors was applied to the unsupervised neural networks which is called NTSO (Neural Text Self Organizer) and created as the approach to the text clustering in 2010 [2]. It was applied for computing the similarity between string vectors in the KNN as the approach to the text categorization in 2018 [28]. The semantic similarity between two strings is basis for deriving more advanced semantic operations.

Let us mention the semantic similarity average into which the semantic similarity is expanded. The operation was described by Jo in 2015 [6]. The semantic similarity average is to compute the semantic similarities of all possible pairs of strings and to average them in its definition. The output value of the semantic similarity average is given as a normalized one between zero and one, and reflects the semantic cohesion of words. As the computation complexity, it takes the quadratic complexity to the number of strings for computing the value.

Let us mention one more semantic operation on strings, which is called semantic similarity variance. The operation is derived from the semantic similarity average, and described by Jo in 2015, together with the semantic similarity average [6]. The operation is to compute the semantic similarity average over strings and to average the difference squares of the individual semantic similarities of all possible pairs from the semantic similarity average. The semantic similarity average and the semantic similarity variance were simulated on the several text collections, presenting the distribution

over them. The statistical analysis on the data which are given as strings is opened by inventing the two semantic operations.

Let us mention some relevancies of the semantic operations on strings to this research. The index optimization which is covered in this research is an instance of the semantic word categorization based on meanings. Because the text mining tasks are based on meanings, rather than on spellings, the word meanings are important for performing the text mining tasks. In this research, the semantic similarity between words will be utilized in using the KNN algorithm for the index optimization. The word collocations in the text is basis for computing the semantic similarity in the previous works and this work.

C. Word Similarity

This section is concerned with the previous works on the similarity metric between words. We need the scheme of computing a similarity between words in using the KNN algorithm for the index optimization as a word classification instance. There are two kinds of word similarity; the syntactic similarity is the similarity between words based on their spellings, and the semantic similarity is one based on their meanings. Because the index optimization is the task based on meanings, the semantic similarity becomes the focus of this research. This section is intended to survey the previous works which provide the schemes of computing the semantic similarity between words.

Let us survey on the previous works where words are encoded into tables and the similarity between tables is computed. In 2006, words are encoded so and the similarity between tables is defined for using the KNN algorithm for the topic based word classification [9]. The similarity metric between tables which represent words was applied in using the AHC algorithm for the semantic word clustering [10]. It was also applied in using the KNN algorithm as the approach to the keyword extraction which is derived from the topic based word classification [11]. In the above literatures, the semantic similarity between words is computed by representing them into tables.

Let us survey on the previous works where words are encoded into string vectors, and the similarity between string vectors is defined. In 2016, words are encoded into string vectors, and the similarity between string vectors is defined as the operation, for modifying the KNN algorithm as the approach to the word classification [12]. The KNN algorithm where the semantic similarity between string vectors was adopted as the word similarity was applied to the keyword extraction as another word classification instance [13]. The semantic similarity between string vectors was applied for modifying the AHC algorithm as the approach to the semantic word clustering [14]. The semantic similarity between words is computed by encoding words into string vectors in the above literatures.

Let us explore the previous works where words are encoded into graphs and the similarity between them is defined. As the approach to the topic based word classification, the KNN algorithm was modified by defining the similarity between graphs as the semantic word similarity [15]. The modified KNN algorithm was applied to the keyword extraction as the task which is derived from the word classification [16]. The AHC algorithm is modified by defining the similarity metric between graphs as the approach to the word clustering [17]. In the above literatures, the semantic similarity between words is computed by encoding them into graphs.

Let us mention some relevance of the previous works which are explored above to this research. In using the KNN for the index optimization, we need to define the semantic similarity metric between words. They are represented into the alternative structured forms to the numerical vectors for computing the semantic word similarity. In this research, the semantic similarity metric between words is proposed by encoding words into numerical vectors. We consider both the feature similarity and the feature value similarity for computing the similarity more reliably.

D. Text Similarity

This section is concerned with the previous works which deals with the schemes of computing the similarity between texts. It is necessary to survey the previous works on the similarity metric between texts for computing it as the feature similarity. In this research, words are encoded into numerical vectors and the features are given as texts. In the previous works, the computation of the similarities among texts is intended to modify the KNN algorithm as the approach to the text mining tasks. This section is intended to survey the previous works which deals with the similarity metrics between texts.

Let us survey the previous works where texts are encoded into tables and the similarity between them is computed. After encoding texts into tables, the similarity metric between tables is defined for modifying the KNN algorithm as the approach to the text classification [18]. The KNN algorithm which uses the similarity metric between tables was applied to the text summarization which was derived from the text classification [19]. The AHC algorithm was modified into the version where the similarity between tables is computed by encoding texts into tables [20]. In the above literatures, the similarity between texts is computed by encoding texts into tables.

Let us survey on the previous works on another scheme of computing the similarity between texts. They are encoded into string vectors and the similarity metric between them is defined for modifying the KNN algorithm as the approach to the text classification [21]. The modified version of the KNN algorithm where the similarity between string vectors representing texts is computed was applied to the text summarization which is mapped into the classification

task [22]. The AHC algorithm where the text similarities are computed by encoding texts into string vectors was applied to the text clustering [23]. The similarities among texts are computed by encoding texts into string vectors in the above literatures.

Let us explore the previous works on the third scheme of computing the text similarities. In using the KNN algorithm for the text classification, texts were encoded into graphs, and the similarity metric between them was defined [29]. The KNN version where the similarities among graphs are computed was applied to the text summarization [30]. The AHC algorithm was modified as well as the KNN algorithm as the approach to the text clustering [31]. In the above literatures, the text similarities are computed by encoding texts into graphs.

Let us mention some relevancies of this research to the previous works which are surveyed above. The texts are used as the attributes in encoding the words into numerical vectors. The text similarities were used for applying the KNN algorithm to the text mining tasks in the above literatures, whereas in this research, the text similarities are used as the feature similarities. In this research, the similarity between texts is computed based on the rate of the shared words; the more shared words, the higher similarity. We will consider the adoption of the schemes which were mentioned in the previous works for the next research.

III. Proposed System

This section is concerned with modifying the KNN (K Nearest Neighbor) algorithm into the version which considers the similarities among features as well as feature values and its application to the index optimization, and it consists of the four sections. In Section III-A, we describe the process of encoding words into numerical vectors. In Section III-B, we do formally the proposed scheme of computing the similarity between two numerical vectors. In Section III-C, we mention the proposed version of KNN algorithm which considers the similarity among features. In Section III-D, we explain the system architecture and execution process of the proposed system.

A. Word Encoding

This section is concerned with the process of encoding a word into a numerical vector. A corpus is prepared and the texts in it are given as the feature candidates. Some are selected among the feature candidates as the features by the text size. For each feature which is given a text, the TF-IDF (Term Frequency and Inverse Document Frequency) is computed as the relationship between a text and a word. This section is intended to describe the process of mapping a word into a numerical vector, referring the corpus.

The process of extracting texts from a corpus as the feature candidates is illustrated in Figure 1. The K words as encoding targets and the corpus as the text collection are prepared. For each word, it is linked to the list of texts which include itself. The text sets which are linked the K words are unioned into a single set of texts, and the texts in the set are given as the feature candidates. It is necessary to define the criteria for selecting some among the feature candidates as the features.

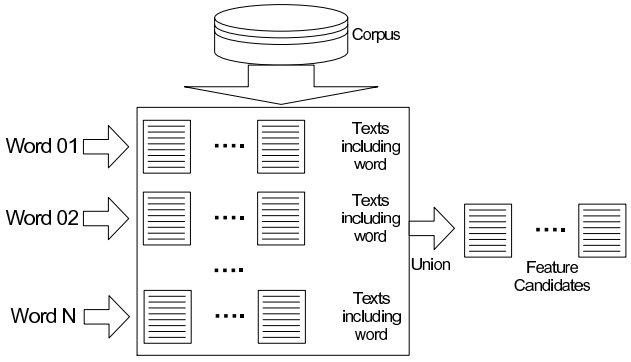

Figure 1. Feature Candidate Generation

The process of selecting the d texts as the features among the N texts which are given as the feature candidates is illustrated in Figure 2. The texts are extracted as the feature candidates by the process which is illustrated in Figure 1. The text size or the sum of the TF-IDF weights of words excluding the stop words is the criteria for selecting texts from the corpus. The texts are ranked by their sizes and their weight sum and the d texts with their highest values are selected. The d texts are used as the attributes for encoding the words into the numerical vectors.

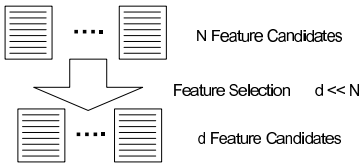

Figure 2. Feature Selection

The process of assigning feature values to the features for each word is illustrated in Figure 3. A word is given as the target to represent into a numerical vector and the texts which are selected by the above process is given as the attributes or the features. The TF-IDF weight is computed for each word and each text by equation which is presented in Figure 3. The frequency or the binary value which indicates the presence or the absence in the text may be used as the alternatives to the TF-IDF weight. In this research, the TF-IDF weighting scheme is adopted for implementing the word encoding system.

Let us make some remarks on the process of encoding words into numerical vectors. The corpus and the K words which are given as the sample words are prepared as the initial status. The words are converted into the numerical

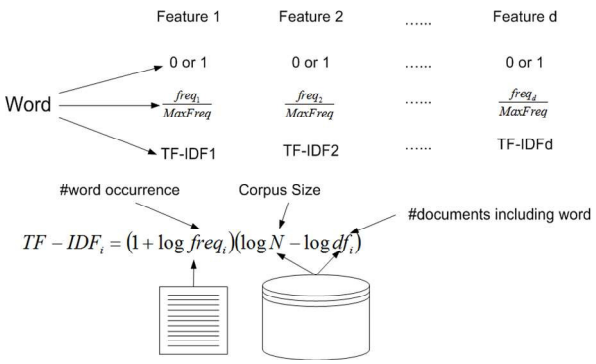

Figure 3. Feature Value Assignment

vectors whose features are given as the texts. Each element of the numerical vector which represents a word indicates the relationship with the text. The similarities among features which are given as texts are considered for computing the similarity between numerical vectors in the next section.

B. Similarity Metric

This section is concerned with the proposed similarity metric between two words. The cosine similarity was used as the traditional similarity metric in the case of encoding words into numerical vector by the process which is described in Section III-A. The semantic similarity between words considering the feature similarities is proposed, in order to avoid the poor discriminations among sparse vectors. The feature similarity means the similarity between features which are given as texts and the feature value similarity means the similarity between two vectors based on their elements. This section is intended to describe the process of computing the similarity metric between two words.

The frame of computing the semantic similarity between words is illustrated in Figure 4. The two vectors, $\mathbf{x}$ and $\mathbf{y}$ represent words, and the feature similarity and the feature value similarity are considered in computing the similarity between them. The feature similarity is the similar among the features, $f_1, f_2, \ldots, f_d$ and the feature value similarity is the similarity between elements in the two vectors, $x_1, x_2, \ldots, x_d$ and $y_1, y_2, \ldots, y_d$. The similarity between vectors is computed by combining the both kinds of similarities as shown in Figure 4. One to one computation happens in the cosine similarity as the traditional one, whereas all possible pair computation of elements of numerical vectors is applied to the proposed similarity metric.

The similarity matrix of the features which are given as texts is illustrated in Figure 5. It is assumed that the texts, $text_1, text_2, \ldots, text_d$ are selected as the features by the process which is described in Section 3.1. A text is indexed into a list of words; T_i and T_j are the sets of words which are indexed respectively from the texts, $text_i$ and $text_j$. The similarity between two texts is computed by equation (1),

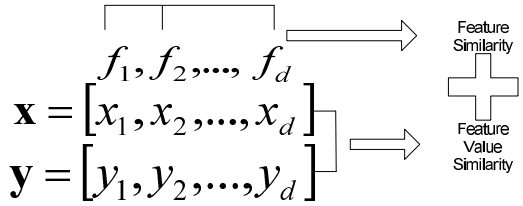

Figure 4. Word Similarity Computation Frame

$$sim(text_i, text_j) = \frac{2 \times |T_i \cap T_j|}{|T_i| + |T_j|} \tag{1}$$

The similarity between two texts in equation (1) is the feature similarity based on the rate of shared words to the words in either of the two texts.

$$\begin{array}{c} \\ text_1 \\ text_2 \\ \cdots \\ text_d \end{array} \begin{array}{c} \begin{array}{cccc} text_1 & text_2 & \cdots & text_d \end{array} \\ \begin{bmatrix} s_{11} & s_{12} & \ldots & s_{1d} \\ s_{21} & s_{22} & \cdots & s_{2d} \\ \ldots & \ldots & \ldots & \ldots \\ s_{d1} & s_{d2} & \ldots & s_{dd} \end{bmatrix} \end{array} \quad s_{ij} = sim(text_i, text_j)$$

Figure 5. $d \times d$ Similarity Matrix

Let us derive the equation for computing the proposed similarity metric with the feature similarity. Equation (1) is simplified into equation (2),

$$sim(text_i, text_j) = sim(f_i, f_j) = f_{ij} \tag{2}$$

Two words are encoded into two d dimensional numerical vectors, $\mathbf{x} = [x_1, x_2, ..., x_d]$ and $\mathbf{y} = [y_1, y_2, ..., y_d]$. The similarity between the two numerical vectors is computed by equation (3),

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{d} \sum_{j=1}^{d} f_{ij} \cdot x_i \cdot y_j}{d\|\mathbf{x}\|\|\mathbf{y}\|} \tag{3}$$

where $\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{d} x_i^2}$ and $\|\mathbf{y}\| = \sqrt{\sum_{i=1}^{d} y_i^2}$. It takes the quadratic complexity to the d dimensional vector for computing the similarity by equation (3).

Let us make some remarks on the proposed similarity metric between two numerical vectors which represent words. The numerical vectors which represent words or texts tend to be sparse. Zero values are very frequent in computing the similarity between two sparse vectors by the cosine similarity. The similarity metric which is expressed in equation (3) prevents from generating zero values. The higher computation complexity in computing the similarity is the payment for getting the solution.

C. Proposed Version of KNN

This section is concerned with the proposed KNN version where the similarities of a novice item with the training examples are computed, considering the feature similarity

and the feature value similarity. In Section III-B, we already described the process of computing the proposed similarity metric between two numerical vectors which represent words. The proposed similarity metric is used for computing the similarities of a novice item with the training examples for taking its nearest neighbors as the modification point of the KNN algorithm. The sample words are encoded into numerical vectors, in advance, and the label of the novice item is decided by voting the labels of the nearest neighbors. This section is intended to describe in detail the modified KNN algorithm which is used as the approach to the index optimization.

Figure 6 illustrated the process of computing the similarities of a novice item with the training examples. The sample words are encoded into numerical vectors, before, and the novice word is encoded into a numerical vector, now. The similarities of the numerical vector representing the novice item with ones representing the sample words is computed by the similarity metric which is described in Section III-B. The similarities of the novice item are assigned as normalized values to the training examples. It takes the quadratic complexity to the numerical vector dimension for computing the similarity between the novice item and a training example.

Figure 7 illustrates the process of selecting the nearest neighbors by their similarities with the novice item. In the process which is presented in Figure 6, the similarities of the notice item with the training examples are computed by encoding them into numerical vectors. The training examples are ranked in the descending order of their similarities, and ones with their higher similarities are taken as the nearest neighbors. In the KNN variant, called RNN (Radius Nearest Neighbor), the training examples with their higher similarity than the threshold are selected, instead of ranking them. In implementing the process of selecting the nearest neighbors, we should adopt the advanced sorting algorithm, such as the quick sorting and the heap sorting.

Figure 8 illustrated the process of deciding the label of the novice item by voting ones of the nearest neighbors. The nearest neighbors are selected from the training examples by the process which is presented in Figure 7. From the nearest neighbors, their labels are collected, and the label with the majority of them is selected as one of the novice one. Here, the nearest neighbors are treated equally for deciding the label, but in the KNN variant, the discriminations may be put among them based on their similarities. The various KNN variants are derived by modifying the scheme of selecting nearest neighbors and one of voting their labels.

Let us make some remarks on the proposed KNN which is described in this section. The sample words and the novice word are encoded into numerical vectors. The similarity metric which is described in Section III-B is used for computing the similarity of each novice item with the training examples. The scheme of selecting nearest neighbors by ranking the training examples is adopted in this research. The unweighted voting of the labels is adopted for deciding the label of the novice item.

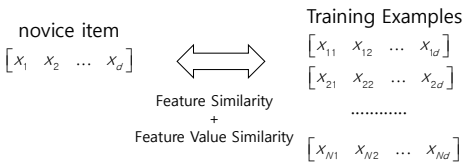

Figure 6. The Process of computing Similarities of a Novice Item with the Training Examples

D. System Architecture

This section is concerned with the architecture and the execution flow of the proposed system. The proposed version of the KNN algorithm as the approach to implementing the system was already described in Section 3.3. The sample words, the system architecture, and the execution flow, for implementing the proposed system are presented, respectively, in Figure 9, 10, and 11. The only brief design of the system is covered in this study, and the source code in Java or Python which implements the system will be presented in the next work. This section is intended to describe the proposed system where the proposed KNN algorithm is adopted as the approach.

Collecting the sample words for implementing the index optimization system is illustrated in Figure 9. The index optimization is interpreted into a domain dependent classification; even same word may be classified into the expansion in one domain and done into the removal in another domain

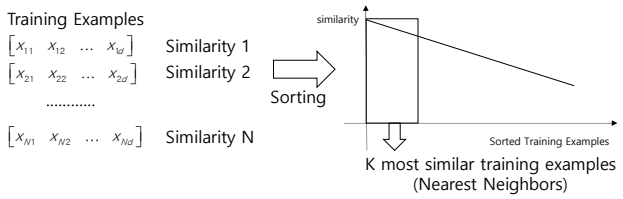

Figure 7. The Process of selecting Nearest Neighbors

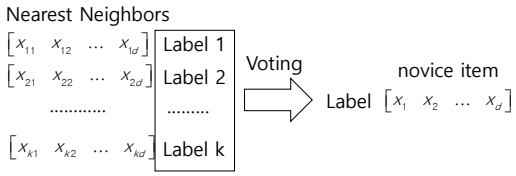

Figure 8. The Process of voting Labels of Nearest Neighbors

in this classification type. The independent classification task where each word is classified into one of the three categories is given for each domain. Several domains are defined and in each domain, words which are exclusively labeled with one of the three categories are collected. It is required to present the domain as a tag for performing the index optimization.

The entire system architecture of the index optimization system which is implemented in this research is illustrated in Figure 10. The words which are labeled with one of the three categories are collected sample words, and they are encoded into numerical vectors by the encoder module. The $d \times d$ similarity matrix which is used for computing the feature similarity is constructed and the similarities of each word which is indexed from a text by the text indexer are computed considering both the feature similarity and the feature value similarity, in the similarity computation module. In the voting module, the nearest neighbors are selected and the category is decided, for each word in the text. The ranking module is nested in the similarity computation module for selecting the nearest neighbors.

The execution process of the index optimization system is illustrated in Figure 11. Texts are collected within the domain and sample words are prepared as numerical vectors. A text is indexed into a list of words and they are encoded into numerical vectors. Each word is classified into one of the three categories; the word group is divided into the three subgroups: the expansion group, the inclusion group, and the removal group. The words in the removal group are excluded, the words in the inclusion group are indexed in the index, and associated words are added from external sources to the words in the expansion group.

Let us make some remarks on the index optimization system which are presented in Figure 10 and 11, respectively as the system architecture and the execution flow. We need two collections for implementing the system: the collection of sample words and the collection of texts within each domain. The system is implemented as multiple independent subsystems, domain by domain, and it is required to present the domain as a tag for performing the index optimization to a text. The proposed system is described staying in the design step; we will present the source code in Java or Python in the next research. We need to implement the additional module for the index expansion to the words in the expansion group by defining its detail algorithm.

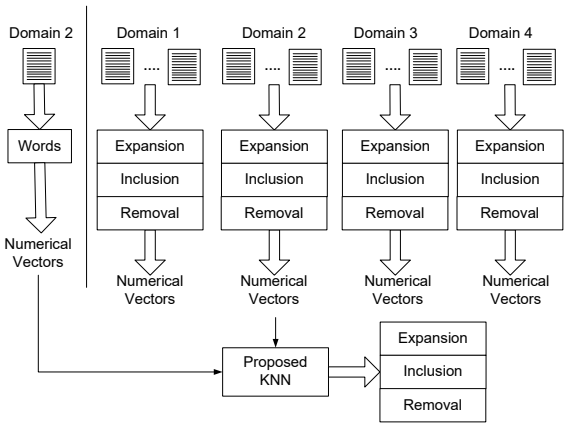

Figure 9. The Process of collecting Sample Words for Index Optimization

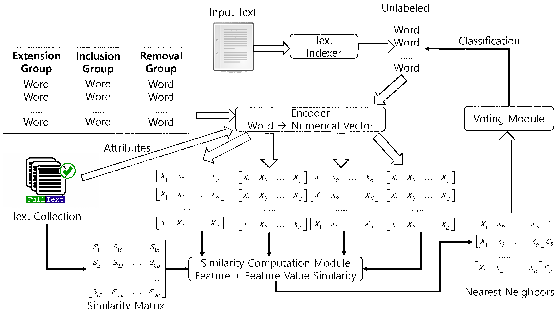

Figure 10. System Architecture of the Index Optimization System

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 index optimization on the collection, NewsPage.com. In Section IV-B and IV-C, we mention the results from comparing the

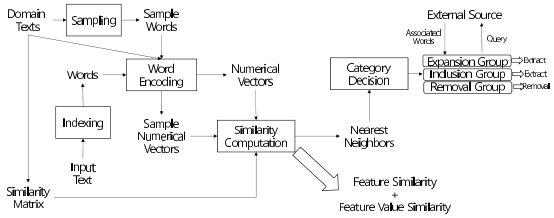

Figure 11. Execution Process of the Index Optimization System

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 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 [**?**]. 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 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 12, 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.

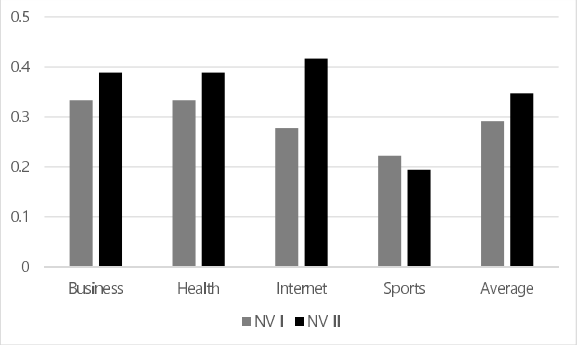

Figure 12. Results from Index Optimization in Text Collection: NewsPage.com

Let us make the discussions on the results from doing the index optimization, using the both versions of KNN algorithm, as shown in Figure 12. The accuracy which is the performance measure of this classification task is in the range between 0.33 and 0.41. The proposed version of KNN algorithm works better in the three domains: Business, Health, and Internet. However, it loses in the domain, Sports. From this set of experiments, we conclude that 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 '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.

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)

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 vectors with the input size fixed to 50. For each test example, we compute its similarities with the 300 training 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 13, 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 13 has the identical frame of presenting the results to those of Figure 12. 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.

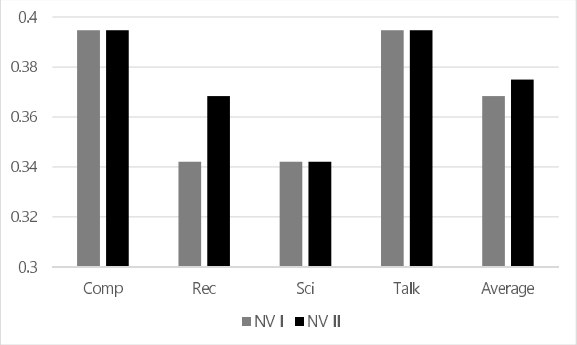

Figure 13. Results from Index Optimization in Text Collection: 20NewsGroup I

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 0.40. The proposed version shows the better performance in the three of the four domains; it does its outstandingly better performance in the domain, sci. However, it shows its competitive performances in the domain, talk. From this set of experiments, the proposed version wins over 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 14, 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'.

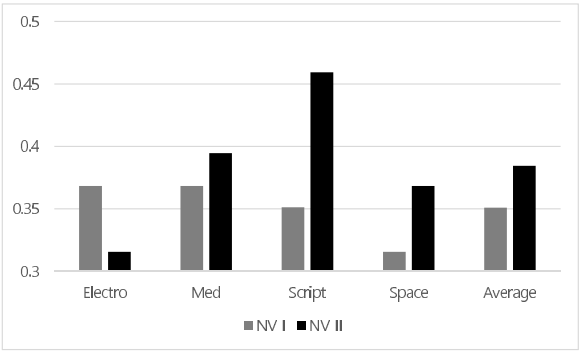

Figure 14. Results from Index Optimization in Text Collection: 20NewsGroup II

Let us discuss on the results from doing the index optimization on the specific version of 20NewsGroups, as shown in Figure 14. The accuracies of both versions of KNN algorithm range between 0.31 and 0.46. The proposed

version shows its better results in three of the four domains. However, it is leaded in the domain, 'electro'. 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.

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 0.31 and 0.41. 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. We need to validate the proposed approach in specific domains such as medicine, engineering, and economics, as well as in generic domains such as ones of news articles. We may consider the computation of similarities among some main features rather than among all features for reducing the computation time. We try to modify other machine learning algorithms such as Na?ve Bayes, Perceptrons, and SVM (Support Vector Machine) based on both kinds of similarities. By adopting the proposed approach, we may implement the index optimization system as a real program.

References

[1] T. Jo, "Modified Version of SVM for Text Categorization", 52-60, International Journal of Fuzzy Logic and Intelligent Systems, Vol 8, No1, 2008.

[2] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", 31-43, Journal of Network Technology, Vol 1, No 1, 2010.

[3] T. Jo, "KNN based Word Categorization considering Feature Similarities", 343-346, The Proceedings of 17th International Conference on Artificial Intelligence, 2015.

[4] T. Jo, "Keyword Extraction by KNN considering Feature Similarities", 64-68, The Proceedings of The 2nd International Conference on Advances in Big Data Analysis, 2015.

[5] T. Jo, "Index Optimization with KNN considering Similarities among Features", 120-124, The Proceedings of 14th International Conference on Advances in Information and Knowledge Engineering, 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, "Extracting Keywords by Graph based KNN", 96-101, The Proceedings of 12th International Conference on Multimedia Information Technology and Applications, 2016.

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

[9] T. Jo, "Table based KNN for Categorizing Words", 696-700, The Proceedings of 18th International Conference on Advanced Communication Technology, 2016.

[10] T. Jo, "Table based AHC Algorithm for Clustering Words", 574-579, The Proceedings of 18th International Conference on Advanced Communication Technology, 2016.

[11] T. Jo, "Table based KNN for Extracting Keywords", 812-817, The Proceedings of 18th International Conference on Advanced Communication Technology, 2016.

[12] T. Jo, "Encoding Words into String Vectors for Word Categorization", 271-276, The Proceedings of 18th International Conference on Artificial Intelligence, 2016.

[13] T. Jo, "String Vector based AHC as Approach to Word Clustering", 133-138, The Proceedings of 12th International Conference on Data Mining, 2016.

[14] T. Jo, "Using String Vector based KNN for Keyword Extraction", 27-32, The Proceedings of 15th International Conference on Advances in Information and Knowledge Engineering, 2016.

[15] T. Jo, "Graph based KNN for Content based Word Classification", 24-29, The Proceedings of 12th International Conference on Multimedia Information Technology and Applications, 2016.

[16] 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.

[17] T. Jo, "Extracting Keywords by Graph based KNN", 96-101, The Proceedings of 12th International Conference on Multimedia Information Technology and Applications, 2016.

[18] T. Jo, "Table based KNN for Article Classification", 271-276, The Proceedings of 19th International Conference on Artificial Intelligence, 2017.

[19] T. Jo, "Table based KNN for Text Summarization", 31-36, The Proceedings of 4th International Conference on Advances in Big Data Analysis, 2017.

[20] T. Jo, "Table based AHC for Text Clustering", 133-138, The Proceedings of 13th International Conference on Data Mining, 2017.

[21] T. Jo, "String Vector based KNN for Text Categorization", 458-462, The Proceedings of 18th International Conference on Advanced Communication Technology, 2017.

[22] T. Jo, "K Nearest Neighbor for Text Summarization using Feature Similarity", DOI: 10.1109/ICCCCEE.2017.7866705, Proceedings of International Conference on Communication, Control, Computing and Electronics Engineering, 2017.

[23] T. Jo, “String Vector based AHC for Text Clustering”, 673-677, The Proceedings of 18th International Conference on Advanced Communication Technology, 2017.

[24] T. Jo, “Word Classification in Domain on Current Affairs by Feature Similarity based K Nearest Neighbor”, 348-351, The Proceedings of International Conference on Artificial Intelligence, 2018.

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

[26] 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.

[27] 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.

[28] 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.

[29] T. Jo, “Graph based KNN for Text Categorization”, 260-264, The Proceedings of IEEE 18th International Conference on Advanced Communication Technology, 2018.

[30] T. Jo, “Graph based KNN for Text Summarization”, 438-442, The Proceedings of IEEE 18th International Conference on Advanced Communication Technology, 2018.

[31] 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.