# String Vector based KNN for Text Categorization

Taeho Jo
*Department of Computer and Information Communication Engineering*
*Hongik University*
*Sejong, South Korea*
*tjo018@hongik.ac.kr*

***Abstract*—This research proposes the string vector based version of the KNN as the approach to the text categorization. Traditionally, texts should be encoded into numerical vectors for using the traditional version of KNN, and encoding so leads to the three main problems: huge dimensionality, sparse distribution, and poor transparency. In order to solve the problems, in this research, texts are encoded into string vectors, instead of numerical vectors, the similarity measure between string vectors is defined, and the KNN is modified into the version where string vector is given its input. As the benefits from this research, we may expect the better performance, more compact representation of each text, and better transparency. The goal of this research is to improve the text categorization performance by solving them.**

***Keywords*-Text Categorization, Semantic Similarity Similarity, String Vector, String Vector based KNN**

## I. Introduction

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

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

Let us consider what we propose in this research as solutions to the above problems. Texts are encoded into string vectors as their structured forms, instead of numerical vectors. The semantic similarity measure between two string vectors is defined as the operation which corresponds to the cosine similarity between two numerical vectors. Using the similarity measure, we modify the KNN (K Nearest Neighbor) into the version where a string vector is given as an input vector. The modified version is applied as the approach to the text categorization.

Let us consider some benefits expected from this research. We may expect the better performance than that of the traditional version of the KNN, by avoiding problems in encoding texts into numerical vectors. We may expect the proposed system to be more transparent because string vectors representing texts are more symbolic than numerical vectors; it is possible to guess the text contents only from their surrogates. It is expected to be more efficient to represent texts into string vectors than into numerical vectors; each string vector which represents a text has only tens of dimensions, whereas each numerical vector representing it has at least several hundreds of dimensions. However, we need to define more advanced operations on string vectors other than the semantic similarity.

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

## II. Previous Works

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

words, this section is intended to explore previous works on solutions to the problems.

Let us mention the popularity of encoding texts into numerical vectors, and the proposal and the application of string kernels as the solution to the above problems. In 2002, Sebastiani presented the numerical vectors are the standard representations of texts in applying the machine learning algorithms to the text classifications [6]. In 2002, Lodhi et al. proposed the string kernel as a kernel function of raw texts in using the SVM (Support Vector Machine) to the text classification [7]. In 2004, Lesile et al. used the version of SVM which proposed by Lodhi et al. to the protein classification [8]. In 2004, Kate and Mooney used also the SVM version for classifying sentences by their meanings [9].

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

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

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

## III. Proposed Approach

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

### *A. Text Encoding*

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

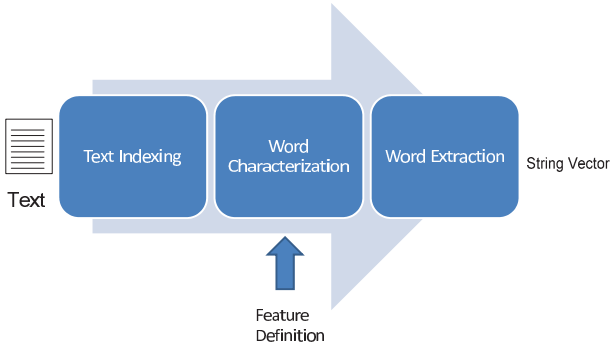


Figure 1. The Process of Text Encoding

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

We need to define the relationships between a word and a text as the features of string vectors, and mention the three types of them. The first type is statistical properties of words in a text such as the highest frequent word and the highest TF-IDF (Term Frequency-Inverse Term Frequency) weighted one. The grammatical properties of a word such as subjective noun, objective noun, and verb may be considered as another feature type. Posting properties of a word which

indicates its position in the given text, such as the first word in the text, the last of in the text, and the first word in the last paragraph, may be regarded as a feature type. In this research, we define the ten features of string vectors as follows:

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

Let us explain the process of encoding a text into a string, once the above features are defined. A text is indexed into a list of words, their frequencies, and their TF-IDF weights, and it is partitioned into a list of paragraphs. Corresponding to the above features, words are extracted as elements of the string vector. As the given text representation, the ten dimensional string vector which consists of the above feature values is constructed. The similarity matrix is required for performing the operation on string vectors, and is described in Section III-B1.

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

### *B. String Vectors*

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

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

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

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

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

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

and if two words have no shared texts, $T_i \cap T_j = \emptyset$ the similarity becomes 0.0 as follows:

$$sim(t_i, t_j) = \frac{2|T_i \cap T_j|}{|T_i| + |T_j|} = 0.0$$

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

From the text collection, we build $N \times N$ square matrix as follows:

$$S = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1d} \\ s_{21} & s_{22} & \cdots & s_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ s_{d1} & s_{d2} & \cdots & s_{dd} \end{pmatrix}.$$

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

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

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

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

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

$$= sim(t_j, t_i) = s_{ji}$$

ISBN 978-89-968650-9-4

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

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

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

The string vector is defined as a finite ordered set of strings as follows:

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

An element in the vector, $str_i$ indicates a text identifier which corresponds to its attribute. The number of elements of the string vector, $\mathbf{str}$ is called its dimension. In order to perform the operation on string vectors, we need to define the similarity matrix which was described in Section III-B1, in advance. Therefore, a string vector consists of strings, while a numerical vector does of numerical values.

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

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

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

where each element, $d_{1i}$ and $d_{21i}$ indicates a text identifier. The operation is defined as equation (2) as follows:

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

The similarity matrix was constructed by the scheme which is described in Section III-B1, and the $sim(d_{1i}, d_{2i})$ is computed by looking up it in the similarity matrix. Instead of building the similarity matrix, we may compute the similarity, interactively.

The semantic similarity measure between string vectors may be characterized mathematically. The commutative law applies as follows:

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

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

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

$$\text{if } \mathbf{str}_1 = \mathbf{str}_2 \text{with } \forall_i sim(d_{1i}, d_{2i}) = 1.0$$

$$\text{then } sim(\mathbf{str}_1, \mathbf{str}_2) = \frac{1}{d}\sum_{i=1}^{d} sim(d_{1i}, d_{2i}) = \frac{d}{d} = 1.0$$

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

$$\text{if } sim(\mathbf{str}_1, \mathbf{str}_2) = 0.0 \text{ and } sim(\mathbf{str}_2, \mathbf{str}_3) = 0.0$$

$$\text{then, not always } sim(\mathbf{str}_1, \mathbf{str}_3) = 0.0$$

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

### C. Proposed Version of KNN

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

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

ISBN 978-89-968650-9-4

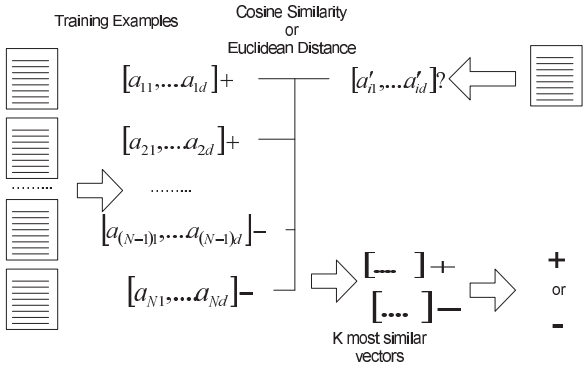


Figure 2. The Traditional Version of KNN

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

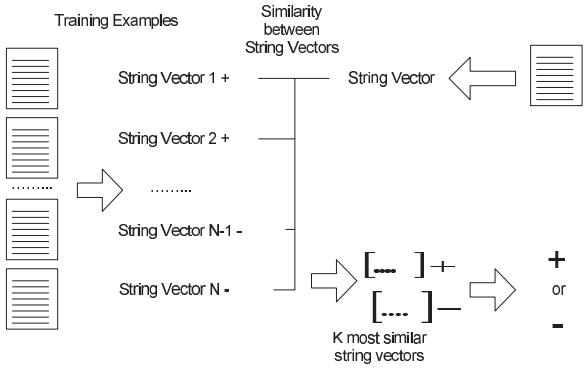


Figure 3. The Proposed Version of KNN

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

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

## IV. Conclusion

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

## V. Acknowledgement

This work was supported by 2016 Hongik University Research Fund.

## References

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

[2] T. Jo and D. Cho, “Index Based Approach for Text Categorization”, 127-132, International Journal of Mathematics and Computers in Simulation, No 2, 2008.

[3] T. Jo, “Table based Matching Algorithm for Soft Categorization of News Articles in Reuter 21578”, pp875-882, Journal of Korea Multimedia Society, No 11, 2008.

[4] T. Jo, “Topic Spotting to News Articles in 20NewsGroups with NTC, , Lecture Notes in Information Technology”, pp50-56, No 7, 2011.

[5] T. Jo, “Definition of String Vector based Operations for Training NTSO using Inverted Index”, pp57-63, Lecture Notes in Information Technology, No 7, 2011.

[6] F. Sebastiani, “Machine Learning in Automated Text Categorization”, pp1-47, ACM Computing Survey, Vol 34, No 1, 2002.

[7] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, “Text Classification with String Kernels”, pp419-444, Journal of Machine Learning Research, Vol 2, No 2, 2002.

[8] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, “Mismatch String Kernels for Discriminative Protein Classification”, pp467-476, Bioinformatics, Vol 20, No 4, 2004.

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

[10] T. Jo, “Single Pass Algorithm for Text Clustering by Encoding Documents into Tables”, pp1749-1757, Journal of Korea Multimedia Society, Vol 11, No 12, 2008.

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

[12] T. Jo, “Normalized Table Matching Algorithm as Approach to Text Categorization”, pp839-849, Soft Computing, Vol 19, No 4, 2015.

[13] T. Jo, “Inverted Index based Modified Version of K-Means Algorithm for Text Clustering”, pp67-76, Journal of Information Processing Systems, Vol 4, No 2, 2008.

[14] T. Jo, “Representationof Texts into String Vectors for Text Categorization”, pp110-127, Journal of Computing Science and Engineering, Vol 4, No 2, 2010.

[15] T. Jo, “NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering”, pp31-43, Journal of Network Technology, Vol 1, No 1, 2010.

[16] T. Jo, “NTC (Neural Text Categorizer): Neural Network for Text Categorization”, pp83-96, International Journal of Information Studies, Vol 2, No 2, 2010.

Taeho Jo (M'97-AM'12) This author became a Member (M) of IEEE in 1997, and an Associate Member (AM) in 2012. He was born in 1970, South Korea. He received his Bachelor degree from Korea University in 1994, his Master degree from Pohang University of Science and Technology in 1997, and his PhD degree from University of Ottawa in 2006. His research area spans mainly over text mining, neural networks, machine learning, and information retrieval. He has the four year experience of working for industrial organizations and ten year experience of working for academic ones. So his research is characterized as the connection from fundamental researches for creating theories and to applied ones for developing products, by his experience of working for both sides.