

Graph based KNN for Text Summarization

Taeho Jo
 School of Games
 Hongik University
 Sejong, South Korea
 tjo018@hongik.ac.kr

Abstract—In this research, we propose that the graph based KNN (K Nearest Neighbor) should be applied to the text summarization tasks. The text summarization tasks may be interpreted into the binary classification tasks, and sentences or paragraphs may be encoded into graphs as well as articles. In this research, we encode sentences or paragraphs into graphs under the view of the text summarization tasks into the task where they are classified into essential parts, or not, modify the KNN into the graph based version where each graph is given as its input data, and apply it to the classification task which is mapped from the text summarization. As the benefits from this research, we expect the more compact, graphical, and symbolic representations of sentences or paragraphs and the improved text summarization performance. Therefore, the goal of this research is to implement the text summarization system with its improved performance and representations of data items.

Keywords—Text Summarization, Graph Similarity, Graph based KNN

I. INTRODUCTION

Text summarization refers to the process of selecting essential parts in each text. Each text is partitioned into sentences or paragraphs by punctuation mark or carriage return, respectively and the task is viewed as the binary classification of partitions into the essence partition or remaining. Each sentence or paragraph is encoded into its own structured form and sample sentences or paragraphs which are labeled with the essence part or the remaining are prepared. By learning sample ones, we construct the classification capacity, classify novice sentences into one of the two categories, and present the sentences or paragraphs which are labeled with the essences as the summary. We need to distinguish the summarization by system from one by human, in that text summarization by human is the process of rewriting a text into its brief version.

Let us mention some points which become the motivations for doing this research. The problems such as huge dimensionality and sparse distribution are caused by encoding texts into numerical vectors in using the traditional machine learning algorithms as the approaches[1]. Graphs are used as the popular representations of knowledge or information in the name of ontology or word net [2][3]. In recent works, various types of algorithms which manipulate

graphs are developed correspondingly [3]. Therefore, by the motivations, in this research, we encode texts into graphs and modify the machine learning algorithm into its graph based version.

Let us mention what we propose in this research as some agenda. Instead of numerical vectors, we encode texts into graphs each of which consists of vertices indicating words and edges indicating their semantic relations. We define the similarity measure between graphs which have different vertices and edges as that between texts. We modify the KNN (K Nearest Neighbors) into its graph based version using the similarity measure, and apply it to the text summarization which is mapped into the binary classification task. The graphs which indicate texts are undirected weighted graphs and are represented into adjacency matrices in the implementation level.

Let us consider some benefits which are expected from this research. From using the proposed KNN version, we expect the better text summarization performance than from using the traditional version. By encoding texts into more graphical forms, we expect more transparency where we are able to guess the text contents only by their representations. We expect also more compactness in encoding texts into graphs than in doing them into numerical vectors; it causes the more efficient text processing. Hence, the goal of this research is to implement the text summarization system which satisfying the benefits.

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 [4]. 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 [5]. In 2004, Lesile et al. used the version of SVM which proposed by Lodhi et al. to the protein classification [6]. In 2004, Kate and Mooney used also the SVM version for classifying sentences by their meanings [7].

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 [8]. In 2008, Jo applied also his proposed approach to the text clustering, as well as the text categorization [12]. In 2011, Jo described as the technique of automatic text classification in his patent document [10]. In 2015, Jo improved the table matching algorithm into its more stable version [11].

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 [12]. 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 [13]. In 2010, Jo proposed the unsupervised neural networks, called Neural Text Self Organizer, which receives the string vector as its input data [14]. 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 [15].

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, we propose that texts should be encoded into graphs, and modify the KNN into the version which processes graphs instead of numerical vectors, as the approach to the text summarization.

III. PROPOSED APPROACH

This section is concerned with encoding words into graphs, modifying the KNN (K Nearest Neighbor) into the graph based version and applying it to the text summarization, and consists of the three sections. In section III-A, we deal with the process of encoding texts into graphs. In section III-B, we describe formally the process of computing the similarity between to graphs. In section III-C, we do the graph vector based KNN version as the approach to the text summarization. In section III-D, we focus on the process of applying the KNN to the given task with viewing it into a classification task.

A. Text Encoding

This section is concerned with the process of encoding a text into a graph as illustrated in Figure 1. The graph is defined into the two sets: vertex set and edge set. Vertices and edges correspond to words and their semantic relationships, respectively. In this research, the graph which represents a text is given as a weighted and undirected one. Therefore, in this section, we will describe in detail the process of representing a text into graph.

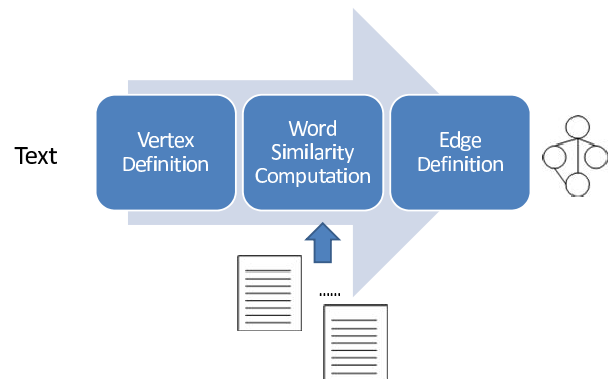


Figure 1. The Process of Encoding Text into Graph

Before encoding a text into its own graph, we need to construct an index list where each text is linked to a list of words. We generate directly a list of texts from a corpus. Each text is indexed into a list of words and it is associated with its own list of its included words. Each word has its weights and posting information as its relationship with a text. A text is indexed basically with the three steps: tokenization, stemming, and stop-word removal.

We need to define vertices which are followed by edges in encoding a text into a graph. The vertices correspond to list of words which are included in the text. Some of them are selected by their weights as vertices; they are notated as follows:

$$D(v) = \{v_{i1}, v_{i2}, \dots, v_{im}\}$$

We consider the ranked selection where a fixed number of texts is selected by ranking them and the score based one

where the texts whose weights are greater than or equal to a threshold are selected as the selection schemes. A set of vertices which indicate words is extracted through indexing.

We need to define the set of edges as well as that of vertices for representing a word into a graph. We compute similarities of all possible pairs of vertices which indicate words. We construct the similarity matrix whose entries are similarities among words from a corpus. We select word pairs whose similarities are more than the given threshold, and define the set which consists of edges as follows:

$$D(e) = \{e_{i1}, e_{i2}, \dots, e_{in}\}$$

The process of building the similarity matrix and computing the similarity between texts will be described in section III-B1.

Let us consider how to represent a graph into its structured form in the implementation level. We may mention the adjacency matrix where vertices correspond to its rows and columns and entries indicate the edge weights. We regard the linked list where vertices are given as nodes and edges are given as pointers between them as another representation of a graph. A graph is represented into a list of edges which are given as pairs of vertex identifiers and each weight is associated with its own weight. In this research, we adopt the third scheme where a graph is represented into a set of edges.

B. Graphs

This section is concerned with the operation on graphs 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 operation on graphs mathematically. Therefore, this section is intended to describe the similarity matrix and the operation on them.

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),

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

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:

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

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 $0 \leq 2|T_i \cap T_j| \leq |T_i| + |T_j|$ from equation (1). It is proved that the similarity matrix is symmetry, as follows:

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

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) *Similarity between Graphs*: This section is concerned with the scheme of computing a similarity between two graphs. Words are encoded into graphs where vertices are text identifiers and edges are the similarity between texts. We assume that each graph is a set of edges and consider the three cases for computing the similarity between graphs: both coincidence, either coincidence, and no coincidence. The similarity between graphs is computed by averaging similarities among edges and is always given as a normalized value between zero and one. Therefore, this section is intended to describe formally the process of computing the similarity between graphs.

We need to consider the similarity between two individual edges, e_i and e_j , which is notated by $sim(e_i, e_j)$, and each weighted edge consist of two nodes and its weight as follows:

$$e_i = \{v_{i1}, v_{i2}, w_i\}$$

and the edge weight is notated by $w(e_i) = w_i$. If no node is shared by two edges like $(A, B, 0.2)$ and $(C, D, 0.4)$, the similarity becomes zero. If only one node is shared by two edges like $(A, B, 0.2)$ and $(B, C, 0.4)$, the similarity becomes the product of two weights as follows:

$$sim(e_i, e_j) = w(e_i)w(e_j).$$

If both nodes are shared, the similarity becomes the average of the two weights as follows:

$$sim(e_i, e_j) = \frac{1}{2}(w(e_i) + w(e_j)).$$

It is assumed that each weight between edges is always given as normalized value between zero and one.

The two graphs, G_1 and G_2 , are expressed into the two sets:

$$G_1 = \{e_{11}, e_{12}, \dots, e_{1n}\}$$

$$G_2 = \{e_{21}, e_{22}, \dots, e_{2n}\}$$

and it is assumed that both graphs have same number of edges. All possible pairs of edges are generated from the two graphs. For each edge in one graph, its similarities with the edges in the other are computed, and the maximum among them is obtained as the similarity between an edge and a graph, by equation (2).

$$sim(e_{1i}, G_2) = \max_{k=1}^n sim(e_{1i}, e_{2k}) \quad (2)$$

The similarity between the two graphs is set by averaging over the maximum similarities of edges with the other by equation (3),

$$sim(G_1, G_2) = \frac{1}{n} \sum_{i=1}^n sim(e_{1i}, G_2) \quad (3)$$

Because the weights of edges are always given as normalized values, the similarity between graphs is always so.

Let us characterize the operation for computing the similarity between graphs, mathematically. If the two graphs, G_1 and G_2 , are identical to each other and all edges are weighted with 1.0 values, $\forall_i, w(e_{1i}) = 1.0$, and $w(e_{2i}) = 1.0$, the similarity between the two graphs becomes 1.0.

$$sim(e_{1i}, e_{2i}) = \frac{1}{2}(w(e_{1i}) + w(e_{2i})) = 1.0$$

$$sim(e_{1i}, G_2) = \max_{k=1}^n sim(e_{1i}, e_{2k}) = 1.0$$

$$sim(G_1, G_2) = \frac{1}{n} \sum_{i=1}^n sim(e_{1i}, G_2) = 1.0$$

If the two graphs, G_1 and G_2 are so with different weights, the similarity between the two graphs is the average over weights of two graphs as follows:

$$sim(e_{1i}, e_{2i}) = \frac{1}{2}(w(e_{1i}) + w(e_{2i}))$$

$$sim(e_{1i}, G_2) = \max_{k=1}^n sim(e_{1i}, e_{2k}) = \frac{1}{2}(w(e_{1i}) + w(e_{2i}))$$

$$sim(G_1, G_2) = \frac{1}{2n} \sum_{i=1}^n (w(e_{1i}) + w(e_{2i}))$$

If there is no shared edge between the two graphs, the similarity becomes zero as follows:

$$sim(e_{1i}, G_2) = \max_{k=1}^n sim(e_{1i}, e_{2k}) = 0.0$$

$$sim(G_1, G_2) = \frac{1}{n} \sum_{i=1}^n sim(e_{1i}, G_2) = 0.0$$

The similarity between the two graphs is always a normalized value between zero and one as proved from the mathematical characterization.

Let us consider the complexity of computing a similarity between graphs to the number of edges in both graphs. The number of all possible pairs becomes $\frac{n(n-1)}{2}$ to the number of edges, n . The similarities of all possible pairs are computed by the above process, $\frac{n(n-1)}{2}$ times. We derive the quadratic complexity, $O(n^2)$, for computing the similarities. Therefore, we need to optimize the number of edges for representing a word into a graph by controlling the threshold between the reliability and the computation speed.

C. Proposed Version of KNN

This section is concerned with the proposed KNN version as the approach to the text categorization. Words are encoded into graphs by the process which was described in section III-A. In this section, we attempt to the traditional KNN into the version where a graph 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.

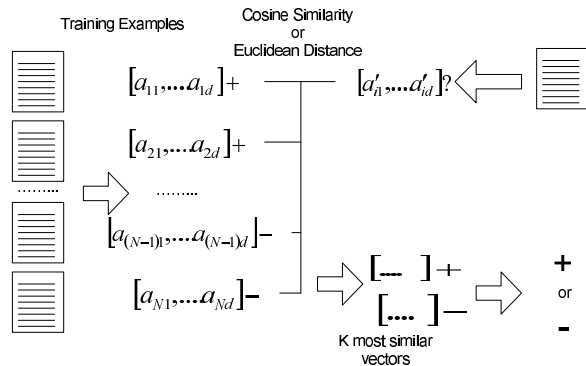


Figure 2. The Traditional Version of KNN

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 graphs by the process described in section III-A. The similarity between two graphs 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 distributions in graphs are never available inherently, the poor discriminations by sparse distributions are certainly overcome in this research.

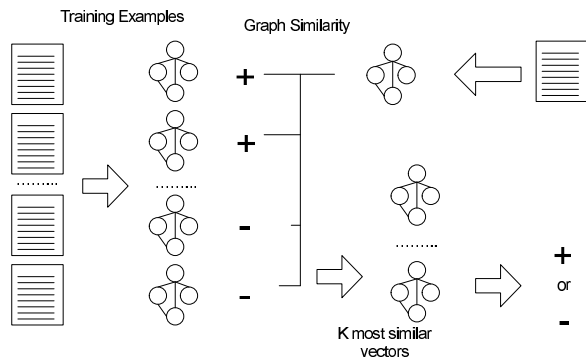


Figure 3. The Proposed Version of KNN

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.

We need to define more operations on graphs for modifying other machine learning algorithms into their graph based versions. In this research, only if we define the similarity measure between graphs, we can modify the KNN. For modifying the k means algorithm, we need to define one more operation which build prototype graph as representative among ones. For modifying the perceptron or the MLP (Multiple Layer Peceptron) where both input and weights are given as graphs, we need the update rule of graphs. In order to define more advanced operations on graphs, we need to do more theoretical research on operations based on the graph theory.

D. Application to Text Summarization

This section is concerned with the scheme of applying the proposed KNN version which was described in section III-C to the text summarization task. Before doing so, we need to transform the task into one where machine learning algorithms are applicable as the flexible and adaptive models. We prepare the paragraphs which are labeled with 'essence' or 'not' as the sample data. The paragraphs are encoded into string vectors by the scheme which was described in section III-A. Therefore, in this section, we describe the process of extracting summaries from texts automatically using the proposed KNN with the view of text summarization into a classification task.

The text summarization is mapped into a binary classification, as shown in Figure 4. A text is given as the input, and it is partitioned into paragraphs by carriage return. Each paragraph is classified into either of the two categories: 'essence' and 'not'. The paragraphs which are classified into 'essence' are selected as the output of the text summarization system. For doing so, we need to collect paragraphs which are labeled with one of the two labels as sample examples, in advance.

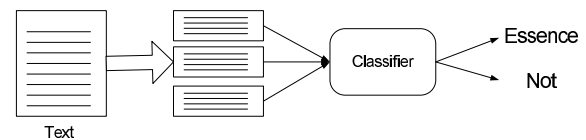


Figure 4. View of Text Summarization into Binary Classification

As sample examples, we need to collect paragraphs which are labeled with one of the two categories, before summarizing a text. The text collection should be segmented into sub-collections which are called domains, by their contents,

manually or automatically. In each sub-collection, texts are partitioned into paragraphs, and they are labeled with one of the two categories, manually. We assign classifier to each domain and train it with the paragraphs in its corresponding domain. When a text is given as the input, we select the classifier which corresponds to the domain which is most similar as the text.

Let us consider the process of applying the KNN to the text summarization which is mapped into a classification. A text is given as the input, and the classifier which corresponds to the subgroup which is most similar to the given text with respect to its content is selected. The text is partitioned into paragraphs, and each paragraph is classified into 'essence' or 'not' by the classifier. The paragraphs which are classified into 'essence' are extracted as results from summarizing the text. Note that the text is rejected, if all paragraphs are classified into 'not'.

Even if the text summarization is viewed into an instance of text categorization, we need to compare the two tasks with each other. In the text categorization, a text is given as an entity, while in the text summarization, a paragraph is done so. In the text categorization, the topics are predefined manually based on the prior knowledge, whereas in the text summarization, the two categories, 'essence' and 'not', are initially given. In the text categorization, the sample texts may span over various domains, whereas in the text summarization, the sample paragraphs should be within a domain. Therefore, although the text summarization belongs to the classification task, it should be distinguished from the topic based text categorization.

IV. CONCLUSION

Let us mention the remaining tasks for doing the further research. We apply and validate the proposed research in summarizing technical documents in specific domains such as medicine or engineering rather than news articles in various domains. We define and characterize more advanced operations mathematically on graphs which represent texts. We modify more advanced machine learning algorithms into their graph based version, using the more sophisticated operations. We implement the text summarization system as a system module or an independent software by adopting the proposed approach.

V. ACKNOWLEDGEMENT

This work was supported by 2017 Hongik University Research Fund.

REFERENCES

- [1] T. Jo, *The Implementation of Dynamic Document Organization using Text Categorization and Text Clustering*, PhD Dissertation of University of Ottawa, 2006.
- [2] N.F. Noy and C. D. Hafner, "State of the Art in Ontology Design", *AI Magazine*, Vol 18, No 3, 1997.
- [3] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologies*, Morgan Kaufmann, 2011.
- [4] F. Sebastiani, "Machine Learning in Automated Text Categorization", pp1-47, *ACM Computing Survey*, Vol 34, No 1, 2002.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] T. Jo and D. Cho, "Index based Approach for Text Categorization", *International Journal of Mathematics and Computers in Simulation*, Vol 2, No 1, 2008.
- [9] 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.
- [10] T. Jo, "Device and Method for Categorizing Electronic Document Automatically", Patent Document, 10-2009-0041272, 10-1071495, 2011.
- [11] T. Jo, "Normalized Table Matching Algorithm as Approach to Text Categorization", pp839-849, *Soft Computing*, Vol 19, No 4, 2015.
- [12] 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.
- [13] T. Jo, "Representation of Texts into String Vectors for Text Categorization", pp110-127, *Journal of Computing Science and Engineering*, Vol 4, No 2, 2010.
- [14] 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.
- [15] 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.