Table based KNN for Text Summarization

Taeho Jo
Department of Computer and Information Communication Engineering, Hongik University, Sejong, South Korea

Abstract— *In this research, we propose the modified version of KNN (K Nearest Neighbor) as an approach to the text summarization. Encoding texts into numerical vectors for using the traditional versions for text mining tasks causes the three main problems: the huge dimensionality, the sparse distribution, and the poor transparency; this fact motivated this research. The idea of this research is to interpret the text summarization task into the text classification task and apply the proposed version of KNN to the task where texts are encoded into tables. The modified version which is proposed in this research is expected to summarize texts more reliably than the traditional version by solving the three problems. Hence, the goal of this research is to implement the text summarization system, using the proposed approach.*

Keywords: Text Summarization, Table Similarity , K Nearest Neighbor

1. Introduction

The text summarization refers to the process of selecting automatically some sentences or paragraphs in the given text as its essential part. As its preliminary task, the text is segmented into the sentences or the paragraphs, by the punctuation mark or the carriage return, respectively. In the task, some sentences or paragraphs are selected as the important part. In this research, the text summarization is interpreted into the binary classification where each sentence or paragraph is classified whether it is the essential part, or not. The automatic text summarization by a system or systems should be distinguished from one by human being which refers to the process of rewriting the entire text into its short version.

Let us consider some motivations for doing this research. In encoding texts into numerical vectors as the traditional preprocessing, the three main problems, such as huge dimensionality, sparse distribution, and poor transparency, may happen[2][3][4][13][6]. Encoding texts into tables showed previously successful results in other tasks text mining: text categorization and clustering [3][4] [7]. Although we proposed previously the alternative representations of texts which were called string vectors, we need to define and characterize them mathematically to make the foundations for creating and modifying string vector based versions of machine learning algorithms [13][6]. Hence, this research is carried out by the motivations; we attempt to encode texts into tables in the text summarization task, as well as the tasks of text categorization and clustering.

This research may be characterized as some agenda. In this research, the text summarization task is interpreted into the binary classification task where each sentence or paragraph is classified into the essence, or not. Each sentence or paragraph is encoded into a table, instead of numerical vectors, to avoid the three problems. The similarity measure between tables which is always given as a normalized value is defined and used for modifying the KNN. The modified version will applied to the binary task which is mapped from the text summarization task.

We will mention some benefits from this research. This research prevents the process of encoding texts from the three main problems mentioned above. By solving the problems, we may expect the better performance than the traditional version of KNN. Since the table is more symbolic representation of each text, we may guess the contents of texts by their representations. However, the table size is given as the external parameter of the proposed text summarization system, it is necessary to be careful for setting its value to optimize the trade-off between the system reliability and speed.

This article is organized into the four sections. In Section 2, we survey the relevant previous works. In Section 3, we describe in detail what we propose in this research. In Section 4, we mention the remaining tasks for doing the further research.

2. 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 [8]. 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 [9]. In 2004, Lesile et al. used the version of SVM which proposed by Lodhi et al. to the protein classification [10]. In 2004, Kate and Mooney used also the SVM version for classifying sentences by their meanings [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[15]. 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 [16]. In 2010, Jo proposed the unsupervised neural networks, called Neural Text Self Organizer, which receives the string vector as its input data [17]. 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 [18].

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 [3]. In 2008, Jo applied also his proposed approach to the text clustering, as well as the text categorization [15]. In 2011, Jo described as the technique of automatic text classification in his patent document [13]. In 2015, Jo improved the table matching algorithm into its more stable version [14].

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 will modify the machine learning algorithm, KNN, into the version which processes tables instead of numerical vectors, and use it as the approach to the text summarization which is mapped into a classification task.

3. Proposed Approach

This section is concerned with the table based KNN (K Nearest Neighbor) as the approach to text summarization, and it consists of the four sections. In Section 3.1, we describe the process of encoding a text into a table. In Section 3.2, we do formally that of computing a similarity between tables into a normalized value between zero and one. In Section 3.3, we mention the proposed version of KNN together with its traditional version. This section is intended to describe in detail the proposed version of KNN as the approach to the text summarization task. In Section 3.4, we mention the scheme of applying the KNN to the task with the view of it into the binary classification task.

3.1 Text Encoding

This section is concerned with the process of encoding texts into tables as illustrated in figure 1. In the process, a text is given as the input and a table which consists of entries of words and their weights is generated as the output. A text is indexed into a list of words through the basic three steps as illustrated in figure . For each word, its weight is computed and assigned to it. Therefore, in this section, we describe the three steps which are involved in text indexing and the scheme of weighting words.

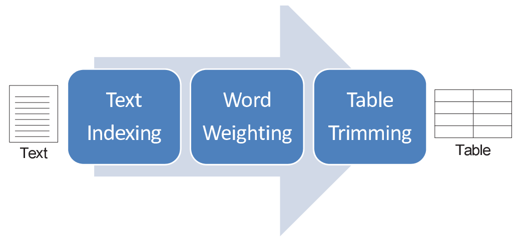


Fig. 1: The Process of Encoding Text into Table

The process of indexing the corpus into list of words is illustrated in figure 2. The first step of the process is to tokenize a string concatenated by texts in the corpus into tokens by segmenting it by white spaces or punctuation marks. The second step called stemming is to map each token into its root form by the grammatical rules. The third step is to remove the stop words which function only grammatically and irrelevantly to contents for more efficiency. The words which are generated the through three steps are usually nouns, verbs, and adjectives.

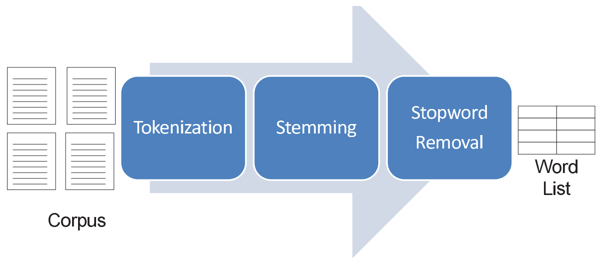


Fig. 2: The Process of Corpus Indexing

The weight of the word, t_j in the text, d_i is denoted by w_{ij}, and let us mention the three schemes of computing it.

We use the relative frequency computed by equation (1)

$$w_{ij} = \frac{TF_{ij}}{TF_{imax}} \tag{1}$$

where TF_{ij} is the frequency of the word,t_j in the text, d_i, and TF_{imax} is the maximum frequency in the text, d_i. We may assign a binary value, zero or one, to weight by equation (2),

$$w_{ji} = \begin{cases} 1 & \text{if } TF_{ji} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

We may use the TF-IDF (Term Frequency - Inverse Document Frequency) weights computed by equation (3),

$$w_{ji} = \begin{cases} log\frac{N}{DF_i}(1 + logTF_{ji}) & \text{if } TF_{ji} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

where N is the total number of texts in the corpus, DF_i is the number of texts including the word, t_{ji}, and TF_{ji} is the frequency of the word, t_{ji} in the given text, D_j. In this research, we adopt the TF-IDF method which is expressed in equation (3) as the scheme of weighting words.

We mention the schemes of trimming the table for more efficiency. The first scheme is to rank the words in the table by their weights and select a fixed number of words with their highest words. The second one is to set a threshold as an external parameter and select words with their higher weights than the threshold, assuming that the weights are always given as normalized values between zero and one. The third one is to cluster words by their weights into the two subgroups: the higher weighted group and the lower weighted group; the words in the former groups are selected. In this research, we adopted the first scheme because it is simplest.

Let us consider the differences between the word encoding and the text encoding. In the word encoding, each word is associated with texts including it, whereas, in the text encoding, each text is done with words included in it. Each table representing a word has its own entries of text identifiers and its weights in the texts, whereas one representing a text has those of words and their weights in the text. Computing similarity between the tables representing both words is based on text identifiers which include them, while doing it between the tables representing both texts is based on words shared by the both texts. Therefore, from the comparisons, reversal of texts and words indicate the essential differences between the text and word encoding.

3.2 Similarity between Two Tables

This section is concerned with the process of computing a similarity between tables. Texts are encoded into tables by the process which was described in Section 3.1. The two tables are viewed into the two sets of words and the set of shared words is retrieved by applying the intersection on the two sets. The similarity between the two tables is based on the ratio of the shared word weights to the total weights of the two tables. Therefore, we intend this section to describe in detail and formally the process of computing the similarity.

A table which represents a word may be formalized as a set of entries of words and its weights. The text, D_j is represented into a set of entries as follows:

$$D_j = \{(t_{j1}, w_{j1}), (t_{j2}, w_{j2}), ..., (t_{jn}, w_{jn})\}$$

where t_{ji} is a word included in the text, D_j, and w_{ji} is the weight of the word, t_{ji} in the text,D_j. The set of only words is as follows:

$$T(D_j) = \{t_{j1}, t_{j2}, ..., t_{jn}\}$$

The TF-IDF (Term Frequency - Inverse Document Frequency) weight, w_{ji} of the word, t_{ji} in the text, D_j is computed by equation (4)

$$w_{ji} = \begin{cases} log\frac{N}{DF_i}(1 + logTF_{ji}) & \text{if } TF_{ji} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

where N is the total number of texts in the corpus, DF_i is the number of texts including the word, t_{ji}, and TF_{ji} is the frequency of the word, t_{ji} in the given text, D_j. Therefore, the table is defined formally as unordered set of pairs of words and their weights.

Let us describe formally the process of computing the similarity between two tables indicating two texts. The two texts,D_1 and D_2 are encoded into the two tables as follows:

$$D_1 = \{(t_{11}, w_{11}), (t_{12}, w_{12}), ..., (t_{1n}, w_{1n})\}$$

$$D_2 = \{(t_{21}, w_{21}), (t_{22}, w_{22}), ..., (t_{2n}, w_{2n})\}$$

The two texts are represented into the two sets of words by applying the operator, $T(\cdot)$, as follows:

$$T(D_1) = \{t_{11}, t_{12}, ..., t_{1n}\}$$

$$T(D_2) = \{t_{21}, t_{22}, ..., t_{2n}\}$$

By applying the intersection to the two sets, a set of shared words is generated as follows:

$$T(D_1) \cap T(D_2) = \{st_1, st_2, ..., st_k\}$$

We construct the table of the shared words and their dual weights among which one is from D_1, and the other is from D_2 as follows:

$$ST = \{(st_1, w_{11}, w_{21}), (st_2, w_{12}, w_{22}), ..., (st_k, w_{k2}, w_{k2})\}$$

The similarity between the two tables is computed as the ratio of the total dual weights of the shared words to the total weights of the ones in both tables, by equation (5).

$$Sim(D_1, D_2) = \frac{\sum_{i=1}^{k}(w_{1i} + w_{2i})}{\sum_{i=1}^{m} w_{1i} + \sum_{i=1}^{m} w_{2i}} \tag{5}$$

It is always given as a normalized value between zero and one; if the two tables, D_1 and D_2 are same to each other, $D_1 = D_2$ the similarity becomes 1.0 as follows:

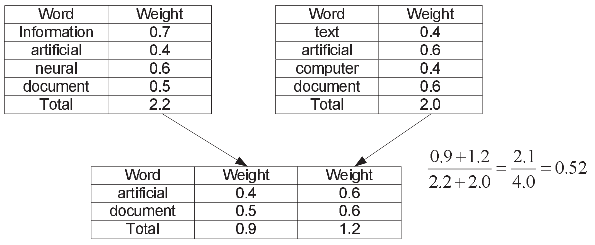
$$Sim(D_1, D_2) = \frac{\sum_{i=1}^{m}(w_{1i} + w_{2i})}{\sum_{i=1}^{m} w_{1i} + \sum_{i=1}^{m} w_{2i}}$$

$$= \frac{\sum_{i=1}^{m} w_{1i} + \sum_{i=1}^{m} w_{2i}}{\sum_{i=1}^{m} w_{1i} + \sum_{i=1}^{m} w_{2i}} = 1.0$$

If they are exclusive, $T(D_1) \cap T(D_2) = \emptyset$ the similarity becomes 0.0 as follows:

$$Sim(D_1, D_2) = \frac{0}{\sum_{i=1}^{m} w_{1i} + \sum_{i=1}^{m} w_{2i}} = 0.0$$

We demonstrate the process of computing the similarity between two tables using the simple example which is presented in Figure 3. The two texts are encoded into the two source tables as shown in Figure 3. In the example, the two words, 'artificial' and 'documents' are shared by the two tables, and each shared ones have their dual weights from the two input tables. The similarity between the two tables is computed to be 0.52 as a normalized value by equation (5). Therefore, the similarity is computed by lexical matching between the two tables.



Word	Weight
Information	0.7
artificial	0.4
neural	0.6
document	0.5
Total	2.2

Word	Weight
text	0.4
artificial	0.6
computer	0.4
document	0.6
Total	2.0

Word	Weight	Weight
artificial	0.4	0.6
document	0.5	0.6
Total	0.9	1.2

$$\frac{0.9+1.2}{2.2+2.0} = \frac{2.1}{4.0} = 0.52$$

Fig. 3: Example of Two Tables

The similarity computation which is presented above is characterized mathematically. The commutative law applies to the computation as follows:

$$Sim(D_1, D_2) = \frac{\sum_{i=1}^{k}(w_{1i} + w_{2i})}{\sum_{i=1}^{m} w_{1i} + \sum_{i=1}^{m} w_{2i}}$$

$$= \frac{\sum_{i=1}^{k}(w_{2i} + w_{1i})}{\sum_{i=1}^{m} w_{2i} + \sum_{i=1}^{m} w_{1i}} = Sim(D_2, D_1).$$

The similarity is always given as a normalized value between zero and one as follows:

$$0 \leq Sim(D_1, D_2) \leq 1.$$

If the weights which are assigned to all words are identical, the similarity between two tables depends on the number of shared words as follows:

$$Sim(D_1, D_2) \leq Sim(D_1, D_3)$$
$$\rightarrow |T(D_1) \cap T(D_2)| \leq |T(D_1) \cap T(D_3)|.$$

The complexity of computing the similarity between two tables is $O(n \log n)$, since it takes $O(n \log n)$ for sorting the entries of two tables using the quick sort or the heap sort, and $O(n)$ for extracting shared elements by the consequential processing [1].

3.3 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 tables by the process which was described in Section 3.2. In this section, we attempt to the traditional KNN into the version where a table 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 4. 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.

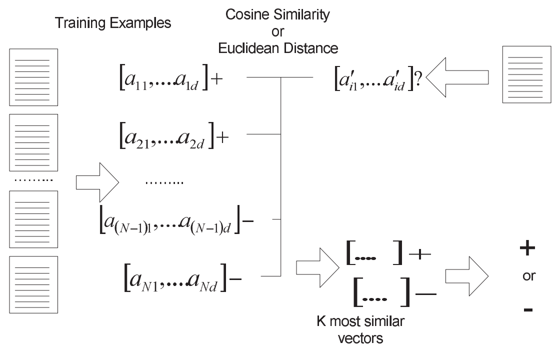

Training Examples
Cosine Similarity
or
Euclidean Distance
$[a_{11},\ldots a_{1d}]+$
$[a'_{i1},\ldots a'_{id}]?$
$[a_{21},\ldots a_{2d}]+$
....
....
$[a_{(N-1)1},\ldots a_{(N-1)d}]-$
$[a_{N1},\ldots a_{Nd}]-$
[----]+
[----]−
K most similar vectors
+
or
-

Fig. 4: The Traditional Version of KNN

Separately from the traditional one, we illustrate the classification process by the proposed version in Figure 5. The sample texts labeled with the positive or negative class are encoded into tables. The similarity between two tables is computed by the scheme which was described in Section 3.2. 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 table is never available inherently, the poor discriminations by sparse distribution are certainly overcome in this research.

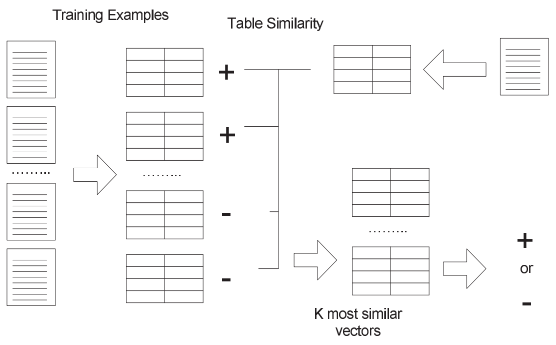


Fig. 5: The Proposed Version of KNN

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 the tables which represent texts are characterized more symbolically than numerical vectors, it is easier to trace results from classifying items. Let us assume that novice tables are classified by voting labels of their nearest neighbors. In each category, the shared words between novice one and its nearest neighbors are extracted. We make a list of shared ones and their weights, and the categorical score, in each category. We present the evidence of classifying an entity into a category, by showing them.

3.4 Application to Text Summarization

This section is concerned with the scheme of applying the proposed KNN version which was described in Section 3.3 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 tables by the scheme which was described in Section 3.2. 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 6. 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.

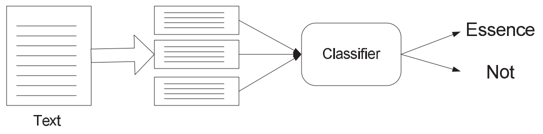


Fig. 6: View of Text Summarization into Binary Classification

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.

4. Conclusion

We need the remaining tasks for doing the further research. We may apply the proposed approach for summarizing texts in the specific domains such as medicine, law, and engineering. We may consider the semantic relations among different words in the tables in compute their similarities, but it requires the similarity matrix or the word net for doing so. We may install the process of optimizing weights of words as the meta-learning tasks. We may implement the text summarization system, adopting the proposed approach.

5. Acknowledgement

This work was supported by 2017 Hongik University Research Fund.

References

[1] M.J. Folk, B. Zoellick, and G. Riccardi, File Structures: An Object Oriented with C++, Addison Wesley, 1998.
[2] T. Jo, "The Implementation of Dynamic Document Organization using Text Categorization and Text Clustering" PhD Dissertation of University of Ottawa, 2006.
[3] T. Jo and D. Cho, "Index Based Approach for Text Categorization", 127-132, International Journal of Mathematics and Computers in Simulation, No 2, 2008.
[4] 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.
[5] T. Jo, "Topic Spotting to News Articles in 20NewsGroups with NTC, , Lecture Notes in Information Technology", pp50-56, No 7, 2011.
[6] T. Jo, "Definition of String Vector based Operations for Training NTSO using Inverted Index", pp57-63, Lecture Notes in Information Technology, No 7, 2011.
[7] T. Jo, "Definition of Table Similarity for News Article Classification", pp202-207, The Proceedings of The Fourth International Conference on Data Mining, 2012.
[8] F. Sebastiani, "Machine Learning in Automated Text Categorization", pp1-47, ACM Computing Survey, Vol 34, No 1, 2002.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] T. Jo, "Device and Method for Categorizing Electronic Document Automatically", Patent Document, 10-2009-0041272, 10-1071495, 2011.
[14] T. Jo, "Normalized Table Matching Algorithm as Approach to Text Categorization", pp839-849, Soft Computing, Vol 19, No 4, 2015.
[15] 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.
[16] T. Jo, "Representationof Texts into String Vectors for Text Categorization", pp110-127, Journal of Computing Science and Engineering, Vol 4, No 2, 2010.
[17] 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.
[18] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", pp83-96, International Journal of Information Studies, Vol 2, No 2, 2010.