

Table based KNN for Index Optimization

Taeho Jo

School of Computer and Information Engineering

Inha University

Incheon, South Korea

tjo018@inha.ac.kr

Abstract—We concern this research with the table based KNN as the approach to the index optimization task. It may be interpreted into an instance of word classification, and the encoding scheme where words are encoded into tables improved the task word classification. In this research, words are encoded into tables and apply the table based KNN to the index optimization task. From this research, we expect the better and more stable performance than the traditional version, in this task. Therefore, this research is intended to provide the improved index optimization scheme.

Keywords—Index Optimization, Table Similarity, Table based KNN

I. INTRODUCTION

Index optimization refers to the process of deciding each word into inclusion, or removal, in order to generate its optimized index for maximizing the information retrieval performance. In this research, in addition to the inclusion and the removal, we consider the expansion which adds more semantically similar words from the given corpus to the essential words. The index optimization is interpreted into the classification task where each word is classified into the three categories: expansion, inclusion, and removal. Under the interpretation, it is assumed that unsupervised learning algorithms will be used as the approach to the task. The scope of this research is restricted to only crisp index optimization which does not allow each word to be overlapped over more than one category.

In encoding words into numerical vectors, some problems are inevitable. If we use texts as features of numerical vectors representing words, many features are required for keeping enough robustness of the system [2]. The discrimination among numerical vectors representing them is very poor because of the sparse distribution in each of them [3]. When using grammatical properties as features, instead of texts, it becomes very difficult and complicated to implement the encoding process. Therefore, this research attempts to solve the above problems by encoding words into tables, instead of numerical vectors.

Let us propose some agenda in this research, in order to solve the above problems. In this research, we propose the index optimization as an instance of classification task for improving the cohesion of indexed words. We encode each

word into a table which consists of entries of texts including it and its weights in them. We define the similarity measure between tables which is always given as a normalized value, and using it, modify the KNN (K Nearest Neighbor) where a table is directly given as the input data. Hence, we apply the modified KNN to the classification task which is transformed from the index optimization.

This research may be expected to provide some benefits for developing the text segmentation system. The table which represents a text may be regarded as the more symbolic and compact text representation; it needs much less than one hundred elements for maintaining the system robustness. We may expect much more discrimination among tables than among numerical vectors, because the sparse distribution does not happen in encoding texts into tables. We expect both better performance and more stability from the proposed KNN version because the similarity measure between tables is always given as a normalized value between zero and one. However, note that the table size is given as the additional parameter, so it should be optimized between the computation speed and the system reliability.

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].

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[11]. 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 [12]. In 2010, Jo proposed the unsupervised neural networks, called Neural Text Self Organizer, which receives the string vector as its input data [13]. 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 [14].

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 [11]. In 2011, Jo described as the technique of automatic text classification in his patent document [9]. In 2015, Jo improved the table matching algorithm into its more stable version [10].

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 index optimization which is viewed into a classification task.

III. PROPOSED SYSTEM

This article is concerned with the index optimization system where the table based KNN (K Nearest Neighbor) is adopted as the approach, and consists of the four sections.

In Section 1, we describe the process of encoding words into tables as the preprocessing step. In Section 2, we cover the scheme of computing a similarity between two tables into a normalized value between zero and one. In Section 3, we mention the proposed KNN where a table is given as the input data instead of a numerical vector. In Section 5, viewing the index optimization into a triary classification task, we explain the scheme of applying the KNN version to the task.

A. Word Encoding

This section is concerned with the process of encoding words into tables as illustrated in figure 1. In the process, a word is given as the input and a table which consists of entries of text identifiers and their weights is generated as the output. In addition, we need a text collection for encoding so; each text links to its own list of included words in the collection. The list of texts each of which is linked with included words is converted into the inverted list where each word links to texts including it. Therefore, in this section we describe the three steps involved in encoding words into tables as illustrated in figure 1.

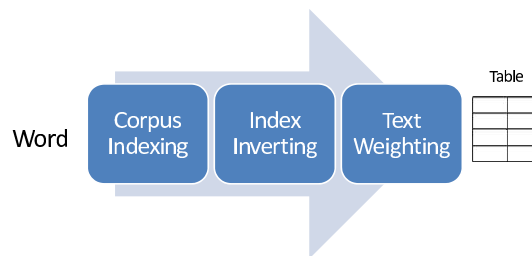


Figure 1. Overall Process of Word Encoding

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.

In Figure 3, we illustrate the inverted list where each word is linked to its list of texts including the word. The index list where each text is linked to its included words may be generated by indexing each text from the text collection. For each word, the inverted list is constructed by retrieving texts including it from the index list. In each word, each text is associated with its weight as the relationship between the word and the text. The association of each word with the list of texts is opposite to the association of each text with its list of words is the reason of calling the list which is presented in figure 3 the inverted list.

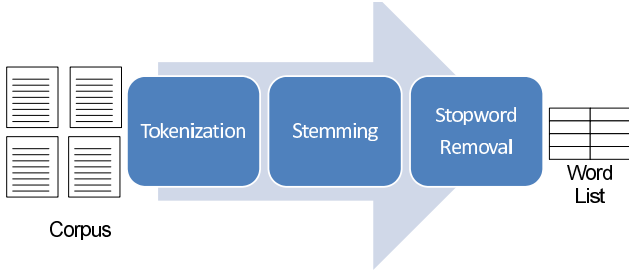


Figure 2. The Process of Corpus Indexing

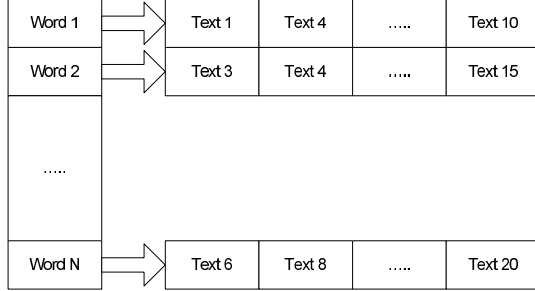


Figure 3. The Inverted Index

We build a table which represents a word from the inverted list which is shown in Figure 3. We fetch the text identifiers which are linked to the given word from the inverted list. We make the list of entries of text identifiers and their weights of words in them. We select entries by their weights and build the constantly sized table which representing the given word. Previously, we mentioned the process of encoding texts into tables; each table consists of words and their weights, instead of text identifiers [10].

Previously, we attempted to encode texts into tables [10], so 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.

B. Similarity between 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 III-A. 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 text identifiers and its weights. The word, T_j is represented into a set of entries as follows:

$$T_j = \{(d_{j1}, w_{j1}), (d_{j2}, w_{j2}), \dots, (d_{jn}, w_{jn})\}$$

where d_{ji} is a text identifier including the word, T_j , and w_{ji} is the weight of the word, T_j in the text, d_{ji} . The set of only text identifiers including the word, T_j is as follows:

$$D(T_j) = \{d_{j1}, d_{j2}, \dots, d_{jn}\}$$

The TF-IDF (Term Frequency - Inverse Document Frequency) weight, w_{ij} of the word, T_j in the text, d_{ji} is computed by equation (1)

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

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

Let us describe formally the process of computing the similarity between two tables indicating two words. The two words, T_1 and T_2 are encoded into the two tables as follows:

$$T_1 = \{(d_{11}, w_{11}), (d_{12}, w_{12}), \dots, (d_{1n}, w_{1n})\}$$

$$T_2 = \{(d_{21}, w_{21}), (d_{22}, w_{22}), \dots, (d_{2n}, w_{2n})\}$$

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

$$D(T_1) = \{d_{11}, d_{12}, \dots, d_{1n}\}$$

$$D(T_2) = \{d_{21}, d_{22}, \dots, d_{2n}\}$$

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

$$D(T_1) \cap D(T_2) = \{sd_1, sd_2, \dots, sd_k\}$$

We construct the table of the shared texts and their dual weights among which one is from T_1 , and the other is from T_2 as follows:

$$SD = \{(sd_1, w_{11}, w_{21}), (sd_2, w_{12}, w_{22}), \dots, (sd_k, w_{k1}, w_{k2})\}$$

The similarity between the two tables is computed as the ratio of the total dual weights of the shared texts to the total weights of the texts in both tables, by equation (2).

$$Sim(T_1, T_2) = \frac{\sum_{i=1}^k (w_{1i} + w_{2i})}{\sum_{i=1}^m w_{1i} + \sum_{i=1}^m w_{2i}} \quad (2)$$

It is always given as a normalized value between zero and one; if the two tables, T_1 and T_2 are same to each other, $T_1 = T_2$ the similarity becomes 1.0 as follows:

$$\begin{aligned} Sim(T_1, T_2) &= \frac{\sum_{i=1}^m (w_{1i} + w_{2i})}{\sum_{i=1}^m w_{1i} + \sum_{i=1}^m w_{2i}} \\ &= \frac{\sum_{i=1}^m w_{1i} + \sum_{i=1}^m w_{2i}}{\sum_{i=1}^m w_{1i} + \sum_{i=1}^m w_{2i}} = 1.0 \end{aligned}$$

If they are exclusive, $D(T_1) \cap D(T_2) = \emptyset$ the similarity becomes 0.0 as follows:

$$Sim(T_1, T_2) = \frac{0}{\sum_{i=1}^m w_{1i} + \sum_{i=1}^m w_{2i}} = 0.0$$

We demonstrate the process of computing the similarity between two tables using the simple example which is presented in Figure 4. The two words are encoded into the two source tables as shown in Figure 4. In the example, the two text identifiers, '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 (2). Therefore, the similarity is computed by lexical matching between the two tables.

Text	Weight	Text	Weight
doc01	0.7	doc05	0.4
doc02	0.4	doc02	0.6
doc03	0.6	doc06	0.4
doc04	0.5	doc04	0.6
Total	2.2	Total	2.0

Text	Weight	Weight
doc02	0.4	0.6
doc04	0.5	0.6
Total	0.9	1.2

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

Figure 4. Example of Two Tables

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

$$\begin{aligned} Sim(T_1, T_2) &= \frac{\sum_{i=1}^k (w_{1i} + w_{2i})}{\sum_{i=1}^m w_{1i} + \sum_{i=1}^m w_{2i}} \\ &= \frac{\sum_{i=1}^k (w_{2i} + w_{1i})}{\sum_{i=1}^m w_{2i} + \sum_{i=1}^m w_{1i}} = Sim(T_2, T_1). \end{aligned}$$

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

$$0 \leq Sim(T_1, T_2) \leq 1.$$

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

$$\begin{aligned} Sim(T_1, T_2) &\leq Sim(T_1, T_3) \\ \rightarrow |D(T_1) \cap D(T_2)| &\leq |D(T_1) \cap D(T_3)|. \end{aligned}$$

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].

C. Proposed Version of KNN

This section is concerned with the proposed KNN version as the approach to the word categorization. Words are encoded into tables by the process which was described in Section III-B. In this section, we attempt to modify 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 words 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 5. 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.

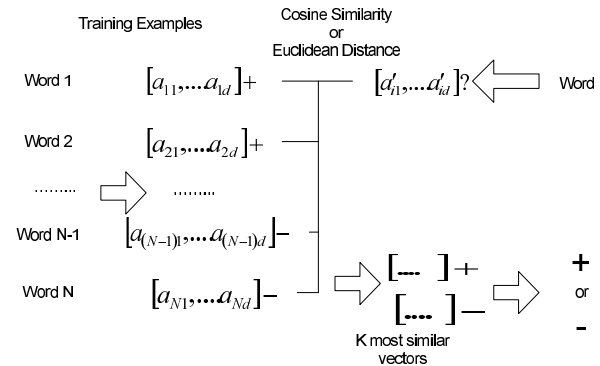


Figure 5. The Traditional Version of KNN

Separately from the traditional one, we illustrate the classification process by the proposed version in Figure 6. 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 III-B. 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.

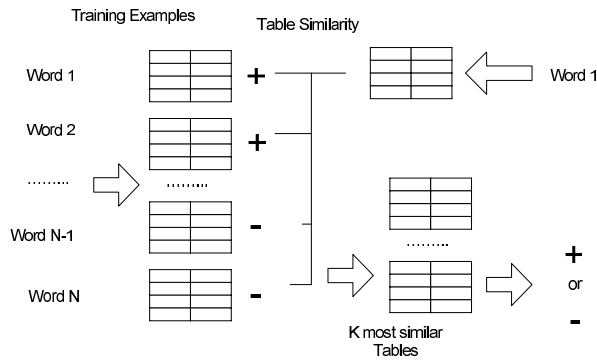


Figure 6. The Proposed Version of KNN

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 words are characterized more symbolically than numerical vectors, it is easier to trace results from classifying items. Let us assume that novice words are classified by voting labels of their nearest neighbors. In each category, the shared text identifiers 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.

D. Application to Index Optimization

This section is concerned with the scheme of applying the proposed KNN version which was described in Section III-C to the index optimization 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 words which are labeled with 'expansion', 'inclusion' or 'removal' as the sample data. The words are encoded into tables by the scheme which was described in section III-B. Therefore, in this section, we describe the process of extracting words which belong to the two categories, 'expansion' and 'inclusion', from texts automatically using the proposed KNN with the view of the index optimization into a classification task.

In this research, the index optimization is viewed into a classification task, as shown in Figure 7. A text is given as the input, and a list of words is extracted by indexing the text. Each word is classified by the classifier into one of the three categories: 'expansion', 'inclusion', or, 'removal'.

In the task, the text is mapped into words which are classified with 'expansion' or 'inclusion'. The similar words to one labeled with 'expansion' will be added from external sources.

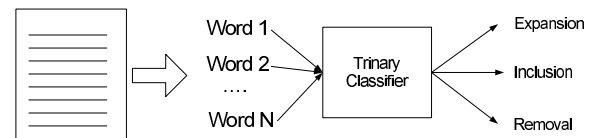


Figure 7. View of Index Optimization into Classification Task

We need to prepare sample words which are labeled with one of the three categories, before classifying a novice one or ones. A text collection is segmented into sub-collections of content based similar words which are called domains, manually or automatically. We prepare sample words which are labeled manually, domain by domain. To each domain, we assign and train a classifier with the words in the corresponding sub-collection. When a text is given as the input, the classifier which corresponds to the most similar domain is selected among them.

Let us consider the process where an article is given as the input and a list of essential words is extracted as the output. We nominate the classifier which corresponds to the subgroup which is closest to the given article with respect to its content. A list of words is extracted by indexing the article, and each word is encoded. The words are classified by the nominated classifier into one of the three categories, and we select ones which are labeled with 'expansion' or 'reservation' as the optimized index. The addition of external words which are semantically similar as ones labeled with 'expansion' is set as the subsequent task.

Even if the index optimization is viewed into an instance of word categorization, it needs to be distinguished from the topic based word categorization. The word categorization is given as a single multiple classification or multiple binary classifications, whereas the index optimization is done as a single triary classification or three binary classification tasks. In the word categorization, each word is classified semantically into one or some of the predefined topics, whereas in the index optimization, it is classified one of the three actions. In the word categorization, each word is classified by its meaning, whereas in the index optimization, it is classified by its importance to the given text. In the word categorization, when the given task is decomposed into binary classification tasks, a classifier is assigned to each topic, whereas, in the index optimization, a classifier is done to each domain.

IV. CONCLUSION

Let us consider the remaining tasks for doing the further research. In one of specific domains such as engineering, science, and medicine, we need to validate and customize the

proposed research in the index optimization, by transforming it into a classification task. Because various schemes of weighting words are available, more than one weight may be assigned to each word, so it need to be considered in computing the similarity between tables. Other machine learning algorithms may be modified as well as KNN into their table based versions. By adopting the proposed approach, we implement the index optimization system as a module of other programs or an independent program.

REFERENCES

- [1] M.J. Folk, B. Zoellick, and G. Riccardi, File Structures: An Object Oriented with C++, Addison Wesley, 1998.
- [2] L.D. Baker and A. K. McCallum , “Distributional clustering of words for text classification”, pp96-103 Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, 1998.
- [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] 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, “Single Pass Algorithm for Text Clustering by Encoding Documents into Tables”, pp1749-1757, Journal of Korea Multimedia Society, Vol 11, No 12, 2008.
- [9] T. Jo, “Device and Method for Categorizing Electronic Document Automatically”, Patent Document, 10-2009-0041272, 10-1071495, 2011.
- [10] T. Jo, “Normalized Table Matching Algorithm as Approach to Text Categorization”, pp839-849, Soft Computing, Vol 19, No 4, 2015.
- [11] 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.
- [12] T. Jo, “Representation of Texts into String Vectors for Text Categorization”, pp110-127, Journal of Computing Science and Engineering, Vol 4, No 2, 2010.
- [13] 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.
- [14] 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.