ACTA SCIENTIFIC MEDICAL SCIENCES

Volume 3 Issue 4 April 2019

Review Article

Text Classification using Feature Similarity based K Nearest Neighbor

Taeho Jo*

School of Game Hongik University Sejong, South Korea

***Corresponding Author:** Taeho Jo, School of Game Hongik University Sejong, South Korea.

Received: February 18, 2019; **Published:** March 01, 2019

Abstract

This article proposes the modified KNN (K Nearest Neighbor) algorithm which considers the feature similarity and is applied to the text categorization. The words which are given as features for encoding texts into numerical vectors are semantic related entities, rather than independent ones, and the synergy effect between the word categorization and the text categorization is expected by combining both of them with each other. In this research, we define the similarity metric between two vectors, including the feature similarity, modify the KNN algorithm by replacing the exiting similarity metric by the proposed one, and apply it to the text categorization. The proposed KNN is empirically validated as the better approach in categorizing texts in news articles and opinions. The significance of this research is to improve the classification performance by utilizing the feature similarities.

Keywords: KNN; Research; Vectors

Introduction

Text categorization refers to the process of assigning a topic or category to each text, as an instance of pattern classification. Its preliminary task is to predefine topics or categories and allocate texts to each of them as the sample data. By learning the sample labeled texts, the classification capacity which is given as symbolic rules, equations or/and statistical model parameters, is constructed. Subsequently, novice texts are classified based on the constructed classification capacity. Even if various types of text categorization are available, the scope of this research is restricted to only hard text categorization where each text is classified into only one topic or category, exclusively.

Some problems are caused by encoding texts into numerical vectors and computing their similarities based on attribute values. Many features are required for maintaining the system robustness in encoding texts into numerical vectors because each feature covers very small proportion of documents [7]. The dominance of zero values in each numerical vector becomes the poor environment for computing their similarities because of the very weak discrimination among numerical vectors [7]. The assumption that the features are independent of each other violates against the reality [25]. Hence, in this research, we consider both features and feature values for compute the similarity between two numerical vectors as the challenge against the problems.

Let us mention what we propose in this research as the idea. In this research, we assume that words are used as features for representing texts into numerical vectors, and attempt to consider their semantic relations. Based on the semantic relations among words, we define the similarity measure which considers both the feature similarity and the feature value similarity between representations of texts. Using the similarity measure, we modify the KNN into the version which computes the similarity between items based on the both similarities, and apply it to the text categorization, as the approach. Hence, we obtain the more discrimination among numerical vectors even in the sparse distribution, and the reduced number of features as the benefits from this research.

Let us mention the benefits expected from this research, based on the above proposals. From this research, we expect the potential possibility of reducing the dimensions of numerical vectors representing texts, by considering the similarity among features as well as among feature values. We may reduce the information loss in computing the similarity between texts by reflecting the semantic similarity among words which are given as features.

We may expect the improved tolerance to the problems from the sparse distributions of numerical vectors, because the similarity between attributes is considered. Therefore, we expect both the better performance of the text categorization and the more efficient representations of texts from this research.

This article is organized into the five sections. In Section II, we survey the relevant previous works. In Section III, we describe in detail what we propose in this research. In Section IV, we validate empirically the proposed approach by comparing it with the traditional one. In Section V, we mention the general discussion on the empirical validations and remaining tasks for doing the further research.

Previous works

This section is concerned with the previous works which are relevant to this research. In Section II-A, we explore the previous cases of applying the KNN algorithm to text mining tasks. In Section II-B, we survey the schemes of encoding texts or words into structured data. In Section II-C, we describe the previous machine learning algorithms which receive alternative structured data such as tables and string vectors to numerical vectors. Therefore, in this section, we provide the history about this research, by surveying the relevant previous works.

Using KNN algorithm to text mining tasks

This section is concerned with the previous cases of applying the KNN algorithm to text mining tasks. Classifying texts or words belong to a text mining task, and the KNN algorithm is adopted as the approach to the task in this research. The KNN algorithm belongs to the lazy learning algorithm which does not learn training examples in advance. The fact that the KNN algorithm is popularly used in classification tasks in any domain, as well as text categorization is the reason of adopting and modifying the KNN algorithm. In this section, we survey cases of applying the KNN algorithm to the word categorization, text categorization, and spam mail filtering.

Let us mention the previous cases of applying the KNN algorithm to the word categorization and its similar tasks. In 2001, Kim., *et al*. translated words using the KNN algorithm between English and Korean [20]. In 2003, Pekar and Staab classified words into their synonyms, using the KNN algorithm [27]. In 2016, Stauffer., *et al*. represented optimal images of handwritten words into graphs and applied the KNN to the word recognition [29]. The word categorization in this research is the task of classifying words based on their topics or meaning; it should be distinguished from ones which are mentioned in the above literatures.

Let us mention the previous cases of applying the KNN algorithm to the text categorization. In 2001, Sam., *et al*. proposed the modified version of KNN which considers the feature importance for computing the similarity between numerical vectors [4]. In 2010, Khan., *et al*. reviewed machine learning approaches including the KNN algorithm to the text categorization [19]. In 2014, Vishwanath., *et al*. proposed the KNN version which computes the similarity between a test document and a class prototype and the Naive Bayes, as the approaches to text categorization [30]. Even recently, texts are still encoded into numerical vectors in using the KNN for the text categorization, in above literatures.

The spam mail filtering refers to a particular text categorization where each email is classified into spam or ham. In 2003, James., *et al*. proposed the neural networks as the approach to the spam mail filtering and compared it with the KNN algorithm as the base approach [5]. In 2004, Lai and Tsai proposed the four machine learning algorithms including the KNN as the email categorization tools [21]. In 2010, Frite., *et al*. used the KNN algorithm for the spam mail filtering with resampling methods [3]. In the above literatures, emails are regarded as texts and encoded into numerical vectors.

We survey the cases of using the KNN for the text mining tasks: word categorization, text categorization, and spam mail filtering. Textual data are encoded into numerical vectors in all above literatures. The KNN algorithm has been used as very popular approach to text mining tasks as well as any other kinds of classification tasks. Even if another approach has been proposed to the word and text categorization, it has been compared with the KNN algorithm. In this research, we adopt the KNN based on the above literatures and modify it into more suitable version for text mining tasks.

Encoding schemes

This is concerned with the previous works on encoding words or texts into structured data. It has been assumed that the input data are always given as numerical vectors in applying machine learning algorithms to applications in any domain. In almost cases of applying machine learning algorithms to text mining texts, texts or words are encoded into numerical vectors. In order to solve the problems in encoding so, there were previous trials of encoding them into alternative structured data to numerical vectors. In this section, we explore the previous schemes of encoding texts or words into structured data.

It is known that texts or words are usually encoded into numerical vectors, by surveying articles on machine learning approaches to text categorization. In 1995, Wiener encoded texts

into approximately three hundreds dimensional numerical vectors, in applying the neural networks to the text categorization[31]. In 1999, Yang encoded texts absolutely into numerical vectors, in evaluating machine learning algorithms in apply them to the text categorization [32]. In 2002, Sebastiani surveyed the machine learning approaches to the text categorization, under the assumption of encoding texts into numerical vectors [28]. The fact that texts should be encoded into numerical vectors for doing the text mining tasks is confirmed through the above literatures.

There existed the previous trials of encoding text into tables, instead of tables for doing the text categorization and clustering. In 2008, Jo modified the single pass algorithm into its table based version for the text clustering [8]. In 2011, Jo invented the table marching algorithm as the method of categorizing texts in his patent [14]. In 2015, Jo improved the table matching algorithm into the stable and robust version [15]. The table becomes the alterative text representation to the numerical vector for doing the both tasks.

There also existed the previous trials of encoding texts into string vectors as one more alternative structured data for doing the both tasks. In 2009, Jo proposed the semantic similarity between string vectors for using the KNN and the SVM for ding the text categorization [10]. In 2010, Jo modified the k means algorithm into its string vector based version as the approach to the text clustering [11]. In 2015, Jo defined and characterized mathematically the numerical operations on string as the fundamental research for modifying other machine learning algorithms into their string vector based versions [16]. It requires to define and characterize more semantic operations for modifying more advanced machine learning algorithms.

In this research, we adopt the scheme where texts are encoded into numerical vectors. The words are defined as features, and TF-IDF (Term Frequency-Inverse Document Frequency) and frequency are given as feature values. In order to avoid the poor discriminations among sparse numerical vectors, we need consider the similarity between words which is the feature similarity. The feature similarity and the feature value similarity are combined with each other for computing the similarity between vectors. We modify the KNN into the version where both similarities are computed and combined with each other as the approach to the text categorization.

Specialized machine learning algorithms

This section is concerned with the previous works on approaches to the text mining tasks where texts are encoded into alternative representations to numerical vectors. In Section II-C, we explored the previous schemes of encoding texts into structured data and the modified versions of existing machine learning algorithms using the scheme. In this section, we introduce the string kernel which is a kernel function on string in using the SVM to the text categorization, and mention the new neural networks, NTC (Neural Text Categorizer) and NTSO (Neural Text Self Organizer) for doing the text categorization and the text clustering. It takes time very much for applying the string kernel version to the text categorization and needs more operations for advancing the created neural networks. In this section, we explore cases of using the string kernel based SVM and the two creative neural networks to text mining tasks.

The string kernel was proposed as a kernel function in using the SVM for classification tasks, in order to avoid problems in encoding texts into numerical vectors. In 2002, Lodhi., *et al*. initially proposed the string kernel, in order solve the problems in doing so [23]. In 2004, Leslie., *et al*. applied the string kernel to protein classification where a protein is given as a string [22]. In 2006, Kate and Mooney used the string kernel for processing semantically sentences, instead of entire full texts [18]. The SVM with the string kernel works not successfully in the text categorization, but it works successfully in the protein classification.

The neural network, NTC (Neural Text Categorizer), was previously invented as a more suitable approach to the text categorization. In 2000, Jo initially proposed the NTC, but it used its weights which are fixed based on word frequencies [6]. In 2008 and 2010, Jo improved the NTC into the version where word weights are updated and applied it to both the exclusive text categorizations and the soft ones [9,12]. In 2012, Pawar and Gawande mentioned the NTC as innovative approach to the text categorization, and in 2015, Abainia., *et al*. applied it for categorizing Arabic texts [1,26]. In future, the NTC needs to be expanded from the swallow version into deep version.

The NTSO (Neural Text Self Organizer) was previously created as a more suitable approach to the text clustering. In 2005, Jo and Japkowicz invented initially as the approach to the text clustering [17]. In 2006, Zehng., *et al*. mentioned the NTSO as one of the main text clustering tools [33]. In 2010, Jo validated empirically the NTSO performance by comparing it with the popular approaches such as the k means algorithm and the Kohonen Networks [13]. The NTSO will be modified into its supervised version and semi- supervised version in future, like the Kohonen Networks.

In this research, we set the text categorization as the task against which we challenge. In the string kernel based SVM,

the lexical similarity between strings is computed, rather than semantic ones; it is more suitable for processing proteins than for doing texts. The two neural networks, NTC and NTSO, take actually much computation time and are very complicated with respect to implementations; they are usually used for implementing heavy versions of text categorization and clustering systems. It requires the mathematical definition and characterizations on more operations on strings for expanding them. Therefore, in this research, we propose the modified KNN version for implementing light versions.

Proposed approach

This section is concerned with the KNN (K Nearest Neighbor) algorithm which considers both the feature similarity and feature value one, and it consists of the three sections. In Section III-A, we describe the process of encoding texts into numerical vectors as the text pre-processing. In Section III-B, we present the equation with which the similarity between two numerical vectors is computed, considering the feature similarity. In Section III-C, we mention the proposed version where the similarity is computed by the proposed scheme with respect to its learning process. Therefore, this section is intended to describe the proposed version of KNN algorithm as the approach to the text categorization task.

Text encoding

This section is concerned with the text encoding where each text is mapped into a numerical vector. By indexing texts in a corpus, we extract words as feature candidates from it. We select some among the feature candidates as attributes of numerical vectors by some criteria. For each text, we assign values as an element of the numerical vector to each attribute. Therefore, in this section, we describe the process of defining the features and mapping each teach into a numerical vector.

Words are generated as feature candidates from the corpus through text indexing. Texts in the collection are concatenated into a big text and it is segmented into tokens by white spaces or punctuation marks. Each token is transformed into its root form through the stemming process; the verbs and nouns are transformed into their root forms and singular forms, respectively. For more efficiency, the stop words which functions only grammatically such as conjunctions, pronouns, and section, are removed from the stemmed tokens. A list of words which consists of verbs, nouns, and adjectives are extracted as the feature candidates.

Once some words are selected as attributes, we need to consider the schemes of defining a value to each attribute. To each attribute, we may assign a binary value indicating whether the word present in the text, or not. We may use the relative frequency of each word in the text. The weight of word to each attribute which is computed by equation (1) may be used as a feature value.

$$w_i = TF_i(\log_2 N - \log_2 DF_i + 1) \quad (1)$$

Where T F_i is the total frequency in the given text, DF_i is the total number of documents including the word, and N is the total number of documents in the corpus. Therefore, the attributes values of a numerical vector which represent a text are relationships between the word and the texts which are selected as features.

Once some words are selected as attributes, we need to consider the schemes of defining a value to each attribute. To each attribute, we may assign a binary value indicating whether the word present in the text, or not. We may use the relative frequency of each word in the text. The weight of word to each attribute which is computed by equation (1) may be used as a feature value. Therefore, the attributes values of a numerical vector which represent a text are relationships between the word and the texts which are selected as features.

The feature selection and the feature value assignment for encoding texts into numerical vectors depend strongly on the given corpus. When changing the corpus, different words are selected by different values of the selection criterion as features. Even if same features are selected, different feature values are assigned. Only addition or deletion of texts in the given corpus may influence on the feature selection and the assignment of feature values. In order to avoid the dependency, we may consider the word net or the dictionary as alternatives to the corpus.

Feature similarity

This section is concerned with the scheme of computing the similarity between numerical vectors as illustrated in figure 1. In this research, we call the traditional similarity measures such as cosine similarity and Euclidean distance feature value similarities where consider only feature values for computing it. In this research, we consider the feature similarity as well as the feature value similarity for computing it as the similarity measure which is specialized for text mining tasks. The numerical vectors which represent texts or words tend to be strongly sparse; only feature value similarity becomes easily fragile to the tendency. Therefore, in this section, as the solution to the problem, we describe the proposed scheme of computing the similarity between numerical vectors.

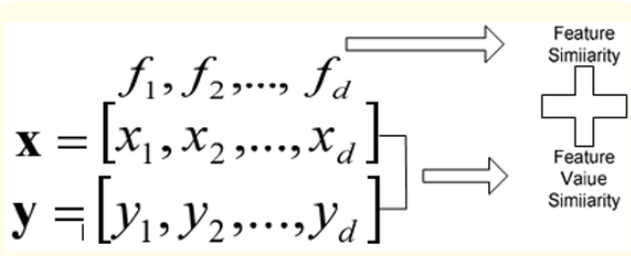

Figure 1: The Combination of Feature and Feature Value Similarity.

Words are given as features for encoding texts into numerical vectors. They are dependent on others rather than independent ones which are assumed in the traditional classifiers, especially in Naive Bayes [25]. Previously, various schemes of computing the semantic similarity between words were developed [24]. We need to assign nonzero similarity between two numerical vectors where non-zero elements are given to different features with their high similarity. It is expected to improve the discriminations among sparse vectors by considering the similarity among features.

We may build the similarity matrix among features automatically from a corpus. From the corpus, we extract easily a list of words. We compute the similarity between two texts by equation (2)

$$s_{ij} = sim(t_i, t_j) = \frac{2 \times df(t_i, t_j)}{df(t_i) + df(t_j)} \quad (2)$$

Figure a

where df (t_i, t_j) is the number of texts which include both words, t_i and t_j, and df (t_i) is the number of texts which includes the word, t_i. We build the similarity matrix which is consists of similarities between text identifiers given as features as follows:

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

Figure b

The rows and columns in the above matrix, S, correspond to the d words which are selected as the features.

The words, t_1, t_2, ..., td are given as the features, and the two texts, d_1 and d_2 are encoded into the two numerical vectors as follows:

d1 = [w11, w12, ..., w1d]
d2 = [w21, w22, ..., w2d].

The features, t_1, t_2, ..., td are defined through the process which was described in Section III-A. We construct the d by d matrix as the similarity matrix of features by the process mentioned above. The similarity between the two vectors are computed with the assumption of availability of the feature similarities, by equation (3)

$$sim(d_1, d_2) = \frac{\sum_{i=1}^{d} \sum_{j=1}^{d} s_{ij} w_{1i} w_{2j}}{d \cdot \|d_1\| \cdot \|d_2\|} \quad (3)$$

$$\|d_1\| = \sqrt{\sum_{i=1}^{d} w_{1i}^2} \text{ and } \|d_2\| = \sqrt{\sum_{i=1}^{d} w_{2i}^2}.$$

Figure c:

We get the value of s_{ij} by equation (2).

The proposed scheme of computing the similarity by equation (3) has very higher complexity as payment for obtaining the more discrimination among sparse vectors. Let us assume that two d dimensional numerical vectors are given as the input for computing the similarity between them. It takes only linear complexity, O(d), to compute the cosine similarity as the traditional one. However, in the proposed scheme takes the quadratic complexity, $O(d^2)$. We may reduce the complexity by computing similarities of some pairs of features, instead of all.

Proposed version of KNN

This section is concerned with the version of K Nearest Neighbor which considers both the feature similarity and the feature value one. The sample texts are encoded into numerical vectors whose features are texts by the scheme which was described in Section III-A. The novice text is given as the classification target, and it is also encoded into a numerical vector. Its similarities with the sample texts are computed by equation (3) for selecting nearest neighbors, in the proposed version. Therefore, in order to provide the detail algorithm, we describe the proposed KNN version, together with the traditional one.

The traditional KNN version is illustrated in figure 2. The sample texts 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 text with those representing sample texts are computed using the Euclidean distance or the cosine similarity. The k most similar sample texts 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.

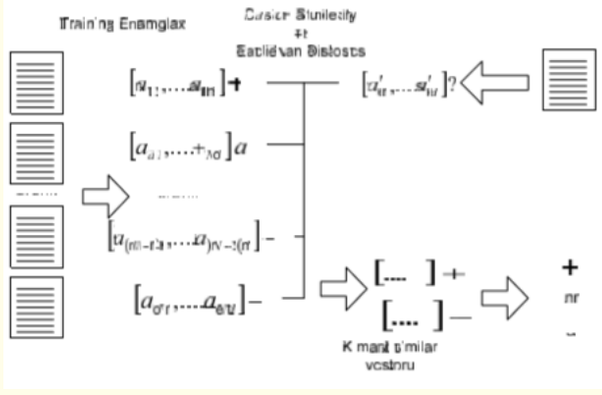

Figure 2: The Traditional Version of KNN.

The proposed KNN version is illustrated in figure 3. Like the traditional version, a text is given as an input and it is encoded into a numerical vector. The similarities of the novice text with the sample ones are computed by equation (3) which was presented in Section III-B. Like the traditional version, k most similar samples are selected as the nearest neighbors, and the label of the novice is decided by voting their labels. The scheme of computing the similarity between numerical vectors is the essential difference between the two versions.

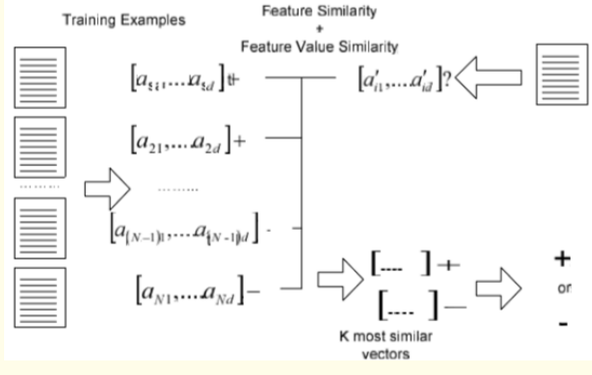

Figure 3: The Proposed Version of KNN.

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.

Let us compare the both KNN versions with each other. In computing the similarity between two numerical vectors, the traditional version uses the Euclidean distance or cosine similarity mainly, whereas the proposed one uses the equation (3). Both versions are common in selecting k nearest neighbors and classifying a novice item by voting the labels of them. However, the proposed version is more tolerant to sparse numerical vectors in computing the similarities among them than the traditional version.

Experiments

This section is concerned with the empirical experiments for validating the proposed version of KNN and consists of the three sections. In Section IV-A, we present the results from applying the proposed version of KNN to the text categorization on the collection, NewsPage.com. In Section IV-B, we show the results from applying it for categorizing texts from the collection, Opinosis.

NewsPage.com

This section is concerned with the experiments for validating the better performance of the proposed version on the collection: NewsPage.com. The four categories are predefined in this collection, and texts are gathered from the collection category by category as labeled ones. Each text is classified exclusively into one of the four categories. In this set of experiments, we apply the traditional and proposed version of KNN to the classification task, without decom- posing it into the binary classifications, and use the accuracy as the evaluation measure. Therefore, in this section, we observe the performance of the both versions of KNN by changing the input size.

In Table 1, we specify the text collection, NewsPage.com, which is used in this set of experiments. This text collection was used for evaluating approaches to text categorization in previous works [15]. In the collection, the four categories are predefined: Business, Health, Internet, and Sports, and 375 texts are selected at random

in each category. In each category, the set of 375 texts is partitioned into the 300 texts as training ones and the 75 texts as test ones. The text collection was built by copying and pasting individual news articles from the web site, newspage.com, in 2005, as plain text files whose extension is 'txt'.

Category	#Texts	#Training Texts	#Test Texts
Business	500	300	75
Health	500	300	75
Internet	500	300	75
Sports	500	300	75
Total	2000	1200	300

Table 1: The number of texts in newspage.com.

Let us mention the experimental process for validating empirically the proposed approach to the task of text categorization. In this collection, the texts are labeled with one of the four categories which are presented in Table 1, and they are encoded into numerical vectors. For each test example, the KNN computes its similarities with the 1200 training examples and selects the three most similarity training examples as its nearest neighbors. Each of the 300 test examples is classified into one of the four categories: Business, Sports, Internet, and Health, by voting the labels of its nearest neighbors. We compute the classification accuracy by dividing the number of correctly classified test examples by the number of test examples, for evaluating the both versions of KNN algorithm.

In Figure 4, we illustrate the experimental results from categorizing texts, using the both versions of KNN algorithm. The y-axis indicates the accuracy which is the rate of the correctly classified examples in the test set. In the x-axis, each group indicates the input size which is the dimension of numerical vectors which represent texts. In each group, the gray bar and the black bar indicate the achievements of the traditional version and the proposed version of KNN algorithm, respectively. In the x-axis, the most right group indicates the average over the accuracies of the left groups.

Let us make the discussions on the results from doing the text categorization using the both versions of KNN algorithm, as shown in Figure 4. The accuracy which is the performance measure of the classification task is in the range between 0.35 and 0.55. The proposed version of KNN algorithm works strongly better in the three input sizes 50, 100, and 200. The both versions matches with each other in the input size, 10. From this set of experiments, we conclude that the proposed version works strongly better than the traditional one, in averaging over the four cases.

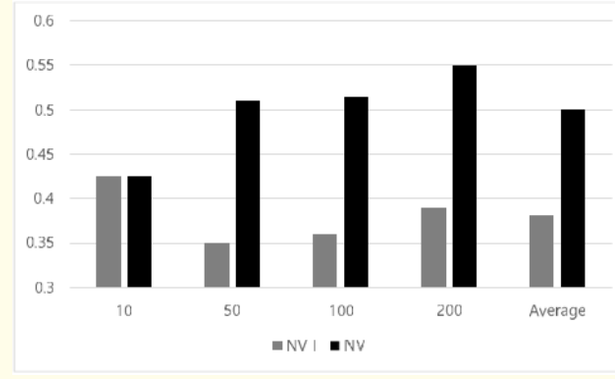

Figure 4: Results from Classifying Texts in Text Collection: News- Page.com.

Opinopsis

This section is concerned with the set of experiments for validating the better performance of the proposed version on the collection, Opinosis. The three categories are predefined in the collection, and labeled texts are prepared from it. Each text is classified exclusively into one of the three categories. We do not decompose the given classification into binary classifications and use the accuracy as the evaluation measure. Therefore, in this section, we observe the performances of the both versions of KNN algorithm with the different input sizes.

In Table II, we specify the text collection, Opinosis, which is used in this set of experiments. The collection was used in previous works for evaluating approaches to text categorization. The three categories, 'Car', 'Electronics', and 'Hotel' are predefined, and all texts are used for evaluating the approaches to text categorization, in this set of experiments. We use six texts in each cate- gory among all texts as the test set as shown in Table 2. We obtained the collection by downloading it from the web site, http://archive.ics.uci.edu/ml/machine-learning- databases/opinion/.

Category	#Texts	#Training Texts	#Test Texts
Car	23	17	6
Electronic	16	10	6
Hotel	12	6	6
Total	51	33	18

Table 2: The number of texts in opiniopsis.

We perform this set of experiments by the process which is described in Section IV-A. We use all of 51 texts which are labeled with one of the three categories and encode them into numerical vectors with the input sizes: 10, 50, 100, and 200. For each test

example, the both versions of KNN computes its similarities with the 33 training examples and select the three most similar training examples as its nearest neighbors. Each of the 18 test examples is classified into one of the three categories, by voting the labels of its nearest neighbors. The classification accuracy is computed by the number of correctly classified test examples by the number of the test examples for evaluating the both versions of KNN algorithm.

In Figure 5, we illustrate the experimental results from categorizing texts using the both versions of KNN algorithm. Like Figure 4, the y-axis indicates the value of accuracy, and the x-axis indicates the group of both versions by an input size. In each group, the gray bar and the black bar indicate the achievements of the traditional version and the proposed version of KNN algorithm, respectively. In Figure 5, the most right group indicates the averages over results over the left four groups. Therefore, Figure 5 presents the results from classifying each text into one of the three categories by the both versions, on the text collection, Opinosis.

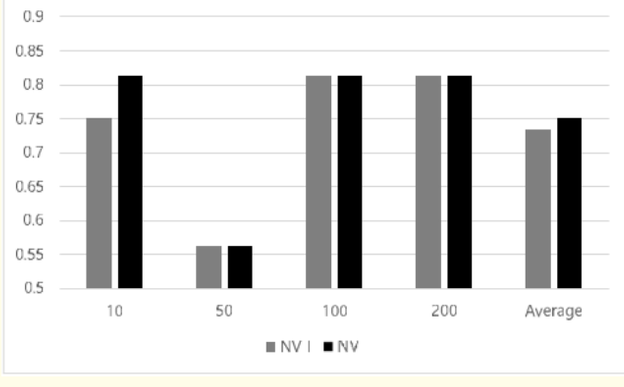

Figure 5: Results from Classifying Texts in Text Collection: Opiniopsis.

We discuss the results from doing the text categorization using the both versions of KNN algorithm, on Opinosis, shown in Figure 5. The accuracy values of the bother versions range between 0.55 and 0.8. The proposed version works better than the traditional one in the input size, 10.

It is comparable with the traditional version in the others. From this set of experiments, we conclude that the proposed version works slightly better than the traditional one, in averaging the four cases.

Conclusion

Let us discuss the entire results from classifying texts us- ing the two versions of KNN algorithm. The both versions are compared with each other in the task of text categorization, in these sets of experiments. The proposed version shows its better results in the both collections. The accuracies of the traditional version range between 0.35 and 0.81, while those of the proposed version range between 0.45 and 0.81. From the two sets of experiments, we conclude that the proposed version improves the text categorization performance, as the contribution of this research.

Let us consider the remaining tasks for doing the further research. The proposed approach should be applied and validated in the specialized domains: engineering, medicine, science, and law, and it should be customized to the suitable version. We may consider similarities among only some essential features rather than among all features, to cut down the computation time. We develop and combine various schemes of computing the similarities among features. By adopting the proposed approach, we will develop the text categorization system as a real version.

Acknowledgement

This work was supported by 2019 Hongik University Research Fund.

Bibliography

1. K Abainia., *et al*. "Neural Text Categorizer for topic identification of noisy Arabic Texts". Proceedings of 12th IEEE Conference on Computer Systems and Applications (2015): 1-8.

2. R Baeza-Yates and B Ribeiro-Neto. "Modern Information Retrieval: The Concepts and Technology behind Search". Addison- Wesley, (2011).

3. L Firte., *et al*. "Spam detection filter using KNN algorithm and resampling". Proceedings of IEEE International Conference on Intelligent Computer Communication and Processing (2010): 27-33.

4. E Han., *et al*. "Text categorization using weight adjusted k-nearest neighbor classification". Proceedings of Pacific-asia conference on knowledge discovery and data mining (2001): 53-65.

5. C James., *et al*. "A neural network based approach to automated e-mail classification". Proceedings of IEEE International Conferences on Web Intelligence (2003): 702-705.

6. T Jo. "Neuro Text Categorizer: A New Model of Neural Network for Text Categorization". *The Proceedings of ICONIP* (2000): 280-285.

7. T Jo. "The Implementation of Dynamic Document Organization using Text Categorization and Text Clustering". PhD Dissertation of University of Ottawa, (2006).

8. T Jo. "Table based Single Pass Algorithm for Clustering News Articles". *International Journal of Fuzzy Logic and Intelligent Systems* 8 (2008): 231-237.

9. T Jo. "Neural Text Categorizer for Exclusive Text Categorization". *Journal of Information Processing Systems* 4 (2008): 77-86.

10. T Jo. "Modification of Classification Algorithm in Favor of Text Categorization". *International Journal of Computer Science and Software Technology* 2 (2009): 13-23.

11. T Jo. "Modification of Clustering Algorithms for Text Clustering". *International Journal of Computer Science and Software Technology* 3 (2010): 21-33.

12. T Jo. "NTC (Neural Text Categorizer): Neural Network for Text Categorization". *International Journal of Information Studies* 2 (2010): 83-96.

13. T Jo. "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering". *Journal of Network Technology* 1 (2010): 31-43.

14. T Jo. "Device and Method for Categorizing Electronic Document Automatically, 10-2009-0041272, 10-1071495, (2011).

15. T Jo. "Normalized Table Matching Algorithm as Approach to Text Categorization". *Soft Computing* 19 (2015): 839-849.

16. T Jo. "Simulation of Numerical Semantic Operations on String in Text Collection". *International Journal of Applied Engineering Research* 10 (2015): 45585-45591.

17. T Jo and N Japkowicz. "Text Clustering using NTSO". *The Proceedings of IJCNN* (2005): 558- 563.

18. RJ Kate and RJ Mooney. "Using String Kernels for Learning Semantic Parsers". Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (2006): 913-920.

19. A Khan., *et al*. "A review of machine learning algorithms for text-documents classification". *Journal of Advances in Information Technology* 1 (2010): 4-20.

20. Y Kim., *et al*. "Collocation dictionary optimization using WordNet and k-nearest neighbor learning". *Machine Translation* 16 (2001): 99-108.

21. C Lai and M Tsai. "An empirical performance comparison of machine learning methods for spam e-mail categorization". Proceedings of IEEE International Conference on Hybrid Intelligent Systems (2004): 44-48.

22. CS Leslie., *et al*. "Mismatch String Kernels for Discriminative Protein Classification". *Bioinformatics* 20 (2004): 467-476.

23. H Lodhi., *et al*. "Text Classification with String Kernels". *Journal of Machine Learning Research* 2 (2002): 419-444.

24. CD Manning and H Schutze. Foundations of Statistical Natural Language Processing, MIT Press, (1999).

25. T Mitchell. Machine Learning, 1st edition McGraw-Hill, (1997).

26. PY Pawar and SH Gawande. "A Comparative Study on Different Types of Approaches to Text Categorization". *International Journal of Machine Learning and Computing* 2 (2012): 423-426.

27. V Pekar and S Staab. " Word classification based on combined measures of distributional and semantic similarity". Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics (2003): 147- 150.

28. F Sebastiani. "Machine learning in automated text categorization". *ACM Computing Survey* (2002): 1-47.

29. M Stauffer., *et al*. "A novel graph database for handwritten word images". Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (2016): 553-563.

30. B Vishwanath., *et al*. "KNN based machine learning approach for text and document mining". *International Journal of Database Theory and Application* 7 (2014): 61-70.

31. ED Wiener. "A Neural Network Approach to Topic Spotting in Text". Master Thesis, the Faculty of the Graduate School of the University of Colorado, (1995).

32. Y Yang. "An evaluation of statistical approaches to text categorization". *Information Retrieval* 1 (1999): 69-90.

33. Y Zheng., *et al*. "A comparative study on text clustering methods". *Advanced Data Mining and Applications* (2006): 644-651.

Volume 3 Issue 4 April 2019