Keyword Extraction by KNN considering Similarity among Features

Taeho Jo
Department of Computer and Information Engineering, Inha University, Incheon, South Korea

Abstract— *In this research, we propose that the K Nearest Neighbors should be used for extracting keywords from articles, considering the feature similarities. In the reality, the relations and the dependencies among features are available; the assumptions that the features are independent of each other violates against the reality. In this research, we define the similarity measure considering both feature values and features, interpret the keyword extraction into the binary classification where each word is classified into a keyword or non-keyword, and use it for modifying the K Nearest Neighbor. As the benefits from this research, we have the chance to represent words into the smaller dimensional numerical vectors and to improve the discriminations among vectors. Therefore, the goal of this research is to implement the keyword extraction systems with the benefits.*

Keywords: Keyword Extraction, Feature Similarity

1. Introduction

The keyword extraction refers to the process of extracting the essential words which reflect the entire content from an article. In this research, the keyword extraction is viewed into a binary classification where each word is classified into one of the two categories: 'keyword', or 'non-keyword'. We prepare sample words which are labeled with one of the two categories and encode them into their structured forms. By learning the sample words, we built the classification capacity, encode novice words into the structured forms, and classify them into one of the two categories. In this research, we use a supervised learning algorithm for the task which is viewed into the classification task.

Let us mention some problems which this research tries to solve. When discovering the dependencies among features of encoding texts or words into numerical vectors, the Bayesian networks was proposed as the approaches to the text mining tasks, but the complicated analysis is required for using them [1]. The assumption that features are independent of each other causes the requirement of many features for the robustness in encoding so. Since each feature has the very weak coverage, we cannot avoid the sparse distribution of each numerical vector where zero values are dominant with more than 95% [3]. Therefore, this research is intended to solve the problems by considering the feature similarity as well as the feature value similarity.

Let us mention what we propose in this research as its idea. In this research, we consider the both similarity measures, feature similarity and feature value similarity, for computing the similarity between numerical vectors. The keyword extraction is viewed into the binary classification where a supervised learning algorithm is applicable. The KNN (K Nearest Neighbor) is modified into the version which accommodates the both similarity measures and applied to the keyword extraction task. Therefore, the goal of this research is to improve the keyword extraction performance by solving the above problems.

We mention what we expect from this research as the benefits. Considering the both similarities which are covered in this research opens the potential way of reducing the dimensionality of numerical vectors for encoding texts. Computing the similarity between two texts by the two measures reflects more semantic similarity between words. It is expected to improve the discriminations among even sparse vectors by using the both kinds of similarities. Therefore, this research pursues the benefits for implementing the keyword extraction systems.

This article is organized into the four sections. In Section **??**, 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 [4]. In 2002, Lodhi et al. proposed the string kernel as a kernel function of raw texts in using the SVM (Support Vector Machine) to the text classification [5]. In 2004, Lesile et al. used the version of SVM which proposed by Lodhi et al. to the protein classification [6]. In 2004, Kate and Mooney used also the SVM version for classifying sentences by their meanings [7].

It was proposed that texts are encoded into tables instead of numerical vectors, as the solutions to the above problems. In

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

Previously, it was proposed that texts should be encoded into string vectors as other structured forms. In 2008, Jo modified the k means algorithm into the version which processes string vectors as the approach to the text clustering[12]. In 2010, Jo modified the two supervised learning algorithms, the KNN and the SVM, into the version as the improved approaches to the text classification [13]. In 2010, Jo proposed the unsupervised neural networks, called Neural Text Self Organizer, which receives the string vector as its input data [14]. In 2010, Jo applied the supervised neural networks, called Neural Text Categorizer, which gets a string vector as its input, as the approach to the text classification [15].

The above previous works proposed the string kernel as the kernel function of raw texts in the SVM, and tables and string vectors as representations of texts, in order to solve the problems. Because the string kernel takes very much computation time for computing their values, it was used for processing short strings or sentences rather than texts. In the previous works on encoding texts into tables, only table matching algorithm was proposed; there is no attempt to modify the machine algorithms into their table based version. In the previous works on encoding texts into string vectors, only frequency was considered for defining features of string vectors. Texts which are used as features of numerical vectors which represent words have their semantic similarities among them, so the similarities will be used for processing sparse numerical vectors, in this research.

3. Proposed Approach

This section is concerned with modifying the AHC (Agglomerative Hierarchical Clustering) algorithm into the version which considers the similarities among features as well as feature values, and it consists of the three sections. In Section 3.1, we describe the process of encoding words into numerical vectors. In Section 3.2, we do formally the proposed scheme of computing the similarity between two numerical vectors. In Section **??**, we mention the proposed version of AHC algorithm which considers the similarity among features as the approach to word clustering. Therefore, this article is intended to describe in detail the modified version of KNN algorithm and its application to the word clustering.

3.1 Word Encoding

This subsection is concerned with the process of encoding words into numerical vectors. Previously, texts each of which is consists of paragraphs were encoded into numerical vectors whose attributes are words. In this research, we attempt to encode words into numerical vectors whose attributes are text identifiers which include them. Encoding of words and texts into numerical vectors looks reverse to each other. In this Section, we describe in detail the process of mapping words into numerical vectors, instead of texts.

In the first step of word encoding, a word-document matrix is constructed automatically from a text collection called corpus. In the corpus, each text is indexed into a list of words. For each word, we compute and assign its weight which is called TF-IDF (Term Frequency-Inverse Document Frequency) weight [2], by equation (1),

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

where TF_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. The word-document matrix consists of TF-IDF weights as relations between a word and a document computed by equation (1). Note that the matrix is a very huge one which consists at least of several thousands of words and documents.

Let us consider the criterion of selecting text identifiers as features, given labeled sampled words and a text collection. We may set a portion of each text in the given sample words as a criteria for selecting features. We may use the total frequency of the sample words in each text as a selection criterion. However, in this research, we decided the total TF-IDF (Term Frequency and Inverse Document Frequency) which is computed by equation (1) as the criterion. We may combine more than two criteria with each other for selecting features.

Once some texts 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 which is given as the attribute, or not. We may use the relative frequency of the word in each text which is an attribute as a feature value. 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 word are relationships between the word and the texts which are selected as features.

The feature selection and the feature value assignment for encoding words into numerical vectors depend strongly on the given corpus. When changing the corpus, different texts 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.

3.2 Feature Similarity

This subsection 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 subsection, as the solution to the problem, we describe the proposed scheme of computing the similarity between numerical vectors.

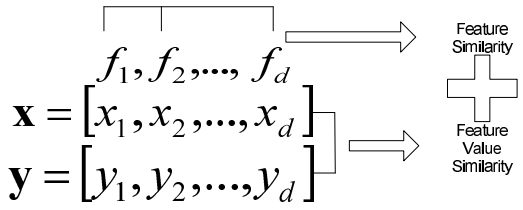

Fig. 1
THE COMBINATION OF FEATURE AND FEATURE VALUE SIMILARITY

Text identifiers are given as features for encoding words into numerical vectors. Texts are dependent on others rather than independent ones which are assumed in the traditional classifiers, especially in Naive Bayes [1]. Previously, various schemes of computing the semantic similarity between texts were developed [2]. 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 text identifiers. We compute the similarity between two texts by equation (2),

$$s_{ij} = sim(d_i, d_j) = \frac{2 \times tf(d_i, d_j)}{tf(d_i) + tf(d_j)} \quad (2)$$

where $tf(d_i, d_j)$ is the number of words which are shared by both texts, d_i and d_j, and $tf(d_i)$ is the number of words which are included in the text, d_i. We build the similarity matrix which is consists of similarities between text identifiers given as features as follows:

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

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

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

$$t_1 = [w_{11}, w_{12}, ..., w_{1d}]$$

$$t_2 = [w_{21}, w_{22}, ..., w_{2d}].$$

The features,$d_1, d_2, ..., d_d$ are defined through the process which was described in Section 3.1. 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),

$$sim(t_1, t_2) = \frac{\sum_{i=1}^{d} \sum_{j=1}^{d} s_{ij} w_{1i} w_{2j}}{d \cdot \|t_1\| \cdot \|t_2\|} \quad (3)$$

where $\|t_1\| = \sqrt{\sum_{i=1}^{d} w_{1i}^2}$ and $\|t_2\| = \sqrt{\sum_{i=1}^{d} w_{2i}^2}$. We get the value of s_{ij} by equation (2).

The proposed scheme of computing the similarity by equation (3) has the 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.

3.3 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 words are encoded into numerical vectors whose features are texts by the scheme which was described in section 3.1. The novice word is given as the classification target, and it is also encoded into a numerical vector. Its similarities with the sample words 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 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.

The proposed KNN version is illustrated in Figure 3. Like the traditional version, a word is given as an input and it is encoded into a numerical vector. The similarities of the novice word with the sample ones are computed by equation (3) which was presented in section 3.2. 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

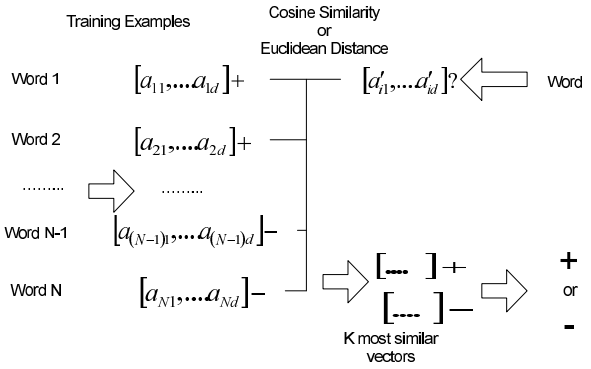

Fig. 2
THE TRADITIONAL VERSION OF KNN

numerical vectors is the essential difference between the two versions.

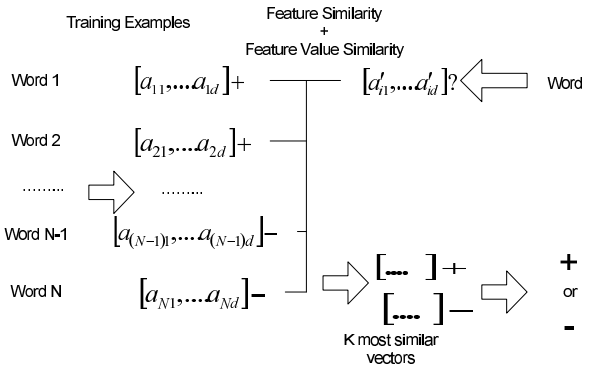

Fig. 3
THE PROPOSED VERSION OF KNN

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.

3.4 The Application to Keyword Extraction

This section is concerned with the scheme of applying the proposed KNN version which was described in section 3.3 to the keyword extraction 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 'keyword' or 'not' as the sample data. The words are encoded into numerical vectors by the scheme which was described in section 3.2. Therefore, in this section, we describe the process of extracting keywords from texts automatically using the proposed KNN with the view of the keyword extraction into a classification task.

In this research, the keyword extraction is viewed into a binary classification task, as shown in Figure 4. 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 either of two labels: 'keyword' or 'not'. The words which are classified into 'keyword' are selected as the output of the keyword extraction system. For doing so, we need to collect words which are labeled with one of the two labels as sample examples, in advance.

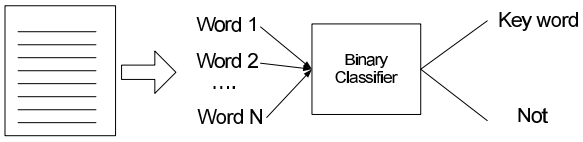

Fig. 4
MAPPING OF KEYWORD EXTRACTION INTO BINARY CLASSIFICATION

We need to prepare sample words which are labeled with 'keyword' or 'not', 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.

We mention the process where an article is given as the input and a list of keywords is generated as the output. We nominate the classifier which corresponds to the sub-group which is similar as the given article, based on its content. A list of words is extracted by indexing the article, and each word is encoded into structured forms. The extracted words are classified by the nominated classifier into 'keyword' or 'not', and the words which are classified into the former are selected. The performance depends on the granularity of each sub-group; it should be optimized between the two factors: the amount of sample examples and the subgroup granularity.

Even if the keyword extraction 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 keyword extraction is fixed only to a single binary classification. In the word categorization, each word is classified semantically into one or some of the predefined topics, whereas in the keyword extraction, it is classified into an essential word, or not. In the word categorization, each word is classified by its meaning, whereas in the keyword extraction, it is classified by its relevancy 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 keyword extraction, a classifier is done to each domain.

4. Conclusion

Let us mention the remaining tasks for doing the further research. We need to validate the proposed approach in specific domains such as medicine, engineering, and economics, as well as in generic domains such as ones of news articles. We may consider the computation of similarities among some main features rather than among all features for reducing the computation time. We try to modify other machine learning algorithms such as Naive Bayes, Perceptrons, and SVM (Support Vector Machine) based on both kinds of similarities. By adopting the proposed approach, we may implement the word clustering system as a real program.

References

[1] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
[2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology behind Search*, Addison-Wesley, 2011.
[3] T. Jo, “The Implementation of Dynamic Document Organization using Text Categorization and Text Clustering" PhD Dissertation, University of Ottawa, Ottawa, Canada, 2006.
[4] F. Sebastiani, “Machine Learning in Automated Text Categorization", *ACM Computing Survey*, Vol. 34, pp. 1-47, 2002.
[5] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, “Text Classification with String Kernels", *Journal of Machine Learning Research*, Vol. 2, pp. 419-444, 2002.
[6] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, “Mismatch String Kernels for Discriminative Protein Classification", *Bioinformatics*, Vol. 20, pp. 467-476, 2004.
[7] R. J. Kate and R. J. Mooney, “Using String Kernels for Learning Semantic Parsers", in *Proc. ICCL ’06*, 2006, pp. 913-920.
[8] T. Jo and D. Cho, “Index based Approach for Text Categorization", *International Journal of Mathematics and Computers in Simulation*, Vol. 2, 2008, pp. 127-132.
[9] T. Jo, “Single Pass Algorithm for Text Clustering by Encoding Documents into Tables", *Journal of Korea Multimedia Society*, Vol. 11, 2008, pp. 1749-1757.
[10] T. Jo, “Device and Method for Categorizing Electronic Document Automatically", South Korean Patent 10-1071495, 2011.
[11] T. Jo, “Normalized Table Matching Algorithm as Approach to Text Categorization", *Soft Computing*, Vol. 19, 2015, pp. 849-849.
[12] T. Jo, “Inverted Index based Modified Version of K-Means Algorithm for Text Clustering", *Journal of Information Processing Systems*, Vol. 4, 2008, pp. 67-76.
[13] T. Jo, “Representationof Texts into String Vectors for Text Categorization", *Journal of Computing Science and Engineering*, Vol. 4, 2010, pp. 110-127.
[14] T. Jo, “NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", *Journal of Network Technology*, Vol. 1, 2010, pp. 31-43.
[15] T. Jo, “NTC (Neural Text Categorizer): Neural Network for Text Categorization", *International Journal of Information Studies*, Vol 2, 2010, pp. 83-96.