Using String Vector based KNN for Keyword Extraction

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

Abstract—In this research, we propose the string vector based KNN as the approach to the keyword extraction. The keyword extraction may be viewed as an instance of word classification, encoding words into numerical vectors may cause the main problems, such as the huge dimensionality, the sparse distribution and the poor transparency, and the problems were solved by encoding texts into string vectors in previous works on text mining tasks. In this research by these motivations, we encode words into string vectors, define the semantic operation on string vectors, and modify the K Nearest neighbor into its string vector based version which is used for the keyword extraction. As the benefits from this research, we expect the better performance and more compact representations than encoding words or texts into numerical vectors. Hence, the goal of this research is to implement the keyword extraction system with the benefits.

Keywords: Keyword Extraction, String Vector, K Nearest Neighbor

1. Introduction

Keyword extraction refers to the process of extracting important words which are called keywords, from an article. The keywords are important indications for performing the information retrieval tasks, so we are interested very much in developing the schemes of extracting them. In this research, the keyword extraction is viewed into a binary word classification where each word is classified into a keyword or a non-keyword. We prepare the sample words which are labeled with 'keyword' or 'non-keyword', and construct the classification capacity by learning them. In this research, we assume that the supervised learning algorithms are used as the approach to the task, even if other types of approaches are available.

We mention some challenges with which this research attempts to tackle. In encodings texts or words into numerical vectors for using the traditional classification algorithms, many features are required, since each feature has very weak coverage[1]. Each numerical vector which represents a text or a word tends to be very sparse; it includes zero values dominantly[5][9]. Even if we proposed that texts or words should be encoded into tables in previous works, it was very expensive to compute the similarity between tables[5][9]. Therefore, in this research, we challenge against the above problems by encoding words into string vectors.

Let us consider some ideas which are proposed in this research. In this research, words are encoded into string vectors which consist of a finite ordered set of text identifiers as alternative representations to numerical vectors. We define the similarity measure between string vectors which is always given as a normalized value between zero and one; it corresponds to the cosine similarity between numerical vectors. The KNN (K Nearest Neighbor) is modified into the string vector based version, and applied to the special instance of word classification which is mapped from the keyword extraction. Note that in this research, the keyword extraction task is interpreted into the classification task.

Let us consider some benefits which are expected from this research. It is expected that string vectors are more compact representations of words which have much less features than numerical vectors. We expect the much better discriminations among string vectors than those among numerical vectors, since the sparse distributions can be avoided almost completely in each string vector. In this research, we expect also the improved performance by solving the above problems in encoding words into numerical vectors. Therefore, the goal of this research is to implement the keyword extraction systems which have the above benefits.

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

2. Previous Works

Let us survey the previous cases of encoding texts into structured forms for using the machine learning algorithms to text mining tasks. The three main problems, huge dimensionality, sparse distribution, and poor transparency, have existed inherently in encoding them into numerical vectors. In previous works, various schemes of preprocessing texts have been proposed, in order to solve the problems. In this survey, we focus on the process of encoding texts into alternative structured forms to numerical vectors. In other words, this section is intended to explore previous works on solutions to the problems.

Let us mention the popularity of encoding texts into numerical vectors, and the proposal and the application of string kernels as the solution to the above problems. In 2002, Sebastiani presented the numerical vectors are the standard
representations of texts in applying the machine learning algorithms to the text classifications [1]. 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 [2]. In 2004, Lesile et al. used the version of SVM which proposed by Lodhi et al. to the protein classification [3]. In 2004, Kate and Mooney used also the SVM version for classifying sentences by their meanings [4].

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 [5]. In 2008, Jo applied also his proposed approach to the text clustering, as well as the text categorization [9]. In 2011, Jo described as the technique of automatic text classification in his patent document [7]. In 2015, Jo improved the table matching algorithm into its more stable version [8].

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[9]. 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 [10]. In 2010, Jo proposed the unsupervised neural networks, called Neural Text Self Organizer, which receives the string vector as its input data [11]. 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 [12].

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, based on [10], we consider the grammatical and posting relations between words and texts as well as the frequencies for defining the features of string vectors, and encode words into string vectors in this research.

3. Proposed Approach

This section is concerned with encoding words into string vectors, modifying the KNN (K Nearest Neighbor) into the string vector based version and applying it to the keyword extraction, and consists of the four sections. In Section 3.1, we deal with the process of encoding words into string vectors. In Section 3.2, we describe formally the similarity matrix and the semantic operation on string vectors. In Section 3.3, we do the string vector based KNN version as the approach to the keyword extraction. In Section 3.4, we focus on the process of applying the KNN to the given task with viewing it into a classification task.

3.1 Word Encoding

This section is concerned with the process of encoding words into string vectors. The three steps are involved in doing so, as illustrated in Figure 1. A single word is given as the input, and a string vector which consists of text identifiers is generated as the output. We need to prepare a corpus which is a collection of texts for encoding words. Therefore, in this section, we will describe each step of encoding the words.

![Fig. 1: Overall Process of Word Encoding](image)

The first step of encoding words into string vectors is to index the corpus into a list of words. The texts in the corpus are concatenated into a single long string and it is tokenized into a list of tokens. Each token is transformed into the root form, using stemming rules. Among them, the stop words which are grammatical words such as propositions, conjunctions, and pronouns, irrelevant to text contents are removed for more efficiency. From the step, verbs, nouns, and adjectives are usually generated as the output. The inverted list where each word is linked to the list of texts which include it is illustrated in Figure 2. A list of words is generated from a text collection by indexing each text. For each word, by retrieving texts which include it, the inverted list is constructed. A text and a word are associated with each other by a weight value as the relationship between them. The links of each word with a list of texts is opposite to those of each text with a list of words becomes the reason of call the list which is presented in Figure 2, inverted list.

Each word is represented into a string vector based on the inverted index which is shown in Figure 3. In this research, we define the features which are relations between texts and words as follows:

- Text identifier which has its highest frequency among the text collection
- Text identifier which has its highest TF-IDF weight among the text collection
- Text identifier which has its second highest frequency among the text collection
Text identifier which has its second highest TF-IDF weight among the text collection
Text identifier which has its highest frequency in its first paragraph among text collection
Text identifier which has its highest frequency in its last paragraph among text collection
Text identifier which has its highest TF-IDF weight in its first paragraph among text collection
Text identifier which has its highest TF-IDF weight in its last paragraph among text collection

We assume that each word is linked with texts including their own information: its frequencies and its weights in the linked texts and their first and last paragraphs. From the inverted index, we assign the corresponding values which are given as text identifiers to each feature. Therefore, the word is encoded into an eight dimensional string vector which consists of eight strings which indicate text identifiers.

Let us consider the differences between the word encoding and the text encoding. Elements of each string vector which represents a word are text identifiers, whereas those of one which represents a text are word. The process of encoding texts involves the link of each text to a list of words, where as that of doing words does the link of each word to a list of texts. For performing semantic similarity between string vectors, in text processing, the word similarity matrix is used as the basis, while in word processing, the text similarity matrix is used. The relations between words and texts are defined as features of strings in encoding texts and words.

3.2 String Vectors

This section is concerned with the operation on string vectors and the basis for carrying out it. It consists of two subsections and assumes that a corpus is required for performing the operation. In Section 3.2.1, we describe the process of constructing the similarity matrix from a corpus. In Section 3.2.2, we define the string vector formally and characterize the operation mathematically. Therefore, this section is intended to describe the similarity matrix and the operation on string vectors.

3.2.1 Similarity Matrix

This subsection is concerned with the similarity matrix as the basis for performing the semantic operation on string vectors. Each row and column of the similarity matrix corresponds to a text in the corpus. The similarities of all possible pairs of texts are given as normalized values between zero and one. The similarity matrix which we construct from the corpus is the $N \times N$ square matrix with symmetry elements and 1’s diagonal elements. In this subsection, we will describe formally the definition and characterization of the similarity matrix.

Each entry of the similarity matrix indicates a similarity between two corresponding texts. The two documents, $d_i$ and $d_j$, are indexed into two sets of words, $D_i$ and $D_j$. The similarity between the two texts is computed by equation (1),

$$sim(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|}$$

where $|D_i|$ is the cardinality of the set, $D_i$. The similarity is always given as a normalized value between zero and one; if two documents are exactly same to each other, the similarity becomes 1.0 as follows:

$$sim(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} = 1.0$$

and if two documents have no shared words, $D_i \cap D_j = \emptyset$ the similarity becomes 0.0 as follows:

$$sim(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} = 0.0$$

The more advanced schemes of computing the similarity will be considered in next research.

From the text collection, we build $N \times N$ square matrix as follows:

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

$N$ individual texts which are contained in the collection correspond to the rows and columns of the matrix. The entry, $s_{ij}$ is computed by equation (1) as follows:

$$s_{ij} = sim(d_i, d_j)$$

The overestimation or underestimation by text lengths are prevented by the denominator in equation (1). To the number of texts, $N$, it costs quadratic complexity, $O(N^2)$, to build the above matrix.

Let us characterize the above similarity matrix, mathematically. Because each column and row corresponds to its same text in the diagonal positions of the matrix, the diagonal elements are always given 1.0 by equation (1).
In the off-diagonal positions of the matrix, the values are always given as normalized ones between zero and one, because of $0 \leq 2|D_i \cap D_j| \leq |D_i| + |D_j|$ from equation (1). It is proved that the similarity matrix is symmetry, as follows:

$$s_{ij} = \text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} = \frac{|D_j \cap D_i|}{|D_j| + |D_i|}$$

$$= \text{sim}(d_j, d_i) = s_{ji}$$

Therefore, the matrix is characterized as the symmetry matrix which consists of the normalized values between zero and one.

The similarity matrix may be constructed automatically from a corpus. The N texts which are contained in the corpus are given as the input and each of them is indexed into a list of words. All possible pairs of texts are generated and the similarities among them are computed by equation (1). By computing them, we construct the square matrix which consists of the similarities. Once making the similarity matrix, it will be used continually as the basis for performing the operation on string vectors.

### 3.2.2 String Vector and Semantic Similarity

This section is concerned with the string vectors and the operation on them. A string vector consists of strings as its elements, instead of numerical values. The operation on string vectors which we define in this subsection corresponds to the cosine similarity between numerical vectors. Afterward, we characterize the operation mathematically. Therefore, in this section, we define formally the semantic similarity as the semantic operation on string vectors.

The string vector is defined as a finite ordered set of strings as follows:

$$\text{str} = [\text{str}_1, \text{str}_2, ..., \text{str}_d]$$

An element in the vector, $\text{str}_i$, indicates a text identifier which corresponds to its attribute. The number of elements of the string vector, $\text{str}$ is called its dimension. In order to perform the operation on string vectors, we need to define the similarity matrix which was described in Section 3.2.1, in advance. Therefore, a string vector consists of strings, while a numerical vector does of numerical values.

We need to define the semantic operation which is called ‘semantic similarity’ in this research, on string vectors; it corresponds to the cosine similarity on numerical vectors. We note the two string vectors as follows:

$$\text{str}_1 = [\text{str}_{11}, \text{str}_{12}, ..., \text{str}_{1d}]$$

$$\text{str}_2 = [\text{str}_{21}, \text{str}_{22}, ..., \text{str}_{2d}]$$

where each element, $d_{1i}$ and $d_{2i}$, indicates a text identifier. The operation is defined as equation (3.2.2) as follows:

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^{d} \text{sim}(d_{1i}, d_{2i})$$

The similarity was constructed by the scheme which is described in Section 3.2.1, and the $\text{sim}(d_{1i}, d_{2i})$ is computed by looking up it in the similarity matrix. Instead of building the similarity matrix, we may compute the similarity, interactively.

The semantic similarity measure between string vectors may be characterized mathematically. The commutative law applies as follows:

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^{d} \text{sim}(d_{1i}, d_{2i})$$

$$= \frac{1}{d} \sum_{i=1}^{d} \text{sim}(d_{2i}, d_{1i}) = \text{sim}(\text{str}_2, \text{str}_1)$$

If the two string vectors are exactly same, its similarity becomes 1.0 as follows:

if $\text{str}_1 = \text{str}_2$ with $\forall i, \text{sim}(d_{1i}, d_{2i}) = 1.0$

then $\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^{d} \text{sim}(d_{1i}, d_{2i}) = \frac{d}{d} = 1.0$

However, note that the transitive rule does not apply as follows:

if $\text{sim}(\text{str}_1, \text{str}_2) = 0.0$ and $\text{sim}(\text{str}_2, \text{str}_3) = 0.0$

then, not always $\text{sim}(\text{str}_1, \text{str}_3) = 0.0$

We need to define the more advanced semantic operations on string vectors for modifying other machine learning algorithms. We define the update rules of weights vectors which are given as string vectors for modifying the neural networks into their string vector based versions. We develop the operations which correspond to computing mean vectors over numerical vectors, for modifying the k means algorithms. We consider the scheme of selecting representative vector among string vectors for modifying the k medoid algorithms so. We will cover the modification of other machine learning algorithms in subsequent researches.

### 3.3 Proposed Version of KNN

This section is concerned with the proposed KNN version as the approach to the text categorization. Raw texts are encoded into string vectors by the process which was described in Section 3.1. In this section, we attempt to the traditional KNN into the version where a string vector is given as the input data. The version is intended to improve the classification performance by avoiding problems from encoding texts into numerical vectors. Therefore, in this section, we describe the proposed KNN version in detail, together with the traditional version.

The traditional KNN version is illustrated in Figure 3. 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.

Because string vectors are characterized more symbolically than numerical vectors, it is easy to trace results from classifying items in the proposed version. It is assumed that a novice item is classified by voting the labels of its nearest neighbors. The similarity between string vectors is computed by the scheme which is described in Section 3.2.2. We may extract the similarities of individual elements of the novice string vector with those of nearest neighbors labeled with the classified category. Therefore, the semantic similarities play role of the evidence for presenting the reasons of classifying the novice one so.

3.4 Application to Keyword Extraction

This section is concerned with the scheme of applying the proposed KNN version which was described in Section ?? 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 tables by the scheme which was described in Section ?? 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 5. 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.

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. The proposed approach should be validated and specialized in the specific domains: medicine, engineering and economics. Other features such as grammatical and posting features may be considered for encoding words into string vectors as well as text identifiers. Other machine learning algorithms as well as the KNN may be modified into their string vector based versions. By adopting the proposed version of the KNN, we may implement the keyword extraction system as a real program.

5. Acknowledgement

This work was supported by 2016 Hongik University Research Fund.

References