String Vector based AHC as Approach to Word Clustering

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

Abstract— *In this research, we propose the string vector based AHC (Agglomerative Hierarchical Clustering) algorithm as the approach to the word clustering. In the previous works on text clustering, it was successful to encode texts into string vectors by improving the performance of text clustering; it provided the motivation of doing this research. In this research, we encode words into string vectors, define the semantic operation on string vectors, and modify the AHC algorithm into its string vector based version. As the benefits from this research, we expect the improved performance and more compact representations of words. Hence, the goal of this research is to implement the word clustering system with the benefits.*

Keywords: Word Clustering, String Vector, AHC Algorithm

1. Introduction

Word clustering refers to the process of segmenting a group of various words into subgroups of content based similar words. In the task, a group of arbitrary words is given as the input, and they are encoded into their structured forms. The similarity measure between the structured forms which represent the words is defined and the similarities among them are computed. The words are arranged into subgroups based on their similarities. In this research, we assume that the unsupervised learning algorithms are used for the task, although other types of approaches exist.

Let us mention the challenges which this research attempts to tackle with. In encoding words into numerical vectors for using the traditional clustering algorithms, we need many features for the robust clustering, since each feature has very weak coverage[1]. Each numerical vector which represents a word or a text tends to have the sparse distribution where zero values are dominant with more than 90%[5][9]. In previous works, we proposed that texts or words should be encoded into tables as alternative representations to numerical vectors, but it is very expensive to compute the similarity between tables[5][9]. Hence, in this research, we solve the problems by encoding words into string vectors.

Let us mention what is proposed in this research as its idea. We encode words into string vectors where elements are text identifiers which are given as symbols or codes as alternative representations to numerical vectors. We define the operation on string vectors which corresponds to the cosine similarity between numerical vectors as the similarity measure between them. We modify the AHC (Agglomerative Hierarchical Clustering) algorithm into the version where input data are given as string vectors. Hence, in this research, the words are clustered by the modified version of AHC algorithm.

Let us consider the benefits which are expected from this research. The string vectors become the more compact representations of words than numerical vectors; it requires much less features in encoding words. We expect the improved discriminations among string vectors since there is very few sparse distributions in string vectors. We expect the improved clustering performance by solving the problems which are caused by encoding words into numerical vectors. Therefore, the goal of this research is to implement the word clustering systems which are improved by the 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 AHC (Agglomerative Hierarchical Clustering) algorithm into string vector based version and applying it to the word clustering, and consists of the three 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 AHC version as the approach to the word clustering. Therefore, this article is intended to describe the proposed AHC version as the word clustering tool.

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.

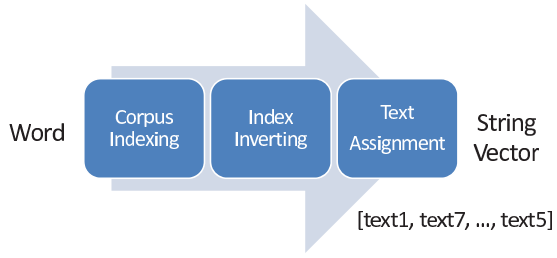

Fig. 1: Overall Process of Word Encoding

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 its 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.

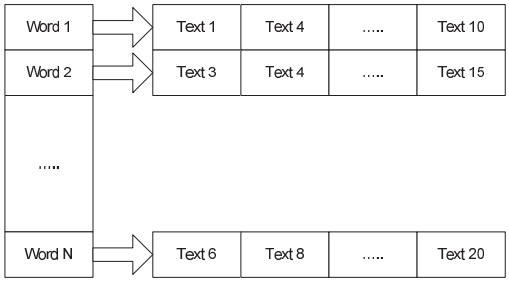

Fig. 2: The Inverted Index

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),

$$sim(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} \tag{1}$$

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:

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

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:

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

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 $0 \leq 2|D_i \cap D_i| \leq |D_i| + |D_j|$ from equation (1). It is proved that the similarity matrix is symmetry, as follows:

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

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:

$$\mathbf{str} = [str_1, str_2, ..., str_d]$$

An element in the vector, str_i indicates a text identifier which corresponds to its attribute. The number of elements of the string vector, $\mathbf{str}$ 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:

$$\mathbf{str}_1 = [str_{11}, str_{12}, ..., str_{1d}]$$

$$\mathbf{str}_2 = [str_{21}, str_{22}, ..., str_{2d}]$$

where each element, d_{1i} and d_{21i} indicates a text identifier. The operation is defined as equation (3.2.2) as follows:

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

The similarity matrix was constructed by the scheme which is described in Section 3.2.1, and the $sim(d_{1i}, d_{2i})$ 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:

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

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

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

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

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

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

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

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

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 The Proposed Version of AHC Algorithm

This section is concerned with the proposed AHC version as the approach to the text clustering. Raw texts are encoded into string vectors by the process which was described in Section 3.2.1. In this section, we attempt to the traditional AHC into the version where a string vector is given as the input data. The version is intended to improve the clustering performance by avoiding problems from encoding texts into numerical vectors. Therefore, in this section, we describe the proposed AHC version in detail, together with the traditional version.

The traditional version of AHC algorithm is illustrated in Figure 3. Words are encoded into numerical vectors, and it begins with unit clusters each of which has only single item. The similarity of every pairs of clusters is computed using the Euclidean distance or the cosine similarity, and the pair with its maximum similarity is merged into a cluster. The clustering by the ACH algorithm proceeds by merging cluster pairs and decrementing number of clusters by one. If the similarities among the sparse numerical vectors are computed, the traditional version becomes very fragile from the poor discriminations among them.

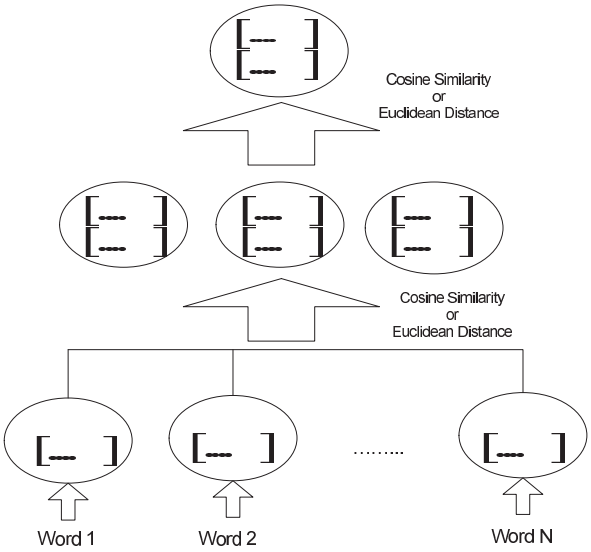

Fig. 3: The Traditional Version of AHC Algorithm

Separately from the traditional version, the clustering process by the proposed AHC version is illustrated in Figure 4. Texts are encoded into string vectors, and the algorithm begins with unit clusters each of which has a single one. The similarities of all possible pairs of clusters are computed and the pair with its maximum similarity is merged into a single cluster. The clustering proceeds by iterating the process of computing the similarities and merging a pair. Because the sparse distribution in each string vector is never available inherently, the poor discriminations by sparse distributions are certainly overcome in this research.

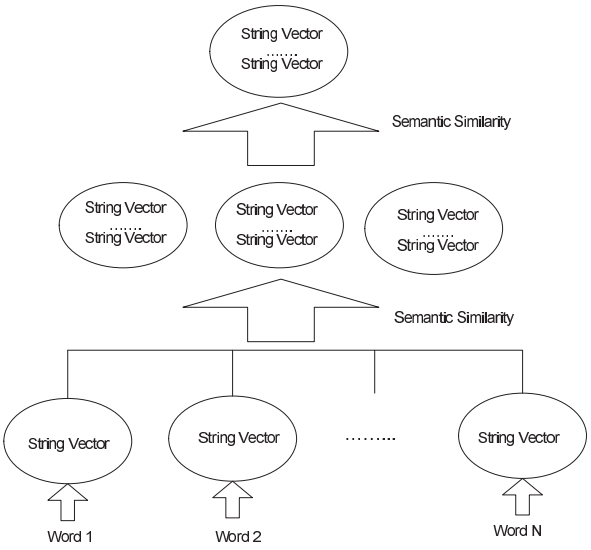

Fig. 4: The Proposed Version of AHC Algorithm

We may consider several schemes of computing a similarity between clusters. We may compute similarities of all possible pairs of items between two clusters and average over them as the cluster similarity. The maximum or the minimum among similarities of all possible pairs is set as the cluster similarity. In another scheme, we may select representative members of two clusters and the similarity between the selected members is regarded as the cluster similarity. In this research, we adopt the first scheme for computing the similarity between two clusters in using the AHC algorithm; other schemes will be considered in next research.

Because string vectors are characterized more symbolically than numerical vectors, it is easy to trace results from clustering items in the proposed version. It is assumed that the clustering proceeds by merging pairs of clusters or data items by their similarities as shown in Figure 3 and 4. The similarity between string vectors is computed by the scheme which is described in Section 3.2.2. A particular entity is nominated and its elements are extracted from it. We present the evidence of clustering entities by associating it with elements from other string vectors within the belonging clustering.

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 AHC may be modified into their string vector based versions. By adopting the proposed version of the AHC, we may implement the word clustering system as a real program.

5. Acknowledgement

This work was supported by 2016 Hongik University Research Fund.

References

[1] F. Sebastiani, “Machine Learning in Automated Text Categorization", pp1-47, ACM Computing Survey, Vol 34, No 1, 2002.
[2] 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.
[3] 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.
[4] 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.
[5] T. Jo and D. Cho, “Index based Approach for Text Categorization",International Journal of Mathematics and Computers in Simulation, Vol 2, No 1, 2008.
[6] 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.

[7] T. Jo, “Device and Method for Categorizing Electronic Document Automatically", Patent Document, 10-2009-0041272, 10-1071495, 2011.
[8] T. Jo, “Normalized Table Matching Algorithm as Approach to Text Categorization", pp839-849, Soft Computing, Vol 19, No 4, 2015.
[9] 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.
[10] T. Jo, “Representationof Texts into String Vectors for Text Categorization", pp110-127, Journal of Computing Science and Engineering, Vol 4, No 2, 2010.
[11] 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.
[12] T. Jo, “NTC (Neural Text Categorizer): Neural Network for Text Categorization", pp83-96, International Journal of Information Studies, Vol 2, No 2, 2010.