

Table based AHC Algorithm for Clustering Words

Taeho Jo

*School of Computer and Information Engineering
Inha University
Incheon, South Korea
tjo018@inha.ac.kr*

Abstract—This research proposes the table based AHC algorithm as the approach to the word clustering task. The results from encoding texts into tables were successful in the previous works on the text categorization and the text clustering, and if oppositely to the case of the text encoding, texts are assumed to be elements of each word, it becomes to be possible to encode words into tables. In this research, we map words into tables in the reverse direction to the text encoding, modify the AHC algorithm into the version where tables are given as its input, and apply the modified version of AHC to the word clustering task. In this research, we expect the better and more stable clustering performance, than the traditional version. Hence, this research provides the improved method of clustering words semantically.

Keywords—Word Clustering, Table Similarity, Table based AHC

I. INTRODUCTION

Word clustering refers to the process of segmenting a group of words into subgroups of similar words. Words are clustered based on their lexical similarities based on their spellings or their semantic ones based on their meanings. The scope of this research is restricted to the latter where words are clustered based on their meanings. Texts in the given corpus are features of representing words, and a similarity between words is computed based on their collocations within each text. Even if various types of clustering are available, this research focus on only hard clustering where each item is allowed to only one cluster.

Let us consider the issues with which this research tries to tackle in encoding words into numerical vectors. When we use texts as features of numerical vectors representing words, we need many features for implementing the robust word clustering systems[2]. Since almost numerical vectors which represent words have their zero values dominantly, what they called the sparse distribution, they have very little discrimination for computing their distances[3]. When we use the grammatical properties as features, implementation of encoding process becomes very difficult and complicated. Therefore, this research attempts to solve the problems by encoding words into tables, instead of numerical vectors.

Let us mention some agenda which are proposed by this research, in order to solve the above problems. Each word is encoded into a table which consists of entries of texts including it and its weights. We define the similarity measure between tables which is always given a normalized value, as the operation on tables. Using the similarity measure, we modify the AHC (Agglomerate Hierarchical Clustering) algorithm into the table based version where each object is given as a table. Therefore, in this research, we apply the modified AHC algorithm to the text clustering as its approach.

Let us mention benefits which are expected from this research. The tables which represent texts may be regarded as the more compact representations than numerical vectors, since the system robustness can be maintained with much less than elements. We expect more discrimination among tables than among numerical vectors in computing the similarity between them, because the sparse distribution is not available in tables. Because of the normalized similarity measure, from the proposed AHC version, we obtain both its better performance and more stability. However, note that in the proposed system, the table size is given as the external parameter, and it should be optimized between the reliability and the computation speed.

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

II. PREVIOUS WORKS

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

words, this section is intended to explore previous works on solutions to the problems.

Let us mention the popularity of encoding texts into numerical vectors, and the proposal and the application of string kernels as the solution to the above problems. In 2002, Sebastiani presented the numerical vectors are the standard representations of texts in applying the machine learning algorithms to the text classifications [4]. In 2002, Lodhi et al. proposed the string kernel as a kernel function of raw texts in using the SVM (Support Vector Machine) to the text classification [5]. In 2004, Lesile et al. used the version of SVM which proposed by Lodhi et al. to the protein classification [6]. In 2004, Kate and Mooney used also the SVM version for classifying sentences by their meanings [7].

Previously, it was proposed that texts should be encoded into string vectors as other structured forms. In 2008, Jo modified the k means algorithm into the version which processes string vectors as the approach to the text clustering[11]. In 2010, Jo modified the two supervised learning algorithms, the KNN and the SVM, into the version as the improved approaches to the text classification [12]. In 2010, Jo proposed the unsupervised neural networks, called Neural Text Self Organizer, which receives the string vector as its input data [13]. In 2010, Jo applied the supervised neural networks, called Neural Text Categorizer, which gets a string vector as its input, as the approach to the text classification [14].

It was proposed that texts are encoded into tables instead of numerical vectors, as the solutions to the above problems. In 2008, Jo and Cho proposed the table matching algorithm as the approach to text classification [3]. In 2008, Jo applied also his proposed approach to the text clustering, as well as the text categorization [11]. In 2011, Jo described as the technique of automatic text classification in his patent document [9]. In 2015, Jo improved the table matching algorithm into its more stable version [10].

The above previous works proposed the string kernel as the kernel function of raw texts in the SVM, and tables and string vectors as representations of texts, in order to solve the problems. Because the string kernel takes very much computation time for computing their values, it was used for processing short strings or sentences rather than texts. In the previous works on encoding texts into tables, only table matching algorithm was proposed; there is no attempt to modify the machine algorithms into their table based version. In the previous works on encoding texts into string vectors, only frequency was considered for defining features of string vectors. In this research, we will modify the machine learning algorithm, AHC algorithm, into the version which processes tables instead of numerical vectors, and use it as the approach to the word clustering.

III. PROPOSED APPROACH

This section is concerned with the table AHC (Agglomerative Hierarchical Clustering) algorithm as the approach to the word clustering tasks. In Section III-A, we describe the process of encoding words into tables, which is called text preprocessing. In Section III-B, we cover the scheme of computing a similarity between two tables into a normalized value between zero and one. In Section III-C, we mention the proposed version of AHC algorithm as the word clustering tool. Therefore, this article is intended to describe the encoding scheme, the similarity computation method, and the proposed version of AHC algorithm for implementing the word clustering systems.

A. Word Encoding

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

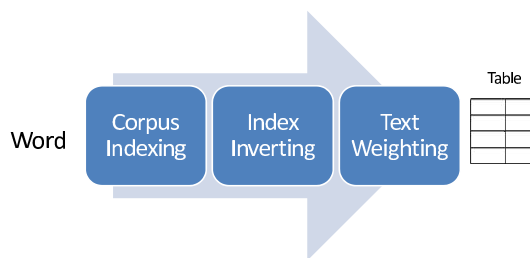


Figure 1. Overall Process of Word Encoding

The process of indexing the corpus into list of words is illustrated in Figure 2. The first step of the process is to tokenize a string concatenated by texts in the corpus into tokens by segmenting it by white spaces or punctuation marks. The second step called stemming is to map each token into its root form by the grammatical rules. The third step is to remove the stop words which function only grammatically and irrelevantly to contents for more efficiency. The words which are generated the through three steps are usually nouns, verbs, and adjectives.

In Figure 3, we illustrate the inverted list where each word is linked to its list of texts including the word. The index list where each text is linked to its included words may be generated by indexing each text from the text collection. For each word, the inverted list is constructed by retrieving texts including it from the index list. In each word, each text is

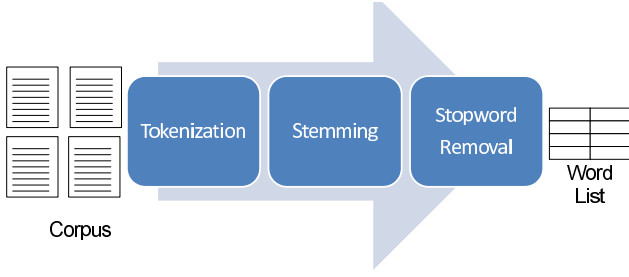


Figure 2. The Process of Corpus Indexing

associated with its weight as the relationship between the word and the text. The association of each word with the list of texts is opposite to the association of each text with its list of words is the reason of calling the list which is presented in figure 3 the inverted list.

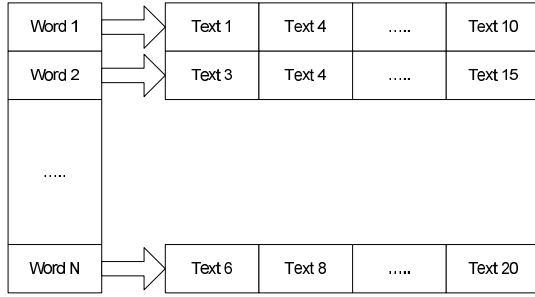


Figure 3. The Inverted Index

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

Previously, we attempted to encode texts into tables [10], so let us consider the differences between the word encoding and the text encoding. In the word encoding, each word is associated with texts including it, whereas, in the text encoding, each text is done with words included in it. Each table representing a word has its own entries of text identifiers and its weights in the texts, whereas one representing a text has those of words and their weights in the text. Computing similarity between the tables representing both words is based on text identifiers which include them, while doing it between the tables representing both texts is based on words shared by the both texts. Therefore, from the comparisons, reversal of texts and words indicate the essential differences between the text and word encoding.

B. Similarity between Tables

This section is concerned with the process of computing a similarity between tables. Texts are encoded into tables by the process which was described in Section III-A. The two tables are viewed into the two sets of words and the set of shared words is retrieved by applying the intersection on the two sets. The similarity between the two tables is based on the ratio of the shared word weights to the total weights of the two tables. Therefore, we intend this section to describe in detail and formally the process of computing the similarity.

A table which represents a word may be formalized as a set of entries of text identifiers and its weights. The word, T_j is represented into a set of entries as follows:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

We demonstrate the process of computing the similarity between two tables using the simple example which is presented in Figure 4. The two words are encoded into the two source tables as shown in Figure 4. In the example, the two text identifiers, 'artificial' and 'documents' are shared by the two tables, and each shared ones have their dual weights from the two input tables. The similarity between the two tables is computed to be 0.52 as a normalized value by equation (2). Therefore, the similarity is computed by lexical matching between the two tables.

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

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

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

Figure 4. Example of Two Tables

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

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

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

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

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

$$Sim(T_1, T_2) \leq Sim(T_1, T_3)$$

$$\rightarrow |D(T_1) \cap D(T_2)| \leq |D(T_1) \cap D(T_3)|.$$

The complexity of computing the similarity between two tables is $O(n \log n)$, since it takes $O(n \log n)$ for sorting the entries of two tables using the quick sort or the heap sort, and $O(n)$ for extracting shared elements by the consequential processing [1].

C. Proposed Version of AHC Algorithm

This section is concerned with the proposed AHC version as the approach to the word clustering. Words are encoded into tables by the process which was described in Section III-B. In this section, we attempt to modify the traditional AHC into the version where a table is given as the input data. The version is intended to improve the clustering performance by avoiding problems from encoding words 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 5. 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 AHC 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.

Separately from the traditional one, we illustrate the classification process by the proposed version in Figure 6. The sample texts labeled with the positive or negative class are encoded into tables. The similarity between two tables is computed by the scheme which was described in Section III-B. Identically to the traditional version, in the proposed version, the k most similarity samples are selected, and the label of the novice one is decided by voting ones of sample entities. Because the sparse distribution in each table is never available inherently, the poor discriminations by sparse distribution are certainly overcome in this research.

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

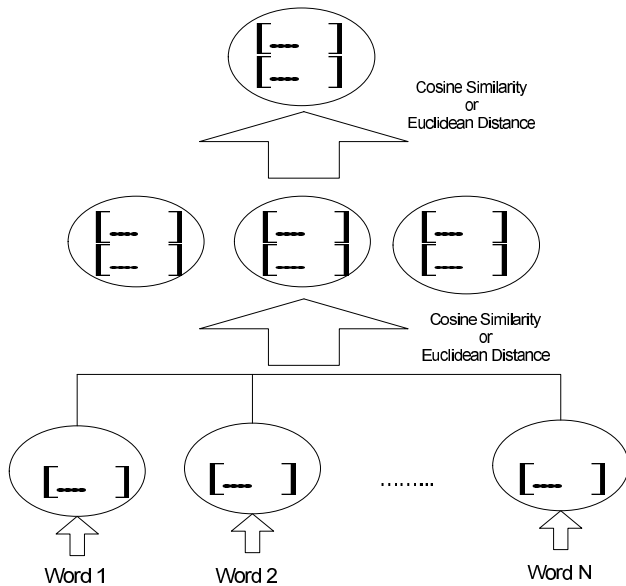


Figure 5. The Traditional Version of AHC Algorithm

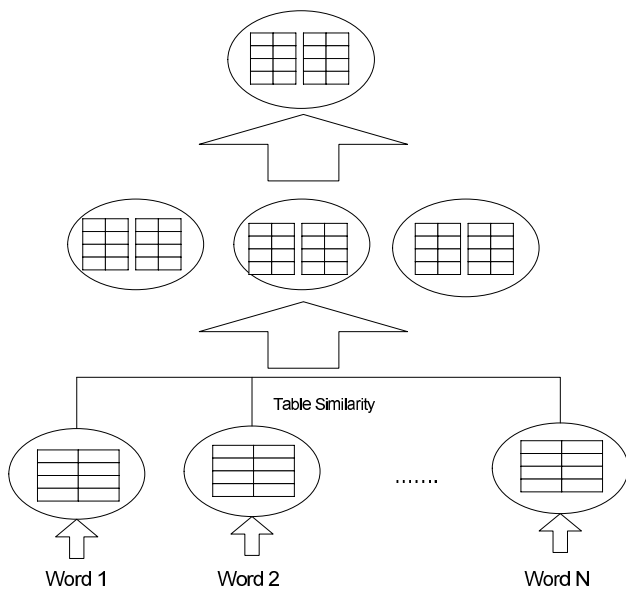


Figure 6. The Proposed Version of AHC Algorithm

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 the tables which represent words are characterized more symbolically than numerical vectors, it is easier to trace results from clustering items. Let us trace why a particular item is arranged into the cluster, by comparing it with the clusters. From each cluster, we extract text identifiers which are shared by it and the cluster members. In each cluster, we present a list of shared ones and their weights,

together with the total weight. Therefore, we present the evidence by highlighting the list which corresponds to the cluster and the total weight.

IV. CONCLUSION

Let us consider the remaining tasks for doing the further research. We need to validate and customize the proposed research in the word clustering in one of specific domains: engineering, science, and medicine. Because various schemes of weighting words are available, more than one weight may be assigned to each word, so it need to be considered in computing the similarity between tables. Other unsupervised machine learning algorithms may be modified as well as AHC into their table based versions. By adopting the proposed approach, we implement the word clustering system as a module of other programs or an independent program.

REFERENCES

- [1] M.J. Folk, B. Zoellick, and G. Riccardi, *File Structures: An Object Oriented with C++*, Addison Wesley, 1998.
- [2] L.D. Baker and A. K. McCallum, "Distributional clustering of words for text classification", pp96-103 *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998.
- [3] T. Jo and D. Cho, "Index Based Approach for Text Categorization", 127-132, *International Journal of Mathematics and Computers in Simulation*, No 2, 2008.
- [4] F. Sebastiani, "Machine Learning in Automated Text Categorization", pp1-47, *ACM Computing Survey*, Vol 34, No 1, 2002.
- [5] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text Classification with String Kernels", pp419-444, *Journal of Machine Learning Research*, Vol 2, No 2, 2002.
- [6] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, "Mismatch String Kernels for Discriminative Protein Classification", pp467-476, *Bioinformatics*, Vol 20, No 4, 2004.
- [7] R. J. Kate and R. J. Mooney, "Using String Kernels for Learning Semantic Parsers", pp913-920, *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 2006.
- [8] T. Jo, "Single Pass Algorithm for Text Clustering by Encoding Documents into Tables", pp1749-1757, *Journal of Korea Multimedia Society*, Vol 11, No 12, 2008.
- [9] T. Jo, "Device and Method for Categorizing Electronic Document Automatically", Patent Document, 10-2009-0041272, 10-1071495, 2011.
- [10] T. Jo, "Normalized Table Matching Algorithm as Approach to Text Categorization", pp839-849, *Soft Computing*, Vol 19, No 4, 2015.

- [11] T. Jo, "Inverted Index based Modified Version of K-Means Algorithm for Text Clustering", pp67-76, Journal of Information Processing Systems, Vol 4, No 2, 2008.
- [12] T. Jo, "Representation of Texts into String Vectors for Text Categorization", pp110-127, Journal of Computing Science and Engineering, Vol 4, No 2, 2010.
- [13] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", pp31-43, Journal of Network Technology, Vol 1, No 1, 2010.
- [14] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", pp83-96, International Journal of Information Studies, Vol 2, No 2, 2010.



Taeho Jo (M'97-AM'12) This author became a Member (M) of IEEE in 1997, and an Associate Member (AM) in 2012. He was born in 1970, South Korea. He received his Bachelor degree from Korea University in 1994, his Master degree from Pohang University of Science and Technology in 1997, and his PhD degree from University of Ottawa in 2006. His research area spans mainly over text mining, neural networks, machine learning, and information retrieval. He has the four year experience of working for industrial organizations and ten year experience of working for academic ones. So his research is characterized as the connection from fundamental researches for creating theories and to applied ones for developing products, by his experience of working for both sides.