IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

Deriving the Partial Order of Documents to Extend Clustering Applications

A. George Louis Raja[1*], F. Sagayaraj Francis[2] and P. Sugumar[3]

[1]Department of Computer Science and Applications, SCSVMV University, Kanchipuram, Tamil Nadu, India
[2]Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry, India
[3]Department of Computer Applications, Sacred Heart College (Autonomous), Tirupattur, Tamil Nadu, India

Corresponding Author: george@shctpt.edu

Available online at: www.ijcseonline.org

Accepted: 21/Jan/2019, Published: 31/Jan/2019

Abstract **–** The exponential growth of text documents over the internet has paved the way for systematic document organization. It is widely accepted that the document clustering has augmented the information retrieval process to a greater extend. Basically all the text clustering algorithms tend to establish more appropriate clusters of text documents, and the accuracy of text clustering algorithms are measured based on cluster cohesion and separation. Keeping to the basic principle of clustering to *minimize cohesion* and *maximize separation*, all the algorithms deploy different strategies to generate better quality clusters. It is observed from the detailed literature survey that *Classification*, *Categorization*, *Plagiarism Detection* and *Clustering* are correlated. All these text mining tasks are performed based on *indexing*, *searching* or *relating* the key terms present in the documents. Moreover, all the text mining methods focuses on establishing the similarity or difference among the text documents, by which they perform their intended tasks. Hence, they tend to limit the application of clustering only to complement information retrieval task. This paper tries to present an algorithm to establish the partial order among the text documents and thus to extend the applications of clustering.

Keywords- Clustering, Partial Ordering, Classification, Categorization, Indexing

I. PARTIAL ORDERING

In mathematics, especially order theory, *a partially ordered set* (also *poset*) formalizes and generalizes the intuitive concept of an *ordering*, *sequencing*, or *arrangement* of the elements of a set [1, 2]. It identifies and concludes the order of precedence among the elements through the defined binary relation between the elements. It is observed that not all the pair of elements in a set need to be comparable, yielding to the concept of total ordering in which all the elements of a set are supposed to be comparable with one another.

When the partial ordering is done for text documents, can be used to extend the application of text clustering. Hence, this paper attempts to construct an partial ordering of text documents. The paper introduces the concept of partial ordering of documents and proposes an algorithm to partially order the documents. The further sections of the paper, illustrates the algorithm and supplements it with the results, and highlight the applications of the algorithm.

II. NEED FOR PARTIAL ORDERING OF DOCUMENTS

In exponentially growing text corpuses, it is highly time consuming to predict the hierarchy of the documents. The hierarchy of the documents once established can help to understand the evolution of the documents [3]. This cumbersome and computationally costly process, if done, can definitely help to organize the documents better, and help to improve the information retrieval process at large [4, 5].

Consider the following group of sentences illustrated in Figure 1 that defines a binary tree, though contextually they are similar, they differ by their entropy.

S1: In a binary tree every node is considered to have a maximum of two children.

S2: A binary tree is a specialization of a tree, in which every node including the root node will have at-most two sub trees, the left sub tree and the right sub tree.

S3: It is observed that in a binary tree every node can have zero, one or two children.

S4: A binary can be empty or can have nodes with at most two children.

Figure 1. Sample Sentences

If the sentences are manually ordered according to their entropy, the partial order may be ***S2, S3, S4, S1***. Obviously this ordering was based on the definition that gives more details on a binary tree.
The partial order can be useful to establish the hierarchy of the sentences based upon their entropy. This hierarchy will reveal the most describing documents in the collection [6, 7].

III. STAGES OF PARTIALLY ORDER DOCUMENT ALGORITHM (PODA)

The Partial ordering algorithm works in five phases as and described in this section.

PHASE I: PREPROCESSING
The Stages I (*Tokenizing*), II(*Removing the Stop Words*), and III (*Stemming*) of *ATSCA* are repeated.

PHASE II: KEY TERM EXTRACTION:
Key terms are the most describing words of a document. The key terms are applied to carry out topic extraction, topic assignment and text summarization. It was established in the previous paper that key term extraction can improve the clustering process. The ATSCA algorithm supported with key graph key term extraction method was proved to have produced better clusters.

The distinct stemmed words of the documents are analyzed for their key terms. The Key Graph Algorithm is applied to extract the key terms from the documents. The keywords $k_{11},k_{12},\ldots\ldots,k_{mn}$ of each of the document $D_1,D_2,\ldots D_n$ of the text corpus is arrived. These keywords are the premises in the Partial order process.

PHASE III: NGD COMPUTATION:
Stage I: Construct the Document Term Matrix

From the reduced set of the *m*x*n* key terms ($k_{11},k_{12},\ldots\ldots,k_{mn}$) from the *n* documents ($D_1,D_2,\ldots D_n$), a Document Term Matrix (*DTM*) is constructed. The DTM is a *K*x*K* matrix, where *k*=*m*x*n*, representing the key words are symmetrically represented in both rows and columns, the key term k_{ij} is presented in i^{th} row and j^{th} column.

Stage II: Estimate the Normalized Google Distances
The Normalized Google Distance (NGD) is a semantic similarity metric, which was applied in the UTSCA clustering algorithm. The application of NGD was found to estimate the semantic similarity of terms with precision. In this stage, the NGD values among the key terms are computed, the NGD value of the key term *i* with key term *j*, NGD(*i*,*j*) is found and stored in the entry DTM[*i*,*j*] of the Document Term Matrix.

Stage III: Reduction:
The NGD values are interpreted to identify the semantic similarity of the terms. It is assumed that when the NGD values of the terms *x* and *y* are closer or equal to zero the terms are similar and greater NGD values deem the terms to be different. This attribute of NGD gives rise to the intuition to deduce the terms with higher NGD values from the Document Term Matrix.
The reduced DTM will have the keyterms which are semantically analogous to one another.

PHASE IV: CLUSTERING
The resultant Document Term Values with the NGD values of the semantically closer terms is put through the centroid based clustering algorithm. The resultant clusters $\{c_1,c_2,\ldots.c_n\}$ along with their relating key terms $\{k_1,k_2,\ldots\ldots k_n\}$ are created. The keyset of the cluster c_i generates its lexicon L_i. Till this stage the steps of Partial Ordering is an amalgamation of Lexicon Extraction and Semantic Clustering processes.

PHASE V: PARTIAL ORDERING:

The documents of a cluster are ranked based on their calculated score. Then, the frequency of distribution with these representative scores results the partial order of the documents.

The following procedure briefs on the frequency distribution calculations:

(i) Identify the number of levels in the hierarchy with $l = \sqrt{n}$, where n is the number of documents in the cluster.

(ii) In every cluster, the range can be defined as range= maximum score – minimum score.

(iii) The number of documents (*nd*) in each range can be fixed as $nd = \frac{range}{l}$.

(iv) Sort the scores of documents in *increasing* order.

(v) Split the documents into two groups initially on the mean value, place the first group of (*n/2*) documents in first level and second group of (*n/2*) documents in the second level.

(vi) Repeat the process (*v*)iteratively until the number of levels equals *nd*.

The process results with the partial order of documents. The Algorithmic interpretation of the process is depicted in Figure 2.

Algorithm *Compute_Frequency_Distribution;*
Input: Dataset *A*, Cluster *C*, Documents *D*, Lexicon Scores *S*
Output: Partially ordered documents of Cluster *C*

```
1. begin
2.      for each cluster c in the data set A
3.              l= √n;
4.              range = maximum(s) - minimum(s);
5.              nd= range/l ;
6.              repeat
7.                      sort the documents in cluster c on scores;
8.                      find the mean range of each cluster;;
9.                      group the documents;
10.                     groups++;
11.             until (groups==nd);
12.     end for;
13.     return;
14. end;
```


Algorithm PODA

Input: Dataset *A* containing the documents to be ordered

Output: Clusters of the partially ordered documents of the dataset *A*

1. *begin*
2. *for each* document d_a in the data set *A*
3. *remove* the punctuators, delimiters and spaces;
4. *for each* token *k*in document d_a
5. *delete* the words specified in the stop words list;
6. *end for;*
7. *for each* term *t* in document d_a
8. *stem* them to the root word with *porter stemming*;
9. *end for;*
10. *end for*;
11. *for each* document *d* in dataset *A*
12. *extract* the keywords from the document *d* using *keygraph* algorithm;
13. *end for*;
14. *for each* keyword $k_{i,j}$ in the *dtm[i,j]*
15. *compute* NGD[*i,j*];
16. *end for;*
17. *for each* entry in *dtm*
18. *if*(NGD[i,j]>=1)*then* remove the entry dtm[*i,j*];
19. *end for;*
20. *for each* entry in *dtm*
21. *find* the closest pair of NGD values to estimate the similar documents;
22. *Mark* the NGD(*x,y*) as a Lexicon entry in l_i*;*
23. *end for;*
24. *call ATSCA_Clustering*;
25. *for each* document *d* in the cluster *c*
26. *compute*
27. *end for;*
29. *call Compute_Frequency_Distribution;*
30. *return* the arranged partial order of the documents.
31. *end;*

Figure 2. PODA Algorithm

IV. EXPERIMENTAL RESULTS

The Partial Ordering algorithm was experimented with a data set with three hundred text documents in the category of Internet of Things, Big Data, Software Engineering and Text Mining.

The algorithm yielded the partial order or hierarchy of documents illustrated in Figure 3, 4, 5 and .6. The Figure 3 illustrates the partial order of documents of Big Data, Figure 4, the partial order of documents of Internet of Things, Figure 5, the partial order of documents of Software Engineering and Figure 6, illustrates the partial order of documents of Text Mining clusters.

Each section marked by a rectangle in the figure 3, 4, 5 and 6 refers to a level of Hierarchy, which is represented from top to bottom. The numbers in each rectangle represents the file numbers given as the input. The top most rectangle in each hierarchy depicts the list of file numbers in the top hierarchy (documents regarded as most important), with the downward rectangles representing files with least importance.

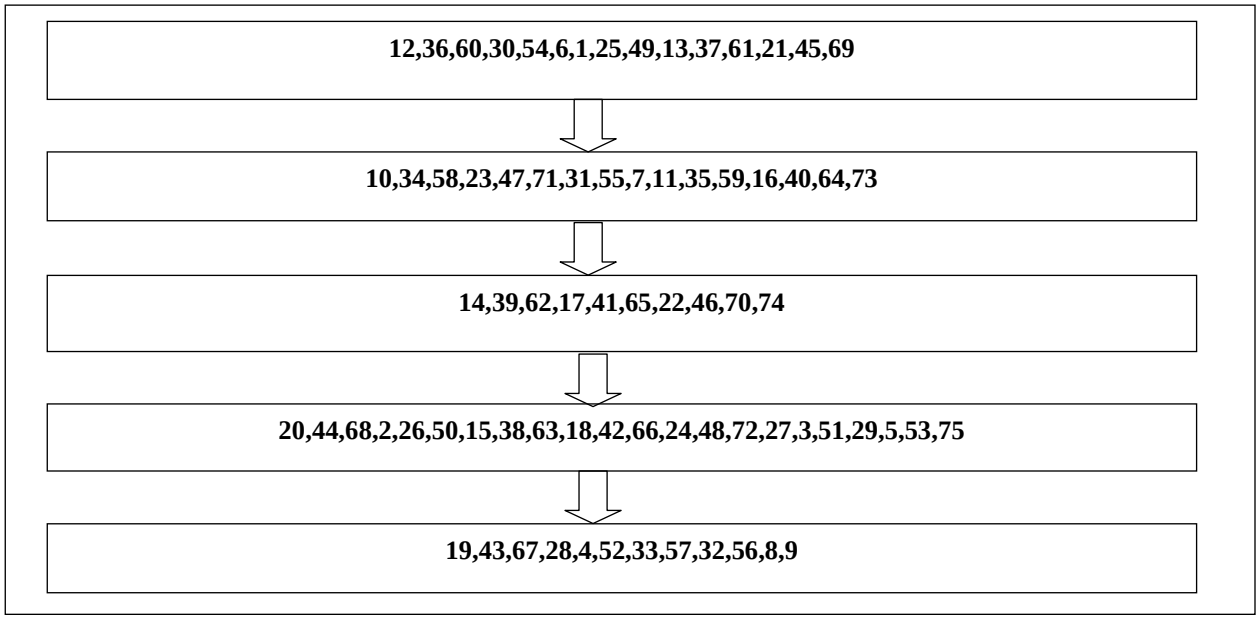

Figure 3. Partial Order of Big Data documents

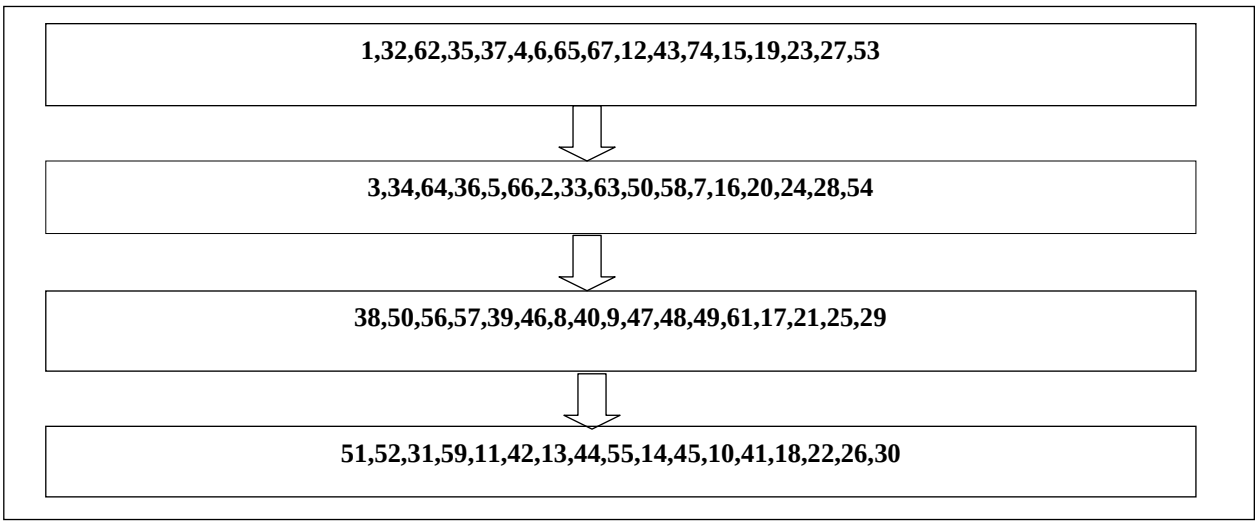

Figure 4. Partial Order of Internet of Things documents

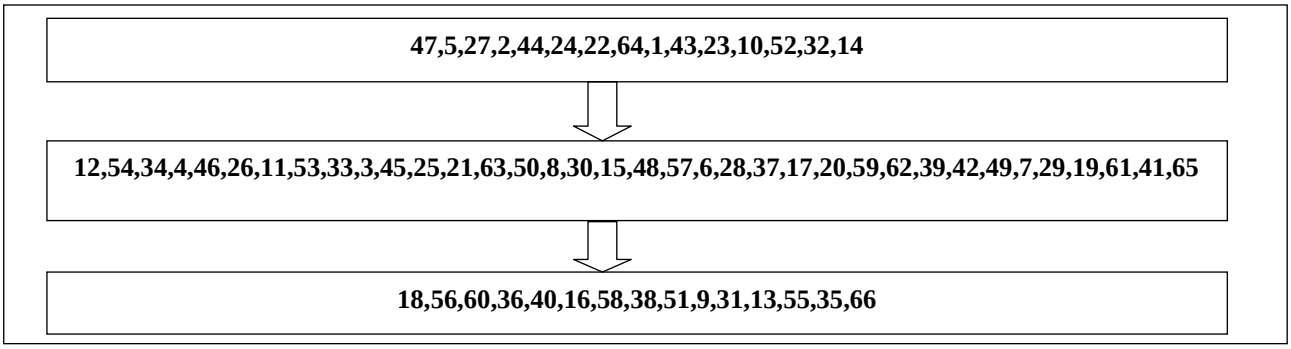

Figure 5. Partial Order of Software Engineering documents

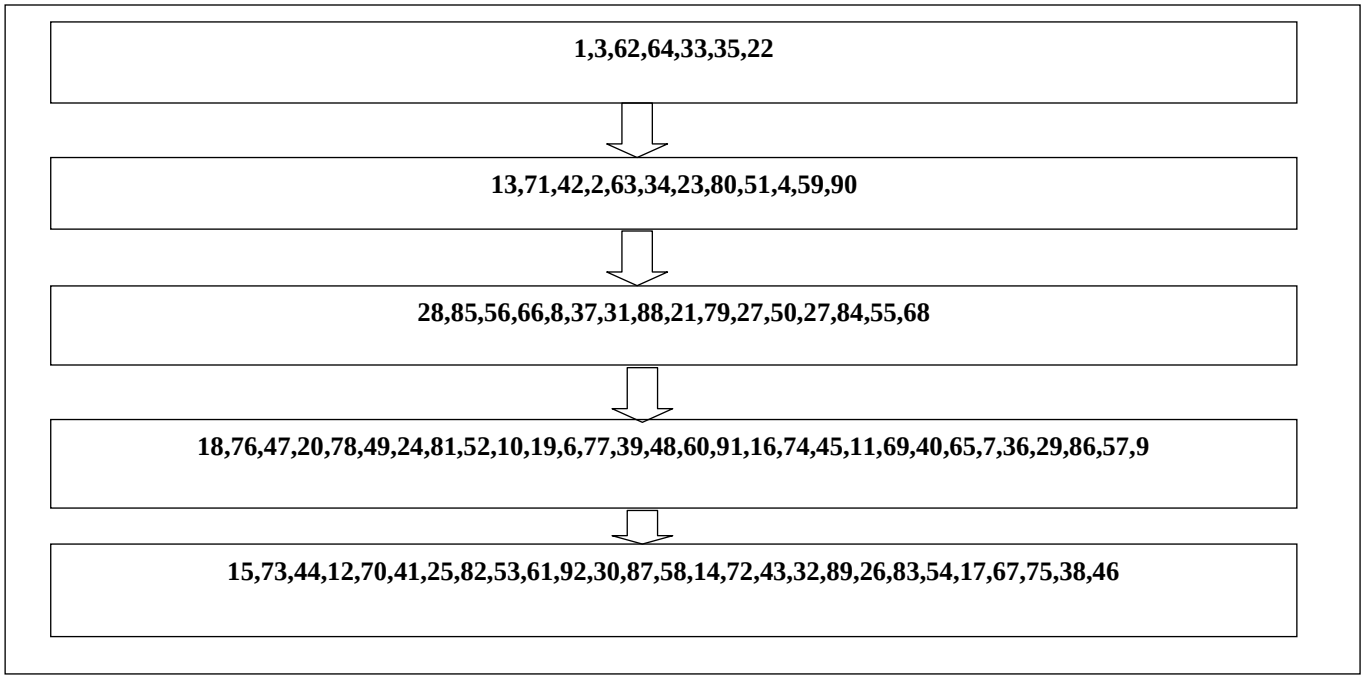

Figure 6. Partial Order of Text Mining documents

The following Table 1 summarizes the number of hierarchy levels in each cluster along with the number of files identified in each level.

Table 1 Summary of Number of Files in each Level

Cluster	Level1	Level2	Level3	Level4	Level5	Total
Big Data	15	16	10	22	12	75
Internet of Things	17	17	16	17	-	67
Software Engineering	15	36	15	-	-	66
Text Mining	7	12	16	30	27	92

The Table 2 correlates the number of levels in each of the clusters with the number of top level clusters generated from *Phase* III using the *ATSCA* algorithm. It can be observed that the number of clusters and levels are approximately equal in number.

Table 2 Comparison of Clusters Vs. Levels

Cluster	Top Level Clusters	Hierarchical Levels
Big data	4	5
Internet of Things	3	4
Software Engineering	3	3
Text Mining	4	5

V. OUTPUT ANALYSIS

The output of the partial ordering algorithm was manually evaluated for its precision. This analysis was done to verify the correctness of the partial ordering of documents produced by the algorithm. Since, the manual method yields 100% precision accuracy in linguistic analysis specifically in partial ordering. The documents were manually ordered and the outputs were used for comparison. The following Table 3 summarizes the outputs of the partial ordering algorithm with the manual outputs.

Table 3 Precision of PODA

Cluster	Level	Number of Documents	Number of Matching Documents	Precision %
Big Data	Level 1	15	12	80
	Level 2	16	14	87.5
	Level 3	10	10	100
	Level 4	22	20	90.9
	Level5	12	10	83.3
Average % of Precision in Big Data Cluster				88
Internet of Things	Level 1	17	15	88.2
	Level 2	17	14	82.3
	Level 3	16	15	93.75
	Level 4	17	16	94.11
Average % of Precision in Internet of Things Cluster				89
Software Engineering	Level1	15	14	93.3
	Level2	36	31	91.1
	Level3	15	14	93.3
Average % of Precision in Software Engineering Cluster				89.9
	Level1	7	7	100

	Level2	12	11	91.6
	Level3	16	14	87.5
	Level4	30	28	93.3
	Level5	27	22	81.4
Average % of Precision in Text Mining Cluster				89.1
Average % of Precision achieved by Partial Order Algorithm				**89**

VI. EXTENDING THE APPLICATIONS OF CLUSTERING

The core intuition behind establishing the partial order of documents is to extend the horizons of clustering applications. The outputs of PODA algorithm augments information retrieval and can be tailored to perform plagiarism detection and categorization.

Plagiarism Detection through PODA

The formulated levels from PODA represent the documents with same frequency of contextually similar terms. This frequency can be deployed to measure the amount of similarity among the documents. In terms of plagiarism detection, similarity is an indicator of plagiarism. The intuition is to identify the contextually best document in each level, which contains the largest frequency of the key terms in that level. Based on the estimation of percentage of terms replicated from this document with each of the rest of documents may determine the amount of similarity or plagiarism observed.

It is observed that arranging the documents according to their proximity in each level may be used to estimate the amount of plagiarism. Each level is illustrated with the most describing document, and the ratio of similarity of all other documents with it. For example it can be observed that in the level 1 (big data) with 15 documents, the document numbered 12 contains the larger number of terms in the corpus and is identified as the contextually significant document. The amount of similarity between the document 12 and other documents in the level is calculated. Thus, this table can be used to observe the amount of intra cluster plagiarism or plagiarism exhibited in the documents of the test corpus.

Text File De-duplication through PODA

The task of file de-duplication is to identify similar files in the given set of input documents. The documents gathered in each level of PODA represent the similar files. Hence, it can be inferred that the documents in the same level of PODA are duplicates

CONCLUSION

The Applications of clustering is bounded to Information Retrieval, though the horizon can be widened to other data mining activities. In this paper, a concept is presented to extend the application of Clustering to partial ordering of Text Documents. When the clustered documents are organized in a hierarchical structure, the structure can be developed to perform document classification and plagiarism detection.

This paper has presented an algorithm to partially order the text documents, and demonstrated the partial ordering of the test corpus. The outputs of the algorithm are found to approximate the manual output of partial ordering.

REFERENCES

[1] 1.www.wikipdia.com/ Hierarchy

[2] 2.www.wikipedia.com/Poset- Wikipedia.html.

[3] 3.Michelangelo Ceci and Donato Malerba, "Classifying web documents in a hierarchyof categories: a comprehensive study", *Journal of Intelligent Information Systems*, ISSN: 0925-9902, Volume 28, Issue 4, pp. 37-78, 2007.

[4] 4.W.T. Chuang, A. Tiyyagura, J. Yang and G. Giuffrida, "A fast algorithm for hierarchicaltext classification",*Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2000)*, pp. 409-418, New York , U.S.A, 2000.

[5] 5.S. D. Alessio, K. Murray, R. Schiaffino, and A. Kershenbau, "The effect of using hierarchical classifiers in text categorization", *Proceedings of the 6thInternationalConferenceonRecherchedInformationAssistdepar Ordinateur*(*RIAO2000*), pp. 302-313, Paris, France,2000.

[6] 6.D. Koller and M. Sahami, "Hierarchically classifying documents using very few words", *Proceedings of the 14th International Conference onMachineLearning* , pp. 170-178, California, U.S.A, 1997.

[7] 7.M.K. M. Rahman and Tony W. S. Chow, "Content based hierarchical document organization using multi layer hybrid network and tree structured features", *Expert Systems with Applications*, ISSN: 2874-2881, Volume 37, 2010

