Exploiting Index Pruning Methods for Clustering XML Collections

Ismail Sengor Altingovde, Duygu Atilgan, and Özgür Ulusoy

Department of Computer Engineering, Bilkent University, Ankara, Turkey
{ismaila,atilgan,oulusoy}@cs.bilkent.edu.tr

Abstract. In this paper, we first employ the well known Cover-Coefficient Based Clustering Methodology (C3M) for clustering XML documents. Next, we apply index pruning techniques from the literature to reduce the size of the document vectors. Our experiments show that for certain cases, it is possible to prune up to 70% of the collection (or, more specifically, underlying document vectors) and still generate a clustering structure that yields the same quality with that of the original collection, in terms of a set of evaluation metrics.

Keywords: Cover-coefficient based clustering, index pruning, XML.

1 Introduction

As the number and size of XML collections increase rapidly, there occurs the need to manage these collections efficiently and effectively. While there is still an ongoing research in this area, INEX XML Mining Track fulfills the need for an evaluation platform to compare the performance of several clustering methods on the same set of data. Within the Clustering task of XML Mining Track of INEX campaign, clustering methods are evaluated according to cluster quality measures on a real-world Wikipedia collection.

To this end, in the last few workshops, many different approaches are proposed which use structural, content-based and link-based features of XML documents. In INEX 2008, Kutty et al. [11] use both structure and content to cluster XML documents. They reduce the dimensionality of the content features by using only the content in frequent subtrees of an XML document. In another work, Zhang et al. [13] make use of the hyperlink structure between XML documents through an extension of a machine learning method based on the Self Organizing Maps for graphs. De Vries et al. [9] use K-Trees to cluster XML documents so that they can obtain clusters in good quality with a low complexity method. Lastly, Tran et al. [12] construct a latent semantic kernel to measure the similarity between content of the XML documents. However, before constructing the kernel, they apply a dimension reduction method based on the common structural information of XML documents to make the construction process less expensive. In all of these work mentioned above, not only the quality of the clusters but the efficiency of the clustering process is also taken into account.

In this paper, we propose an approach which reduces the dimension of the underlying document vectors without change or with a slight change in the quality of the output clustering structure. More specifically, we use a partitioning type clustering algorithm, so-called Cover-Coefficient Based Clustering Methodology (C^3M) [7], along with some index pruning techniques for clustering XML documents.

2 Approach

2.1 Baseline Clustering with C^3M Algorithm

In this work, we use the well-known Cover-Coefficient Based Clustering Methodology (C^3M) [7] to cluster the XML documents. C^3M is a single-pass partitioning type clustering algorithm which is shown to have good information retrieval performance with flat documents (e.g., see [6]). The algorithm operates on documents represented by vector space model. Using this model, a document collection can be abstracted by a document-term matrix, D; of size m by n whose individual entries, d_{ij} ($1<i<m$; $1<j<n$), indicate the number of occurrences of term j (t_j) in document i (d_i). In C^3M, the document-term matrix[1] D is mapped into an m by m cover-coefficient (C) matrix which captures the relationships among the documents of a database. The diagonal entries of C are used to find the number of clusters, denoted as n_c; and to select the cluster seeds. During the construction of clusters, the relationships between a nonseed document (d_i) and a seed document (d_j) are determined by calculating the c_{ij} entry of C; where c_{ij} indicates the extent to which d_i is covered by d_j.

A major strength of C^3M is that for a given dataset, the algorithm itself can determine the number of clusters, i.e., there is no need for specifying the number of clusters, as in some other algorithms. However, for the purposes of this track, we cluster the XML documents into a given number of clusters (for several values like 1000, 10000, etc.) using C^3M method, as required. In this paper, we simply use the content of XML documents for clustering. Our preliminary experiments that also take the link structure into account did not yield better results than just using the content. Nevertheless; our work in this direction is still under progress.

2.2 Employing Pruning Strategies for Clustering

From the previous works, it is known that static index pruning techniques can reduce the size of an index (and the underlying collection) while providing comparative effectiveness performance with that of the unpruned case [2, 3, 4, 5, 8]. In a more recent study, we show that such pruning techniques can also be adapted for pruning the element-index for an XML collection [1]. Here, with the aim of both improving the quality of clusters and reducing the dataset dimensions for clustering, we apply static pruning techniques on XML documents. We adapt two well-known pruning techniques, namely, term-centric [8] and document-centric pruning [5] from the literature to obtain more compact representations of the documents. Then, we cluster documents with these reduced representations for various pruning levels, again using C^3M algorithm. The pruning strategies we employ in this work can be summarized as follows:

[1] Note that, in practice, the document-term matrix only includes non-zero term occurrences for each document.

- *Document-centric pruning (DCP):* This technique is essentially intended to reduce the size of an inverted index by discarding unimportant terms from each document. In the original study, a term's importance for a document is determined by that term's contribution to the document's Kullback-Leibler divergence (KLD) from the entire collection [5]. In a more recent work [2], we show that using the contribution of a term to the retrieval score of a document (by using a function like BM25) also performs quite well. In this paper, we again follow this practice and for each term that appears in a given document, we compute that term's BM25 score for this document. Then, those terms that have the lowest scores are pruned, according to the required pruning level. Once the pruned documents are obtained at a given pruning level, corresponding document vectors are generated to be fed to the C^3M clustering algorithm.
- *Term-centric pruning (TCP)*: This method operates on an inverted index, so we start with creating an index for our collection. Next, we apply term-centric pruning at different pruning levels, and once the pruned index files are obtained, we convert them to the document vectors to be given to the clustering algorithm[2]. In a nutshell, the term-centric pruning strategy works as follows [8]. For each term t, the postings in t's posting list are sorted according to their score with respect to a ranking function, which is BM25 in our case. Next, the k^{th} highest score in the list, z_t, is determined and all postings that have scores less than $z_t * \varepsilon$ are removed, where ε specifies the pruning level. In this paper, we skip this last step, i.e. ε-based tuning, and simply remove the lowest scoring postings of a list for a given pruning percentage.

3 Experiments

In this paper, we essentially use a subset of the INEX 2009 XML Wikipedia collection provided by XML Mining Track. This subset, so-called small collection, contains 54575 documents. On the other hand, the large collection contains around 2.7 million documents and takes 60 GB. It is used only in the baseline experiments for various number of clusters.

As the baseline, we form clusters by applying C^3M algorithm to XML documents represented with the bag of words representation of terms, as provided by the track organizers. For several different number of output clusters, namely 100, 500, 1000, 2500, 5000 and 10000, we obtain the clusters and evaluate them at the online evaluation website of this track. The website reports the standard evaluation criteria for clustering such as micro purity, macro purity, micro entropy, macro entropy, normalized mutual information (NMI), micro F1 score and macro F1 score for a given clustering structure. However, only purity measures are used as the official evaluation

[2] It is possible to avoid converting the index to the document vectors by slightly modifying the input requirements of the clustering algorithm. Anyway, we did not spend much effort in this direction as this conversion stage, which is nothing but an inversion of the inverted index, can also be realized in an efficient manner.

Table 1. Micro and macro purity values for the baseline C^{3}M clustering for different number of clusters using the small collection

No. of clusters	Micro Purity	Macro Purity
100	0.1152	0.1343
500	0.1528	0.1777
1000	0.1861	0.2147
2500	0.2487	0.3031
5000	0.3265	0.4160
10000	0.4004	0.5416

Table 2. Micro and macro purity values for the baseline C^{3}M clustering for different number of clusters using the large collection

No. of clusters	Micro Purity	Macro Purity
100	0.1566	0.1234
1000	0.1617	0.1669
10000	0.1942	0.2408

criteria for this task. In Tables 1 and 2, we report those results for clustering small and large collection, respectively. For the latter case, due to time limitations, we experimented with three different numbers of clusters such as 100, 1000 and 10000. A quick comparison of the results in Tables 1 and 2 for corresponding cases imply that purity scores are better for the smaller dataset than that of the larger dataset, especially for large number of clusters. We anticipate that better purity scores for the large collection can be obtained by using a higher number of clusters.

Next, we experiment with the clusters produced by the pruning-based approaches. For each pruning technique, namely, TCP and DCP, we obtain the document vectors at four different pruning levels, i.e., 30%, 50%, 70% and 90%. Note that, a document vector includes term id and number of occurrences for each term in a document, stored in the binary format (i.e., as a transpose of an inverted index). In Table 3, we provide results for the small collection and 10000 clusters. Our findings reveal that up to 70% pruning with DCP, quality of the clusters is still comparable to or even superior than the corresponding baseline case, in terms of the evaluation measures.

Regarding the comparison of pruning strategies, clusters obtained with DCP yield better results than those obtained with TCP up to 70% pruning for both micro and macro purity measures. For the pruning levels higher than 70%, DCP and TCP give better results interchangeably for these measures. In [1], we observed a similar behavior regarding the retrieval effectiveness of indexes pruned with TCP and DCP.

From Table 3, we also deduce that DCP-based clustering at 30% pruning level produce the best results for both of the evaluation measures in comparison to the other pruning-based clusters. For this best-performing case, namely DCP at 30% pruning, we also provide performance findings with varying number of clusters (see Table 4).

The comparison of the results in Tables 1 and 4 shows that the DCP-based clusters are inferior to the corresponding baseline clustering up to 10000 clusters, but they provide almost the same performance for the 10000 clusters case.

Table 3. Comparison of the purity scores for clustering structures based on TCP and DCP at various pruning levels using the small collection. Number of clusters is 10000. Prune (%) field denotes the percentage of pruning. Best results for each measure are shown in bold.

Pruning Strategy	**Prune (%)**	**Micro Purity**	**Macro Purity**
No Prune	0%	0.4004	**0.5416**
DCP	30%	**0.4028**	0.5400
TCP	30%	0.3914	0.5229
DCP	50%	0.4019	0.5375
TCP	50%	0.3870	0.5141
DCP	70%	0.4016	0.5302
TCP	70%	0.3776	0.5042
DCP	90%	0.3783	0.4768
TCP	90%	0.3639	0.5073

Table 4. Micro and macro purity values for DCP at 30% pruning for different number of clusters

No. of clusters	**Micro Purity**	**Macro Purity**
100	0.1021	0.1265
500	0.1347	0.1539
1000	0.1641	0.1917
2500	0.2234	0.2737
5000	0.2986	0.3854
10000	0.4028	0.5400

In this year's clustering task, other than the standard evaluation criteria, the quality of the clusters relative to the optimal collection selection goal is also investigated. To this end, a set of queries with manual query assessments from the INEX Ad Hoc track are used and each set of clusters obtained is scored according to the result set of each query. According to the clustering hypothesis [10], the documents that cluster together have similar relevance to a given query. Therefore, it is expected that the relevant documents for ad-hoc queries will be in the same cluster in a good clustering solution. In particular, mean Normalised Cluster Cumulative Gain (nCCG) score is used to evaluate the clusters according to the given queries.

In Table 5, we provide the mean and the standard deviation of nCCG values for our baseline C^3M clustering on the small data collection. Regarding the pruning-based approaches, the mean nCCG values obtained from the clusters produced by TCP and DCP for various pruning levels are provided in Table 6. In parallel with the findings obtained by the purity criteria, mean nCCG values of the clusters obtained by DCP are still better than or comparable to the ones obtained by the baseline approach up to 70% pruning level. On the other hand, TCP approach yields better mean nCCG values even at 90% pruning level.

In Table 7, we provide the mean nCCG values obtained from different number of clusters formed by the DCP approach at 30% pruning level. A quick comparison of the results in Table 7 with those in Table 5 reveals that clusters obtained after DCP pruning are more effective than the clusters obtained by the baseline strategy for various number of clusters.

Table 5. Mean and standard deviation of nCCG values for the baseline C^{3}M clustering for different number of clusters using the small collection

No. of clusters	Mean nCCG	Std. Dev. CCG
100	0.7344	0.2124
500	0.6258	0.2482
1000	0.5986	0.2790
2500	0.5786	0.2352
5000	0.5918	0.2395
10000	0.4799	0.2507

Table 6. Comparison of the mean and standard deviation of nCCG values for clustering structures based on TCP and DCP at various pruning levels using the small collection. Number of clusters is 10000. Prune (%) field denotes the percentage of pruning. Best results for each measure are shown in bold.

Pruning Strategy	Prune (%)	Mean nCCG	Std. Dev. CCG
No Prune	0%	0.4799	0.2507
DCP	30%	0.4950	0.2549
TCP	30%	0.4940	0.2370
DCP	50%	0.4828	0.2467
TCP	50%	0.4618	0.2236
DCP	70%	0.4601	0.2207
TCP	70%	0.5075	0.2176
DCP	90%	0.4613	0.2343
TCP	90%	**0.5132**	0.2804

Table 7. Mean and standard deviation of nCCG values for DCP at 30% pruning for different number of clusters

No. of clusters	Mean nCCG	Std. Dev. CCG
100	0.7426	0.1978
500	0.6424	0.2326
1000	0.5834	0.2804
2500	0.5965	0.2504
5000	0.5929	0.2468
10000	0.4950	0.2549

Finally, in Figure 1 we compare the performance of the C^{3}M clustering with the other runs submitted to INEX 2009 in terms of the mean nCCG. For each case (i.e., number of clusters), we plot the highest scoring clustering approaches from each group. Note that, for cluster numbers of 100, 500, 2500, and 5000, our strategy (denoted as Altingovde et al.) corresponds to DCP based clustering at 30%; and for the cluster number of 10000 we report the score of TCP based clustering at 90%. For one last case where cluster number is set to 1000, we report the baseline C^{3}M score, which turns out to be the highest.

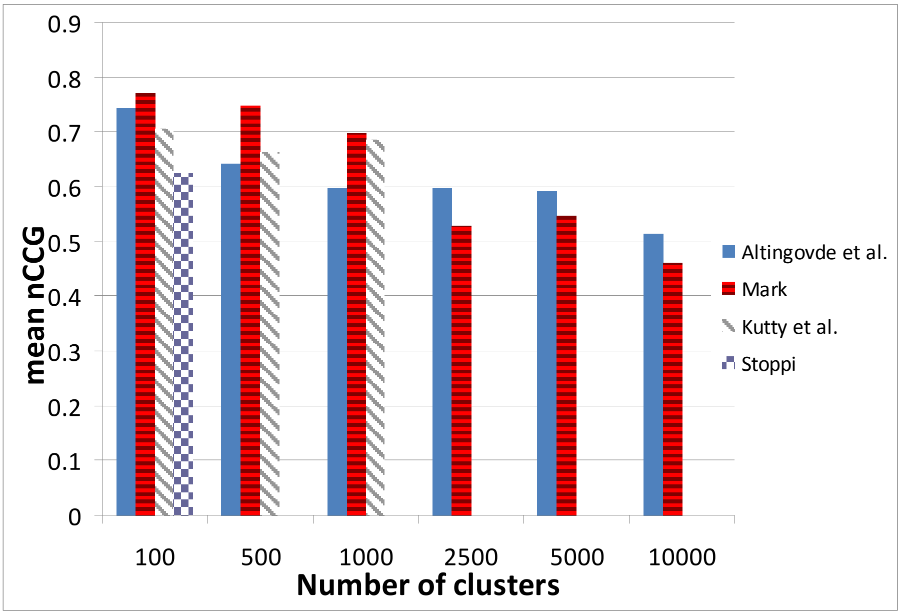

Fig. 1. Comparison of the highest scoring runs submitted to INEX for varying number of clusters on the small collection

4 Conclusion

In this paper, we employ the well-known C^3M algorithm for content based clustering of XML documents. Furthermore, we use index pruning techniques from the literature to reduce the size of the document vectors on which C^3M operates. Our findings reveal that, for a high number of clusters, the quality of the clusters produced by the C^3M algorithm does not degrade when up to 70% of the index (and, equivalently, the document vectors) is pruned.

Our future work involves repeating our experiments with larger datasets and additional evaluation metrics. Furthermore, we plan to extend the pruning strategies to exploit the structure of the XML documents in addition to content.

Acknowledgments. This work is supported by TÜBİTAK under the grant number 108E008.

References

1. Altingovde, I.S., Atilgan, D., Ulusoy, Ö.: XML Retrieval using Pruned Element-Index Files. In: Proc. of ECIR 2010, pp. 306–318 (2010)
2. Altingovde, I.S., Ozcan, R., Ulusoy, Ö.: A practitioner's guide for static index pruning. In: Proc. of ECIR 2009, pp. 675–679 (2009)

3. Altingovde, I.S., Ozcan, R., Ulusoy, Ö.: Exploiting query views for static index pruning in web search engines. In: Proc. of CIKM 2009, pp. 1951–1954 (2009)
4. Blanco, R., Barreiro, A.: Boosting static pruning of inverted files. In: Proc. of SIGIR 2007, The Netherlands, pp. 777–778 (2007)
5. Büttcher, S., Clarke, C.L.: A document-centric approach to static index pruning in text retrieval systems. In: Proc. of CIKM 2006, pp. 182–189 (2006)
6. Can, F., Altingövde, I.S., Demir, E.: Efficiency and effectiveness of query processing in cluster-based retrieval. Information Systems 29(8), 697–717 (2004)
7. Can, F., Ozkarahan, E.A.: Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. ACM Transactions on Database Systems 15, 483–517 (1990)
8. Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y.S., Soffer, A.: Static index pruning for information retrieval systems. In: Proc. of SIGIR 2001, pp. 43–50 (2001)
9. De Vries, C.M., Geva, S.: Document Clustering with K-tree. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 420–431. Springer, Heidelberg (2009)
10. Jardine, N., van Rijsbergen, C.J.: The Use of Hierarchic Clustering in Information Retrieval. Information Storage and Retrieval 7(5), 217–240 (1971)
11. Kutty, S., Tran, T., Nayak, R., Li, Y.: Clustering XML documents using frequent subtrees. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 436–445. Springer, Heidelberg (2009)
12. Tran, T., Kutty, S., Nayak, R.: Utilizing the Structure and Content Information for XML Document Clustering. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 460–468. Springer, Heidelberg (2009)
13. Zhang, S., Hagenbuchner, M., Tsoi, A.C., Sperduti, A.: Self Organizing Maps for the Clustering of Large Sets of Labeled Graphs. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2008. LNCS, vol. 5631, pp. 469–481. Springer, Heidelberg (2009)