Efficient Processing of Category-Restricted Queries for Web Directories

Ismail Sengor Altingovde, Fazli Can, and Özgür Ulusoy

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

Abstract. We show that a cluster-skipping inverted index (CS-IIS) is a practical and efficient file structure to support category-restricted queries for searching Web directories. The query processing strategy with CS-IIS improves CPU time efficiency without imposing any limitations on the directory size.

1 Introduction

Web directories typically involve a hierarchy of categories and employ human editors who assign Web pages to corresponding categories. Web surfers make use of such directories either for merely browsing, or issuing a query under a certain category that they have chosen (i.e., a category-restricted search [4, 5]).

In the earlier works [4, 5], a simple way of processing the category-restricted queries is described as follows. The system first determines the categories that are under the user specified category, i.e., the sub-tree (or graph, more generally) rooted at the user's initial category selection (step 1). This set constitutes the *target categories*. Next, the query is processed using an inverted index over the *entire* document collection in the directory and a candidate result set is obtained (step 2). Finally, to obtain the query output, the documents that are not from the target categories found in the first step are eliminated from the candidate result set (step 3). In this paper, this is referred to as the *baseline* method.

The baseline method is not very efficient: The document selection step uses the entire index without making use of the target categories, which is known at that time. This means several accumulators are updated and extracted, just to be eliminated at the very end. Furthermore, the candidate elimination step requires to check the category of (at least) the top-*N* candidate documents, and this would require *N* separate disk accesses if the data structure mapping documents to categories is kept on disk. In [1], several alternatives to the above baseline query processing strategy are discussed to allow the use of the target categories as early as possible in the document selection stage. In particular, if it is possible to store the entire mapping of documents to categories in the main memory, then the query processor can avoid computing partial similarities for documents that are not in target categories without making any disk accesses. However, this approach would again suffer if the inverted index is compressed, which is a typical practice. In this case, the query processor would still waste CPU cycles for decoding some postings, just to be discarded when it is realized that they are not from the target categories.

In the literature, cluster-skipping inverted index structure (CS-IIS) is introduced for efficient cluster-based retrieval [2, 3, 6]. In this paper, we propose to adapt the CS-IIS

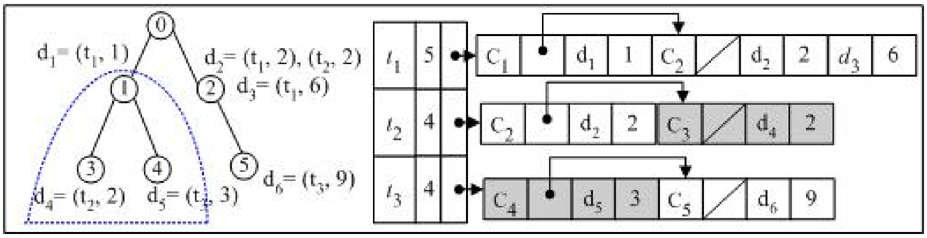

Fig. 1. A hierarchical taxonomy and the corresponding CS-IIS. Given the query = {t_2, t_3} that is restricted to C_1, the query processor first identifies the target categories (C_1, C_3 and C_4, as shown within dotted lines) and then processes posting lists. Note that, only the shaded parts of the posting lists are processed and the rest is skipped.

as a practical and efficient choice to combine the last two steps of the baseline approach. Thus, the major contribution of this work is demonstrating how CS-IIS can be employed in a hierarchical clustering framework, such as a Web directory, and how exactly the gains or costs are affected due to some unique properties of this framework. The experiments are held using the largest available Web directory dataset as provided by Open Directory Project (ODP). This work differs from the earlier works [3, 6] in the following ways: *i)* in the earlier works, an automatic and partitioning clustering structure is assumed, whereas the Web directory domain involves a hierarchical taxonomy, *ii)* the previous works involve moderate number of categories (although they were quite large figures in the automatic text clustering literature) whereas Web directories involve hundreds of thousands of categories, and *iii)* both the data, categorization and queries are *real*, which makes this environment a unique opportunity to show the applicability of the CS-IIS approach.

2 Category-Restricted Query Processing with CS-IIS

In CS-IIS, the <*document, term frequency*> pairs in a posting list are reorganized such that all documents from the same category are grouped together, and at the beginning of each such group an extra element is stored in the form of <*category id, next category address*>. While constructing the CS-IIS for a hierarchy as in the case of a Web directory, documents in a posting list are grouped with the categories under which they immediately appear (see Figure 1). This is different from an earlier proposal where the signature of the full category path is stored for each document [5].

During query processing, the target category set is identified by expanding the initial category given by the user (e.g., see [4]). Next, for each query term, the corresponding posting list is brought to the memory. By comparing the category ids in the posting list and in the target categories in a merge-join fashion, accumulators for only those documents that are from the target categories are updated. That is, if the category id in an inverted list element is not found in the target categories, the succeeding documents in the list (for that category) are skipped and the query processor jumps to the next category pointed by the "next category address." When all query terms are processed, the non-zero accumulators are only from the targets.

3 ODP Dataset Characteristics and Experimental Results

Dataset. For this study, we use the largest publicly available category hierarchy as provided by ODP Web site (www.dmoz.org). After preprocessing and cleaning data files, we end up with a category hierarchy of approximately 719K categories and 4.5 million URLs. For most of the URLs, a one- or two-sentence length description is also provided in the data file. In this paper, we use these descriptions as the actual documents. Note that, this yields significantly shorter documents (with a few words on the average) than usual. Our on-going work involves downloading the actual documents from the Web.

While constructing the hierarchy using the data files, we decided to use *narrow*, *symbolic* and *letterbar* tags in the data file as denoting the children of a category. The resulting hierarchy is more like a graph than a tree in that only 36% of the categories have a single parent. This indicates that, it would be better to keep track of the immediate category of a document as in our adaptation in Section 2 (and also the approach in [4]) with respect to keeping the entire path (e.g., see [5]), as there may be several paths to a particular document.

We find that a great majority of categories are rather small, i.e., 98% of them include less than 50 documents. Furthermore, 93% of the documents (about 4.2 million) belong to only one category, whereas 6% of the documents belong to two parents and only the remaining 1% of the documents appears in three or more categories. These numbers are important for CS-IIS, since a posting list needs to store the same documents as many times as they appear in different categories. The above trends conform the observations in earlier works [4, 5], and show that the waste of storage space due to overlapping documents among categories would not be high.

Indexing. After preprocessing, the document description file takes 2 GB on disk. During inverted index creation, all words (without stemming) are used except numbers and stopwords, yielding 1.1 million terms at the end. The resulting size of the typical inverted file (i.e., to be used by the baseline approach) is 342 MB whereas the size of the CS-IIS file is 609 MB. Note that, the additional space used in CS-IIS is unusually large in comparison to the earlier works (i.e., only 26% more space usage was observed in [6]). We attribute two reasons for this outcome, and state their remedies as follows: *i)* the dataset includes too many categories with respect to the number of documents. In [6], for instance, a collection of approximately 400K documents yields only 1357 clusters, whereas here approximately 4.5M pages are distributed to 719K categories. We believe this situation would change for our benefit in time, as the growth rate of hierarchy may possibly be less than that of the collection. Furthermore, the taxonomy may be populated to reach to a much larger collection size using automatic classification techniques. *ii)* the documents are unusually short, as we use just the summaries in this initial stage of our work.

Queries and query processing efficiency. We use two methods for obtaining category-restricted queries. First, we prepare a Web-based system which allows users (graduate students) to specify queries along with categories and evaluate the results (available at http://139.179.11.31/~kesra/bir2/).

For this paper, we only use 64 category-restricted queries from this system and refer to them as *manual-category* queries. Additionally, we employ the efficiency task

Table 1. In-memory query processing efficiency (all average values)

Query set	Strategy	Query evaluation time (sec)	No. of non-zero accumulators	No. of postings read
Manual-category	Baseline	0.128	17,219	17,339
	CS-IIS	**0.109 (15%)**[*]	11,758	28,913
Automatic-category	Baseline	0.158	19,900	20,367
	CS-IIS	**0.100 (37%)**[*]	250	33,271

[*] Percentage improvement w.r.t. baseline.

topics of TREC 2005 terabyte track. This latter set includes 50K queries, and 46K of them are used in the experiments after those without any matches in the collection are discarded. This set is referred to as *automatic-category* queries.

Notice that, the latter query set lacks any initial target category specification, so we had to match the queries to categories automatically. To achieve this, we use all terms in categories to compute query-category similarities. At this stage, the well-known TF-IDF term weighting with the cosine measure is employed. Next, for each query, we find the top-10 highest scoring category and choose a single one with the shortest distance to the root (i.e., imitating the typical user behavior of selecting a category as shallow as possible [5] while browsing).

For both query sets, this initial target category is then further expanded, i.e., the sub-graph is obtained. In the following experiments, the time cost for obtaining target categories is not considered, as this stage is exactly the same for both of the compared strategies and can be achieved very efficiently by using the method in [4]. The query-document matching stage also uses the TF-IDF based weighting scheme and cosine similarity measure [6]. Top 100 results are returned for each query. The in-memory average query processing (CPU) times are reported in Table 1, as well as the number of non-zero accumulators at the end, and the average length of posting lists read.

Table 1 reveals that for both query sets, using CS-IIS improves the efficiency of work done in main memory. This gain is caused by two factors: first, skipping irrelevant clusters reduces the redundant partial similarity computations. Secondly, but equally importantly, the number of non-zero accumulators at the end of query, which are to be inserted into and extracted from a min-heap, is considerably reduced. We even favor the baseline strategy by assuming that the document-category mapping is in the memory. Note that, the gains would be more emphasized if compression had been used, as skipping would also reduce the burden of decoding operations [2]. A second observation is that, the manual-category queries apparently cover a larger sub-graph and thus process more documents for both strategies. Indeed, in that query set, 55% of the queries are restricted to categories at depth 1. In contrary, the automatic-category queries usually locate the initial target category in a deeper position in the graph. That is why the latter makes much less operations and obtains more gains. Nevertheless, we used the same automatic category computation technique for the manual-category query set, and observed that most of the returned categories are reasonably relevant to queries, but not necessarily the same as the ones as specified by the user. Our current work involves a quantitative analysis of target category selection and using more sophisticated term weighting schemes to represent categories.

For the disk access issues, we assume that posting lists are brought to memory entirely and discarded once they are used (i.e., no caching). It is possible to read only a fraction of the posting lists in the baseline strategy. This is also possible in our approach. Indeed, if the skipping elements are kept at the beginning of each list instead of being intertwined with the document postings, reading only a relevant part from the disk would be straightforward. Furthermore, it is also possible to sort each category's list with respect to, say, frequency, and dynamically prune the search. Lastly, caching (if used) would provide similar improvements for both approaches. In Table 1, the difference between the list lengths fetched from the disk is around 12 K postings (for manual-category query set), adding up to 96 KB (i.e., 8 bytes/posting). Considering a typical disk with the transfer rate of 20 MB/s, the additional sequential read cost is only 5 ms, which is clearly less than the in-memory gains for this case.

4 Discussions and Conclusion

CS-IIS has some other advantages in comparison to the earlier works in the literature. We observe that the real life hierarchies are quite large (in contrast to those in [4, 5]). So, it may be difficult to use the signature-file based system as in [5]. The approach discussed in [4] enforces an upper limit on the number of categories (e.g., 1024). Furthermore, both of these earlier works involve using a part of document id to represent its categories, which would require bitwise operations during query processing and may complicate the use of typical index compression schemes. On the other hand, CS-IIS imposes no limits on neither the size of category nor the number of documents and can be practically used in existing systems, even with compression.

In this paper, the CS-IIS is adapted for hierarchical categories in Web directories to allow efficient processing of category-restricted queries. Our preliminary results show that, despite the use of very short document descriptions and the imbalance between the number of categories and documents, the proposed strategy is quite promising.

Acknowledgments. This work is partially supported by The Scientific and Technical Research Council of Turkey (TÜBİTAK) under the grant numbers 105E024 and 105E065. We thank to İnci Durmaz and Esra Küçükoğuz for processing ODP data.

References

1. Altingovde, I.S., Can, F., Ulusoy, Ö.: Algorithms for within-cluster searches using inverted files. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 707–716. Springer, Heidelberg (2006)
2. Altingovde, I.S., Demir, E., Can, F., Ulusoy, Ö.: Incremental cluster-based retrieval using compressed cluster-skipping inverted files. ACM TOIS (to appear)
3. Altingovde, I.S., Ozcan, R., Ocalan, H.C., Can, F., Ulusoy, Ö.: Large-scale cluster-based retrieval experiments on Turkish texts. In: Proc. of SIGIR 2007, pp. 891–892 (2007)
4. Cacheda, F., Baeza-Yates, R.: An optimistic model for searching Web directories. In: McDonald, S., Tait, J.I. (eds.) ECIR 2004. LNCS, vol. 2997, pp. 364–377. Springer, Heidelberg (2004)
5. Cacheda, F., Carneiro, V., Guerrero, C., Viña, Á.: Optimization of restricted searches in Web directories using hybrid data structures. In: Sebastiani, F. (ed.) ECIR 2003. LNCS, vol. 2633, pp. 436–451. Springer, Heidelberg (2003)
6. Can, F., Altingovde, I.S., Demir, E.: Efficiency and effectiveness of query processing in cluster-based retrieval. Information Systems 29(8), 697–717 (2004)