Case-Base Maintenance: A Streaming Approach

Yang Zhang, Su Zhang, and David Leake

School of Informatics and Computing, Indiana University,
Bloomington, IN 47405, U.S.A
yz90@umail.iu.edu,zhangsu@indiana.edu,leake@cs.indiana.edu

Abstract. Case Base maintenance can be crucial to CBR system performance. A central current of case base maintenance research focuses on competence-based deletion. Traditionally, deletion is done periodically, pausing CBR system processing to examining the entire case base. However, for streaming data, as often arises in big data contexts, such an approach may be expensive or infeasible. To address this problem, this paper proposes that CBR maintenance can draw on advances from data discovery research. In particular, it presents a case study applying the recent *Sieve-Streaming* algorithm [2] to enable continuous streaming CBR maintenance to reduce demands on case storage and provide efficient continuous maintenance. The paper presents a preliminary evaluation of this method on the Travel Agent Case Base, and compares it to traditional methods. The experiments are encouraging for the practicality and benefits of the approach for scaleup of case-base maintenance in settings with large-scale data streams.

Keywords: Case-Base Maintenance, Case Deletion, Streaming Algorithm.

1 Introduction

Case-based Reasoning(CBR) is the process of reasoning by adapting prior relevant cases to solve new problems (e.g., [7]). As CBR systems are used over long time frames, system knowledge must be maintained over time. For maintenance of the case base, a common question is how maintenance strategies can adjust knowledge to balance system efficiency and solution quality [6]. This has often focused on how to increase retrieval efficiency, by decreasing case base size, while maintaining system competence. Extensive CBR research has addressed this through methods for competence-based deletion, building on the seminal work of Smyth and Keane [10]. Such methods have been shown to provide good results for compression while retaining system accuracy.

As the CBR community addresses issues for growing data sets, issues arise making such models more difficult to apply. Competence-based deletion approaches often use greedy methods to selecting a subset of cases to retain from a case base to maximize competence, starting from a "complete" case base whose

competence they aim to retain; processing is done in a maintenance step to which it is assumed that considerable resources can be devoted, pausing the CBR cycle. Such policies can be characterized as *synchronic*, processing a snapshot of the case base, and *periodic* [12] with comparatively large periodicity.

However, CBR applications may need to be fielded with small sets of seed cases, with cases acquired from processing over time, and may be applied in domains that change over time. For such systems, a "complete" case base may never exist. Even if a complete case base might exist in principle, such a case base might be prohibitively large. Large data sets are acquired on a daily basis by streaming systems. For example, in the context of e-commerce, on "Cyber Monday" of 2016, Amazon U.S. reported that it processed 23 million orders. Building a comparatively complete case base from large-scale steaming data could require enormous storage and expensive processing to compress the case base. Even if storage were available for all cases in a large scale system, it might be undesirable in real-time CBR systems to interrupt system processing for frequent compression of the case base, if cases were received at a high rate.

The general problem of dealing with large scale streaming data is of course not new to CBR. In fact, the question of summarizing large-scale streaming data is an active research topic for the knowledge discovery community. This raises the question of whether approaches from that community provide solutions that the CBR community can exploit to improve case base maintenance performance. This paper presents a case study of the application of a recent method for massive data summarization, *Sieve-Streaming* [2], to case-base maintenance. Sieve streaming is a knowledge discovery method for selecting representative elements of large data sets in a single pass through data, with fixed memory requirements, with performance guarantees to provide guarantees on level of approximation of an optimal solution. The approach is been tested successfully for applications such as on-the-fly clustering.

This paper presents a preliminary exploration of the use of Sieve-Streaming for continuous incremental case base maintenance of cases from a case stream, without access to the full case base. It describes this approach and evaluates it for the *Travel Agents Case Base*, comparing its performance in terms of case base size, competence and retrieval quality of case base with two baseline methods.

2 Related Work

Case-base maintenance has been the subject of extensive study, summarized in the literature (e.g. [7, 12]). Most relevant to this paper are the two approaches we will take as baselines. Condensed Nearest Neighbor (CNN) [3] is a classic method for compaction of data sets to be solved by Nearest Neighbor algorithms, but CNN is order-dependent and requires multiple loops over data set to guarantee the consistency.

Smyth and McKenna[8] developed a competence-based model of CBM and defined the concept of coverage, reachability and relative coverage which modeled local competence contributions of cases interact as well as global competence

properties of a case base. This gave rise to several competence-guided editing solutions, called *footprinting* techniques, such as COV-FP, RFC-FP, RC-FP, etc, and the *footprint deletion*(FD) algorithm.

3 Adapting the Sieve-Streaming Algorithm for Case Base Maintenance

3.1 The Sieve-Streaming Algorithm

The Sieve-Streaming algorithm is a method to "summarize a massive data set "on the fly" [2] by selecting a high utility subset. The utility of the subset reflects the utilities of its members, according to a utility function whose values depend not only on the member being evaluated but on the other elements of the subset. The task addressed by Sieve-Streaming maps naturally to the task of selecting a high competence subset of the case base, with overall utility corresponding to overall competence, and the value of including particular cases depending on the other cases in the case base.

Sieve-Streaming addresses the problem of selecting elements to retain for very large data sets, for which storing the entire set is not possible or practical. If the optimal solution (for the case-base maintenance problem, the optimal competence) were known, it would be possible to strategically retain only those elements in the stream that provide sufficient gain towards the optimal solution. As the optimal solution will generally not be known, Sieve-Streaming approximates knowledge of the optimum by using a collection of estimations refined as new elements are seen. Each particular estimation corresponds to a set of elements, called a sieve. The number of sieves used, and the particular sieves which are retained, changed dynamically as the data stream is processed and more information becomes available. Incoming data is processed by each sieve, and the algorithm's result is the set contained in the sieve with maximal value. This approach enables calculating a reasonable solution from a subset of the values. In a CBR context, we can consider each sieve be a candidate case base, built in parallel, with the most successful returned by the algorithm.

The Sieve-Streaming algorithm assumes that a maximal data set (case base) size, k, is predefined. It tracks m, the maximum utility value of all individual examples seen, using that to estimate an upper bound on the marginal utility contribution of each added instance. As m changes, the algorithm adjusts the set of candidate thresholds determining the marginal utility to retain an example. For each input element e_i, the algorithm calculates the marginal gain $\Delta_f(e_i|S_v)$ of adding the element e_i to sieve S_v, given the a predefined utility function $f(S)$. If the marginal gain is more significant than the marginal value threshold, calculated as $\frac{\frac{v}{2}-f(S_v)}{k-|S_v|}$, and the maximum sieve size k has not yet been reached, the data point will be added to the sieve; otherwise, it will be discarded. Finally, the algorithm outputs the elements in the best sieve as output. Full details, including performance guarantees, are provided by Badanidiyuru et al. [2].

3.2 Adding Sieve-Streaming Maintenance to the CBR Cycle

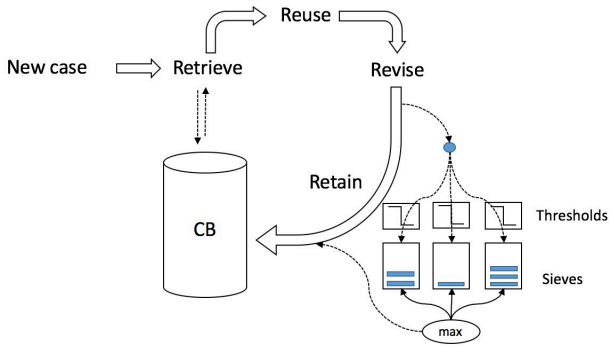


Fig. 1. Illustration of CBR Framework with Adapted Sieve-Streaming Algorithm (CBR cycle adapted from [1] and Sieve-Streaming process adapted from [2])

A motivation for incorporating Sieve-Streaming into CBR is to enable efficient ongoing competence-based case base updates at every execution of the CBR cycle. Sieve-Streaming can be applied to case-base maintenance as follows. First, a competence-based utility function is defined to calculate marginal gain for each case. The following subsection presents a simple domain-independent sample choice, but domain-specific utility functions could be defined as well.

Given an input problem, the system retrieves the most similar case in its case base. If the problem solved by the retrieved case is not identical to the new problem, the new problem is filtered by Sieve-Streaming to determine whether to add it to the case base. The case's marginal gain is compared to that of sieves whose capacity constraint has not yet been reached. If its marginal gain is larger than this sieve's marginal value threshold, the case is added to this sieve. If case's marginal gain value smaller than every sieve's marginal value threshold, the case is not added and processing continues with the next problem.

This approach enables the system to maintain a dynamic case base within desired size limits, with coverage satisfying the sieve streaming algorithm's guarantees, with ongoing rather than periodic maintenance.

3.3 A Sample Competence-based Utility Function

In order to apply Sieve-Streaming, a utility function must be defined, based on the system's similarity metric, to develop a submodular objective function f reflecting competence contributions. The goal for case base maintenance is to find a compact representative subset, which has characteristics similar to exemplar selection. Thus, we model our application on the example of Badanidiyuru et al. for exemplar selection, with *K-medoid loss function* as the utility function.

K-medoid Loss Function The *K-medoid* method selects elements by minimizing the sum of pairwise distances[4]. Following Badanidiyuru et al. [2], here $distance(e, v)$ is used to encode distances between exemplar and element. This yields Badanidiyuru et al.'s loss function:

$$L(S) \doteq \frac{1}{|V|} \sum_{e \in V} \min_{v \in S} distance(e, v) \tag{1}$$

For each element in V, the algorithm calculates distance between it and any element in given set S and selects the minimum distance among the results. Then it calculates the average of this minimum distance and used as the *loss* of set S. We want to minimize this loss. Following Krause and Gomes [5], $L(S)$ can be transformed into a monotone submodular way with an auxiliary element e_0, resulting in the following definition from Badanidiyuru et al.:

$$f(S) \doteq L(\{e_0\}) - L(S \cup \{e_0\}) \tag{2}$$

Here, $f(S)$ is always greater than 0 no matter what e_0 is chosen. Then minimizing L is equal to maximizing f. Because we also use similarity to measure the competence, we could use this $f(S)$ as the utility function in the Sieve-Streaming algorithm.

The calculation of *loss* relies on V–the entire data set–which could never be accessed during processing. However, we can address this in practice by using a sample of cases from the initial case base as an evaluation set.

4 Evaluation

The evaluation of streaming case retention explored four questions, each examining behavior as a function of the number of problems processed:

1. How does maintenance processing time vary for streaming case retention compared to baselines?
2. How does case base size vary for streaming case retention?
3. How does case base competence vary, and how does it compare to that with baseline methods?
4. How does accumulated retrieval distance vary, and how does it compare to that with baseline methods?

4.1 Experimental Design

Test case base: Tests were performed using the Travel Agents Case Base,[1] commonly used as a benchmark within the CBR community. This case base contains 1470 cases, each with a case identifier plus 9 features: *Journey Code, Holiday Type, Price, Number of Persons, Region, Transportation, Duration, Season, Accommodation, Hotel.* There are 6 categorical features and 3 numerical features

[1] `http://cbrwiki.fdi.ucm.es/mediawiki/index.php/Case_Bases`

Table 1. Parameters

1. Sampling parameters
   (a) n: Size of input stream.
   (b) *evaluation size*: Size of evaluation set. This sampled from same domain but was exclusive of the input stream.
   (c) *init size*: Size of initial case base (randomly selected)
2. Sieve-Streaming parameters
   (a) k: Restricted maximum size of sieves' capacity, also the size of case base.
   (b) ϵ: Scale of threshold set $O, \epsilon \in [0, 1]$.
3. CNN-FP/RC-FP parameters
   (a) *max*: Maintenance threshold. If Case-Base size increased to this value, CNN-FP or RC-FP maintenance method will be triggered.
   (b) *min*: Maintenance threshold. After maintenance, Case Base size should always no larger than this value.
   (c) *solve threshold*: Cases within this threshold of each other are considered to "solve" each other

left in each case. Similarity of categorical data was measured with *Huffman Code Similarity* based on Huffman coding [9]; this was used to translate categorical data to binary strings then converted to decimal values, enabling measuring similarity between cases simply by calculating Euclidean distance.

Competence Calculation: Smyth and McKenna's *Relative Coverage* [11] is an influential method for calculating competence contributions. In general, the calculation of *Relative Coverage* relies on determination of the the Coverage Set and Reachability Set for a case. However, in travel domain, our CBR system is a simple recommendation system which accepts trip characteristics and retrieves a similar case to return. This simplifies the calculation of *Relative Coverage*; we simply use similarity measurements to represent competence.

4.2 Test Runs and Parameter Settings

Experimental runs were conducted on four input streams. These were generated by randomly selecting an ordered set of 1000 of the 1470 cases in *Travel Agent Case Base*, and taking prefixes of lengths 100, 200, 500, and 1000 cases respectively.

The tests assessed the performance of Sieve-Streaming Algorithm along with CNN-FP and RC-FP as benchmark methods. Table 1 describes the parameters of the algorithms; Table 2 reports their settings for all runs. Note that for different size streams, different parameter settings were used for initial case base sizes and the minimum and maximum values of the case base size for non-streaming methods (always varying from 5% to 10% of the entire case base).

Table 2. Parameter Settings

	Parameters	Exp_1	Exp_2	Exp_3	Exp_4
Sampling	n	100	200	500	1000
	evaluation size	5	10	25	50
	init size	3	6	15	30
Sieve-Streaming	k	5	10	25	50
	ϵ	0.01	0.01	0.01	0.01
CNN-FP/RC-FP	*max*	10	20	50	100
	min	5	10	25	50
	solve threshold	0.5	0.5	0.5	0.5

4.3 Comparison and Analysis

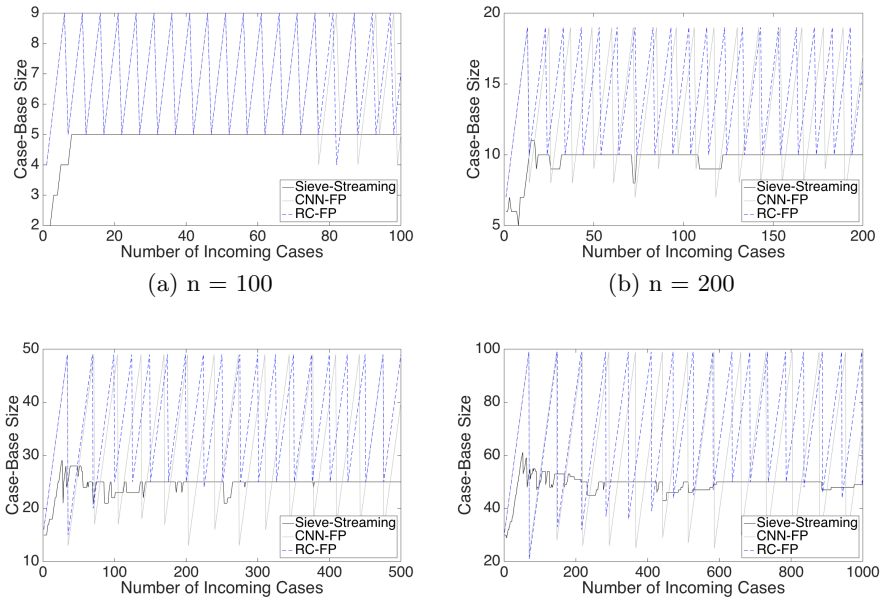
Question 1: Time Comparison: Table 3 shows that in our tests, CNN-FP and Sieve-Streaming had similar time performance, with Sieve-Streaming more expensive. This was a surprising result. However, timings may depend on implementation factors and no attempt was made to optimize the Sieve-Streaming algorithm; more investigation is needed. Both CNN-FP and Sieve-Streaming ran much faster than RC-FP, and RC-FPs growth rate was much higher than for CNN-FP and Sieve-Streaming.

Table 3. System Wall Time Comparison

Input Stream Size	CNN-FP	RC-FP	Sieve-Streaming
n = 100	124.23 s	374.01 s	139.44 s
n = 200	488.62 s	1561.40 s	594.23 s
n = 500	2904.61 s	9407.22 s	3953.24 s
n = 1000	11120.26 s	35598.39 s	18470.49 s

Question 2: Case Base Size: Figure 2 shows changing case base size during processing. CNN-FP and RC-FP size fluctuates, growing from the minimum to the limit, while Sieve-Streaming tends to maintain a smaller and more stable Case-Base size than CNN and RC.

Question 3: Competence: Figure 3 shows competence with the three methods. Competence is assessed using a randomly selected evaluation set, drawn from cases outside the case stream (the same set is used for all streams). Competence is calculated with the evaluation set based on the k-medoid utility function. CNN-FP and RC-FP show better initial performance for all streams, with some fluctuation as case base size changes. The competence of Sieve-Streaming increases gradually, finally surpassing CNN-FP and RC-FP.



(a) n = 100 (b) n = 200

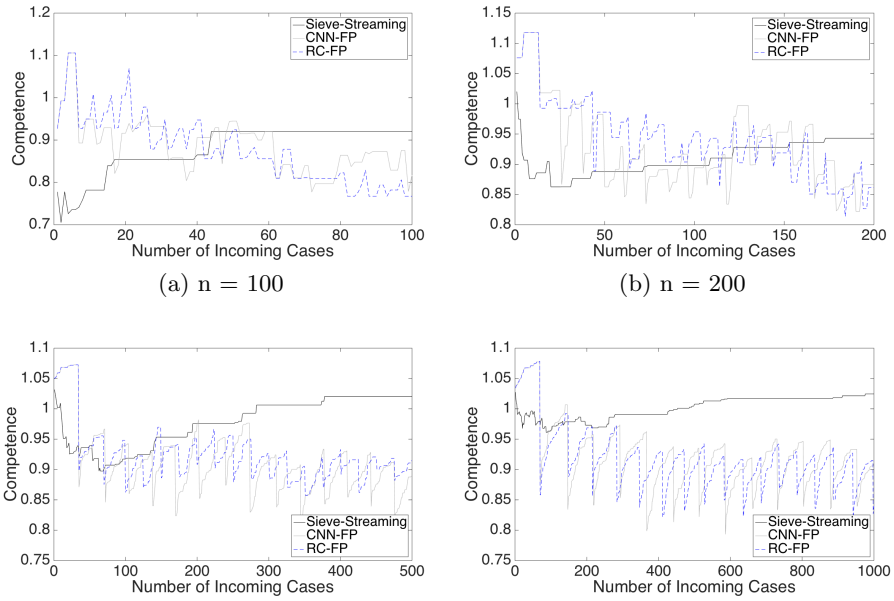
(c) n = 500 (d) n = 1000

Fig. 2. Case Base Size

Question 4: Retrieval Quality: We use accumulated distances between each incoming case and correspond retrieval case to measure retrieval accuracy to reflect the efficiency of maintenance strategies. Figure 4 shows that RC-FP always has the lowest accumulated retrieval distances. CNN has a lower accumulated retrieval distances over Sieve-Streaming for n=100 and n=200, however, when n=500, accuracy of Sieve-Streaming is better.

5 Conclusion

Knowledge discovery methods provide an important potential resource for CBR. This paper has explored the use of Sieve-Streaming for on-the-fly, incremental competence-based case deletion. The results suggest that it provides high quality case base compression, trading off some accuracy loss compared to the gold standard method RC-FP for much improved efficiency. We noted that the performance of Sieve-Streaming improved when the size of the case stream increased. A surprising result was that Sieve-Streaming required more processing time than CNN-FP. This may be an artifact of implementation, especially given that the implementations were not optimized. We are currently re-implementing all methods to enable more uniform comparison. The superior competence results suggest that the use of Sieve-Streaming compared to CNN-FP would still be justified,



(a) n = 100 (b) n = 200

(c) n = 500 (d) n = 1000

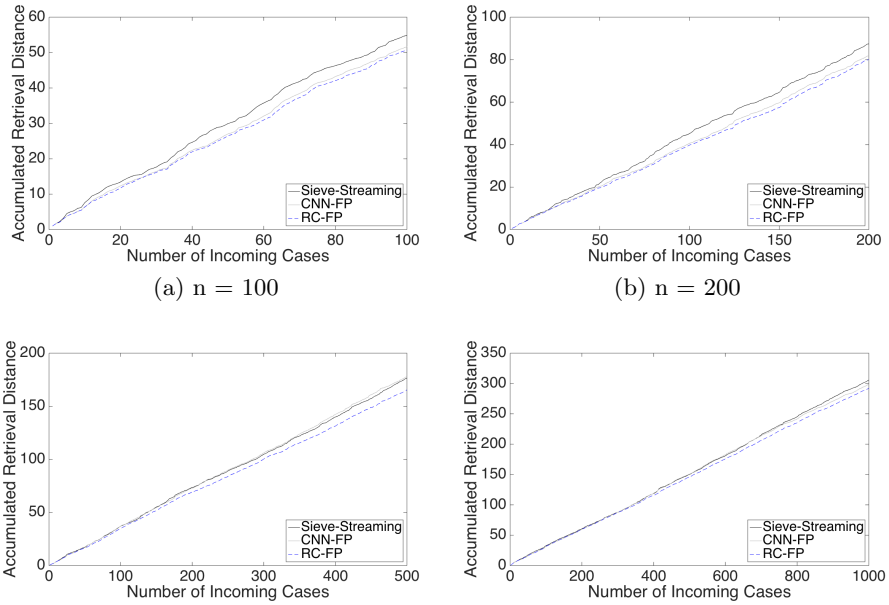
Fig. 3. Competence

but additional examination is needed to better understand the processing times required for all methods.

The results presented here are preliminary in that they report a limited set of trials, on a small case base, without a systematic effort to tune parameters. In future work, we plan to test the strategies with multiple large scale data streams and to examine further how the approach is affected by changing parameter settings.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–52 (1994), http://www.iiia.csic.es/People/enric/AICom.pdf
2. Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., Krause, A.: Streaming submodular maximization: Massive data summarization on the fly. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 671–680. ACM (2014)
3. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory 14, 515–516 (1968)
4. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis, vol. 344. John Wiley & Sons (2009)



(a) n = 100 (b) n = 200

(c) n = 500 (d) n = 1000

Fig. 4. Accumulated Retrieval Distance

5. Krause, A., Gomes, R.G.: Budgeted nonparametric learning from data streams. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 391–398 (2010)
6. Leake, D.B., Smyth, B., Wilson, D.C., Yang, Q.: Introduction to the special issue on maintaining case-based reasoning systems. Computational Intelligence 17(2), 193–195 (2001)
7. López de Mántaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M., Cox, M., Forbus, K., Keane, M., Aamodt, A., Watson, I.: Retrieval, reuse, revision, and retention in CBR. Knowledge Engineering Review 20(3) (2005)
8. McKenna, E., Smyth, B.: Building compact competent case-bases. In: Case-based reasoning research and development, pp. 329–342. Springer (1999)
9. Prasad, B.V.V.S.: A novel distance similarity measure on learning techniques and comparison with image processing. International Journal of Engineering Research and Development 2(8), 29–32 (2012)
10. Smyth, B., Keane, M.: Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. pp. 377–382. Morgan Kaufmann, San Mateo (1995)
11. Smyth, B., McKenna, E.: Competence models and the maintenance problem. Computational Intelligence 17(2), 235–249 (2001)
12. Wilson, D., Leake, D.: Maintaining case-based reasoners: Dimensions and directions. Computational Intelligence 17(2), 196–213 (2001)