CBR Meets Big Data: A Case Study of Large-Scale Adaptation Rule Generation

Vahid Jalali and David Leake

School of Informatics and Computing, Indiana University
Bloomington IN 47408, USA
vjalalib@indiana.edu, leake@indiana.edu

Abstract. Adaptation knowledge generation is a difficult problem for CBR. In previous work we developed *ensembles of adaptation for regression* (EAR), a family of methods for generating and applying ensembles of adaptation rules for case-based regression. EAR has been shown to provide good performance, but at the cost of high computational complexity. When efficiency problems result from case base growth, a common CBR approach is to focus on case base maintenance, to compress the case base. This paper presents a case study of an alternative approach, harnessing big data methods, specifically MapReduce and locality sensitive hashing (LSH), to make the EAR approach feasible for large case bases without compression. Experimental results show that the new method, BEAR, substantially increases accuracy compared to a baseline big data k-NN method using LSH. BEAR's accuracy is comparable to that of traditional k-NN without using LSH, while its processing time remains reasonable for a case base of millions of cases. We suggest that increased use of big data methods in CBR has the potential for a departure from compression-based case-base maintenance methods, with their concomitant solution quality penalty, to enable the benefits of full case bases at much larger scales.

Key words: Case-Based Reasoning, Ensemble of Adaptations for Regression, Locality Sensitive Hashing

1 Introduction

The growth of digital data is widely heralded. A 2014 article estimates that "[A]lmost 90% of the world's data was generated during the past two years, with 2.5 quintillion bytes of data added each day" [1]. Individual organizations collect data sets on an unprecedented scale. For example, in 2013, a single health care network in the U.S. state of California was estimated to have over 26 petabytes of patient data from electronic health records alone [2]. Big data methods and resources have changed the practicality of using such large-scale data, with inexpensive cloud computing services enabling processing data sets of unprecedented scale. However, these are not a panacea: making good use of large-scale data remains a challenge (e.g., [3]).

Case-based reasoning's ability to reason from individual examples and its inertia-free learning make it appear a natural approach to apply to big-data problems such as predicting from very large example sets. Likewise, if CBR systems had the capability to

handle very large data sets, that capability could facilitate CBR research on very large data sources already identified as interesting to CBR, such as cases harvested from the "experience Web" [4], cases resulting from large-scale real-time capture of case data from instrumented systems [5], or cases arising from case capture in trace-based reasoning [6].

However, realizing the potential of CBR to have impact on big data problems will depend on CBR systems being able to exploit the information in case bases with size far beyond the scale now commonly considered in the CBR literature. The case-based reasoning community has long been aware of the challenges of scaling up CBR to large case bases. The primary response has been case-base maintenance methods aimed at reducing the size of the case base while preserving competence (e.g., [7, 8]). Such methods have proven effective at making good use of case knowledge within storage limits. However, because compression methods delete some of the CBR system's knowledge, they commonly sacrifice some solution quality.

A key factor in success of CBR when applied to big data is efficient retrieval of cases. As CBR does not generalize beyond cases, it is extremely important to the success of a CBR system to be able to find required cases rapidly. In this paper we illustrate the practicality of applying big-data tools to increase the speed and scalability of CBR, using MapReduce and Locality Sensitive Hashing for finding nearest neighbors of the input query.

In previous work, we introduced and evaluated a method for addressing the classic CBR problem of acquiring case adaptation knowledge with *ensembles of adaptations for regression* (EAR) [9]. This work demonstrated the accuracy benefits of EAR [9–12], but also identified important efficiency concerns for large case bases. This paper presents a case study applying big data methods to addressing EAR's scale-up, leveraging techniques and frameworks well known to the big data community to enable large-scale CBR. It presents a new algorithm, BEAR,[1] applying the EAR approach in a MapReduce framework. The paper demonstrates that the use of big data methods substantially extends the size of case base for which the EAR approach is practical, to case bases of millions of cases even on a small Amazon Elastic MapReduce (EMR) cluster.[2]

The paper begins with a discussion of the relationship of big data and CBR, contrasting the "retain and scale up" approach of big data to the compression-based focus of case-base maintenance. It next introduces the EAR family of methods and the two big data methods applied in its new version, locality sensitive hashing [13] and MapReduce. With this foundation it introduces BEAR, a realization of EAR for big data platforms, and presents an experimental evaluation assessing BEAR's accuracy for a case base of two million cases. To assess the benefit of BEAR's ensemble approach it compares it to a baseline of a big data version of k-NN, using Locality Sensitive Hashing for implementing nearest neighbor search. It also shows that BEAR's approach helps alleviate the accuracy penalty that can result from using LSH instead of a traditional (exhaustive) approach for finding nearest neighbors, thus compensating for a potential drawback of using LSH. To assess the need for big data methods for BEAR's task and BEAR's

[1] Big data ensembles of adaptations for regression
[2] http://aws.amazon.com/elasticmapreduce/

scaleup potential, it also compares BEAR's scaleup performance to that of k-NN using a traditional (exhaustive) approach for finding nearest neighbors.

The evaluation supports the accuracy benefits of BEAR and that the speedup benefits of big data methods are sufficient to counterbalance the computational complexity of BEAR's rule generation and ensemble solution methods. Thus the use of big data methods may have benefits for CBR beyond simple speedups, by making practical the use of richer methods which can increase accuracy. Two million cases is large by the standards of current CBR practice, but true "big data" CBR will involve much larger data sets. The paper closes with a discussion of BEAR's potential for scaleup to such data sets.

2 Scaling CBR to Big Data

Big data has had a transformative effect on data management, enabling many enterprises to exploit data resources at previously unheard-of data scales. Large data sets such as electronic medical records collections may naturally be seen as containing cases; routine data capture in many domains could provide rich case bases. If cases can be retrieved sufficiently efficiently, CBR is an appealing method for large-scale reasoning because its lazy learning avoids the overheads associated with traditional rule mining approaches enables inertia-free adjustments to additional data, without the need for retraining.

However, CBR systems have seldom ventured into the scale of big data. For example, calculating metrics such as number of visitors or page views for a social media or e-commerce web site with hundreds of million users is a common practice at industry, but in current CBR research, experiments with tens of thousands of cases, or even much fewer, are common. Few CBR projects have considered scales up to millions of cases [14, 15], and to our knowledge, none have explored larger scales except a few exceptions such as a recent effort to apply big data methods focused on exact match only, rather than similarity-based retrieval [15].

When CBR research has addressed increased data sizes, the primary focus has been compression of existing data rather than scale-up. Considerable CBR research has focused on the efficiency issues arising from case-base growth. As the case base grows, the swamping utility problem can adversely affect case retrieval times, degrading system performance [16, 17]. Within the CBR community and the machine learning community studying instance-based learning, extensive effort has been devoted to addressing the swamping utility problem for case retrieval with case-base maintenance methods for controlling case-base growth, with the goal of generating case bases that are compact but retain coverage of as many problems as possible. Methods for developing compact competent case bases include selective deletion (e.g., [7, 18]), selective case retention (e.g., [19–21]), and competence-aware construction of case bases [8, 22–25]. Such methods generally trade off size against accuracy; they aim to retain as much competence as possible for a given amount of compression. This tradeoff has been seen as the price of making CBR feasible for domains in which the set of possible cases is large, but storage and processing resources are limited.

This paper argues that applying big data methods can change this calculus; that even for case bases on the order of millions of cases, big data methods can make the best case-base compression strategy *no compression at all*. If big data methods can enable CBR scale-up, dramatically increasing the feasibility of handling very large case bases, compression methods will be required only for extreme scale case bases—and, even for very large cases, might not be required at all in practice. Already, it has been observed that for common practical CBR tasks, even with conventional methods, case base size may not be an issue [26]; big data methods could bring CBR to bear on a new class of problems, at much larger scale.

3 Foundations of the Proposed Method

The case study in this paper focuses on applying CBR to numerical prediction tasks under big data settings, demonstrating the feasibility of big data approaches to provide good performance at scales on the order of millions of cases, with minimal quality loss. The method proposed in this paper builds on three currents of research. The first, from CBR, is the EAR family of methods [9] for case-based regression using ensembles of adaptations. The second, from big data is Locality Sensitive Hashing (LSH), a method for nearest neighbor search in big data platforms. The third is MapReduce, a popular framework for parallel processing of data.

3.1 The EAR Family of Methods

The acquisition of case adaptation knowledge is a classic problem for CBR. A popular approach to this problem, for numerical prediction (regression) tasks, is to generate adaptation rules automatically from the case base. The EAR family of methods solves numerical prediction problems using automatically-generated ensembles of adaptations to adapt prior solutions.

The EAR approach applies to any adaptation generation method, but it has been tested for a popular case-based rule generation method, the *Case Difference Heuristic*, which generates rules based on comparing pairs of cases. Given two cases A and B, with problem parts Prob(A) and Prob(B), and solution parts Sol(A) and Sol(B), the case difference heuristic approach assumes that problems with similar difference in their problem descriptions will have similar differences in their solutions. For example, for predicting apartment rental prices from a case base of rental properties and prices, if one apartment's monthly rent is $300 more than the rent of an otherwise highly similar apartment, and their difference is that the more expensive apartment has an additional bedroom, the comparison might suggest a general rule: *When the previous apartment case has one bedroom fewer, predict that the new apartment's rent will be $300 more than the rent of the previous apartment* (We note that many possible rules could be generated; the choice of rules is outside the scope of this paper.)

More precisely, for cases A and B, the case difference heuristic approach generates an adaptation rule applicable to a retrieved case C and problem P, for which the difference in problems of A and B is similar to the difference between the problem of C and P, i.e., for which diff(Prob(C),P) is similar to diff(Prob(A),Prob(B)). The new

Algorithm 1 EAR's basic algorithm

Input:
Q: input query
n: number of base cases to adapt to solve query
r: number of rules to be applied per base case
CB: case base
Output: Estimated solution value for Q

$CasesToAdapt \leftarrow$ NeighborhoodSelection(Q,n,CB)
$NewRules$: $\leftarrow$ RuleGenerationStrategy(Q,$CasesToAdapt$,CB)
for c in $CasesToAdapt$ **do**
    RankedRules $\leftarrow$ RankRules($NewRules$,c,Q)
    $ValEstimate(c) \leftarrow$ CombineAdaptations($RankedRules$, c, r)
end for
return CombineVals($\cup_{c \in CasesToAdapt} ValEstimate(c)$)

rule adjusts Sol(C) to generate a new solution N, such that diffSol(C),P is similar to diff(Sol(A),Sol(B)). For a more detailed description, see Hanney and Keane [27].

The results of the case difference heuristic depend on the cases from which rules are generated; the final results depend on the cases to which they are applied. The EAR methods estimate the solution of a case by retrieving a set of similar cases, adjusting their values by applying an ensemble of adaptation rules and combining the adjusted values to form the final prediction. Algorithm 1 explains the overall approach of EAR. In Algorithm 1, $NeighborhoodSelection$, $RuleGenerationStrategy$, and $RankedRules$ respectively denote methods for finding nearest neighbors, generating adaptation rules and adaptation retrieval in EAR4. More details are provided in [9].

EAR has different variations based on the subsets of cases it uses as source cases for solving input problems and the cases it selects as the basis for building adaptation rules. Different variants use different combinations of local and global cases. For example, EAR4, selects cases for both building solution and adaptation rules from the local neighborhood of the input problem. In this paper we focus on big data versions of EAR4 a family of EAR methods that generates both solutions and adaptations from the local neighborhood of the input query.

EAR has been shown to provide significant gains in accuracy over baseline methods [9]. However, because it depends on multiple case retrievals to generate adaptation rules for multiple case neighborhoods, its application for large case bases, using conventional CBR techniques, can be expensive. We have developed compression-based methods to help alleviate this [11], but like all compression-based methods, these trade off accuracy for compression. This motivated us to explore the application of big data techniques to the EAR approach.

3.2 Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) [13] was developed to decrease the time complexity of finding nearest neighbors for an input query in d-dimensional Euclidean space. LSH

achieves this goal by approximating the nearest neighbor search process; it uses families of hashing functions for which the probability of collision is higher for cases which are similar (in terms of their input features). Since the introduction of LSH, various schemes have been proposed to improve various aspects of the core method [28–30].

LSH groups similar items into different buckets by maximizing the probablity of collsion for similar items. In contrast to nearest neighbor search, LSH does not require comparing a case with other cases to find its nearest neighbors. Instead, if an appropriate hashing function is used it is expected that a case and its nearest neighbors end up in the same bucket. LSH is an approximation method and it does not guarantee grouping a case and its nearest neighbors into the same bucket. However, though LSH sacrifices accuracy for efficiency, it has been demonstrated that LSH can be sufficiently accurate in practice [30].

Previous experiments have studied the performance of k-NN using locality-sensitive hashing to retrieve nearest neighbors (e.g. [29]), showing that LSH achieves higher efficiency compared to linear k-NN with the expected loss in accuracy. In this paper, we explore both the efficiency benefits of LSH for EAR's ensemble method, and the ability of EAR's ensemble method to provide good performance despite the approximations made by LSH.

3.3 MapReduce

MapReduce is a framework that enables parallel processing of data. The "map" step reads and filters data. Next, data is distributed among different nodes/reducers based on a particular field (key) where data is summarized and desired metrics are calculated for the subset of data in each reducer. Different implementations of the MapReduce framework are available. A popular open source implementations of MapReduce framework which is commonly used in industry is *Apache Hadoop*. Recent work by Beaver and Dumoulin [15] has applied MapReduce for CBR, but only for retrieval of exact match cases, rather than for similarity-based retrieval.

4 BEAR: A General Approach to Applying EAR Family Methods to Big Data

4.1 Overview

BEAR (Big-data Ensembles of Adaptations for Regression) is a realization of the EAR family of case-based regression methods in a big data platform, aimed at decreasing the cost of finding nearest neighbors, a process for which the computational expense may become serious issue for very large case bases. The EAR family of methods must identify nearest neighbors at three steps in their processing: to select source cases to adapt, to select candidate cases to build adaptation rules, and to retrieve adaptation rules. Among these three steps, retrieving adaptations is potentially the most challenging, because, for a case base of size n, the upper bound on the number of possible adaptation rules to generate is $O(n^2)$. However, EAR4 mitigates this by limiting the cases to participate in rule generation process to the cases in the local neighborhood of the input query.

Therefore, in EAR4 it is likely that source case retrieval will be a more serious resource issue than the adaptation retrieval.

To overcome the challenges raised by the size of the case/rule base, we have investigated minimizing the size of the case base [31] and adaptation rule set [12] to improve performance (in terms of the required computing resources). Although these methods can be useful for reducing the case/rule base size, the process of case/rule base size reduction can still be time consuming and costly; they have not been applied to case bases with more than a few thousands cases. In contrast to these methods, BEAR aims to mitigate challenges brought by the size of the case base by leveraging existing frameworks and algorithms for processing big data to yield accurate estimates rapidly, using locality sensitive hashing on top of a MapReduce framework.

4.2 BEAR's Architecture

BEAR consists of two main modules: LSH for retrieving similar cases and EAR for rule generation and value estimation. The architecture of the system is designed to work in a MapReduce framework. In the map step cases and queries are read and hashed to different buckets using LSH. Cases and queries with identical hashed keys are sent to the same reducer node. In the reduce step two main activities are done: First, the nearest neighbors of each query (from the cases in the same reducer) are determined; Next, depending on the selected EAR method (i.e. EAR4), the adaptation rules are also generated.

For EAR4, which only uses local cases, adaptation rules are only generated and retrieved within the same bucket (for some other variations of EAR, e.g. EAR5, which uses global case information, adaptations would be generated within all buckets and the generated adaptations from different buckets unioned together to form the rule base from which adaptations are retrieved). The final estimates are generated by applying an ensemble of adaptation rules for adjusting the base cases' values and combining those adjusted values. Figure 1 summarizes BEAR's process for estimating case solutions.

In Figure 1, circles represent cases in the case base and the square represents the input query. Cases are hashed and transferred to different reducers based on their hashed keys. Next, adaptation rules are generated based on the cases hashed to the same reducer as the input query. Finally EAR4 is used to estimate the solution of the input query. Depending on the implementation of BEAR, there could be another step in its process flow (not depicted), using a similarity measure such as Euclidean distance to filter out cases in the same bucket as the input query based on a distance threshold or a predefined number of nearest neighbors.

The use of MapReduce offers the advantage of being able to process multiple queries simultaneously, enabling, for example, millions of queries to be processed in parallel. Also, even when single queries are processed sequentially, the use of MapReduce enables processing the cases in the case base in parallel on multiple nodes, rather than having to sequentially process all cases to select those whose LSH hashing keys match that of the query, which would not be scalable.

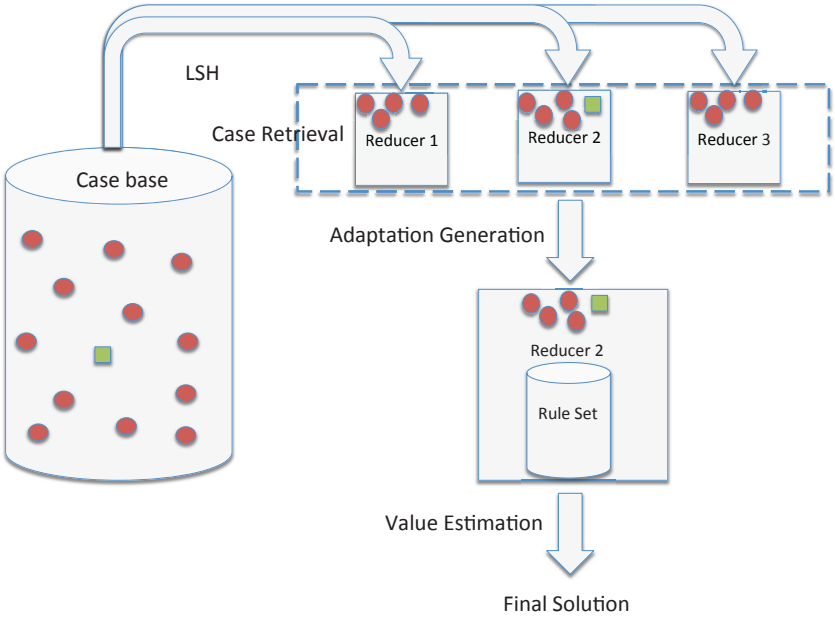

Fig. 1. Illustration of Global BEAR process flow.

5 Evaluation

Our evaluation tests the execution time benefits associated with big data methods for scaled up case-based regression with ensembles of adaptations for regression, and studies whether BEAR's use of an ensemble of adaptations improves accuracy compared to applying a non-ensemble approach, when both methods use LSH for retrieval.

Because LSH is not guaranteed always to retrieve the optimal neighbors, we expect that using LSH rather than exhaustive search for nearest neighbors will somewhat degrade overall performance. Consequently, another question is whether the ensemble of adaptations approach, applied in the context of LSH, helps to mitigate this drawback.

This involves two types of tests. The first is a test of the accuracy of "traditional" k-NN, for which neighbors are selected exhastively, compared to that when LSH is used to select (an approximate set of) neighbors, and when LSH is used in conjunction with BEAR. The second, is ablation study to determine how much of the performance of BEAR can be ascribed to its ensemble method, as opposed to the fact that it uses adaptations, while k-NN does not. For the purposes of this case study, we test for a particular LSH implementation, described below, which we refer to as LSH1.

Specifically, our experiments address the following questions:

Table 1. Characteristics of the test domains

Domain name	# features	# cases	Avg. cases/solution	sol. sd
Auto	13	195	1.1	8.1
MPG	7	392	3.1	7.8
Housing	13	506	2.21	9.2
Power	7	2,049,280	1.09	1.06

1. Q1: How does the accuracy of BEAR compare to that of a baseline of k-NN using LSH1 for finding nearest neighbors?
2. Q2: How does the ensemble approach of BEAR increase accuracy compared to applying single adaptations for adjusting base case values?
3. Q3: How does the accuracy of BEAR using LSH1 compare to that of exhaustive k-NN?
4. Q4: How does execution efficiency of BEAR compare to that of traditional (non-LSH) k-NN?

5.1 Experimental Design

We evaluated BEAR on four sample domains from the UCI repository [32]: Automobile (Auto), Auto MPG (MPG), Housing, and electric power consumption (Power). The goal for these domains is respectively to predict the auto price, fuel efficiency (miles per gallon), property value, and household global minute-averaged active power (in kilowatts). For all data sets records with unknown values are removed and feature values are normalized by subtracting feature's mean from the value and dividing the result by standard deviation of the feature's values. (Cases with missing features could be handled by standard feature imputation methods, but this is beyond the scope of our experiment.) In addition, for domains with non-numeric features, only numeric features are used. The accuracy is measured in term of Mean Absolute Error (MAE) in all experiments and ten-fold cross validation is used for conducting the experiments. For all domains parameters are tuned using hill climbing. In all experiments BEAR's performance is compared with that of an implementation of k-NN based on LSH which we refer to simply as k-NN from this point forward. Sample domains are chosen so that they cover both smaller and huge case bases. Table 1 summarizes the characteristics of the sample domains.

All records and features were used for the Housing (506) domain. Auto, MPG, and Power contained some records with unknown feature values, which were removed (46 out of 205 for Auto, 6 out of 398 for MPG and 25979 out of 2075259 for Power). For all domains only numeric attributes are used in the experiments to enable the application of p-stable locality sensitive hashing. All features of MPG and Housing were numeric, but 10 non-numeric features were removed from Auto and 1 from Power. We note that the numeric features are not required by the general BEAR method.

We note that LSH is a family of methods. Our implementation of BEAR uses Apache DataFu, originally introduced in [33], to support locality sensitive hashing. The corresponding class from Apache DataFu used in BEAR is $L2PStableHash$, with a

Domain name	MAE for k-NN	MAE for BEAR	% improvement over k-NN
Auto	2.04	1.18	42.14%
MPG	2.62	2.06	21.40%
Housing	3.73	2.84	23.98%
Power	0.15	0.10	36.01%

Table 2. Accuracy comparison for k-NN and BEAR

2-stable distribution and default parameter settings. It is important to note that because the focus of our experiments is primarily the comparison of BEAR to LHS-based k-NN, and both methods are based on the same version of LSH, the specific variant chosen is not significant to our results.

It also uses EAR4's Weka plugin's code [34], combined with some common functionality from Weka [35] to generate adaptation, retrieve and apply them and build the final prediction. The experiments are run on an EMR amazon cluster with one m3.xlarge master node and ten c3.2xlarge core nodes.

5.2 Experimental Results

Q1: How does the accuracy of BEAR compare to that of a baseline of k-NN using LSH1 for finding nearest neighbors? To address Q1, we conducted experiments to compare BEAR with k-NN using LSH1. In all experiments BEAR's estimations are generated using EAR4 to generate adaptation rules, retrieve adaptations and build final estimations based on nearest neighbors retrieved by LSH1. The experiments report estimation error in terms of Mean Absolute Error. Table 2 summarizes the results for four sample domains. In all domains BEAR outperforms k-NN by substantial margins. A one side paired t-test with 95% confidence interval was used to assess the statistical significance of results achieved by BEAR in the smaller case bases (we excluded the Power domain from the statistical significance analysis because of the very large size of the case base). The null hypothesis is that the MAE of BEAR is greater than that of k-NN. The results of the t-test showed that $p<.01$, so the improvement of BEAR over k-NN is significant.

Q2: How does the ensemble approach of BEAR increase accuracy compared to applying single adaptations for adjusting base case values? To study the effect of applying an ensemble of adaptations on estimations' accuracy we implemented an ablated version of BEAR, *BEAR1* in which only one adaptation is applied to adjust case values. Figure 2 shows the percent of improvement in MAE over k-NN for BEAR and BEAR1. BEAR1 outperforms k-NN in all domains, but the improvement is less than that of BEAR over k-NN, which shows the benefit of ensemble approach of BEAR.

Q3: How does accuracy of k-NN and BEAR using LSH1 compare to that of traditional k-NN? Because LSH-based retrieval does not guarantee always selecting the true nearest neighbors to a case, some accuracy penalty may be expected. However, we hypothesize that BEAR's ensemble method helps alleviate the associated quality degradation. We tested this hypothesis by comparing the performance of traditional k-NN on the Auto,

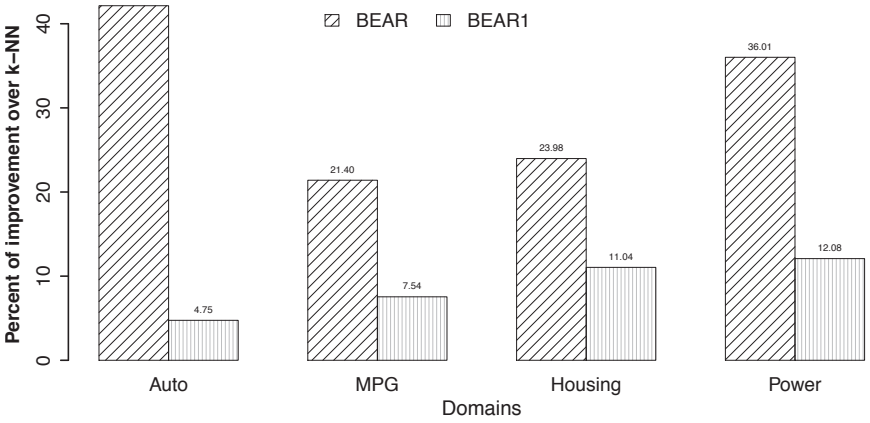

Fig. 2. BEAR vs. BEAR1

MPG, and Housing domains, with the previously-described testing scenario (because the relatively large size of the Power domain made traditional k-NN excessively expensive, we did not compare performance in the Power domain). MAE of traditional k-NN for auto and mpg domains was 1.31 and 2.14 respectively, compared to 1.18 and 2.06 for BEAR. However, in the housing domain, traditional k-NN slightly outperformed BEAR, with an MAE of 2.68, versus 2.84 for BEAR (approximately a 5% drop). Thus BEAR retains accuracy comparable to traditional k-NN.

Q4: How does execution efficiency of BEAR compare to that of traditional k-NN? Figure 3 shows the run time in seconds of traditional (non-LSH) k-NN and BEAR on different subsets of the Power domain, ranging from 20,000 to 820,000 randomly selected cases. The recorded run times are the total time for conducting ten fold cross validation. All experiments were run on a single machine with 16 GB memory and 2.8 GHz Intel Core i7 processor. Weka's [35] IBK package is used as the implementation of k-NN. For smaller case base sizes (e.g. 20,000 cases) k-NN is quite fast; a ten-fold cross validation test takes on the order of 1 second. For a case base approximately 31 times larger (615,000 cases) the test takes 4.5 hours—approximately 16,000 times longer. When the size is increased to 820,000 cases, time increases to 21 hours.

On the other hand, using LSH and parrallelizing the process over different nodes enables EAR4 to process same sizes of the case based in significantly less time. An interesting observation is that for a case base of 20,000 cases it actually takes less time for k-NN to yield the results than BEAR (it takes k-NN 10 seconds while takes BEAR 265 seconds). This is because of the communication overhead of MapReduce framework which makes applying big data techniques less efficient when applied to small case bases. However, k-NN run time increases very rapidly compared to BEAR as case

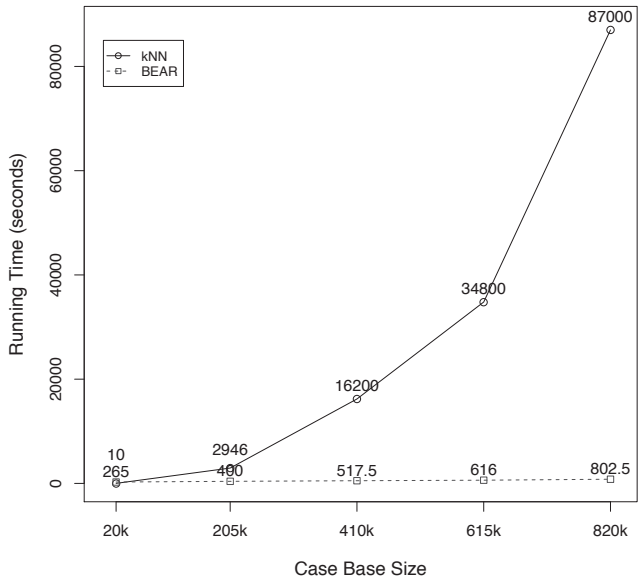

Fig. 3. Running time of k-NN and BEAR for different sizes of Power domain

base size increases to 205,000 cases, while the run time of BEAR increases at a much lower rate. Even when the entire power case base is used (over 2 million cases), BEAR takes less than 20 minutes to complete the experimental run on an EMR cluster with the configuration described in section 5.1. We note that this corresponds to less than .1 second per problem solved, even on a small cluster. This supports the need to move to big data methods for practical large-scale CBR.

5.3 Overall Perspective: Scale-Up, Time, Space, and Accuracy

The experiments in this case study illustrate the ability of the BEAR approach, which combines ensemble adaptations with locality-sensitive hashing, both to remain efficient for large scale data and to provide substantial accuracy increases compared to non-ensemble adaptation of cases retrieved by LSH, and for k-NN using LSH. More generally, it illustrates the potential of CBR's reasoning capability (in the form of case adaptation) to provide strong benefits not present in big data/retrival-only methods. The largest test case base used in our experiments has two million cases, and was run on a small cluster (with ten core nodes). However, BEAR could easily be applied to substantially larger case bases with tens or even hundreds of millions of cases, and expected running times comparable to that reported in this paper, by increasing the computational

resources to the level common in industrial settings (e.g. a cluster with hundred nodes or more).

In many treatments of case-based maintenance in CBR, having/maintaining a large number of cases is assumed to correspond to degraded retrieval and processing time, potentially requiring sacrificing information by case deletion. However, leveraging big data platforms and techniques it is possible to avoid information loss, and consequently yield more accurate solutions, by retaining full case bases impractical for conventional methods and using them efficiently. The ability of big data to integrate both with flexible similarity-based retrieval and case adaptation is promising for the general ability of much scaled up CBR. This, in turn, could open the door to very large-scale CBR, with near-instant retrieval from case bases with millions of cases, plus the potential accuracy benefits of avoiding the need for case-base compression in many domains.

The previous experiments focus on the ability of big data methods to enable using full case bases. However, given the speed of those methods, for time-critical tasks it could even be feasible to sacrifice additional space for the sake of time. As a concrete example, for numerical prediction using BEAR in a domain with millions of cases (e.g. the Power domain), it would be possible to pre-process the data to generate the LSH keys for each case and store all cases with their corresponding hash keys in a NoSQL database. Because, in LSH, each record can be hashed with a set of hash families, this results in having case bases of size orders of magnitude greater than the original case base. However, with this NoSQL design, applying a method such as EAR4 on top of big data methods could enable processing thousands of queries in a matter of a few seconds even without MapReduce. Even for millions of queries, using MapReduce for query processing only and using the NoSQL database for case retrieval, average response time per query could still be in range of a few milliseconds.

6 Conclusion and Future Directions

In this paper we illustrated the practicality of a big-data version of ensembles of adaptation for regression, implemented in BEAR, which uses MapReduce and Locality Sensitive Hashing for finding nearest neighbors of the input query. We consider the results encouraging for the application of big data methods to the fuller CBR process, to exploit not only larger case bases but also collections of adaptation rules, without compression. Such methods might also present opportunities for CBR approaches to big data problems more generally, as an alternative to rule mining. In addition, the BEAR approach improves performance compared to the big data baseline k-NN with LSH1, and preserves comparable performance to that of much more costly traditional k-NN.

As future directions, we intend to compare accuracy and speed performance achieved by case base compression to those of BEAR, to better understand the trade-offs between traditional and big data methods for CBR. Given BEAR's efficiency, we also intend to extend our methods to test more computationally expensive variations of the EAR family of methods as the case-base estimator module in BEAR. For example, generating rules from neighborhoods other than the local neighborhood of the input query—which requires consideration of many more cases—and adding contextual considerations in adaptation retrieval, have produced good small-scale results [10],

but with high costs that raised concerns for their large-scale applicability by conventional CBR methods. The BEAR framework suggests a path for making practical such case-intensive methods.

Previous CBR research has applied big data methods to CBR when case retrieval relies on exact match (string-based) retrieval [15]; BEAR enables similarity-based matching. However, an importaant problem is how to apply these techniques to structured cases.

References

1. Kim, G.H., Trimi, S., Chung, J.H.: Big-data applications in the government sector. Communications of the ACM **57**(3) (2014) 78–85
2. Hoover, W.: Transforming health care through big data. Technical report, Institute for Health Technology Transformation (2013)
3. Greengard, S.: Weathering a new era of big data. Communications of the ACM **57**(9) (2014) 12–14
4. Plaza, E.: Semantics and experience in the future web. In: Proceedings of the Ninth European Conference on Case-Based Reasoning, Springer (2008) 44–58
5. Ontañón, S., Lee, Y.C., Snodgrass, S., Bonfiglio, D., Winston, F., McDonald, C., Gonzalez, A.: Case-based prediction of teen driver behavior and skill. In: Case-Based Reasoning Research and Development, ICCBR 2014, Berlin, Springer (2014) 375–389
6. Cordier, A., Lefevre, M., Champin, P.A., Georgeon, O., Mille, A.: Trace-based reasoning – modeling interaction traces for reasoning on experiences. In: Proceedings of the 2014 Florida AI Research Symposium, AAAI Press (2014) 363–368
7. 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, San Mateo, Morgan Kaufmann (1995) 377–382
8. Smyth, B., McKenna, E.: Building compact competent case-bases. In: Proceedings of the Third International Conference on Case-Based Reasoning, Berlin, Springer Verlag (1999) 329–342
9. Jalali, V., Leake, D.: Extending case adaptation with automatically-generated ensembles of adaptation rules. In: Case-Based Reasoning Research and Development, ICCBR 2013, Berlin, Springer (2013) 188–202
10. Jalali, V., Leake, D.: A context-aware approach to selecting adaptations for case-based reasoning. In: Modeling and Using Context. Springer, Berlin (2013) 101–114
11. Jalali, V., Leake, D.: Adaptation-guided case base maintenance. In: Proceedings of the Twenty-Eighth Conference on Artificial Intelligence, AAAI Press (2014) 1875–1881
12. Jalali, V., Leake, D.: On retention of adaptation rules. In: Case-Based Reasoning Research and Development, ICCBR 2014, Berlin, Springer (2014)
13. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. STOC '98, New York, NY, USA, ACM (1998) 604–613
14. Daengdej, J., Lukose, D., Tsui, E., Beinat, P., Prophet, L.: Dynamically creating indices for two million cases: A real world problem. In: Advances in Case-Based Reasoning, Berlin, Springer (1996) 105–119
15. Beaver, I., Dumoulin, J.: Applying mapreduce to learning user preferences in near real-time. In: Case-Based Reasoning Research and Development, ICCBR 2014, Berlin, Springer (2014) 15–28

16. Francis, A., Ram, A.: Computational models of the utility problem and their application to a utility analysis of case-based reasoning. In: Proceedings of the Workshop on Knowledge Compilation and Speed-Up Learning. (1993)
17. Smyth, B., Cunningham, P.: The utility problem analysed: A case-based reasoning perspective. In: Proceedings of the Third European Workshop on Case-Based Reasoning, Berlin, Springer (1996) 392–399
18. Craw, S., Massie, S., Wiratunga, N.: Informed case base maintenance: A complexity profiling approach. In: Proceedings of the Twenty-Second National Conference on Artificial Intelligence, AAAI Press (2007) 1618–1621
19. Muñoz-Avila, H.: A case retention policy based on detrimental retrieval. In: Proceedings of ICCBR-99. (1999)
20. Ontañón, S., Plaza, E.: Collaborative case retention strategies for CBR agents. In: Case-Based Reasoning Research and Development: Proceedings of the Fifth International Conference on Case-Based Reasoning, ICCBR-03, Berlin, Springer-Verlag (2003)
21. Salamó, M., López-Sánchez, M.: Adaptive case-based reasoning using retention and forgetting strategies. Know.-Based Syst. **24**(2) (March 2011) 230–247
22. Zhu, J., Yang, Q.: Remembering to add: Competence-preserving case-addition policies for case base maintenance. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1999) 234–241
23. Angiulli, F.: Fast condensed nearest neighbor rule. In: Proceedings of the twenty-second international conference on Machine learning, New York, ACM (2005) 25–32
24. Wilson, D., Martinez, T.: Reduction techniques for instance-based learning algorithms. Machine Learning **38**(3) (2000) 257–286
25. Brighton, H., Mellish, C.: Identifying competence-critical instances for instance-based learners. In: Instance Selection and Construction for Data Mining. Volume 608 of The Springer International Series in Engineering and Computer Science. Springer, Berlin (2001) 77–94
26. Houeland, G., Aamodt, A.: The utility problem for lazy learners - towards a non-eager approach. In Bichindaritz, I., Montani, S., eds.: Case-Based Reasoning Research and Development, ICCBR 2010, Berlin, Springer (2010) 141–155
27. Hanney, K., Keane, M.: The adaptation knowledge bottleneck: How to ease it by learning from cases. In: Proceedings of the Second International Conference on Case-Based Reasoning, Berlin, Springer Verlag (1997) 359–370
28. Gionis, A., Indyk, P., Motwani, R., et al.: Similarity search in high dimensions via hashing. In: VLDB. Volume 99. (1999) 518–529
29. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: IEEE International Conference on Computer Vision ICCV. (2009)
30. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry. SCG '04, New York, NY, USA, ACM (2004) 253–262
31. Jalali, V., Leake, D.: Adaptation-guided case base maintenance. In: Proceedings of the Twenty-Eighth Conference on Artificial Intelligence, AAAI Press (2014) 1875–1881
32. Frank, A., Asuncion, A.: UCI machine learning repository (2010) `http://archive.ics.uci.edu/ml`.
33. Hayes, M., Shah, S.: Hourglass: A library for incremental processing on hadoop. In: Big Data, 2013 IEEE International Conference on. (Oct 2013) 742–752
34. Jalali, V., Leake, D.: Manual for EAR4 and CAAR weka plugins, case-based regression and ensembles of adaptations, version 1. Technical Report TR 717, Computer Science Department, Indiana University, Bloomington, IN (2015)
35. Witten, I., Frank, E., Hall, M.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Third edn. Morgan Kaufmann, San Francisco (2011)