SCALING PERSONALIZED HEALTHCARE WITH BIG DATA

Keith Feldman
University of Notre Dame,
Notre Dame, Indiana
USA
kfeldman@nd.edu

Nitesh V. Chawla
University of Notre Dame,
Notre Dame, Indiana
USA
nchawla@nd.edu

Abstract

Today the healthcare industry is undergoing one of the most important and challenging transitions to date, the move from paper to electronic healthcare records. While the healthcare industry has generally been an incrementally advancing field, this change has the potential to be revolutionarily. Using the data collected from these electronic records exciting tools such as disease recommendation systems have been created to deliver personalized models of an individual's health profile. However despite their early success, tools such as these will soon encounter a significant problem. The amount of healthcare encounter data collected is increasing drastically, and the computational time for these applications will soon reach a point at which these systems can no longer function in a practical timeframe for clinical use. This paper will begin by analyzing the performance limitations of the personalized disease prediction engine CARE (Collaborative Assessment and Recommendation Engine). Next it will detail the creation and performance of a new single patient implementation of the algorithm. Finally this work will demonstrate a novel parallel implementation of the CARE algorithm, and demonstrate the performance benefits on big patient data.

Keywords: personalized healthcare, big data, CARE

Introduction

Medical research has taken place for decades. It has provided what we as a society feel are some of the greatest modern achievements, from the discovery of bacteria and viruses to the development of antibiotics. Today, as the healthcare industry begins its transition into the digital age it is easy to see the event as a mere coming of age, simply the transformation of the medical community's paper documentation into electronic form. However, it provides so much more. This transition has laid the foundation for another fundamental advancement in the field of healthcare, the evolution from preventative care into personalized treatment plans.

It has been well documented that early detection and treatment of many diseases is directly correlated with improved health outcomes for the patient (Etzioni et al., 2003) (Wilkinson, Donaldson, Hurst, Seemungal, & Wedzicha, 2004) (Lard et al., 2001). As a result, regular so called "wellness-visit programs" have been implemented by many companies and care providers in order to promote preemptive testing for certain conditions (Kickbusch & Payne, 2003; Ozminkowski et al., 2002). However, as the identification and treatment of these diseases are performed in the same manner for multiple individuals based primarily on their current health state, i.e. age, gender, race, prior lab results, etc. this type of care falls closer to preventative medicine than personalized care.

Where as prior studies have guided preventative medicine treatment strategies by providing historical probabilistic models based on the outcomes of patients who developed similar conditions, new predictive techniques can help create personalized models of a patients future health risks tailored to the individual's health profile. In order to create these personalized models, data mining techniques have been applied to population-level health data aggregated from electronic healthcare records (EMR).

While classical data mining methodology such as clustering, decision trees and cohort analysis produced encouraging results, there was unfortunately a problem (Jensen, Jensen, & Brunak, 2012; Bellazzi & Zupan, 2008). As with paper records, each additional medical encounter by a patient resulted in additional data added to their electronic health record, and the quantity of data soon exceeded the ability of standard data processing techniques. In response, new data processing techniques and algorithms are being created, such as Google's MapReduce, Yahoo's Hadoop, etc. (Mayer-Sch¨onberger & Cukier, 2013). These techniques utilize the concepts of task segmentation and distributed computing in order to alleviate some of the computational load from a single machine, and allow for significantly improved runtimes for parallelizable tasks. Due to the time critical nature of medical conditions, the utility of any model created is directly proportional to the time taken to create it. As such we must focus on training time of a model order to allow personalized healthcare models to be created within a useful timeframe.

Among the most notable examples from emerging Electronic Medical Records (EMR) based technology, is the disease prediction model. These models utilize a patient's personal healthcare data in order to rank the likelihood of the individual obtaining specific diseases. One such idea came from the University of Notre Dame in the form of a disease prediction technique called CARE (Collaborative Assessment and Recommendation Engine) (Davis, Chawla, Christakis, & Barabási, 2010). The CARE algorithm in its current state is extremely accurate, with an implementation already being licensed for clinical use.

However despite CARE's effectiveness, one of the algorithm's foundational features, the ability to train risk models from population level healthcare records, has the potential to be- come one of the greatest implementation weaknesses. The CARE algorithm utilizes immense amounts of individual healthcare encounters in order to build a detailed similarity model for a specific individual, and collaborative filtering is intrinsically a computationally expensive algorithm. These facts combined with the ever-increasing amount of EMR encounter data present in hospital databases create a major operational issue.

This paper will focus on the primary issue of CARE's usability in a clinical setting. It will begin by identifying the limitations of the current CARE algorithm, and aim to provide a set of optimal performance parameters. Next some of the accuracy limitations will be addressed through the creation of a single patient version of CARE. Finally this work will demonstrate a novel distributed computing implementation of the CARE algorithm. This implementation will address issues with both execution time and disease coverage while attempting to provide near industry level performance.

Related Work

At the time of its publication, the CARE algorithm was the first of its kind, receiving a United States patent. However, to date many other diseases recommendation systems have been created (Tassy & Pourqui´e, 2013; Jensen et al., 2012; Austin, Tu, Ho, Levy, & Lee, 2013; AbuKhousa & Campbell, 2012). While these systems utilize many different machine learning and data mining techniques in order to produce their recommendations, each still potentially suffers from the dependence on high volume datasets. Typically these systems fall into two main categories, making use of either a patient's phenotypic profile, or their medical, disease and family histories as the training set of disease occurrences. Among the most widely known is the system HARM (McCormick, Rudin, & Madigan, 2012). Similarly to CARE, HARM is a personalized disease recommendation system, but rather than of using collaborative filtering HARM utilizes a significantly more complex mathematical model based on association rules. However as with CARE, and many of the other systems mentioned above, the authors of HARM do not discuss the potential for parallelization or distributed computing in their paper. Conversely, it has already been well established that distributed computing can provide significant improvement in runtime for computationally expensive systems (Goil & Choudhary, 1997; Stonebraker et al., 2010).

Collaborative filtering techniques like those employed by CARE have been used for some time in online product recommendation systems such as Amazon.com (Linden, Smith, & York, 2003). However, their application to disease prediction is relatively new. This practice has been brought about by a fundamental shift in how we think about diseases. Recently there has been a focus on modeling diseases as a network rather than isolated instances, allowing for the utilization of numerous networking-modeling techniques (Steinhaeuser & Chawla, 2009). However, healthcare information is extremely private, and the difficulties associated with of transporting and housing large scale healthcare data platforms have been some of the major obstacles preventing techniques such as these from widespread adoption.

There exists some prior work evaluating privacy when using collaborative filtering techniques on distributed data sets, such as the work done by Berkvosky el. in (Berkovsky, Eytani, Kuflik,& Ricci, 2007). This paper details the concern of passing around sensitive information just to perform calculations on the data. However in his implementation, Berkvosky details a method for subset data selection in order to pass a minimal amount of identifiable information to the system. The solution proposed in our paper aims to take the method one step further, and rather than distribute a minimal data subset for computation, distribute the computation to each data site. Further, this paper also aims to address privacy concerns by transmitting only the result of calculations over the network. The architecture provided in this paper is more akin to the work described in class MapReduce problem, where the data are summarized at each node and then these summary results are returned to the requester (Dean & Ghemawat, 2008).

Additional work related to the concept of personalized distributed data can be seen in the Lathia et. paper (Lathia, Hailes, & Capra, 2007). Lathia details a method for creating a custom similarity ranking based on random instances to protect the privacy of data. This similarly data could then be passed around without fear of revealing personalized information.

Current CARE Architecture

The current CARE architecture can be seen in Figure 1 and is fairly straightforward. The basic steps for the algorithm are detailed below.

1. CARE begins with an individual presenting a set of diseases. This set is the accumulation of diseases over their personal medical history.

2. The individual's disease similarity is then compared to all other patients in the provider's existing database and an initial filtering is done.

   (a) This filtering partitions the total dataset to include only those patients with whom the current test patient has some disease similarity, as collaborative filtering will yield no benefit between two individuals who do not have any disease in common.

3. Collaborative filtering is then performed on this filtered dataset.

4. Finally a probabilistic ranking of diseases for the individual is returned.

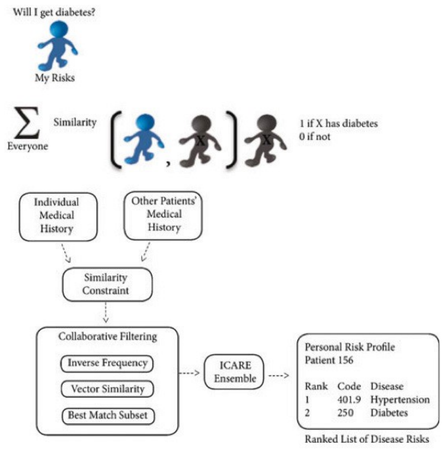

Figure 1: Current CARE Implementation (Chawla & Davis, 2013).

Data

The data used in this analysis is the same dataset utilized within the CARE research. The data consists of anonymized Medicare claims records collected by the Harvard University Medical School. There are approximately 13 million individual patients, accounting for just over 32 million hospital visits, and contains a total of 18 thousand unique disease codes. Each record represents a single hospital visit and is comprised of a patient ID number and up to 10 individual diagnoses from the visit. The diagnosis codes are defined by the International Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM), published by the World Health Organization (WHO) (Slee, 1978).

Through the ICD-9-CM code each disease is given a unique code, which can be up to 5 characters long. These codes may include specifics of the condition, such as the anatomical location. However, these fine-grained details are not required for the CARE algorithm, and as a result the 5 digit diagnosis codes can be collapsed to a 3-digit generalization of the diseases. For example codes 461.0 and 461.1 can be collapsed into the generic diagnosis code 461. The correctness of this generality is documented within the CARE paper, and as such will be used going forward in this work as well (Davis et al., 2010).

It is important to note that a disease may be diagnosed to an individual multiple times throughout their medical history. However, as multiple diseases are not useful when comparing patient's disease sets, only unique diseases are required for recommendations. Figure 2 shows that the average number of unique diseases converges to approximately 7 per patient over the full dataset. This value will be used when identifying outliers from the randomly selected patients, helping to reduce the bias between datasets and execution time.

Sequential CARE Implementation

As stated prior the current implementation of CARE is executed in a sequential manner. Our evaluation began with a detailed benchmarking on the existing CARE architecture. The goal of these benchmarks was to identify areas of computational resource restrictions, as well as any areas of algorithmic complexity that could not be solved with improved hardware.

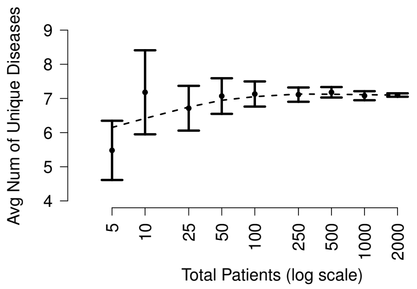

Figure 2: Average Number of Unique Diseases Per Patient

CARE was designed to operate within a clinical healthcare environment, and as this environment operates in a time-critical manner the primary benchmark metric used was execution time. Further a secondary metric for evaluation is the total number of patients used for training, as a proxy for recommendation accuracy. This proxy is a result of the sparse nature of disease networks. A large sampling of patients is required in order to ensure sufficient similarity comparisons for the disease ranking calculations.

Evaluation Environment

To perform the evaluation of CARE the Opteron machine at the Notre Dame Center for Re- search Computing (CRC) was utilized. This machine contains 4, 16-core 2.3 GHz AMD Opteron processors with 128 GB RAM.

As this machine is a time-shared system it is likely to experience performance fluxuations, as a result of user process load. In order to account for this all simulations were run 5 times, and the results averaged to obtain stable performance estimates. Additionally, prior to each benchmark CARE was run once in an attempt warm the cache for the new set of diseases and visits.

Further to ensure that the performance benchmarks were accurate and repeatable CARE was complied using the -O0 flag to disable any specific complier optimizations. This was decided as the parallel CARE implementation was run on the CRC Sun Grid Engine (SGE). Given that CARE has no specific system requirements, SGE workers could theoretically be dispatched to any machine with a spare core. As such, optimizations were removed to ensure uniform execution patterns across all worker machines. Finally all extraneous system calls, branching and logging were all removed from the CARE source code. This was done in an effort to stabilize the code from any branch prediction within processors, to ensure as consistent a pipeline as possible between each subsequent benchmark.

Execution Mode

In the course of this evaluation it became clear that CARE was constructed with some specific design considerations. The current implementation CARE effectively performs an all-pairs computation for diseases similarity against every patient within the database. This design has since been designated *Batch Mode*, as this is a process that would likely be executed as an overnight batch job. In contrast this work details a new version of CARE, which performs

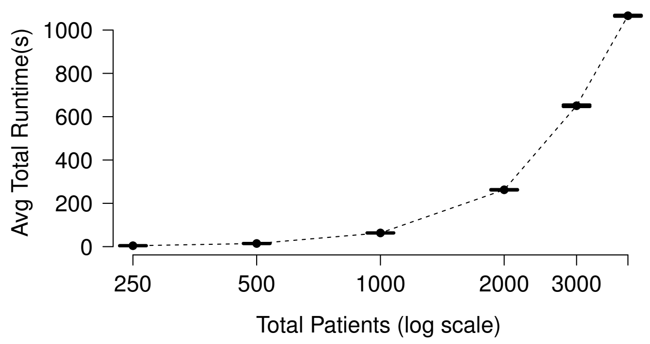

Figure 3: Average Total Runtime of Unmodified CARE

Total	I/O Execution	CPU Computation
5	13.27	24.95
10	8.49	61.31
25	1.73	88.55
50	0.73	90.96
100	0.35	92.64
250	0.15	93.00

Table 1: I/O Vs. CPU Breakdown as a Percent of Total Execution Time

on-demand recommendations for a single patient. This implementation is similarly designed *Individual Mode*, and is detailed below. The following sections will detail the performance evaluations of each recommendation mode.

Batch Mode

As started prior, *Batch Mode* is akin to a daily overnight job, run by a medical provider to preprocess disease rankings for all current patients. These preprocessed results could then be used to provide instant queryable results. Further these results could be used to generate automated reports to alert doctors if a fast acting disease obtains a high probability ranking.

Results and Analysis

The initial performance analysis was performed on an unmodified version of the CARE algorithm, and base runtimes were established over a varying number of patients (Figure 3).

Further as the goal of this work was to demonstrate the implementation of a scalable disease prediction algorithm, which could utilize big data from the largest of provider networks within an industry feasible timeframe, it was important to first determine which computational area would benefit most from optimization. In order to achieve this the major internal CARE functions were divided into the two fundamental computational classes, I/O and CPU bound, and their percent of total computation time was calculated (Table 1). Note the remaining percentage of execution time for smaller patient datasets was consumed by standard system calls and data structure initialization. As you can see these calls are not a bounding feature as they reduce to under six percent with a dataset of only 250 patients, well below even the smallest provider networks.

The primary product of these initial evaluations was the confirmation that the CARE algorithm is highly CPU bound. Further, the program was not only CPU bound, but worse the all-pairs computation for disease rankings resulted in an exponential runtime as a factor of total patients. In fact as you can see from the comparison between Figure 3 and Table 1, the percent of execution time from I/O becomes minimal before the runtime even begins it's rapid increase. This further reinforces the need to focus on CPU bound functionality.

The standard CARE algorithm computes the disease rankings for all patients contained in a four thousand person dataset in just over 17 minutes. At only four thousand patients this dataset is a reasonable analog for even the smallest of medical practices. As stated earlier patient counts can be used as a proxy for CARE accuracy, and a dataset this small is a significant concern. Additionally these base implementation results highlight another major issue. Studies have shown that an entire clinical encounter, including recording a patient's medical history and vitals lasts around just 17 minutes (Mechanic, McAlpine, & Rosenthal, 2001). At a runtime of matching this figure, these results effectively eliminate the usage of *Batch Mode* as a feasible method for intra-day and on-demand disease rankings.

Individual Mode

While the batch method is useful for preprocessing diseases rankings, there are two significant drawbacks to utilizing this method. First *Batch Mode* does not take into account new patients, operating under the assumption that the patient is already contained within the provider's database. This implies that new patients will not be able to achieve their disease rankings until the following day. This delay then limits the ability of CARE to be utilized in many healthcare settings, particularly those which operate in a time-critical manner such as intensive care units.

The second drawback focuses on the accuracy of the CARE recommendations. Healthcare providers are constantly updating patients medical records with new and potentially vital information. By choosing to utilize CARE recommendations based on the prior days records you are potentially creating a situation where an individual's recommendations are incomplete or incorrect.

In order to address these concerns this work has implemented an extension of CARE, which calculates disease rankings for a single patient. This *Individual Mode* of CARE utilizes the current state of the database when the rankings are requested, and can be run on any patient, prior or new, as long as their diagnosis history is provided.

Results and Analysis

After modifying the CARE algorithm to perform on-demand computation, the logical first step was to compare the execution time to that of *Batch Mode* (Figure 4). By removing the all-pairs comparison, CARE is able to handle substantially larger datasets. It was able to process just under 400 thousand patients in the equivalent timespan of the four thousand patient dataset, at 17 minutes. Again, as the number of patients in the provider database can be used as a proxy for ranking the *Individual Mode* would provide disease rankings at a significantly higher confidence level. It is important to note that the *Batch Mode* would create disease rankings for 3,999 more patients than would the *Individual Mode* implementation. However when diagnosing a patient, accuracy for that individual may be more important than would be the diagnosis for multiple potentially unrelated patients at once. In fact, if *Individual Mode* CARE were initialized at the start of a patients primary care visit, their diseases rankings could be calculated against a dataset of up to 300 thousand patients. After execution there would be almost one third of the total encounter time, 5 of the total 17 minutes, remaining for medical staff to analyze and

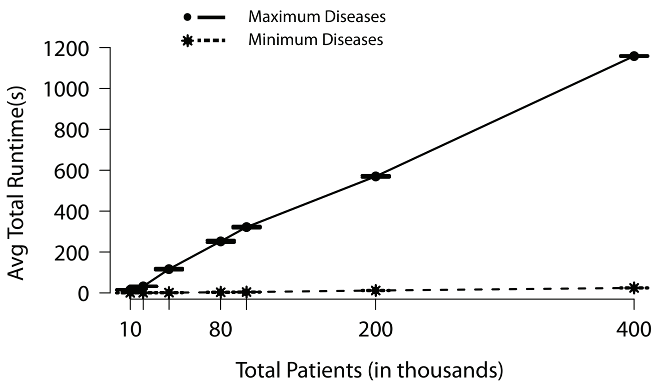

Figure 4: Average Total Runtime of Unmodified CARE

discuss the results. Further, the *Individual Mode* implementation directly addresses the issue of utilizing CARE in a time-sensitive environment. In these scenarios CARE could be initialized at the time of the patients admission. CARE would then be able to produce a preliminary set diseases risk within the early minutes of treatment, a critical period of time for high-risk patients.

Analyzing the outcome from *Individual Mode* yields an interesting result as related to the general CARE algorithm. The results show that execution time is not independently correlated with the size of the patient dataset used. Instead, the execution time is also directly correlated to the number of diseases per patient. This comes as a result of the preprocessing step taken by CARE, noted earlier, where only those patients that share a common disease are utilized for the collaborative filtering algorithm. The results of this implementation decision are shown by Figure 4 where the patient with the maximum number of diseases has a significantly higher runtime, as does the patient with the minimum number of diseases when run on the same dataset.

Parallel CARE Implementation

While the work of creating a single user implementation of CARE shows that it is possible to improve the execution time, the computational requirements of collaborative filtering limit the maximal performance gains that can be achieved by the current CARE architecture. After evaluating both the current implementation and improved single patient CARE algorithms, it became clear that the fundamental CARE architecture would need to be changed to obtain any further performance improvements. In order to understand how the CARE algorithm could benefit from optimizations, such as parallel execution, it was first important to understand where internally did CARE stall.

This paper has previously shown that CARE is CPU bound, but now going further it is important to define where this occurs. In order to answer this the CPU bound components were broken down into the individual functions as a percent of total runtime (Table 2). It is clear that the CPU bounding is dominated by one function, Best Match.

Total	**Best Match**	**Load Patient**	**Load Disease**
5	65.28	15.86	18.86
10	87.84	6.69	5.47
25	98.08	1.36	0.55
50	99.20	0.64	0.15
100	99.62	0.33	0.04
250	99.88	0.15	0.01

Table 2: Function Breakdown of CARE Execution

	Percent of Time Per Function		
Total	**Vector Similarity**	**Merge Visits**	**System Calls**
25	41.78	10.32	47.90
50	41.69	10.05	48.26
100	42.24	8.22	49.54
250	42.13	7.52	50.36
Average	*42.51*	*7.85*	*49.64*
SD	*1.30*	*2.19*	*1.35*

Table 3: Component Breakdown of Best Match Function

Taking this further the Best Match function was broken down to analyze exactly what was causing the bottleneck (Table 3). Note that due to the short execution time and sparse nature of the disease network, analyzing datasets containing less than 10 patients creates highly variable and non-convergent results. Thus all datasets below 25 patients were excluded from this evaluation.

Looking at the table it is interesting to see that the percent of time spent in each component of the function remains unchanged as a product of number of patients in the dataset. This result lends itself well to the potential benefits of parallelization as it shows that even though the algorithm has an exponential runtime the amount of time spent in each function is stable.

Distributed CARE

Taking a deeper look at the sequential *Batch Mode* implementation it becomes possible to utilize the exponential increase in runtime to our advantage. While many see only the increased runtimes, the real value comes in the other direction. The exponential decay of runtime as a function of decreasing dataset sizes. By breaking down the Best Match function on smaller subsets of the total patient list it becomes possible to achieve significant performance benefits. These benefits can manifest as a minimization of computation time, or as the increase in size of the possible training datasets within the encounter timeframe.

Distributed Architecture

The distributed model for the CARE algorithm is a result of the existing CARE framework, where each patient's disease ranking is calculated independently. While all patient's medical histories are used as training for the model, creating the disease network for a patient is an independent event. As a result, the distributed version of CARE partitions the set of all patients into equal size subsets. These subsets are then distributed to up to 50 individual worker nodes

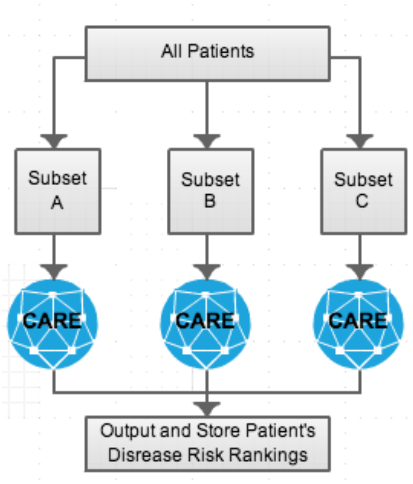

Figure 5: Distributed CARE Architecture

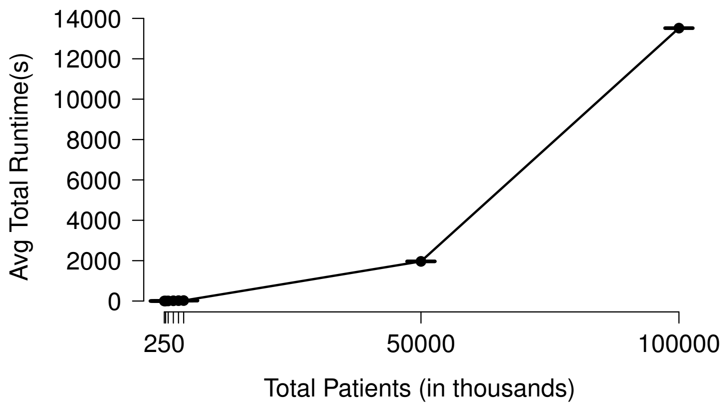

Figure 6: Average Total Runtime of Distributed CARE

using the WorkQueue algorithm created by the Cooperative Computing Lab (CCL) at Notre Dame (Bui, Rajan, Abdul-Wahid, Izaguirre, & Thain, 2011). Under this framework each worker node receives a copy of the CARE algorithm as well as information about the subset it is tasked to compute. After completion only the disease risk rankings need to be returned and stored in a database. This process is shown in Figure 5.

Evaluation Metric Extension

For the analysis of *Distributed CARE* model this paper will utilize an additional performance metrics designated *Derived Time*. Due to the time-shared nature of tasks on the CRC machines an accurate time for large-scale tasks cannot be measured. However, accurate execution times can be measured for each task. *Derived Time* then is an extension of standard time and is calculated assuming each task will begin as the task in front of it ends. Each block of tasks will execute in parallel and the total *Derived Time* will be a result of the average I/O overhead added to the average computation time for a task multiplied by the number of worker cycles that must be completed for the entire dataset. This method is utilized for all datasets over 10^4 patients.

Analysis and Results

By utilizing the exponential decay aspect of the *Batch Mode* implementation it is clear that the execution time of CARE can be significantly improved. Additionally this improvement

Task Count	Avg IO Overhead (s)	Avg CPU Time (s)	Num Worker	Derived Execution
10000	0.017	134.358	200	7.465
25000	0.021	43.811	500	6.088
50000	0.098	19.412	1000	5.419
100000	0.101	6.658	2000	3.755

Table 4: Avg Total Runtime per Task of Distributed CARE

Task Count	1000	10000	25000	50000
Patients Per Task	50	5	2	1
No Cache	43618.23	43609.78	46276.27	57924.65
Cache First Access	58597.37	50521.59	57008.15	54396.94
Cache Subsequent Access	29.92	32.11	29.89	31.43

Table 5: Avg IO Overhead for 50000 Patient Dataset in μs

does not suffer from the trade off of reduced numbers of patient disease risk calculations, as does the current *Individual Mode* improvement. As you can see Figure 6 demonstrates the significant improvements of *Distributed CARE.* It is important to note that *Distributed CARE* still exhibits an exponential execution time as a function of total patients in the dataset. This is expected, as *Distributed CARE* makes no effort to change the internal framework of CARE. The goal was to process significantly larger patient datasets in a practical timeframe, both of which *Distributed CARE achieves.*

While another exponential program may not seem like much of an improvement the following shows that it is not only an improvement, but also necessary for widespread adoption of CARE. Using the standard *Batch Mode* execution times above we were able to fit the exponential function $60.3882e^{0.00073x}$ to model the projected runtime of CARE. At this estimation the computation for 100000 patients, a number easily achievable by major practices, would take $7.04e^{25}$ years. This figure effectively eliminates CARE's usage on a practice wide basis, forcing it to be used only in specialized cases. This figure effectively eliminates CARE's usage on a practice wide basis, forcing it to be used only in specialized cases. As you can see through Table 4 increasing the task count, and thus decreasing the patients per worker task dramatically decreases the computation time down to just under 4 hours. With runtimes at just 1/6 of a day, significantly more patients could benefit from the usage of CARE.

However in order to fully utilize the *Distributed CARE* algorithm, you must look further and consider optimizations within each run. This work decided to utilize as many workers as allowed, in this case 50, under the assumption that the any medical practice using CARE will have had over 50 patients pass through their system, allowing for at least a minimum of one patient per worker. As such, all optimizations will come through variations in the distributed subset size. When analyzing a distributed system for performance optimizations it becomes important to look not only at the execution time for a task, but also at the time needed for communication overhead between the master and worker nodes. Again, as you can see from Table 4 the majority of a task's execution time is spent in CPU bound calculations. It should be noted that while computation time exponentially decreases with the task count, I/O overhead increases. This shows that in theory for large datasets if the task count is too high I/O overhead may actually increase the total execution time of *Distributed CARE.* Unfortunately it was not possible to reach that limit even when utilizing all patients with over 5 diseases. However, it should be considered that medical practices with datasets containing millions of patients might be achieving sub-optimal performance by trying to partition the dataset too tightly.

As an additional note, distributed systems today are becoming significantly more common, and as such many controller systems are implementing tools to improve performance of these distributed systems. For example WorkQueue algorithm has the ability to cache input files for distribution to worker nodes. As you can see in Table 5 enabling caching effectively eliminates all overhead from the master, reducing the need for dataset specific optimizations. While there may be diminishing returns on reduced execution time, a practice may utilize as many computational resources as available and be confident that they are operating at optimal performance levels.

Summary and Recommendations

To recap, this paper has shown the performance limitations of the current CARE algorithm. While some claim that an overnight batch execution is sufficient, as it can process a large patient dataset with a high degree of accuracy, this method is non-viable for medical usage. Big data providers such as Facebook do utilize similar batch events to help with data processing, but the information generated does not have the safety-critical nature of healthcare data. In the event that a disease is incorrectly recorded, a patient may have to wait up to 24 hours to receive updated disease risks. This turnaround time may be unacceptable, especially for time critical units.

In order to solve the issue of computation time this paper has outlined two distinct methods. First a single patient version of CARE, which can be utilized to perform disease risk rankings on demand with a fairly high degree of accuracy. This method is intended to be utilized in the case above where updated rankings must be regenerated due to error, or for a new patient who was not present in the database when the last batch job was run. The second method is a distributed computation of the CARE algorithm. This implementation can be used to generate on-demand rankings for a single patient with a high degree of accuracy, or executed as a nightly batch job on significantly larger patient sets for large practices or hospitals.

Future Work

As stated earlier the current CARE implementation has been licensed to an external entity, and as such we cannot comment on any current development efforts. However, in the academic setting we are currently investigating the possibility of augmenting the CARE algorithm with additional EMR data. This data aims to move beyond ICD-9 disease comparisons and include features such as patient medications and procedures. Although the additional features in the data may serve to improve performance, they will further exacerbate the issue of computational complexity. Each additional feature will be included as a part of the all pairs comparison performed by the collaborative filtering, compounding the exponential runtime, and further reinforcing the need for a distributed computing framework such as that introduced in this work.

Acknowledgements

We acknowledge financial support from grant FA9550-12-1-0405 from the U.S. Air Force Office of Scientific Research (AFOSR) and the Defense Advanced Research Projects Agency (DARPA).

References

AbuKhousa, E., & Campbell, P. (2012). Predictive data mining to support clinical decisions: An overview of heart disease prediction systems. In *Innovations in information technology (iit), 2012 international conference on* (pp. 267–272).

Austin, P. C., Tu, J. V., Ho, J. E., Levy, D., & Lee, D. S. (2013). Using methods from the data- mining and machine-learning literature for disease classification and prediction: a case study examining classification of heart failure subtypes. *Journal of clinical epidemiology* .

Bellazzi, R., & Zupan, B. (2008). Predictive data mining in clinical medicine: current issues and guidelines. *international journal of medical informatics*, *77* (2), 81–97.

Berkovsky, S., Eytani, Y., Kuflik, T., & Ricci, F. (2007). Enhancing privacy and preserving accuracy of a distributed collaborative filtering. In *Proceedings of the 2007 acm conference on recommender systems* (pp. 9–16).

Bui, P., Rajan, D., Abdul-Wahid, B., Izaguirre, J., & Thain, D. (2011). Work queue+ python: A framework for scalable scientific ensemble applications. In *Workshop on python for high performance and scientific computing at sc11.*

Chawla, N. V., & Davis, D. A. (2013). Bringing big data to personalized healthcare: A patient-centered framework. *Journal of general internal medicine*, *28* (3), 660–665.

Davis, D. A., Chawla, N. V., Christakis, N. A., & Barab´asi, A.-L. (2010). Time to care: a col- laborative engine for practical disease prediction. *Data Mining and Knowledge Discovery* , *20* (3), 388–415.

Dean, J., & Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters.
Communications of the ACM , *51* (1), 107–113.

Etzioni, R., Urban, N., Ramsey, S., McIntosh, M., Schwartz, S., Reid, B., . . . Hartwell, L. (2003). The case for early detection. *Nature Reviews Cancer* , *3* (4), 243–252.

Goil, S., & Choudhary, A. (1997). High performance olap and data mining on parallel comput- ers. *Data Mining and Knowledge Discovery* , *1* (4), 391–417.

Jensen, P. B., Jensen, L. J., & Brunak, S. (2012). Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, *13* (6), 395–405.

Kickbusch, I., & Payne, L. (2003). Twenty-first century health promotion: the public health revolution meets the wellness revolution. *Health promotion international* , *18* (4), 275–278. Lard, L. R., Visser, H., Speyer, I., vander Horst-Bruinsma, I. E., Zwinderman, A. H., Breedveld,
F. C., & Hazes, J. M. (2001). Early versus delayed treatment in patients with recent-
onset rheumatoid arthritis: comparison of two cohorts who received different treatment strategies. *The American journal of medicine*, *111* (6), 446–451.

Lathia, N., Hailes, S., & Capra, L. (2007). Private distributed collaborative filtering using esti- mated concordance measures. In *Proceedings of the 2007 acm conference on recommender systems* (pp. 1–8).

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE* , *7* (1), 76–80.

Mayer-Sch¨onberger, V., & Cukier, K. (2013). *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt.

McCormick, T. H., Rudin, C., & Madigan, D. (2012). Bayesian hierarchical rule modeling for predicting medical conditions. *The Annals of Applied Statistics*, *6* (2), 652–668.

Mechanic, D., McAlpine, D. D., & Rosenthal, M. (2001). Are patients' office visits with physicians getting shorter? *New England Journal of Medicine*, *344* (3), 198–204.

Ozminkowski, R. J., Ling, D., Goetzel, R. Z., Bruno, J. A., Rutter, K. R., Isaac, F., & Wang,
S. (2002). Long-term impact of johnson & johnson's health & wellness program on health care utilization and expenditures. *Journal of Occupational and Environmental Medicine*, *44* (1), 21–29.

Slee, V. N. (1978). The international classification of diseases: ninth revision (icd-9). *Annals of internal medicine*, *88* (3), 424–426.

Steinhaeuser, K., & Chawla, N. V. (2009). A network-based approach to understanding and predicting diseases. In *Social computing and behavioral modeling* (pp. 1–8). Springer.

Stonebraker, M., Abadi, D., DeWitt, D. J., Madden, S., Paulson, E., Pavlo, A., & Rasin, A. (2010). Mapreduce and parallel dbmss: friends or foes? *Communications of the ACM* , *53* (1), 64–71.

Tassy, O., & Pourqui´e, O. (2013). Manteia, a predictive data mining system for vertebrate genes and its applications to human genetic diseases. *Nucleic acids research*, gkt807.

Wilkinson, T. M., Donaldson, G. C., Hurst, J. R., Seemungal, T. A., & Wedzicha, J. A. (2004). Early therapy improves outcomes of exacerbations of chronic obstructive pulmonary dis- ease. *American journal of respiratory and critical care medicine*, *169* (12), 1298–1303.