

R. Alonso, J. A. Bloom, H. Li, and C. Basu, “An Adaptive Nearest Neighbor Search for a Parts Acquisition ePortal”, in the Proceedings of the ninth ACM International Conference on Knowledge Discovery and Data Mining, SIGKDD'03, Washington, August 2003.

ACM COPYRIGHT NOTICE. [Copyright © 2003 by the Association for Computing Machinery, Inc.](#) Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept., ACM, Inc., fax +1 (212) 869-0481, or permissions@acm.org.

© ACM, 2003. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in the Proceedings of the ACM SIGKDD, {VOL#, ISS#, (DATE)} <http://doi.acm.org/10.1145/nnnnnn.nnnnn>

An Adaptive Nearest Neighbor Search for a Parts Acquisition ePortal

Rafael Alonso

Jeffrey A. Bloom

Hua Li

Chumki Basu

Sarnoff Corporation
201 Washington Road
Princeton, NJ 08543
609-734-2172

{ralonso, j bloom, hli, cbasu}@sarnoff.com

ABSTRACT

One of the major hurdles in maintaining long-lived electronic systems is that electronic parts become obsolete, no longer available from the original suppliers. When this occurs, an engineer is tasked with resolving the problem by finding a replacement that is "as similar as possible" to the original part. The current approach involves a laborious manual search through several electronic portals and data books. The search is difficult because potential replacements may differ from the original and from each other by one or more parameters. Worse still, the cumbersome nature of this process may cause the engineers to miss appropriate solutions amid the many thousands of parts listed in industry catalogs.

In this paper, we address this problem by introducing the notion of a parametric "distance" between electronic components. We use this distance to search a large parts data set and recommend likely replacements. Recommendations are based on an adaptive nearest-neighbor search through the parametric data set. For each user, we learn how to scale the axes of the feature space in which the nearest neighbors are sought. This allows the system to learn each user's judgment of the phrase "as similar as possible."

Categories and Subject Descriptors

H.2.8 [Database Applications]: Data Mining, H.3.3 [Information Search and Retrieval]: Search Process, Clustering, Information Filtering, I.2.6 [Learning]: Knowledge Acquisition.

General Terms

Algorithms, Management, Measurement, Performance, Design, Experimentation.

Keywords

Adaptive Search, k -Nearest Neighbor classification, User Profiling, Query by Example.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGKDD '03, August 24-27, 2003, Washington, DC, USA.

Copyright 2003 ACM 1-58113-737-0/03/0008...\$5.00.

1. INTRODUCTION

The motivating application for the work described in this paper is the search for "similar" electronic parts to replace electronic parts that are no longer available from their original sources. In this context, we use a nearest neighbor technique to identify parts that are parametrically most similar. This alone is novel in the state-of-practice. We extend this further by allowing the feature space to dynamically adapt to each individual user. We describe a new method for this adaptation.

1.1 Obsolete Part Problem

In the current world of high-tech electronic gadgets, the lifecycle of a typical electronic part can run as short as 18 months. This poses a significant challenge to the U.S. government and its technology suppliers as products for the government market are many years in development and testing and, once deployed, are typically fielded for 10 to 20 years. During that time, additional devices must often be built and fielded devices must often be repaired.

Suppliers, repair depots, and technicians in the field need access to a supply of components found in a military system. When a component is no longer available through the original source, then it is deemed "obsolete" and a process is put in place for resolution. By the year 2000, the obsolescence rate had risen to over 66,000 electronic components per year [1].

1.2 Current State of the Art in Obsolete Parts Resolution

For the purposes of this paper, we will denote as an "Application Engineer", or simply AE, the person responsible for resolving the obsolete parts problem. It is important to note that there are many AEs addressing the resolution of the same obsolete part. These AEs are distributed among the various government product suppliers and supply depots as well as field technicians. Further, as each AE is potentially addressing the problem in a different context (a given obsolete part is a component in many different electronic devices), there will be many different resolutions. The part that one AE selects for one application, may be an inappropriate resolution for another.

In order to resolve a particular obsolete part problem, the AE must first gather all of the parametric information describing the obsolete part and then search for possible replacement parts with which the system will be able to function as designed. Currently, there is no single on-line source for this data. There are web sites maintained by individual manufacturers, aftermarket suppliers, parts distributors, and others. At each of these sites, the AE must

submit a query and evaluate and compare the results. Few of these sites allow searching on parametric data and even fewer will return close matches.

In response to this problem, a group of companies, including Sarnoff Corporation, Rockwell Collins, Altarum, and Aquilent, have been working with the Defense Sustainment Consortium, funded by the Defense Logistics Agency, to create an "ePortal for Parts Acquisition" which will be a single point of entry for AEs. This ePortal will allow an AE to search a number of heterogeneous, distributed parts databases with a single query and receive the results in a unified format for easy comparison.

As is the case with all large data sets, searching for the right information becomes a challenge. This is especially true when the search is for a close neighbor and when different users have different notions of closeness. This paper describes the development of a personalized tool that will reduce the cost of finding a resolution and will improve the chances of finding the best replacement.

2. SEARCH BASED ON ADAPTIVE USER PROFILE

Our approach to dealing with this particular information overload problem is to define a distance measure between the obsolete part, called the target, and each of the other parts in a large data set. This distance measure is guided by a user profile. Potential replacement parts are sorted by distance and presented to the user. Feedback, obtained when the user selects an acceptable replacement part, is used to update the user profile.

A block diagram of our application algorithm is shown in Figure 1. Block 1 obtains a parametric description of the target. Block 2 provides a high-level filtering of the entire catalog of data (all heterogeneous parts databases taken together) to remove parts that are clearly unacceptable (can't replace a memory chip with a voltage regulator). Blocks 3-7 represent calculation and presentation of the distance measure, feedback, and profile adaptation.

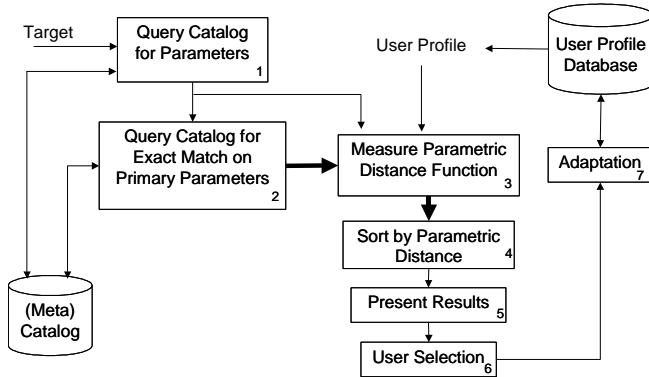


Figure 1. Adaptive Search Algorithm

2.1 Parametric Distance

The distance between any two parts is a weighted combination of parameter difference functions across all parameters. The weights represent the user profile. Each user will have a different set of weights and these weights will adapt, over time, in response to feedback obtained from the user. The difference functions are

determined by domain experts¹ to appropriately represent differences in each parameter (e.g., What is the "cost" of replacing an ceramic package with a plastic package? Is this cost commutative?)

The problem of defining distance measures for non-numeric features has been studied in other contexts as well ([3], [4]). Stanfill and Waltz [4] used a "value difference metric", a non-Euclidean distance metric for working with symbolic features, which says two values are similar if they occur with the same relative frequency for all classifications.

For the i^{th} parameter, we define a parameter difference measure, d_i , between parts P_a and P_b ,

$$d_i = d_i(P_a(i), P_b(i)).$$

Each parameter difference is normalized such that differences measured along a number of different parameters can be meaningfully compared and combined.

One example of parameter difference function for a numeric-valued parameter is the absolute difference between the parameter values. For non-numeric-valued parameters, the difference functions are more difficult to specify and require the assistance from the domain experts.

The parametric distance, $D(a,b)$, between two parts is then some weighted combination of these parameter difference measures. In this paper we use the weighted summation in which case the part distance is a linear combination of parameter differences,

$$D(a,b) = \sum_i \alpha_i d_i. \quad (1)$$

2.2 User Profile

The need to maintain explicit user profiles is especially relevant to systems that recommend items to users, from Web pages to music [5], [6], [7], [8]. Traditionally, these profiles are based on simple vector representations of data.

Our profile, assuming n physical and functional attributes or parameters, is represented as a set of weights, i.e. an n -dimensional vector $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_n]$. These weights are real, non-negative numbers constrained to be less than, or equal to some number m :

$$0.0 \leq \alpha_i \leq m.$$

Each element of the weight vector can be interpreted as indicating the relative importance of the corresponding parameter in determining the similarity of two parts. Parameters with larger weights are considered more important in this determination.

2.3 Profile Adaptation: Bubble Up Algorithm

Profile adaptation is based on user feedback. User feedback based on user selection, sometimes referred to as *clickthrough* data, have been analyzed in the KDD community to learn ranking functions [9]. As in that work, we also rely on the idea that clickthrough data gives us partial information about the user's judgments regarding relevance.

We obtain feedback when the user selects one of the parts on the list as an appropriate replacement for the target and the selected

¹ The domain expert may be an Application Engineer or one of the engineers who designed the assembly.

part is *not* the first item presented. In other words, we obtain feedback when the selected item was not the part that had the minimum distance to the target part. An example of this feedback is shown below where $D(t,a) < D(t,b) < D(t,c) < D(t,d)$ and part P_d is selected by the user.

Table 1. User Feedback

Recommended Replacement For Target Part, P_t	Parametric Distance	Selected
P_a	$D(t,a)$	No
P_b	$D(t,b)$	No
P_c	$D(t,c)$	No
P_d	$D(t,d)$	Yes

This feedback is used to modify the values of the parameter weights such that, if the same query were run again, part P_d would be found earlier in the part ranking.

The selection of part P_d and rejection of parts P_a , P_b , and P_c give us the following three elements of feedback.

1. $D(t,a) < D(t,d)$ but $D(t,d)$ should have been smaller
2. $D(t,b) < D(t,d)$ but $D(t,d)$ should have been smaller
3. $D(t,c) < D(t,d)$ but $D(t,d)$ should have been smaller

We cannot reach any conclusions about the relationships between the three unselected parts (e.g., between P_a and P_b), nor can we form any conclusions about other parts further down the list below P_d , as they may not have been examined by the user.

The goal of the adaptation algorithm is to modify the parameter weights in response to the three statements above. We can imagine addressing all three simultaneously, but the algorithm presented here addresses them individually starting with the part one position higher than the selected part on the sorted list. Once the weights have been changed, we recalculate the distances and iterate. As the iterations proceed, the selected part “bubbles up” through the list. Thus, we fondly call this algorithm the *Bubble Up* algorithm².

The stopping criterion for the iterations can include the following: stop when the change in the weights has reached a maximum, stop when the selected part appears higher than some threshold ranking in the sorted list, and stop when each item from the feedback set has been examined at least once.

The Bubble Up algorithm can be simply stated in the pseudo code statements below where $\{P_f\}$ is the set of parts involved in the feedback, that is, the set of parts ranked higher than that selected.

```

 $P_s = \text{selected part}$ 
while ( not stop )
  set  $P_f | ( \text{rank}( P_f ) = \text{rank}( P_s ) - 1 )$ 
   $\alpha = \text{modify}( \alpha, P_s, P_f, P_t )$ 
  recalculate Parametric Distance and sort
end while

```

The function called *modify*(\cdot) in this pseudo code is designed to minimize the change to the weights necessary to effect the desired

² We have studied several other profile adaptation algorithms in our work that are not described here. Refer to [2] for the details.

change in rank order. In the first iteration of the example, the algorithm makes the minimum change in the weights such that $D'(t,d) < D'(t,c)$. Here we've introduced the distance $D'(\cdot)$ to indicate the part distance between the two parts after we modify the weights.

By minimum change in the weights we mean that we will minimize the maximum change across all parameters. It can be shown that one way to do this is to change all the weights by the same amount, δ . If we specify the amount by which $D'(t,d) < D'(t,c)$, we essentially require that $D'(t,d) - D'(t,c) = -k$ where k is a positive number. Since the parametric distances are linear combinations of parameter distances, we can derive the required value of δ as follows.

From Equation 1 we can write

$$D(t,d) - D(t,c) = \sum_i \alpha_i g_i(d,c),$$

where $g_i(d,c) = g_i(P_d, P_c) = \bar{d}_i(P_t, P_d) - \bar{d}_i(P_t, P_c)$.

The goal of the first iteration of the bubble up algorithm is to determine the value of δ such that

$$\begin{aligned} D'(t,d) - D'(t,c) &= -k \\ \sum_i (\alpha_i \pm \delta) g_i(d,c) &= -k \end{aligned} \quad (2)$$

where we've incorporated the restriction that all weights change by the same amount, δ , only the sign of that change can vary by parameter.

Note that with $\delta=0$, as is the case prior to modification, the sum on the left-hand side of Equation 2 evaluates to a positive number while the target on the right, $-k$, is a negative number. In order to make the sum more negative, we wish to add δ to those α_i corresponding to negative values of g_i and to subtract δ from those α_i corresponding to positive values of g_i . This can be formalized by representing g_i as

$$g_i(d,c) = \text{sign}(g_i(d,c)) \times |g_i(d,c)|$$

Then the value of δ can be determined as follows

$$\begin{aligned} \sum_i (\alpha_i - \text{sign}(g_i(d,c))\delta) g_i(d,c) &= -k \\ \delta &= \frac{k + \sum_i \alpha_i (d_i(t,d) - d_i(t,c))}{\sum_i |d_i(t,d) - d_i(t,c)|} \end{aligned} \quad (3)$$

Equation 3 provides a closed form solution for the smallest constant value of δ that can be added to or subtracted from each of the α_i such that the distance between part P_d and the target part, part P_t , will be smaller (by k) than the distance between part P_c and part P_t .

In summary, the new set of α_i necessary to ensure that $D(t,c) - D(t,d) = k$ is simply

$$\alpha_i^n = \alpha_i^{n-1} - (\text{sign}(d_i^{n-1}(t,d) - d_i^{n-1}(t,c)) \times \delta) \quad (4)$$

where the superscripts on α and δ indicate the iteration number and δ is determined by Equation 3. Rewritten with the notation of the pseudo code example we have

```

modify(  $\alpha_i$ ,  $P_s$ ,  $P_t$ ,  $P_t$  )
{

$$\delta = \frac{k + \sum_i \alpha_i (d_i(t,s) - d_i(t,f))}{\sum_i |d_i(t,s) - d_i(t,f)|}$$

return  $\alpha_i - (\text{sign}(d_i(t,s) - d_i(t,f)) \times \delta)$ 
}

```

3. EXPERIMENTAL RESULTS

We have tested various algorithms with simulated users and both synthetic data and real data. This testing, detailed in [2], allowed for some fine tuning of the algorithms. Once we were satisfied with the system performance on simulated data, we ran experiments with real electronic parts data and parts selection data provided by actual Application Engineers. In this section we describe these real-user tests.

The parts data consisted of 4842 Static RAM (SRAM) parts, each described by 25 parameters. This data, provided by Information Handling Services (IHS), is a snapshot of the data available through their CAPS Expert™ product.

Given the time consuming nature of finding replacement parts, we limited our testing to four searches. Four parts from the data set were artificially marked as obsolete. These were used by each of three users (all professional AEs) who were asked to find acceptable replacements. The AEs accessed the data through the CAPS Expert™ interface. Independently, we downloaded the data set to our local system for analysis of our algorithms.

3.1 Method

We evaluate our algorithms in three stages. First, in Section 3.3, we use a default profile, initialized so that all parameters are weighted equally. Given each of the target parts, we use this default profile to rank the entire database of parts and then identify where, on that list, the selections of the users lie. This will give an indication of the utility of a static parametric distance function. We then allow the profile to adapt in response to user selections. In many cases, each user has identified two acceptable solutions for each target part. It is not uncommon, in fact, for an engineer searching for solutions to identify multiple options. In the Section 3.4 below, we provide our system one user-specified solution, allow the system to adapt to that feedback, and then observe the change in ranking of the second user-specified solution using the adapted profile. Finally, in Section 3.5, we allow the system to adapt to two target / selection pairs and then observe the change in ranking of the parts selected to replace the third target.

3.2 User Data

Rather than list the manufacturers part names, we will reference each part by the index number from our data set.

The four target parts have indices, 4823, 4827, 4838, and 4839.

For the first of these four parts we have data from three users regarding acceptable replacements in the data set. For the other three parts, we have data from two users. These data are shown in Table 2. As can be seen, users identified two acceptable replacement parts for most targets.

Table 2. User Data
(N/S=no selection)

Target	User 1	User 2	User 3
4823	4824	4826	4825
4823	4835	4837	4836
4838	4830	4829	N/S
4838	4831	4830	N/S
4827	4828	4829	N/S
4827	N/S	4830	N/S
4839	4832	4454	N/S
4839	4833	4834	N/S

3.3 Parametric Distance with Default Profile

In this experiment, we initialize the profile to its default value. Each element of the profile is set to 5.0 and bounded to a range of 0.0 to 10.0. Given this default profile, we search for parts similar to part 4823. The 34 parts closest to the target are shown in Figure 2. The plot shows the 34 parts arranged vertically in 8 groups. All parts in each group are the same distance from the target and that distance is indicated below the x-axis. The 5 parts in the first group (shown on the left) are a distance of 0.0 from the target. The 8 parts in the second group have a distance of 0.29 to the target. We refer to each grouping as a cluster and note the number of parts in each cluster. Note that since all of the parts in any one cluster are the same distance from the target, the order in which they are listed in Figure 2 is arbitrary.

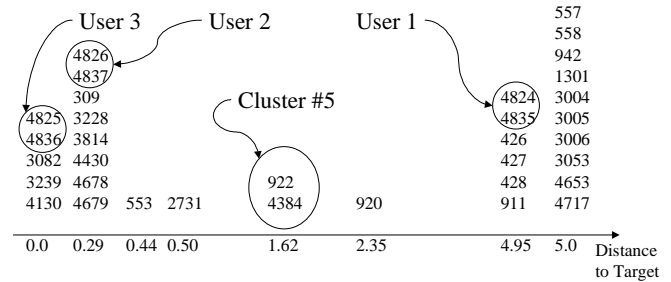


Figure 2. Parts Similar to 4823 Using a Default Profile

In order to assess the “ranking” of a part, we need to consider three values: the rank of the cluster in which it lies, the size of the cluster in which it lies, and the total number of parts in all clusters closer to the target. For example, the distance between part 4384 and the target, 4823, is 1.62. There are two parts that both have this distance and they are considered a cluster of size two. This is the fifth cluster and there are 15 parts closer to the target.

Highlighted in this figure are the parts that were selected by the three users. Significantly, the default profile ranks all the user-identified parts in one of the top 7 clusters and in the top 24 of all 4842 parts. This demonstrates the utility of using the parametric distance function for this application.

Mixed in among the parts selected by the three users are many other parts that are the same distance or closer to the target part. These parts represent potential replacement parts that were not found by the users. For example, the three parts not selected in the first cluster appear to be drop-in replacements (with some

parameters possibly upgraded) and these were not found by any of the users.

It is interesting that, for each user, the distance values of the two selected parts are the same, yet those value differ from one user to the next. This suggests that there is some consistency in each user’s selection process and that the parametric distance captures this consistency.

Similar results are found for two more SRAM parts, the 4838 and the 4839. These results are not presented here. For details, please see [2].

There is a fourth SRAM part for which we have user data: part 4827. User 1 identified one replacement for this part and User 2 identified two as shown in Table 2.

In this case, the results of our algorithm differed significantly from the selections of the two users. While these users identified parts 4828, 4829, and 4839 as the best replacement parts they could find, our algorithm with the default profile ranked these parts 3875, 593, and 499 respectively. In other words, there are 3874 parts that are closer to the target than the part that User 1 selected (4828).

Why did the default profile perform so poorly? To examine this, we look at the target and the part selected by User 1, side by side. The parameters in which these two parts differ are shown in Table 3.

We see that the target part was built to military specifications or mil spec (as indicated by the MHR and RL parameters) with significantly less current draw (MC and MII) and lower min. supply voltage (MSV). A casual observer (as the default profiles represents) might not expect 4828 to be an appropriate replacement. In fact, the difference function for military specifications assigns a high penalty to replacing a mil spec part with a non-mil spec part. Thus, the algorithm correctly ranked it low.

However, after receiving just one iteration of user feedback indicating that this user is willing to replace mil spec parts with non-mil spec parts, part 4828 moves up to the 3rd cluster with only 3 parts closer to the target.

Table 3. Parameter Comparison between Parts 4827 & 4828

	4827 (Target)	4828
AT	120n	35n
MHR	Y	N
RL	38510B/34H/35Q	Comm
MC	7m	120m
MSV	2	4.5
MII	25u	15m

3.4 Intra-Target Adaptation

For most target parts, each user has identified two potential replacements. In this test, we select one of these replacements, allow the profile to adapt, and then measure the improvement in ranking of the second part compared to it’s position using the default profile. The results are shown in Table 4.

The default profile does quite well, always ranking the selected parts one of the first 7 clusters and always with fewer than 20 parts ranked higher. When the selected part is ranked in the top cluster, it provides no feedback for adaptation. This is the case in

6 of the 14 cases. Of the remaining 8 cases, there are 4 in which the cluster ranking and number of parts above did not change, but in each of those, the part was initially ranked in the 2nd or 3rd cluster. In four cases, the ranking of the second part was significantly improved after feedback from the first (cluster 7 to cluster 1 or cluster 6 to cluster 1).

Table 4. Intra-Target Adaptation Results

	Target	Selected	Test Part	Initial Cluster (# in Cluster)	Parts Above	Adapted Cluster (# in Cluster)	Parts Above
User 1	4823	4824	4835	7 (6)	18	1 (23)	0
		4835	4824	7 (6)	18	1 (23)	0
	4838	4830	4831	2 (9)	12	2 (9)	12
		4831	4830	1 (12)	0	1 (21)	0
	4839	4832	4833	3 (1)	3	3 (1)	3
4833		4832	1 (2)	0	1 (3)	0	
User 2	4823	4826	4837	2 (8)	5	2 (8)	5
		4837	4826	2 (8)	5	2 (8)	5
	4838	4829	4830	1 (12)	0	1 (12)	0
		4830	4829	1 (12)	0	1 (12)	0
	4839	4834	4454	6 (3)	7	1 (5)	0
4454		4834	6 (3)	7	1 (5)	0	
User 3	4823	4825	4836	1 (5)	0	1 (5)	0
		4836	4825	1 (5)	0	1 (5)	0

3.5 Inter-Target Adaptation

Now we investigate how well adaptation based on one or two target/selected pairs can improve the ranking of the third (for Users 1 & 2). For the data collected, excluding the mil spec part, there are 48 possible combinations of training pairs and test pairs. Table 5 below, presents a randomly selected sampling of those combinations.

Table 5. Inter-Target Adaptation Results

	Training Data	Target	Test Part	Initial Cluster (# in Cluster)	Parts Above	Adapted Cluster (# in Cluster)	Parts Above						
User 1	4823 / 4824 4838 / 4831	4839	4833	3 (1)	3	1 (7)	0						
								4823	4824	7 (6)	18	3 (6)	18
	4838 / 4831 4839 / 4833	4823	4835	7 (6)	18	3 (6)	18						
	4823 / 4835 4839 / 4833	4838	4831	2 (9)	12	1 (26)	0						
User 2								4823 / 4826 4838 / 4829	4839	4834	6 (3)	7	6 (3)
	4823 / 4837 4838 / 4830	4839	4454	6 (3)	7	6 (3)	7						
								4839 / 4834 4838 / 4829	4823	4837	2 (8)	5	2 (8)
	4839 / 4834 4838 / 4829	4823	4826	2 (8)	5	2 (8)	5						

For User 1, the 2-sample training set always improved the cluster ranking of the test part. For User 2, the cluster ranking remained unchanged. Rather than indicating a difference between the two users, this difference may be an artifact of sparse data. Many of these target / selection pairs do not cause a change to the profile. We believe that with a larger training set, we would see more adaptation and would see improvements in the cluster ranking.

3.6 Summary of Experimental Results

The use of a parametric distance measure for similarity measurements can significantly improve the search efficiency by finding appropriate replacement parts (as indicated by our test users) and presenting them to a user. All replacement parts were moved to the top ranked 21 parts (or top 0.5% of the entire dataset) with the default profile. It can also identify potential replacement parts that would not otherwise be found by increasing the size of the top-ranked clusters. In other words, this approach increases the recall of replacement parts, which gives the application engineer more options in finding a solution that is cost-effective. Starting with the default profile, once a user identifies one acceptable replacement and the profile is allowed to adapt, other acceptable replacements move into the top three clusters (if they were not already there). The data collected is too sparse to conclusively prove the hypothesis that learning on one target part can improve the results for a different target part. However, these preliminary experiments using real user data are encouraging.

4. CONCLUSIONS

In this paper we have sketched the obsolete parts acquisition process and presented a solution addressing that problem. Our technical approach to the problem makes use of machine-learning based techniques to profile the application requirements of various classes of engineers, and uses those profiles to tune the behavior of our nearest-neighbor based algorithms. We have shown that our approach works well in a set of experiments employing both real-world parts databases and realistic user queries. We have developed and tested our algorithms in collaboration with a community of actual users, who have validated our research directions. Finally, the central benefit of our approach is that it reduces the time required by an engineer to locate a replacement part and also increases the likelihood that a replacement part can be found and an expensive redesign avoided.

5. FUTURE WORK

In this work we have shown the success of our algorithms in stand-alone experiments. The next step will consist of an implementation of our profiling and clustering work in an actual commercial strength research prototype. We expect to commence that stage of our work in the near future.

In addition, we plan to explore the incorporation of other types of learning techniques into our search algorithms. In our current work we have employed profiling algorithms that leverage information about the previous behavior patterns of a given engineer to help that same engineer complete his or her task. We have not attempted to employ collaborative filtering techniques (e.g. [7], [8]) that make use of information about a community of users since in our initial prototype there was only a small group of

(relatively) dissimilar users. As a community of users develops around our ePortal, we expect to pursue this direction in future work.

6. ACKNOWLEDGMENTS

The authors would like to thank Application Engineers Kevin Kovar, Steve Leith, and Tom Smith from Rockwell Collins for providing invaluable insight into the obsolescence problem and resolution process and for providing us with the user data needed to evaluate our algorithms. Kevin also provided the domain expertise required for this project. The parts data set used for our experiments was a snapshot from CAPS Expert™, provided courtesy of IHS Corporation. The authors would further like to thank Don O'Brien and Leo Plonsky from the DLA and Curtis Holcomb from ATI Corp. for their guidance and feedback in this project. Finally, we thank Diane Snyder and Mike McGrath for their leadership and encouragement. This work was completed under the sponsorship of the Defense Sustainment Consortium (DSC) as part of Defense Logistics Agency Contract No.: N00140-00-BAA3339.

7. REFERENCES

- [1] Collaborative Opportunities in DMSMS: A Study Report, Industrial Base Information Center, AFRL/MLME, <http://www.dtic.mil/natibo/docs/dmsms-report-2.pdf>, 2001.
- [2] Alonso, R., Bloom, J., and Li, H. SmartSearch for Obsolete Parts Acquisition. *Technical Report*, Sarnoff Corporation, Princeton, NJ, 2002.
- [3] Cost, S. and Salzberg, S. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning* 10, 1993, 57-78.
- [4] Stanfill, C. and Waltz, D. Toward memory-based reasoning. *Communications of the ACM*, 29(12), 1213-1228.
- [5] Balabanovic, M., Shoham, Y., and Yun, Y. An Adaptive Agent for Automated Web Browsing. *Journal of Visual Communication and Image Representation* 6(4), 1995.
- [6] Pazzani, M.J. and Billsus, D. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning* 27, 313-331.
- [7] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of CSCW'94*, Chapel Hill, NC, 1994.
- [8] Shardanand, U. and Maes, P. Social Information Filtering for Automating "Word of Mouth". *Proceedings of CHI-95*, Denver, CO, May, 1995.
- [9] Joachims, T. Optimizing Search Engines using Clickthrough. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2002.