Audience Segment Expansion Using Distributed In-Database K-Means Clustering

Archana Ramesh
nPario Inc., Redmond, WA
archana@npario.com

Ankur Teredesai
Center for Web and Data Science, University of Washington
ankurt@uw.edu

Ashish Bindra, Sreenivasulu Pokuri, Krishna Uppala
nPario Inc., Redmond, WA
{ashish, pokuri, krishna}@npario.com

ABSTRACT

Online display advertisers extensively use the concept of a user segment to cluster users into targetable groups. When the sizes of such segments are less than the desired value for campaign budgets, there is a need to use probabilistic modeling to expand the size. This process is termed look-alike modeling. Given the multitude of data providers and online data sources, there are thousands of segments for each targetable consumer extracted from billions of online (even offline) actions performed by millions of users. The majority of advertisers, marketers and publishers have to use large scale distributed infrastructures to create thousands of user segments on a daily basis. Developing accurate data mining models efficiently within such platforms is a challenging task. The volume and variety of data can be a significant bottleneck for non-disk resident algorithms, since operating time for training and scoring hundreds of segments with millions of targetable users is non-trivial.

In this paper, we present a novel k-means based distributed in-database algorithm for look-alike modeling implemented within the nPario database system. We demonstrate the utility of the algorithm: accurate, invariant of size and skew of the targetable audience(very few positive examples), and dependent linearly on the capacity and number of nodes in the distributed environment. To the best of our knowledge this is the first ever commercially deployed distributed look-alike modeling implementation to solve this problem. We compare the performance of our algorithm with other distributed and non-distributed look-alike modeling techniques, and report the results over a multi-core environment.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.2.8 [**Database Management**]: Database Applications-Data mining

1. INTRODUCTION

In this paper we propose a novel distributed in-database clustering algorithm for look-alike modeling, and demonstrate that the proposed algorithm is scalable for very large sized segments, can handle hundreds of input attributes as modeling parameters (high-dimensional), and is comparably accurate with baseline previously reported approaches for this task.

In Internet advertising, a group of users that form a targetable audience is termed a segment. This notion of a user segment is central to ad targeting in both contextual and display advertising. The entire revenue growth of the online ad-technology industry is driven by advances in the ability to segment users efficiently and accurately into cohesive targetable groups, i.e., targetable segments [10]. As opposed to early stage online targeting mechanisms, which relied only on user demographics, recent online targeting relies on mining large volumes of user (consumer) events for insights that can be then monetized. This event data is often augmented with additional information available about the consumer from various third-party data providers who continuously track and probabilistically estimate a particular user's interests. This enriched consumer profile then enables advertisers to determine the exact combinations of criteria they want to target for a particular display advertising campaign. In spite of the multitude of data providers and large volumes of behavioral data, advertisers are often faced with a shortage of targetable consumers since ad-campaigns are often short-term, wide reaching and intended for a broad set of audiences. Hence, advertisers and marketers have come to rely on data mining models to enable them to expand the targetable audience based on the characteristics of the baseline segment [11]. This is termed look-alike modeling.

Online advertisers today use advanced tools to create a segment - a homogeneous group of users sharing several behavioral characteristics - to indicate a targetable set of users. In order to be useful, a given segment needs to meet with three criteria - homogeneity, identifiability and interpretability [16]. Segments are usually created towards specific campaigns and the total number of segments can be very large. Three challenges exist in the context of segmenting big data - a large number of attributes increasing the dimensionality of segmentation, a sparse distribution of segments across users and dynamic segments requiring segmentation to be done quickly on a daily basis.

In order to make useful conclusions from this data, traditionally, the data is loaded into a relational database or

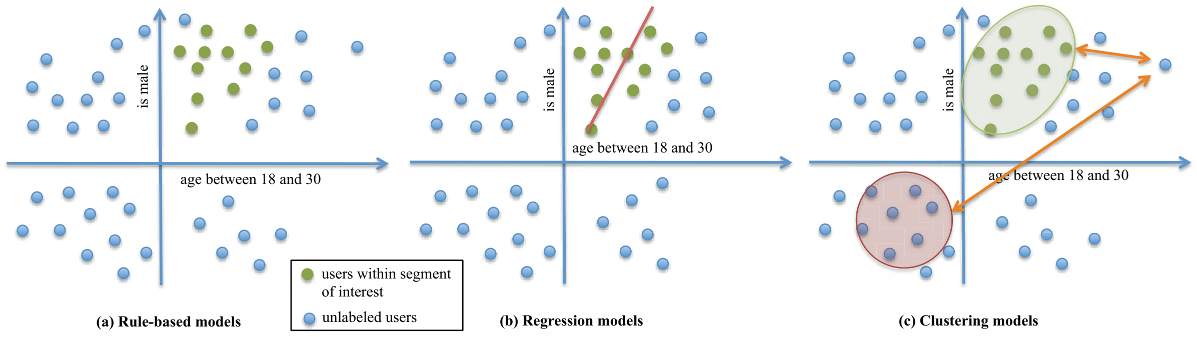


Figure 1: Modeling the segment "sports enthusiast" using rule-based approaches (a), regression approaches (b) and clustering approaches (c).

a warehouse. Most data warehouses do not natively support mining and analytics, so the data is usually exported and then processed by external applications (such as the R statistical computing framework or Matlab). After processing, results are imported into the data warehouse again. While this works well for small data sets, current day data sets far exceed the capabilities of this process and need to be processed in a distributed environment. Exporting and importing adds significant time to the analysis workflow, is cumbersome and error prone. Thus over the last decade data analytics are migrating closer to the database [6, 8] motivating the in-database approach adopted in this work. However, existing libraries tend to be generic approaches and fail to cope with the challenges posed by online advertising data.

Online advertising data contains membership information for billions of users across several thousand segments. To complicate matters further, membership information is typically only positive, implying that for a given segment, while we have a clear definition of who belongs to the segment, we do not know for sure whether users not belonging to the segment are truly negative or just unlabeled data. In addition, the data is also very sparse - very few conversions per impressions. This introduces a fairly well understood problem in marketing and ad targeting of finding too few users to make up a targetable audience for a given campaign or ad. This problem is usually caused by complex campaign rules, lack of density of third-party data and incomplete or conflicting user profile information. Consequently, advertisers are often looking to expand a given segment containing a small number of users who have responded well to a given campaign. To achieve expansion, advertisers and marketers rely on look-alike modeling to predict the likelihood of a given user belonging to a particular segment, based on known attributes about the user. This prediction task is non-trivial since the datasets often lack negative class labels, are of low-quality and density and often involve conflicting information about the same attribute from different data providers.

From a data mining perspective look-alike modeling translates into a predictive modeling task - given a segment definition in terms of a group of users sharing behavior, our goal is to assign to all users that do not belong to the segment, a score indicating how likely the user is to belong to the segment. The characteristics of a useful segment - homogeneity (a high similarity amongst the users within the segment) and identifiability (a low similarity between users of different segments) naturally lead us towards clustering approaches for segment expansion.

In this paper we present a fast, accurate algorithm for expanding segments and identifying other segments, which are, most related to the segment in question. We make several contributions through this paper :

- we develop a novel distributed in-database k-means algorithm, which computes segment expansions efficiently
- we compare our algorithm to the current state-of-the-art in look-alike modeling and demonstrate how our approach outperforms existing approaches
- we develop a novel approach to compute related segments with a low overlap in user membership.

2. RELATED WORK

Modeling users based on their propensity to respond to an advertisement campaign is a problem that has been of interest to advertisers for several decades. Given a segment that responds well to a campaign, advertisers are interested in expanding the segment to other users who would also be likely to respond well to the same campaign. Conventional approaches rely on manual rule-based models of segments, defined by those driving the ad campaigns. Such approaches characterize segments based on specific demographic rules applicable to the users within the segment (e.g., age between 18 and 24, annual income lesser than 80,000) [4]. Models are usually formulated by domain experts and thus, are commonly used in scenarios where such domain knowledge is readily available.

On the contrary, look-alike models allow for a fuzzy definition of segment membership, by identifying users which resemble those in the segment of interest. As a result, a look-like model could result in fuzzy definition of segments combining multiple attributes to differing extents. Several approaches for look-alike modeling have been developed in the last decade including regression approaches and frequent pattern mining approaches. Regression models [17] begin with a segment of users who respond to a certain ad campaign and fit a description (computationally modeled as a weighted sum of attributes). Additional users are then assigned to the segment based on how well they fit the description. On the other hand, there have been multiple endeavors applying frequent pattern mining to build look-alike models [3, 11] though none of these models have been implemented over a distributed big data framework or reported ability to process segments containing millions of users. Such approaches derive rules from pairs of features that define a segment. The rules are then applied to expand the segment. Behavioral data is characterized by a very small number of conversions or positive class labels in a given campaign, making it very challenging to use classification approaches such as regression or support vector machines in this context.

What the segment expansion process needs is a fast approach that will capture the behavior of users within the segments using a limited number of positive class labels, take into account the high-dimensional variability in those users, and assign a score to users outside the segment, based on how similar the user is to the segment.

Our attention now turns towards clustering approaches for segment expansion. Clustering is defined as the unsupervised classification of patterns (observations, data items or feature vectors) into groups (clusters) [9]. Clustering has been used in several related marketing applications [15, 13]. In this work, we develop an algorithm for segment expansion and look-alike modeling. We model segments as clusters of users with different characteristics, which we learn through the k-means algorithm. We then score users based the similarity between the user and the segment cluster.

An important aspect in dealing with big advertising data is dealing with the volume of the data. In the recent years, there has been a migration of data mining operations from externals applications to within the database [8, 6]. Additionally, columnar databases are becoming increasingly popular as the data management platform for advertising and marketing datasets due to the increasing amount of enterprises adopting key-value pair storage over relational data models [2, 14, 1]. At the same time, distributed data mining efforts such as Apache Mahout [12] and similar data mining techniques using Hadoop and Map-Reduce [5] based implementations are also becoming popular. Distributed databases have a sophisticated query processing engine that is able to partition data across multiple nodes very similar to the Map Reduce paradigm. Emerging in-database analytics [6, 8] are trying to exploit this highly optimized parallel query processing engine by writing algorithms which do not have to worry about distribution since the query processor will do that quite efficiently on its own.

For instance, MADlib [8] is an open source project working towards a suite of machine learning, mining, statistical and mathematical operations for databases completely based on data-parallel algorithms. Similarly, Bismarck [6] utilizes an efficient in-database implementation of the incremental gradient descent algorithm to develop numerous data mining algorithms such as regression and support vectors machines. Bismark is now a part of the MADLib library. MADlib works on PostgreSQL (which currently supports a single node) and on a distributed database based on PostgreSQL working on horizontally partitioned data sets (Greenplum community edition). However both of these are row stores, i.e., the query processor pipeline works on a record at a time. Our algorithm relies on a columnar store database so the data can be physically partitioned both horizontally and vertically across a very large number of nodes. The query processor works on a vector of rows at a time, providing drastic improvements in performance. Motivated by this we developed a distributed in-database implementation of k-means for segment expansion.

In this paper we present a look-alike model to learn the clustering model of segments and then use this model to make class predictions on the additional users that would belong to this segment. Our algorithm is one of the pioneers in distributed in-database look-alike modeling using k-means. As opposed to existing approaches, which extract the dataset and perform the mining process, or undergo multiple passes over the dataset, we perform our analysis within the database. As seen in the following sections, such an approach both minimizes latency and provides a high accuracy.

3. DISTRIBUTED IN-DATABASE CLUSTERING

Let U denote the population of users ($U = \{u_1, u_2, \cdots u_n\}$) typically identified by cookies or similar identifiers, S denote the set of segments ($S = \{s_1, s_2, \cdots s_m\}$) such as sports enthusiast or gender, and $f : S \to U$ denote an n:m mapping between the segments and the users where $f(s_i) = (u_{i1}, u_{i2}, \cdots u_{ij})$ represents those users that belong to the segment s_i. For implementation, we assume that this mapping works both ways i.e., data structures for both the users ($f(s_i)$) belonging to a specified segment (s_i) as well as the segments ($f^{-1}(u_i)$) that a specified user (u_i) belongs to can be constructed.

Algorithm 1 Clustering based Look-Alike Modeling

```
Input: segment of interest, $s_i$, user-segment mapping, $f(s_i)$, number of clusters, $k$, cross-validation folds, $c$
Output: expanded segment $s_i^{'}$, score $\alpha_j : \forall u_j \in (U \setminus U_i)$ accuracy of model $acc$
 1: $s_i^{'} \leftarrow \emptyset$
 2: $U_i \leftarrow f(s_i)$
 3: Split $U_i$ into $c$ groups $(U_i^1, U_i^2, \cdots U_i^c)$ for X validation:
 4: for $p = (1, 2, \cdots c)$ do
 5:     $U_i^{train} \leftarrow u_j : u_j \in U_i; u_j \notin U_i^p$
 6:     $U_i^{test} \leftarrow u_j : u_j \in U_i^p$
 7:     In-database k-means:
        $C_i \leftarrow KMeansTraining(U_i^{train}, k)$
 8:     $\overline{U}_i \leftarrow x \in U : x \notin f(s_i), |\overline{U}_i^{'}| = |U_i^{train}|$
 9:     $\overline{C}_i \leftarrow KMeansTraining(\overline{U}_i^{'}, k)$
10:     $count \leftarrow 0$
11:     for $u_j \in U_i^{test}$ do
12:         if $\min(\cos(u_j, C_i)) > \min(\cos(u_j, C_i^{'}))$ then
13:             $count \leftarrow count + 1$
14:         end if
15:         $acc_p \leftarrow \frac{count}{|U_i^{test}|}$ is accuracy of this run
16:     end for
17: end for
18: $acc \leftarrow \frac{\sum_{i=1}^{p} acc_p}{c}$
19: Perform scoring using the model:
20: for $u_j \in (U \setminus U_i)$ do
21:     if $\cos(u_j, C_i) > \cos(u_j, C_i^{'})$ then
22:         $s_i^{'} \leftarrow s_i^{'} \cup u$
23:     end if
24:     $\alpha_j \leftarrow \cos(u_j, C_i)$
25: end for
```

Given a segment of interest s_i consisting of users $f(s_i) = (u_{i1}, u_{i2}, \cdots u_{ij})$ to expand, the goal of look-alike modeling is to compute:

- for every user $u'_i \notin f(s_i)$, a **score** $\alpha \rightarrow [0,1]$ quantifying the likelihood of this user u'_i to belong to s_i.
- for every segment $s'_i \in (S \setminus s_i)$, the **lift** $l \rightarrow \mathbb{R}^+$, indicative of the extent to which the group of users in segment s'_i, resemble the segment s_i relative to all the segments in S.

3.1 Clustering based Look-Alike Modeling for Segment Expansion

Our model leverages the inherent overlapping nature of segment definitions as illustrated in Figure 2. Algorithm 1 illustrates the steps involved in expanding a segment using such a clustering model. Given a segment of interest s_i consisting of users $f(s_i) = (u_{i1}, u_{i2}, \cdots u_{ij})$, our algorithm begins with splitting these users into c folds for cross-validation. We use $c-1$ folds for training and 1 fold for testing. We then identify all the segments that these users belong to and train the user-segment profile to obtain a set of k centroids. Next, we sample a sub-set of users that do not belong to the segment (unlabeled users) of the same size as the previous training set and compute the centroid for this set. We term this set as "unlabeled" as opposed to a "negative" as this set consists of both users that do not belong to the segment of interest as well as users who resemble the segment of interest (that would be assigned a high score by algorithm 1). Our goal is to build models for both the users within the segment and users that do not belong to the segment. All candidate users (users that we are expanding to) are then compared to these two models and assigned to the segment, if the user is closer to the segment model than the other model. Using the test set, we compute the similarity of each user in the test set with both the positive and unlabeled centroids. If a user is found to be closer to the positive centroid than the unlabeled centroid, we consider it to be correctly classified.

Users in our scenario are identified by the segments they belong to, indicating the need for a distance metric that will compute distance based on the membership properties of elements such as cosine and Jaccard similarity. We used these to optimize on computation time and accuracy. First, the cosine similarity is used in training and validation as it provides finer granularity of the hypothetical means getting generated while computing the candidate centroids at each iteration. Once the centroids are computed, it is sufficient to just check the set overlap between iterations to indicate if iterations have converged using a simpler binary distance metric such as Jaccard. This helps save a few microseconds for each query which can add up very quickly for large datasets and commercial systems such as ours, when we have to expand hundreds if not thousands of segments daily.

The procedure *KMeansTrain* invokes a novel distributed implementation of the well known basic k-means algorithm [7] in which the initial centroid is used to train d different partitions of the data. This results in d different centroids. Before proceeding to describe the query that has been developed and the k-means algorithm, we first describe the model obtained by our clustering algorithm. Figure 2 aids in illustrating this concept. We use the overlapping nature of segments to define a segment in terms of several other segments. Thus our model takes the form of a set of clusters

Algorithm 2 K-Means Training

```
Input: training set, $U_i^{train}$, user-segment mapping, $f$, number of clusters, $k$
Output: centroids, $C_i$, a set of size $k$ where each element is composed of a set of segment-similarity pairs $(s_j, d_j)$ where $d_j$ is the cosine similarity of the segment $s_j$ with the segment $s_i$
1: $U_i^{initial} \leftarrow (u_1, u_2, \cdots u_k) : u_j \in U_i^{train}$
2: $C_i \leftarrow (c_1, c_2, \cdots c_k)$ :
     $c_j = (s_{j1}, 0), (s_{j2}, 0), \cdots (s_{jg}, 0)$;
     $f^{-1}(U_i^{train}) = (s_{j1}, s_{j2}, \cdots s_{jg})$
3: for $u_j \in U_i^{train}$ do
4:     $c_b \leftarrow c_b \cup f^{-1}(u_j)$:
           $cos(f^{-1}(u_j), c_a) = \max_{1 \le a \le k} cos(f^{-1}(u_j), c_a)$
5: end for
```

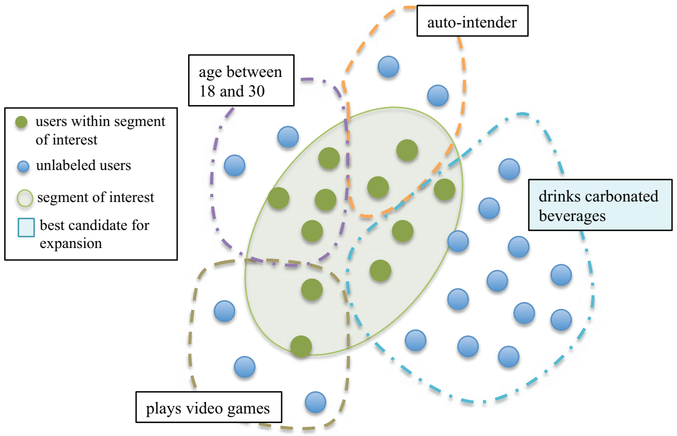

auto-intender
age between 18 and 30
● users within segment of interest
● unlabeled users
segment of interest
best candidate for expansion
drinks carbonated beverages
plays video games

Figure 2: Clustering model of the segment "sports enthusiast" illustrating the different related segments that users in this segment also belong to. Section The metric "lift" identifies the best candidate for expansion - the segment which has the most number of potential users.

(such as "auto-intender", "plays video games") each consisting of a set of segments and their individual similarities to the segment of interest. Our algorithm thus learns a set of k sub-clusters where each cluster consists of segment similarity pairs.

3.2 In-Database Distributed Query Syntax

Recall that our algorithm executes as an in-database query. Next, we describe the query syntax for training the k-means model. The query takes as input the user segment profile data and a set of S segments (typically selected as S distinct users and the segments they belong to) from the training set.

```
select segments as iteration_i_segments, similarity
from kmeansiteration (select
'<iteration_(i-1)_segments>', segments from
<user_segment_profile>);
```

The query returns a list of segments and the similarity of each segment with the segment of interest. The list of segments is organized into k clusters and the similarity is computed as in algorithm 2. We compare the centroids obtained at iteration i with the centroids obtained by iteration $i-1$ and check if the Jaccard similarity (as explained previously) between these centroids is greater than a certain threshold (typically 0.9). Convergence parameters are: all centroids unchanged (standard metric) or reaching the maximum number of iterations. We report these results in the subsequent experiments section.

The lack of negative training examples (examples on users who do not belong to a segment) poses a large challenge while measuring the performance of the look-alike model. Conventional metrics such as the precision and recall are no longer applicable due to the lack of any negative training examples (cannot identify the true negatives) to compare against. Hence, instead, we adopt alternative performance measures - the accuracy and confidence.

The accuracy of the test set is computed as:

$$accuracy = \frac{\text{number of correctly classified users}}{\text{total number of users}} \quad (1)$$

This measure is useful since most advertisers are interested primarily in classifiers that predict which users belong to a segment (rather than which users do not). Additionally, we also use the confidence to assess the performance of our methods. The confidence is measured after expanding the base segment (to sizes 2X, 3X and so on). The confidence at 2X is defined as the *minimum* confidence of the classifier for any user within the expanded segment twice as large as the original segment of interest i.e., if a segment was expanded to a size twice as large as the base size, confidence reflects the minimum similarity of a user (within this expanded set) to the base set. Typically, competing approaches often show average confidence since the average is higher than the minimum. In our case we use a conservative approach and report the minimum confidence. Further, the confidence was also computed at 3X expansion to demonstrate the power for look alike expansion.

3.3 Insights

One of the key contributions of our approach for look-alike modeling is the interpretation component which provides actionable insights for advertisers and marketeers. Next, we describe the approach to determine related segments for a base segment, and identify two efficiently computed metrics which permit ranking of the related segments: (a) overall similarity of a segment to the base segment and (b) lift. Our customers have indicated these two as very useful features of our application. Algorithm 3 shows the steps involved in computing the overall segment similarity and lift.

Since the centroids are a list of clusters each containing segment-similarity pairs, we first iterate through each segment within all pairs and compute for the segment the overall weighted average of similarities, weighted by the size of each cluster that the segment would be a part of. This process is also distributed and allows for normalization of the similarity score based on the cluster sizes giving equal importance to both segments within large clusters and those within smaller clusters. Thus we compute the overall segment similarity for each segment.

To compute the lift, we first score a random set of users that belong to the segment and determine how many belong to the segment of interest. We then scale this proportion by the average proportion of users in the related segment who belong to the segment of interest to obtain the lift. This metric allows advertisers to determine how good their return-on-investment would be if they were to target the related segment.

Algorithm 3 Insights and Lift Computation

```
Input: segment of interest, $s_k$,
    user-segment mapping, $f$,
    centroids, $C_i$, a set of size $k$ where each element is composed of a set of segment-similarity pairs $< s_j, d_j >$
Output: overall similarity, $r_i, \forall s_i \in (c_1 \cup c_2 \cdots \cup c_k)$,
    lift, $l_i, \forall s_i \in (c_1 \cup c_2 \cdots \cup c_k)$
 1: for $(s_j, d_j) \in (c_1 \cup c_2 \cdots \cup c_k)$ do
 2:     $r_j \leftarrow 0$
 3:     for $c_a \in (c_1, c_2 \cdots c_k)$ do
 4:         if $(s_j, d_j) \in c_a$ then
 5:             $r_j \leftarrow r_j + (d_j \times |c_a|)$
 6:         end if
 7:     end for
 8:     $r_j \leftarrow \sum_{c_a \in (c_1, c_2 \cdots c_k):(s_j, d_j) \in c_a} |c_a|$
 9: end for
10: for $(s_j, d_j) \in (c_1 \cup c_2 \cdots \cup c_k)$ do
11:     $t_j \leftarrow \frac{\sum_{u_j \in U: u_j \in f(s_j), cos((u_j)C_i)>0.5} 1}{\sum_{u_j \in U: u_j \in f(s_j)} 1}$
12: end for
13: for $(s_j, d_j) \in (c_1 \cup c_2 \cdots \cup c_k)$ do
14:     $l_j \leftarrow \frac{l_j}{\frac{\sum_{j:(s_j, d_j) \in (c_1 \cup c_2 \cdots \cup c_k)} t_j}{\sum_{j:(s_j, d_j) \in (c_1 \cup c_2 \cdots \cup c_k)} 1}}$
15: end for
```

4. EXPERIMENTS AND RESULTS

In the following subsections we describe the application of our algorithms to online retail data, experiments conducted and the performance results obtained through our experiments.

4.1 Dataset Description

We used online retail data from a digital marketing company for the experiments in this paper. On a daily basis, the data consists of ˜600 million impressions, ˜1.2 billion events, ˜513 million third-party cookies, ˜2700 segments and ˜200 gigabytes of new data. On average, a user was associated with ˜25 segments. Figure 3 shows the distribution of users across the segments. Our experiments were also tested on a second dataset, consisting of 125 million unique users and ˜2000 segments and found to yield similar results. In all our experiments we used the accuracy obtained through 5 fold cross validation in order to select the best parameter. Our experiments were conducted on a three node cluster with a 16-core AMD Opteron Processor 6128 and each node consisting of 8 x 2 GB data disks and 128 GB RAM.

4.2 Characterizing Parameters for K-Means

We conducted several experiments to attempt to reduce the time taken for segment expansion, by varying different parameters. In the following sub-sections we describe the relationships that the Jaccard convergence threshold, number of clusters (k) and the amount of training data used had on the performance of the algorithm.

4.2.1 Jaccard Similarity

The Jaccard similarity metric is used to determine whether the K-Means algorithm has converged after every iteration.

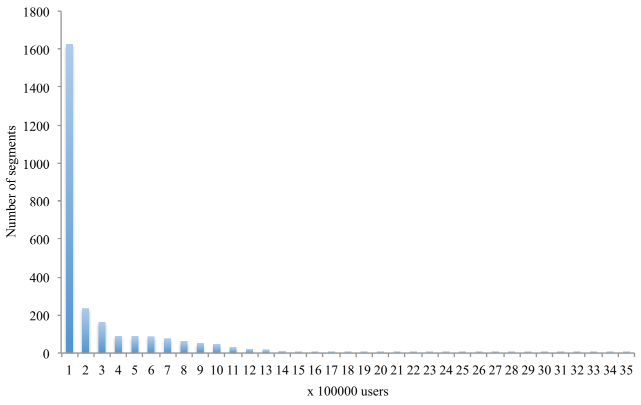


Figure 3: Histogram of the user segment distribution. This shows a long-tailed distribution with a steep descent. This plot illustrates the need for segment expansion.

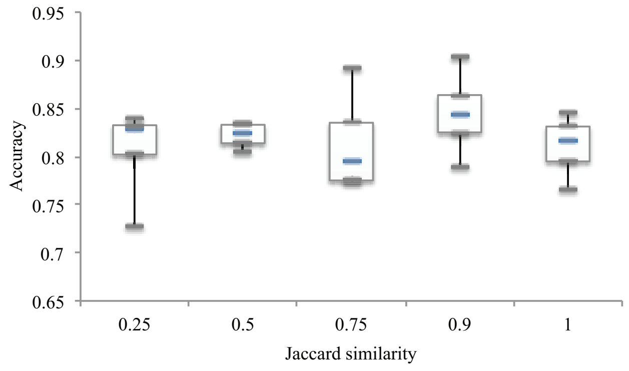


Figure 4: Average accuracy of the K-Means lookalike model across 5-fold cross-validation while varying the Jaccard convergence threshold. The box plot shows both the variance across the 5 folds and different segment sizes as well as the average accuracy. The highest accuracy is obtained with a similarity of 0.9.

Consequently, we anticipate that the lower the Jaccard similarity, the faster the convergence and possibly, the lower the accuracy, since we are relaxing the criteria for how consistent the results of subsequent iterations should be. We varied the Jaccard similarity between 0.25 and 1 and the results are shown in the box plot in Figure 4. Surprisingly, we noticed that lower Jaccard convergence thresholds did not necessarily imply a lower accuracy. This is attributed to the inability of the training to converge within the specified number of maximum iterations yielding a poor model for validation. The best accuracy was achieved at a Jaccard convergence threshold of 0.9 and this was used for all subsequent experiments. Further our experiments were conducted on segments of varying sizes and the average results are reported.

4.2.2 Number of Clusters (k)

The next parameter we studied was the number of clusters. For our data, we varied k between 3 and 40 and found that k = 10 provides the best average performance and with k > 10, the performance tended to fall. This was an interesting find as this indicates that even with large numbers of users, most users tend to fall under similar profiles with similar behaviors. Our results are shown in Figure 5.

4.2.3 Training Data

Motivated by the small number of clusters (relative to the segment sizes), our motivation was to determine if accurate models could be obtained from smaller sample sets of the data. This would then significantly reduce memory requirements and improve performance. We conducted experiments on samples of varying sizes representing 12.5%, 25 %, 27.5 % and 50 % of the data (totally consisting of 120 million users). Surprisingly, we found that even with a 4-fold increase in training size, the accuracy remained with a 0.01 margin. This finding is significant as it shows that random sampling has tremendous potential in building predictive models from big data.

4.2.4 Performance Analysis and Comparisons

As discussed earlier, frequent pattern mining through association rule mining is a popular mechanism used for segment expansion. In this section, we compare our k-means algorithm with association rule mining [3]. Further, we compare two variants of the algorithm, a distributed one and non-distributed one. Our results are shown in tables 1 and 2. Our system runs on thousands of segments in production, but we constrained the experiment to 10 segments of varying sizes, with 5 fold cross validation to elucidate the spectrum of results. Further, we used a subset of the data (12.5% of the original data) for the experiments in this section for two reasons - a. we identified (refer figure 6) that training samples as small as 12.5% of the data are adequate to build accurate models and b. we compare against association rule mining which is time-consuming for large segments.

In table 1, we show the variation in size of the segments and the overall accuracy. We use multiple measures to assess the performance of our algorithms, including the accuracy obtained on the test set and confidence at 2X and 3X. Table 2 shows the performance in terms of the time taken by the three algorithms. For association rule mining, we provide the total time taken (both training and scoring). For the k-means modeling, we provide more granular timings capturing both training and scoring time as well as the number of iterations for building the two models. We implemented and tested two variations of the in-database strategy as outlined earlier. In the first strategy we implemented the algorithm as in-database but without distribution. In the second case we extended the algorithm to not only be in-database but also distributed across nodes available to the distributed query processor. Table 2 provides the number of iterations to converge and the time per iteration for both the positive and unlabeled case.

Our results were interesting on multiple levels. First, we noticed that k-means consistently outperformed in-database association rule mining in both accuracy and time taken to generate a model. Secondly, we also noticed that, although the confidence obtained at 2X and 3X expansions did decrease as the original segment size increases, overall k-means based expansion results in lesser decline in confidence com-

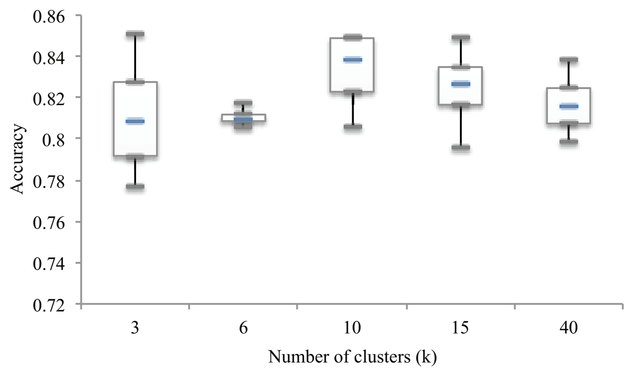


Figure 5: Average accuracy of the K-Means look-alike model across 5-fold cross-validation while varying the number of clusters (k). The box plot illustrates both the average accuracy obtained as well as the variance across the 5-fold cross validation runs for segments of varying sizes. The plot illustrates two metrics that could be used to pick the best number of clusters (k) - the accuracy and the variance of accuracy across predictions. While k = 10 shows the highest overall accuracy, k = 6 shows the most robust model.

pared to the association rule mining model. The results indicate that the in-database implementation combined with distribution is a sound strategy when base segment sizes are large. The savings in terms of time compared to the overhead of distribution is not significant for very small segment sizes (less than thousand users) which are fairly rare for targeting purposes. In the future we intend to automate these cut-off points empirically for a given dataset to determine if we should employ a distributed or non-distributed strategy for a given segment.

4.2.5 *Insights*

Using the trained models obtained for each segment, we analyzed the insights obtained from each segment. These are listed in table 3. Insights, as described before (also illustrated in figure 2), are segments related to the segment of interest. The lift value denotes how much the related segment resembles the segment of interest relative to other segments in the data set. It is interesting to see that iPhone owners with a mortgage tend to travel internationally, be upscale and like to gamble. Similarly those who gamble or are in good financial health tend to be top tier spenders on computer & software, home & garden, and movies and also use rewards cards. This provides advertisers a good idea of which related segments to target for a specific campaign.

5. CONCLUSION

Our paper makes a novel contribution to the field of scalable consumer intelligence and data science by providing a fast, scalable algorithm to model user segments. There is tremendous utility in developing in-database audience intelligence mining models, due to the ability of this architecture to scale while maintaining the flexibility of performing database queries. In this paper, we describe a real-world, deployed system that successfully helps advertisers and marketers create segments and then expand them in a matter of seconds, using a novel in-database implementation of k-means on the nPario massively parallel distributed columnar database. We describe the details of our algorithm and demonstrate the parameters that need to be tuned in order to achieve the most accurate segment expansions. We find that the in-database implementation combined with distribution is sound when base segment sizes are large. We also provide conclusive evidence to support the claim that with a random sample of as little as 12.5 % of the original big data set, one can still build accurate predictive models. Thus, the decision to use either the distributed or non-distributed algorithm for look-alike modeling is influenced by several factors including the capacity of the distributed environment and the segment size.

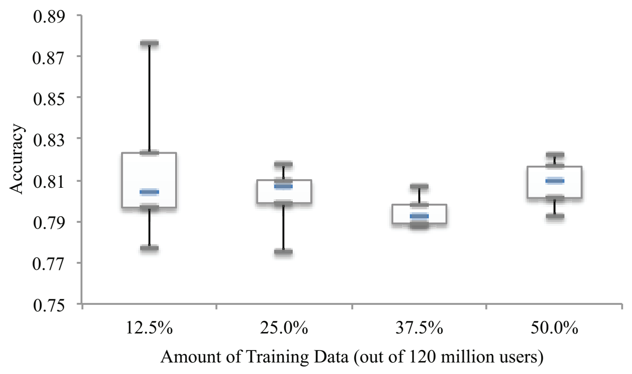


Figure 6: Box plot showing the effects of the training data size on accuracy. The plot illustrates both average accuracy as well as the variance in the accuracy across 5-fold cross-validation applied to segments of varying sizes. An interesting find from this figure is that a training sample as small as 12.5% of the total data would be adequate to build an accurate predictive model.

6. ACKNOWLEDGMENTS

The authors would like to thank Arta Shayandeh, Wenke Zhang, Dave Taniguchi and Ramesh Pabbati for their technical contributions and insights and Christa Carpentiere for editorial support.

7. REFERENCES

[1] D. J. Abadi, P. A. Boncz, and S. Harizopoulos. Column-oriented database systems. *Proceedings of the VLDB Endowment*, 2(2):1664–1665, 2009.

[2] D. J. Abadi, S. R. Madden, and N. Hachem. Column-stores vs. row-stores: How different are they really? In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 967–980. ACM, 2008.

[3] A. Bindra, S. Pokuri, K. Uppala, and A. Teredesai. Distributed big advertiser data mining. In *2012 IEEE 12th International Conference on Data Mining Workshops (ICDMW)*, pages 914–914. IEEE, 2012.

[4] A. Broder and V. Josifovski. Computational advertising MS&E239. *Stanford University Course*

Table 1: Performance (Accuracy) Comparisons between Association Rule Mining, In-Database K-Means and Distributed In-Database K-Means.

Algorithm	Segment ID	1	2	3	4	5	6	7	8	9	10
	Size	19935	25857	47514	52558	76273	91447	100578	238397	951434	1016882
In-Database Association Rule Mining	Accuracy	0.40	0.77	0.73	0.81	0.55	0.42	**0.83**	0.51	0.21	0.34
	Confidence (2X)	0.43	0.09	0.32	0.12	0.39	0.18	0.22	0.27	0.001	0.001
	Confidence (3X)	0.38	0.07	0.22	0.03	0.25	0.11	0.18	0.17	0.001	0.001
In Database K-Means	Accuracy	**0.82**	**0.88**	0.83	**0.97**	0.80	0.85	0.73	**0.87**	0.79	**0.73**
	Confidence (2X)	**0.72**	**0.73**	**0.67**	**0.63**	**0.67**	**0.66**	0.66	0.55	**0.25**	0.13
	Confidence (3X)	**0.71**	**0.71**	**0.65**	**0.59**	**0.64**	**0.61**	0.62	0.46	**0.12**	0.08
Distributed In Database K-Means	Accuracy	0.78	**0.88**	**0.91**	0.96	**0.81**	**0.89**	0.86	0.84	**0.80**	**0.73**
	Confidence (2X)	**0.72**	**0.73**	**0.67**	0.62	**0.67**	**0.66**	**0.67**	**0.56**	**0.25**	**0.16**
	Confidence (3X)	0.70	**0.71**	**0.65**	0.58	**0.63**	**0.61**	0.63	**0.47**	**0.12**	**0.09**

Table 2: Performance (Time) Comparisons between Association Rule Mining, In-Database K-Means and Distributed In-Database K-Means.

Algorithm	Segment ID	1	2	3	4	5	6	7	8	9	10
	Size	19935	25857	47514	52558	76273	91447	100578	238397	951434	1016882
In-Database Association Rule Mining	Total Time (seconds)	580	462	734	1449	344	989	1025	816	421	1485
In-Database K-Means	Number of Iterations (Positive)	9	6	8	7	6	10	7	7	6	12
	Number of Iterations (Unlabeled)	8	8	8	14	8	9	7	6	10	9
	Time (seconds) per Iteration	0.467	0.414	0.610	1.006	0.829	0.711	0.588	0.812	1.715	3.059
	Scoring Time (seconds)	10.412	10.568	11.125	9.860	9.671	9.106	11.126	10.856	10.252	9.602
Distributed In-Database K-Means	Number of Iterations (Positive)	6	9	6	5	6	6	9	7	6	6
	Number of Iterations (Unlabeled)	7	8	6	7	10	6	4	7	8	7
	Time (seconds) per Iteration	0.434	0.423	0.538	0.974	0.561	0.623	0.566	0.849	1.081	2.207
	Scoring Time (seconds)	5.643	5.657	5.461	5.364	5.551	6.760	6.700	5.383	5.570	6.175

Table 3: Insights Obtained Using Clustering Model.

Segment	**Insights (Related Segments)**	**Lift**
iPhone Owners With Mortgage (52558 users)	Intent : Retail : Propensity - Retail - Upscale	1.48
	Intent : Personal Tech : Propensity - Personal Tech - Mobile - BlackBerry	1.43
	Intent : Personal Tech : Propensity - Personal Tech - Mobile - Smartphone	1.41
	Intent : Retail : Propensity - Travel - Likely Gamblers	1.29
	Intent : Retail : Propensity - Travel - International	1.26
Male Truck/ SUV/ Minivan Buyer (91447 users)	Interest : Purchase Behavior : Purchase Behaviors - Shopping - Home and Garden - Appliances	1.35
	Interest : Purchase Behavior : Purchase Behaviors - Shopping - Books	1.34
	Interest : Purchase Behavior : Purchase Behaviors - Shopping - Personal Tech	1.3
	Interest : Purchase Behavior : Purchase Behaviors - Shopping - Home and Garden - Decor	1.28
	Branded : Hobbies - Reading	1.22
Gamblers or Good Financial Health (1016882 users)	Interest : Purchase Behavior : Purchase Behaviors - Shopping - Home and Garden	1.33
	Branded : USE THIS: MasterCard - Top Tier Spender - Retail - Computer & Software Sales	1.22
	Branded : USE THIS: MasterCard - Top Tier Spender - Entertainment - Movies	1.2
	Branded : USE THIS: MasterCard - Top Tier Spender - Premium Cards	1.17
	Branded : USE THIS: MasterCard - Top Tier Spender Rewards Cards	1.17

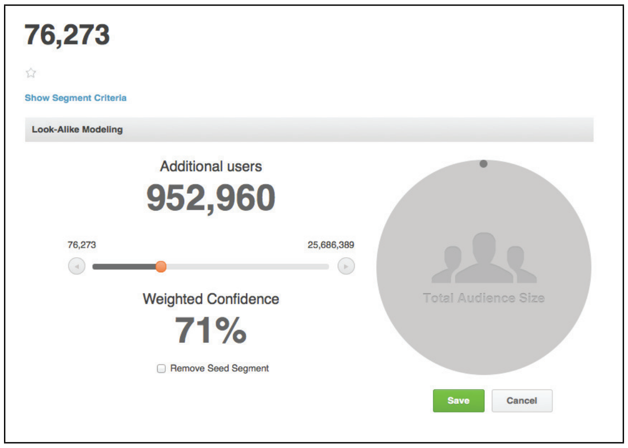
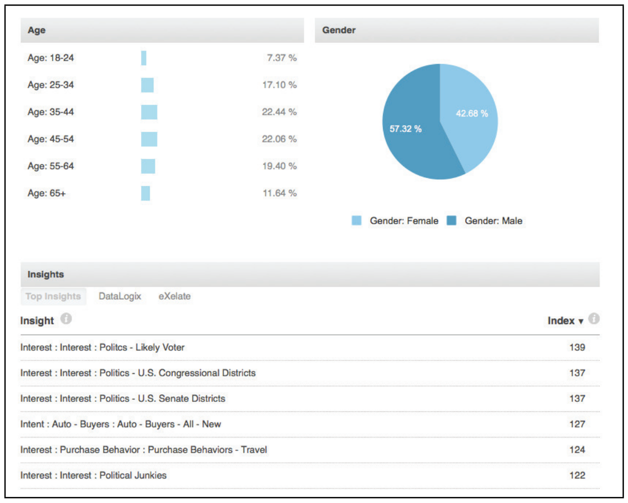


Figure 7: Snapshot of the application interface showing expansion of the segment "Between 18 - 24 with a wireless plan". The circle on the right shows how many users are in the segment in relation to the entire population. The slider allows users to interactively explore how much confidence they would need to compromise to reach a certain expansion.



Insight	Index ▾
Interest : Interest : Politcs - Likely Voter	139
Interest : Interest : Politics - U.S. Congressional Districts	137
Interest : Interest : Politics - U.S. Senate Districts	137
Intent : Auto - Buyers : Auto - Buyers - All - New	127
Interest : Purchase Behavior : Purchase Behaviors - Travel	124
Interest : Interest : Political Junkies	122

Figure 8: Snapshot of the application interface showing insights for the segment "College graduates and IT professionals". The pie chart on the right shows the gender distribution within the segment, while on the left one can see the age distribution within the data. The table shows a list of the segments related to this segment along with the lift in percent.

Materials, 2011.

[5] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[6] X. Feng, A. Kumar, B. Recht, and C. Ré. Towards a unified architecture for in-RDBMS analytics. In *Proceedings of the 2012 International Conference on Management of Data*, pages 325–336. ACM, 2012.

[7] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, pages 100–108, 1979.

[8] J. M. Hellerstein, C. Ré, F. Schoppmann, D. Z. Wang, E. Fratkin, A. Gorajek, K. S. Ng, C. Welton, X. Feng, K. Li, et al. The MADlib analytics library: or MAD skills, the SQL. *Proceedings of the VLDB Endowment*, 5(12):1700–1711, 2012.

[9] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[10] I. A. B. . P. C. LLP. IAB Internet advertising revenue report. www.iab.net, 2011.

[11] A. Mangalampalli, A. Ratnaparkhi, A. O. Hatch, A. Bagherjeiran, R. Parekh, and V. Pudi. A feature-pair-based associative classification approach to look-alike modeling for conversion-oriented user-targeting in tail campaigns. In *Proceedings of the 20th International Conference Companion on World Wide Web*, pages 85–86. ACM, 2011.

[12] S. Owen, R. Anil, T. Dunning, and E. Friedman. *Mahout in Action*. Manning Publications Co., 2011.

[13] N. Sinha, V. Ahuja, and Y. Medury. Cluster analysis for consumer segmentation using a brand customer centricity calculator. *Apeejay Business Review*, page 68.

[14] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, et al. C-store: a column-oriented DBMS. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 553–564. VLDB Endowment, 2005.

[15] H. Wang, D. Huo, J. Huang, Y. Xu, L. Yan, W. Sun, and X. Li. An approach for improving k-means algorithm on market segmentation. In *2010 International Conference on System Science and Engineering (ICSSE)*, pages 368–372. IEEE, 2010.

[16] M. Wedel and W. A. Kamakura. *Market Segmentation: Conceptual and Methodological Foundations*, volume 8. Springer, 2000.

[17] J. Yan, D. Shen, T. Mah, N. Liu, Z. Chen, and Y. Li. Behavioral targeting online advertising. *Online Multimedia Advertising: Techniques and Technologies*, pages 213–232, 2011.