Score Look-alike Audiences

Qiang Ma, Eeshan Wagh, Jiayi Wen, Zhen Xia[†], Robert Ormandi[‡] and Datong Chen
Yahoo!, Sunnyvale, California
{qiang,eeshanwagh,jiayi,datong}@yahoo-inc.com [†] zhenxia@utexas.edu [‡] ormandi.robert@gmail.com

***Abstract*—Look-alike models, which are efficient tools for finding similar users from a smaller user set, are quickly revolutionizing the online programmatic advertising industry. The datasets in these contexts exhibit extremely sparse feature spaces on a massive scale, so traditionally, the state-of-the-art look-alike models have used pairwise similarities to construct these similar user sets. One of the key challenges of the similarity-based models is that they do not provide a way to measure the potential value of the users to an advertiser, which is crucial in an advertising context. We propose methods to *score* users within the expanded audience in a way which relates directly to the business metric that the advertiser wants to optimize. We present three scoring models and show that, through empirical evaluation using real-world, large-scale data, by incorporating the potential value of a user to an advertiser into our scoring model, we can significantly improve the performance of the look-alike models over methods which only use pairwise similarities of users.**

I. Introduction

Look-alike models are increasingly popular and effective tools for online programmatic advertising supported by many popular advertising platforms. With these models, an advertiser can specify a set of users, e.g. a set of previous customers, then ask the system running such models to algorithmically generate an expanded audience of "look-a-like" users who have similar behavioral patterns to the original user set. This is a powerful tool as advertisers can market their product to users who have similar purchasing patterns to those who have already engaged with their product and, as a consequence, achieve higher conversion[1] rates than traditional targeting methods [1], [2]. The key benefit of look-alike modeling is that it does not need the advertiser to explicitly specify the target user set, instead, the look-alike model, by construction, infers important user behavior and can model more complex user interactions and patterns than an advertiser would manually be able to specify.

Look-alike modeling presents an interesting new direction for recommender systems. Traditionally, we are given a single user, and we want to recommend some small number of items for that particular user. For example, this could mean recommending the most relevant movies, news articles, songs, etc. for a single user given their preferences. With look-alike modeling, we are given a large set of users, and we would like to create another, even larger, expanded set of users who have similar behavioral patterns to our input set of users. In the context of advertising, we also want the output of our model to have reasonable size on the entire population of users so that we can identify as many relevant users to target as possible without sacrificing accuracy. Therefore, in contrast to the traditional recommender systems, a usual look-alike system models a many-to-many type of relationship between users where both the input, and the expanded user set can have extremely large sizes, even tens of millions of users.

The above mentioned many-to-many type of relationship, along with the massive scale, and sparse feature sets, which are quite common in the advertisement related problems [3], [4], leads to approaches which use pairwise similarities between users [2], [1]. In these approaches, users are represented as feature vectors reflecting their activities within the advertisement system. Then, traditionally, the audience expansion is carried out by selecting the top-M most similar users to the input user set from the user base of the advertisement system measured on the feature vectors [2]. In order to decrease the computational complexity associated with computing similarities over billions of users, the locality-sensitive hashing (LSH) [5] technique is often applied to approximate these pairwise similarities, and speed up the search of the most similar users.

Due to their simplicity and the achieved, significant performance lift, look-alike models are revolutionizing the industry of the online advertisement [6], [7], [2]. However, these models mainly rely only on the similarity assumption (i.e. similar behavior to the behavior of the input users, leads to higher conversion rate), and they are not optimized against any business metrics (e.g. conversion rate, that it the ratio of the actual number of clicking users divided by the total number of users who have seen the ad served) of the advertiser. In other words, these methods directly model the expanded user set as a set with a membership measure (similarity) only, and there is no direct way to estimate the *potential value* of a given user within the expanded user set for a particular advertiser.

The main contributions of our paper include the followings:

1) ScoringLook-alike, a novel and generic approach for fitting look-alike models at scale.
2) The introduction of three novel scoring models to rank users within a look-alike audience which integrate the similarity based approaches with methods that estimate the potential value of a user to an advertiser.
3) Empirical analysis of proposed methods by comparing them to multiple baseline methods on real-world, large-scale campaigns data at *Yahoo!*'s advertisement platform.

The outline of the paper is as follows. Section II specifies the problem we consider, and define a few preliminary notations used throughout the paper. In Section III, we summarize the related work and elaborate on some common algorithmic techniques used to build our algorithm. Section IV presents our core contributions by introducing the algorithm ScoringLook-alike along with the three scoring models. It also discusses the main intuitions behind the proposed models. Section V contains the experimental evaluation. Finally, Section VI concludes the paper.

II. Problem Statement

As we have briefly outlined in the previous section, the high-level definition of look-alike modeling can be summarized as follows. Given a set of seed users S, our goal is to find a set of expanded users R from a universe U in a way that the users of the set R exhibit similar behavioral patterns to the users of the seed set S. The universe U can be considered as the total user base of an advertisement system in which the model is executed. A user is represented by a feature vector which reflects the user's past activities collected by the advertisement system. The feature vector usually has very large dimensions (e.g. millions of binary features) with extreme sparsity (e.g. 0.1% of non-zero elements). In general, throughout the paper, we refer to this dimension as D.

[1] In online advertising, conversion usually refers to as some specific user reactions to advertisement, such as purchases, subscription to services, etc.

2375-9259/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDMW.2016.62

Moreover, we would like to *rank order* the recommended users of the set R with meaningful, business value-related (e.g. estimated conversion rate) scores through a scoring function Score : $R \to \mathbb{R}$. This score is supposed to measure the potential value of each recommended user for the advertiser which can play a different role than the behavioral similarities measured between the users of the input and expanded user sets, though they could be correlated depending on how the score function is constructed. We expect that a higher value of the Score function suggests a stronger potential value for the advertiser.

Additionally, the problem we target has a massive scale. A typical configuration of the set sizes above could be $|S| = 1M(Million)$, $|R| = 20M$, and $|U| = 3000M$. In general, we impose the constraints regarding size of the participant sets: $1 \ll |S| \ll |R| \ll |U|$. Throughout the paper, generally, we refer to the size of the set S as N ($N = |S|$), the size of the set R as M ($M = |R|$), and the size of the set U as T ($T = |U|$). We assume that all the data stored within a distributed file (HDFS [8]) or database (like Cassandra [9] or BigTable [10]) system, and all the computation has to be expressed within a distributed system. Our computational model consists of multiple computational units, called workers, handling a limited amount of local memory and storage space, communicating with a limited amount of other workers (distributed in-memory computational environment [11]). Spark [12], [13] is the particular system that we used to implement our algorithm. However the algorithm can be easily ported to other architectures (like Flink [14] or similar architectures systems [15]) with similar computational and communication capabilities.

III. Related Work

Look-alike audience expansion systems are available in various forms in the online advertising industry. Classification-based methods [6], [16] assume separability of seed-like users from the remaining ones within the user feature space. They directly model the probability of a user being a "look-alike" compared to the seed users by maximizing the log-likelihood of the model considering the seed users as positive examples while randomly sampling negative users following different strategies. One naive strategy is to handle the non-seed users as negative samples which lead to a highly imbalanced learning problem in which potential converters are considered as negative examples with high probability [3]. A more sophisticated strategy involves sampling the negative examples from the past non-converter users regarding the same or similar ad campaigns [2], [16]. In these methods, the audience is expanded by applying the classification model on all users of the advertising system, and selecting the ones with high predicted probability of belonging to the positive class membership.

Rule-based models [7], [1] learn weights for the features instead of modeling the connection between the features and the binary label of being a seed-like user. In these approaches, the weight function of a feature is designed to reflect the conversion performance of users who have the particular feature[2]. Then, greedily, the best performing features are gathered, and users who have those features are collected as the recommended audience.

Similarity-based models [2] realize that the problem of look-alike modeling is a recommendation problem and tackle the problem by recommending an expanded user set by applying similarity measures. Several variants of these methods known including the naive similarity-based [2], segment-based [1], and locally sensitive hashing (LSH)-based [2] look-alike models. The naive similarity-based approaches compute directly the pairwise similarities between the seed and all users seen by the advertisement system. Then, they directly choose the expanded user set by applying a variant of k nearest neighbor (kNN) search. The main drawback of these techniques is the runtime complexity of $\mathcal{O}(NTD)$ which is clearly not tractable in a typical online advertisement scenario where each component (N, T, and D) is large. The segment-based methods mitigate the runtime complexity by applying predefined buckets, called segments, to speed up the similarity search. While the LSH-based methods approximate the similarity computation and the kNN search by applying hash-based computational tricks. Since our proposed scoring methods heavily rely on this technique, we take a closer look at it in the following section.

A. Background

The LSH technique is a general method to speed up the kNN search. It has been successfully applied in many areas, such as duplicate image detection [17], [18]; topic extraction from documents [19]; web page clustering [20], genomic sequence comparison [21]; 3D object indexing [22].

To find the top-M most similar users to a seed set S efficiently at scale, one needs to approximate the pairwise similarities, and the search for those users to mitigate the squared runtime complexity associated with the a naive implementation of the model. A common approximation technique is the LSH [5] technique. The particular application of LSH in the look-alike modeling is summarized in our previous work [2] while the general concept of LSH to approximate kNN search were developed in [5].

The application of LSH for look-alike modeling is twofold: (1) at model building time, all the users of the advertisement system (U) are mapped into a lower dimensional, hashed space which approximates the similarities measured in the original space (Algorithm 1); while, (2) at prediction time, the seed users are mapped into the same space, and the expanded user set is collected (Algorithm 2).

The Algorithm 1 applies the LSH technique through the function $transformUser$. This function calculates the min-hash of the user's feature vector which approximates the Jaccard similarity of the original feature vectors [23]. By applying different hashing schemes, other similarity metrics can be approximated in a similar way [23]. After having the signature vector, the method updates the mapping between the bucket IDs and the users having those bucket IDs within their signature vector at line 6 of the function *signatureMatrix*. Here $\mathcal{H}$ is a set of K independent hash functions $h_j : \{1, \ldots, D\} \to \{1, \ldots, Z\}$ where Z denotes the range of the hash functions. As we detailed in our previous work [2], tuning the range parameter Z of the hash functions applied (banding technique [23], [2]) provides a way of controlling the granularity of the similarity approximation. That is, setting the value of the parameter Z higher leads to a larger similarity threshold within the seed set expansion at Algorithm 2.

The expansion of a user seed set S is shown in Algorithm 2. The method simply calculates the signature vectors of the users provided at the seed set S, then calculates the expanded user set E by taking all the users from the buckets assigned to the signatures corresponding to the users in the seed set. Based on the theory behind LSH and the banding technique mentioned above, this approach approximates the kNN similarity search with a given granularity.

IV. Main Algorithm

As we have summarized in Section III, look-alike models based on pairwise similarities rely on the assumption that targeting users with similar behavioral patterns to the input set of

[2] These methods assume that a binary feature set is applied.

Algorithm 1 LSH—Model Fitting

```
procedure $signatureMatrix(U, \mathcal{H})$
    $M \leftarrow \emptyset^Z$                          ▷ initialization
    for $k \in U$ do
        $m_k \leftarrow transformUser(k, \mathcal{H})$
        for $b \in m_k$ do
            $M_b \leftarrow M_b \cup k$        ▷ add user $k$ to bucket $b$
        end for
    end for
    return $M$                                   ▷ signature matrix
end procedure
```

```
procedure $transformUser(k, \mathcal{H})$
    $B \leftarrow 0^{|\mathcal{H}|}$                              ▷ initialization
    $x_k \leftarrow userFeatures(k) \in \mathbb{R}^D$         ▷ sparse, binary
    for $h_j \in \mathcal{H}$ do
        $B_j \leftarrow \infty$                        ▷ $j$th component of $B$
        for non-zero index $i$ of $x_k$ do
            $B_j \leftarrow \min(B_j, h_j(i))$      ▷ update min-hash $j$
        end for
    end for
    return $B$                 ▷ buckets assigned to the user $k$
end procedure
```

Algorithm 2 LSH—Expansion

```
procedure $expand(S, \mathcal{H}, M)$
    $E \leftarrow \emptyset$
    for $u \in S$ do
        $m_u \leftarrow transformUser(u, \mathcal{H})$
        for $b \in m_u$ do
            $E \leftarrow E \cup M_b$          ▷ add users from bucket $b$
        end for
    end for
    return $E$
end procedure
```

users will correlate with the advertisers' goal. However, these goals could vary significantly between advertisers operating in different contexts, so we need a way to explicitly incorporate the potential value of a user to an advertiser in these models. On the other hand, one can consider the classification-based approaches which apply conversion-based label generation that identifies the expanded user set by relying heavily on the conversion information of the ad campaigns, and tend to ignore more the user characteristics provided by the seed set S.

We propose a way to address this by providing a general pairwise similarity-based approach, called SCORINGLOOKALIKE. This method was inspired by the LSH algorithm showed in Algorithm 1 and 2. However, it was redesigned to take into account the specifics of the in-memory computational model that we work in, and provides a way to efficiently calculate the distributed signature matrix and other side information within this computational environment. This side information (e.g. conversion related information) directly relates to the advertiser's goal leading to a more balanced model which incorporates that side information with the similarities to calculate the final scores measuring the users potential value for the advertiser. Within the general framework, we derive three particular scoring models. The first method scores users purely based on similarity to the input seed set. The second method ranks users based on the importance of their features for predicting their value to the advertiser. The third method combines the above two inspired by the idea behind the collaborative filtering methods from the area of recommender systems.

Let us first elaborate on the fundamentals of the proposed scoring models, then combine them together into an efficient distributed algorithm.

A. Score by Seed Similarity

As we highlighted earlier, the similarity measured between an arbitrary user u of the advertisement system, and the users of a seed set S can directly be considered as a measure of the user's value, since we assume that the seed set S indirectly encodes the hidden valuation of the advertiser through the initial assumption of the similar behavior of a user u to the users in the seed set leads to higher conversion probability. Hence, considering the pure similarities measured between the users of the advertisement system and the seed users seems a reasonable scoring model.

In order to measure the similarity between an arbitrary user u and a set of users S, we need to involve the computation of the similarity between u and a single seed user $s_i \in S$, and aggregate those elementary similarities measured between u and s_i to get the final similarity between the user u and seed user set S. Also, from the viewpoint of computation, we need to be able to calculate those similarities efficiently, since, as it was stated in the problem statement in Section II, both the size of S and the dimension of the feature space D, in which the similarities are calculated, are large.

Considering these requirements, we have many alternatives for the particular similarity measures such as Jaccard similarity, Cosine similarity, Dice similarity, etc. Also, these similarities can be efficiently approximated by applying various LSH schemas [24]. In our particular case, we applied Jaccard similarity throughout the paper which defined in Eq. 1. Here F_q denotes the set created from the feature vector x_q of the user q by collecting its non-zero indices for both $q \in \{u, s_i\}$.

$$\text{Jaccard}(u, s_i) = \frac{|F_u \cap F_{s_i}|}{|F_u \cup F_{s_i}|} \tag{1}$$

Having the elementary Jaccard similarity, the score function according to our first scoring model, called model "Scoring by Seed Similarity", is defined in Eq. 2 which is an unnormalized measure between an arbitrary user u and a user set S.

$$\text{Score}(u) = \sum_{s_i \in S} \text{Jaccard}(u, s_i) \tag{2}$$

Within the LSH framework, the Jaccard(u, s_i) similarity can be estimated by the number of overlapping buckets between u and all users s_i in LSH signature matrix. Therefore, Eq. 2 can be efficiently estimated as shown in Eq. 3 where B_u and B_{s_i} are the signature vectors calculated by the function *transformUser* from the Algorithm 1.

$$\text{Score}(u) = \sum_{s_i \in S} |B_u \cap B_{s_i}| \tag{3}$$

We can normalize this score by the total number of buckets. However, since the number of buckets a user falls in is a fixed constant specified through the signature matrix construction in Algorithm 1, we can ignore the normalization factor here. Since in LSH signature matrix construction process, only the users with similarity above a given threshold (i.e. controlled by the banding parameters [23]) are likely to get associated with the same buckets. The users in the same bucket do not have much variance in the Jaccard similarity. Therefore, Eq. 3 effectively counts the number of seeds a user u is similar to.

B. Score by Feature Information Value

The previous method models the user's value for the advertiser using the seed similarities as proxy whereas the similarity metric is calculated within a large dimensional feature space. With the use of the similarity metrics, the above method, inherently, takes into account each feature from that feature space with equal weight. However, in practice, certain features can be more predictive than others. According to this observation, a candidate user u_i should be ranked higher than u_j if u_i exhibits more important features than u_j does.

To quantify the importance of a particular feature f, intuitively, we want to assign higher importance for features that can better distinguish future converters and non-converters. Particularly, the features have higher coverage in converters, but lower coverage in non-converters are preferred. Information value (IV), shown in Eq. 4, is a natural choice to measure feature importance under this intuition. Here p_i, and q_i denote the proportion of positive and negative samples, respectively, with feature $f = 1$, and labels y in a dataset D.

$$\begin{aligned} \mathrm{IV}_D(f, y) &= (p_i - q_i) \log \left(\frac{p_i(1-q_i) + \epsilon}{q_i(1-p_i) + \epsilon} \right) \\ p_i = p_i(f, y) &= \frac{\sum_D \mathbb{1}(y=1)\mathbb{1}(f=1)}{\sum_D \mathbb{1}(y=1)} \\ q_i = q_i(f, y) &= \frac{\sum_D \mathbb{1}(y=0)\mathbb{1}(f=1)}{\sum_D \mathbb{1}(y=0)} \end{aligned} \tag{4}$$

Based on this, we can define our second scoring model referred to as "Score by Feature Information Value" presented in Eq. 5. Here $\mathcal{F}$ is the index set of the features, $x_{u,i} \in \{0, 1\}$ is an indicator variable of being the feature i turned on for user u, and IV as defined above.

$$\mathrm{Score}(u) = \sum_{i \in \mathcal{F}} x_{u,i} \cdot \mathrm{IV}(f_i, y) \tag{5}$$

As a generalization of this scoring component, one can easily apply any metric which measures the correspondence between a given feature f, and the conversion labels y. For example, one can fit a generalized linear model, and use the learned coefficients as weights that measure the strengths of the features. However, we prefer to use the much simpler IV measure, since it can be calculated in a parallel manner and hence allows us to scale up our algorithm, compared to the more sophisticated models probably larger runtime complexities especially within the distributed setting we have. We present results showing the effectiveness of Information Value as a feature selection tool in Section V-C (Table III).

C. Score by Combining Seed Similarity and Feature Information Value

So far we have introduced the scoring method "Scoring by Seed Similarity" which relies completely on the user similarities, and the scoring method "Score by Feature Information Value" which prefers the converters through measuring the information value of the features. However, intuitively, both aspects seem important to derive a balanced scoring component. According to this, here we introduce the "Combined Scoring Model" which, similarly to the idea of collaborative filtering, mixes the benefits of the above mentioned two scoring approaches, while it keeps the calculation scalable. The actual scoring is defined in Eq. 6 for this component.

$$\mathrm{Score}(u) = \sum_{s_i \in S} |B_u \cap B_{s_i}| \cdot \mathrm{IV}(s_i) \tag{6}$$

Here B_q is the set of buckets associated to user $q \in \{u, s_i\}$, while $\mathrm{IV}(s_i)$ is the aggregated feature importance value for the seed user s_i. $\mathrm{IV}(s_i)$ is computed as shown in Eq. 7 where $\mathcal{F}$ is the set of features. The function $\mathrm{IV}(f_k, y)$ applied here is the feature's information value introduced in Eq. 4, while $x_{s_i,j} \in \{0, 1\}$ refers to the jth component of the feature vector associated with the seed user s_i.

$$\mathrm{IV}(s_i) = \sum_{j \in \mathcal{F}} x_{s_i,j} \cdot \mathrm{IV}(f_j, y) \tag{7}$$

The interpretation of the combined model completely aligns with the idea of the collaborative filtering which averages out the ratings of the users with similar taste using the similarities as weights [23]. Similar to this, our combined scoring model computes the weighted average of the feature importance values, as proxies to the conversion, using the seed similarity values as weights.

We have just introduced the fundamentals of our scoring methods, in the following, let us elaborate on the distributed algorithm SCORINGLOOK-ALIKE which provides the glue between these components, and results in our scalable look-alike model with scoring support.

D. SCORINGLOOK-ALIKE*—The Distributed Algorithm*

At the scale we work, the data associated with the user set U is stored within a distributed system. To efficiently build an LSH-based model with the scoring support proposed earlier, we need to express the computation using efficiently the operations provided by the distributed computational model. Algorithm 3 provides the main skeleton of such an implementation using the distributed primitives: *map*, *flatten*, *reduce*, and *zip* which are often provided by the in-memory distributed computational frameworks [11], [12].

The definition of these primitives is as follows. The distributed primitive called *map* is supposed to take a distributed collection as its first parameter, and a function as its second one. Then, it iterates over the distributed collection (parallel execution over the workers of the system) and transforms the individual data units of the collection by applying the function received in the second parameter. The execution of this primitive takes $\mathcal{O}(\frac{TD}{W})$ time assuming that the data size is proportional to T, the execution of the function on a data unit takes time proportional to D, and the number of worker machines in the system is W; and there is no network communication involved except transmitting executable code of the function to the worker machines which is negligible. Due to the parallel execution and minimal network communication, this is a very efficient primitive.

The distributed operation *flatten* takes a structured data unit as parameters, then it decomposes it according to the top level structure locally. That is, if a distributed collection of vectors is passed to this method, which is the case in our use case, it decomposes it to component index, and value pairs; and produces new, less structured distributed collection from the result. Since this operation can be seen as a special case of the *map* primitive which operates on the structure of the data, the execution is fully parallel with the same runtime complexity, and also there is no communication cost involved.

The distributed method *reduce* supports the implementation of key-based aggregation over a distributed collection. That is,

Algorithm 3 ScoringLook-alike—Distributed Model Fitting

```
1: procedure $distributedSignatureMatirx(U; \mathcal{H})$
2:     $X \leftarrow map(U, userFeatures(.))$                  ▷ feature extraction
3:     $B \leftarrow map(X, transformUser(., \mathcal{H}))$    ▷ calculating the LSH-hashes
4:     $B_F \leftarrow flatten(B)$                             ▷ storing as id, value pairs
5:     $M_F \leftarrow map(B_F, swap(.,.))$                    ▷ performing the actual transpose operation
6:     $M \leftarrow reduce(M_F, collectValuesAsSet(.))$       ▷ collecting the users assigned to the same bucket
7:     $Y \leftarrow map(U, pastConversions(.))$               ▷ obtaining the labels
8:     $SI \leftarrow sideInformation(X, Y)$                   ▷ calculating the side information
9:     return $(B, M, SI)$
10: end procedure
```

Algorithm 4 Distributed IV Calculation

```
procedure sideInformation—IV$(X, Y)$
    return $reduce(zip(X, Y), \mathrm{IV}(.,.))$
end procedure
```

Algorithm 5 ScoringLook-alike—User Expansion

```
procedure $expand(S, \mathcal{H}, B, M, IV)$
    $X_S \leftarrow map(S, userFeatures(.))$
    $B_S \leftarrow map(X_S, transformUser(., \mathcal{H}))$   ▷ hashing
    $SC \leftarrow map(zip(B, IV, B_S), score(.))$            ▷ joining
    $E \leftarrow map(SC, filterTopM(.))$                     ▷ collecting
    return $E$
end procedure
```

it receives a distributed collection and an aggregation function, and it performs a network-wise aggregation of the values with the same key by applying the provided aggregation function. The runtime complexity of the method is the same as the complexity of the *map* primitive. However, it involves network communication at the size of the total dataset $\mathcal{O}(TD)$ which is transmitted in parallel. Usually, this is the most expensive primitive.

The operation *zip* vertically aligns two distributed datasets. There are two use cases of this method. In the case of general one, this is an expensive operation, since it needs to perform network communication to find the matching pieces of the data. In this case, both the time and communication complexity are proportional to the size of the larger and the smaller datasets, respectively. However, in our particular case in the Algorithm 4, unlike in Algorithm 5, the raw dataset U from which the two collections X and Y received as parameters of the primitive were generated is the same meaning that the allocation of the data units within this collection over the workers are already aligned. That is, in this case, we do not need to perform any communication through carrying out the operation. The time complexity associated with this use case is $\mathcal{O}(\frac{T}{W})$ where T is the total number of data units, and W is the number of workers like before.

Having the above definition of the distributed computational primitives in hand, it is easy to prove that Algorithm 3, ScoringLook-alike, computes the same model within the distributed setting, as Algorithm 1 does in the centralized computational environment. To see this, one needs to observe that line 3 of the ScoringLook-alike directly maps to the line 4 of the function *signatureMatrix* of the centralized algorithm. While, the section between line 4 and 6 of the Algorithm 3 implements a distributed transpose operation by first exploding the structure of the LSH vectors at the line 4, then swapping the role of keys and values at line 5, and finally collecting the distributed bucket ID, user ID[3] pairs corresponding to the same bucket IDs at line 6. Finally, at line 8, the conversion related side information is calculated. The actual implementation of this function, when the side information consists of the IV measure, can be seen at Algorithm 4. Summarizing the observation of the above analysis, the aggregated time complexity of the proposed method ScoringLook-alike presented at Algorithm 3 is $\mathcal{O}(\frac{TD}{W})$ while the associated communication complexity is $\mathcal{O}(TD)$ assuming the side information can be calculated using the above mentioned distributed primitives constant times.

It is important to note that, this guarantee does not necessary hold if one applies a more complex side information calculation method. That is, if one applies the generalized linear model-based feature importance measure mentioned at the end of the Section IV-B, the resulted algorithm will be certainly out of this complexity range since the model fitting algorithm of the generalized linear model, the often used distributed stochastic gradient descent method, needs non-constant number of iterations to converge.

Having the distributed representation of the model, the implementation of the user expansion function along with the scoring components proposed in Section IV-C is provided at Algorithm 5. The implementation involves a join operation implemented by using the general use case of the distributed operation *zip* which dominates both the time and communication complexities leading to time complexity $\mathcal{O}(\frac{TD}{W})$ and communication complexity $\mathcal{O}(ND)$. We provide the distributed user expansion algorithm only for the scoring component presented in Section IV-C. Based on this the remaining algorithms can be derived easily.

V. Experimental Evaluation

Here we summarize the results of the experiments we performed to evaluate the proposed algorithm ScoringLook-alike by comparing it to multiple baseline algorithms on various large-scale, real-world databases collected from currently running campaign logs of the *Yahoo!*'s advertisement system.

A. Scoring Methods

Let us first define the set of baseline methods we applied through the evaluation.

`O2O` is our simplest baseline method. The corresponding Score function assigns uniform random value independently to each user from the look-alike candidate audience set after executing the look-alike algorithm proposed in [2] which does not support scoring. Those candidate users have Jaccard similarity to seeds users above the threshold configured in the LSH used in look-alike algorithm. The corresponding scoring function is shown in Eq. 8.

[3] Anonymized user IDs.

TABLE I
CONVERSION RATE COMPARISON OF DIFFERENT TARGETING METHODS

Targeting Method	Conversion Rate Normalized by No-targeting
No Targeting	100%
Demographics	105%
Interests	161%
Re-targeting	188%
Look-alike Audience	206%

$$\text{Score}(u) = \text{Uniform(0,1)} \tag{8}$$

M2O, or "Scoring by Seed Similarity" as it was referred to earlier, is our next baseline. This method follows exactly the scoring proposed in Section IV-A. Though, we consider this scoring component as part of our contribution, during the evaluation we consider it as a baseline method since it can be implemented as a natural extension of pairwise similarity-based look-alike models proposed by e.g. [2]. The exact scoring function is presented earlier in Eq. 3.

All the other methods proposed in the paper are considered as non-baseline methods. Through the evaluation, the algorithm applying the scoring component "Score by Feature Information Value" presented in Section IV-B within the algorithm SCORINGLOOK-ALIKE is referred to as M2M; while the scoring component "Score by Combining Seed Similarity and Feature Information Value" proposed in Section IV-C is referred to as MM2M. The corresponding scoring functions are presented earlier in Eq. 5, and Eq. 6, respectively.

B. Evaluation Methodology

We performed the evaluation of the models by considering conversion logs from the advertisement system corresponding to a certain time interval around a time moment t from the recent past. Throughout the evaluation, we assumed that the past converters of advertisement campaigns, that is the ones converted before the time moment t (a period defined by look back window size, W_b), are the seed users for those particular campaigns. Then, we separately generated the look-alike audiences applying the different algorithms and evaluated them based on how many true converters observed after the time moment t (a period defined by look ahead window size W_f) were included in the recommended audiences. Note that the main drawback of this evaluation technique is that, without actually applying the models in the production system on running campaigns, we do not know which and how many users of the recommended audiences would be truly available for ad serving component within the live system. That is, the presented evaluation results can be considered as somewhat biased here. However, according to our preliminary observations on look-alike modeling, the provided results can be considered as reasonable estimations of the unbiased ones, which are quite expensive since they consume real campaign budgets.

Given a particular campaign c, we followed the exact methodology specified below to perform the evaluation of a given look-alike model:

- Seed user set generation: We selected the evaluation time point t, then took all converters from the time interval $[t{-}W_b, t]$ as seed user set S.
- Candidate user pool generation: We considered all the users, U, reached by the ad serving platform in the time interval $[t{-}W_f, t]$.
- Prediction: We executed the model building and the audience expansion algorithms of the given look-alike model, then, based on the scoring, we recommend the top-M user as the recommended audience set R.
- Evaluation: We took the C set of real converters of the campaign c observed in the time interval $(t, t{+}W_f)$, and computed the evaluation metric specified below to measure the performance of the given model.

TABLE II
PERCENT CAMPAIGNS WHICH HAVE LONG TAIL OVERLAP

Audience Size	10M	20M	30M
Percent Campaigns with Long Tail Overlap	63%	63%	68%

In our evaluations, we varied the value of M to evaluate the model performances for different size of recommended audience. We expect a performance decay as M gets larger.

As a performance evaluation metric, we applied the ratio of the true converters in the recommended audience as it is shown in Eq. 9. Here C is the set of unique true converters, while R is the set of unique audience recommended as defined above.

$$CR(R) = \frac{|R \cap C|}{|R|} \tag{9}$$

To perform the evaluation on multiple campaigns, we generated independently look-alike audience for each campaign, then computed their corresponding performance metric separately. Finally, we calculated the average of those performance scores corresponding to the same recommended audience size.

Regarding the exact dataset we used, we selected one particular day in March 2016 as the time point t to generate training and control datasets. Then, we sampled 200 campaigns from ad servers randomly for evaluation, whose campaign goals are to get conversions as many as possible while meeting their cost-per-conversion goal. Converters from this data according to the above-defined methodology with look back window size W_b=*7 days* are collected as seed user set for each campaign, while billions of users seen by ad servers corresponding to the same time window were gathered as candidate users. Finally, the true converters according to the methodology with the look ahead window size W_f=*7 days* are grabbed to form the ground-truth converters C.

C. Analysis of the Performance

In a wider picture of online advertising, there are multiple choices of how to target a campaign, some common choices are no-targeting (i.e., targeting everyone), demographics based targeting (i.e., targeting by user age and/or gender), interests (i.e., users who are interested in specific topics online), re-targeting (i.e., users who went to the advertiser's web pages in the past), and look-alike audience (i.e., the targeting method proposed in this paper). To give a feeling of the conversion performance of those targeting methods, Table I illustrates the relative performance over no-targeting from 20 online campaigns (with different targeting methods) of one advertiser. These campaigns ran at the same time over 30 days. As we can see, no-targeting performs the worst as expected, and interests based targeting works much better than demographics based targeting since it considers users fine granular intentions. Re-targeting works better than traditional targeting methods. Look-alike audience can outperform all of them regarding conversion rate. Using look-alike, an ad campaign can reach more audience than re-targeting and generate more ad values for both advertiser and ad platform.

In Figure 1, we show the main performance comparisons of the proposed scoring methods against a baseline generated by randomly selected audience from the entire user pool,

TABLE III
FEATURES WITH TOP-IVS FOR PREDICTING AGE RANGE

Young	Working-Age	Older
PlayStation 4	Kate Spade	Internet Explorer 9
Xbox One	Salman Khan	Cascading Style Sheets
Fallout 4	Batman vs Superman	Internet Explorer 8
Xbox 360	Liberty Mutual	Myocardial Infarction
PlayStation 3	Marvel Comics	Warning Sign

referred as Random. The CR performance is reported as a function of various generated audience sizes on 465 real ad campaigns. As expected, the Random has the lowest CR overall recommendation algorithms. Even the O2O is about 4x outperformed to Random, which proves the power of look-alike recommendation.

A better scoring method within recommendations can significantly lift the performance in the top millions of audience in comparison to a simple O2O scoring. The performance gaps (3x-10x) between the result of O2O and the proposed M* scoring methods at the moderate audience size regions indicate that the M* scoring functions capture important aspects as the higher-ranked users convert more often than the lower-ranked ones. The CR performance from all M* scoring algorithms monotonically decrease as the audience size increases, and finally, converge to the O2O performance. This gives an advertiser a flexibility to focus its ad dollars only on the desired size of recommended audience with the highest performance.

Although M2O does not have the best performance in all proposed scoring models, it still has significant performance lifts (3x-8x) in comparison to O2O. This result indicates that users-2-users similarity is a strong signal in look-alike recommendation even without any further knowledge on the predictive power of user features. M2O has an advantage on complexity by only considering the number of seeds a user is similar to, without learning any models on user features. This property is a strong advantage in many applications where recommendation needs to be done with tight constraints on memory usage and response time interval. However, a disadvantage of this model is that for some campaigns, many users may only share 1-3 seeds in the seed set. This long tail becomes a problem as these users all have the same model score. Table II shows that a large percent of campaigns have an overlap with these users which is undesirable as we cannot distinguish the quality users within the tail.

By estimating the prediction power of user features, M2M further improve the M2O performance by 30% at the smallest evaluated audience size. This additional performance lift comes from better knowledge on user features. For an ad campaign, some features have predictive power than the others. Users who have more predictive features are ranked higher than the ones lack of these good features.

We chose IV as a measure of the predictive power of a feature not only because it is naive and easy-to-learn, but also because it is a good approximation for the seed-count when the number of converters and the number of users triggered by each feature is very very small in comparison to the total number of users. Table III shows an example of the features with top IVs for predicting users' age ranges. Younger users are interested in video game related content; Working-age users are interested in movies and insurance; Older users concern more about life and health. These top-IV ranked features in the example align very well with common senses.

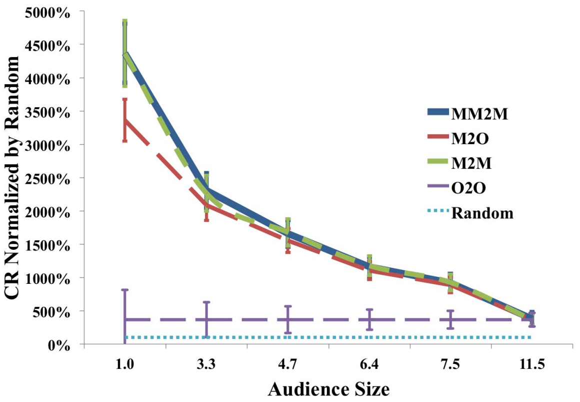

Fig. 1. Performance comparison of the proposed scoring algorithms within top millions of audiences. The X-axis (Y-axis) is scaled by the size (CR) of the Random with the smallest audience size to protect the business sensitive information.

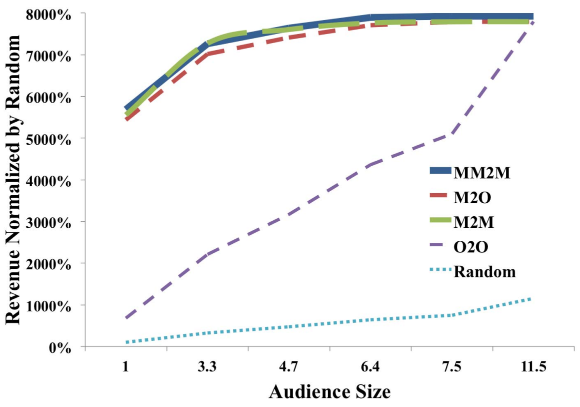

Fig. 2. Revenue comparison of the proposed scoring algorithms. The X-axis (Y-axis) is scaled by the size (revenue) of the Random with the smallest audience size to protect the business sensitive information. X-axis buckets are not linearly aligned.

In Figure 1, the performance of M2M is consistently higher than M2O, and empirically proves that the use of IV, in fact, improves the performance of scoring as well further justifying its effectiveness. The performance gap between M2M and M2O becomes marginal as the recommended audience size increases indicating the feature power is limited in recommending the very top audience only. Seeds are still the most critical factor for recommending large audience. Other than IV, we also tried many other naive models and generalized linear models. The performance of these models is on-par with M2M. Though, M2M has the lowest computational complexity among these.

MM2M, a combination of similar seed-count with feature importance, also performs on-par with M2M. An advantage of MM2M over M2M is that it is considerably more efficient to compute IV values only for seed users than for all candidate users. In practice, the computation saving achieved by MM2M is very significant as the average number of seeds is on the scale of tens of thousands per campaign, but look-alike candidate audience size is on the scale of tens of million.

For some campaigns, advertisers set their dollar values for users' conversions[4], regarding cost-per-action (CPA). If we assume advertisers pay the advertisement platform by the total value of conversions generated, we can measure the monetization impact of different algorithms by estimating the revenue as *Number of conversions* × *CPA*. Figure 2 illustrates the projected revenue number with fixed CPA for the proposed algorithms. At each audience size point, the revenue of each algorithm is the sum of revenues generated by all tested campaigns. The improvements generated by the scoring algorithms directly lead to more monetization dollars in online advertising. We can see in the figure that all the three M* algorithms can generate more than 60x revenue of the Random baseline. Although M2M and MM2M show only about 5% lift over M2O, considering the revenue base in the advertisement platform, the impact is significant.

VI. Conclusions

Look-alike modeling is an interesting, new direction in the field of recommender systems. It extends the notion of traditional recommender systems to model a many-to-many type of relationship between the input user set and output, recommended user set where both of these sets can be extremely large. It raises new challenges such as the ability to efficiently compute the expanded audiences at a massive scale where the universe could be billions of users. In this paper, we illustrated the power of many-to-many scoring in large scale look-alike recommendations with 3 M* algorithms. Through empirical analysis, we show that M2O type of scoring has about 3x-8x performance improvements over the O2O type of scoring. By learning the power of features, M2M type of scoring can achieve a 30% lift as opposed to using only the user pairwise similarity model to an advertiser and user pairwise similarity, we can further achieve a 30% lift vs. M2O. Indeed, a hybrid scoring model which incorporates both the pairwise similarities of users and a score which measures the potential value of the user to the advertiser performs significantly better than purely relying on pairwise similarities. This empirically validates the intuition that a good look-alike should balance the implicit signals captured through pairwise similarities with the explicit signals which are important to a particular advertiser (which is captured through the feature information value) to be truly successful.

We also discuss how such models can be implemented at scale using the state of the art methods in distributed computing, and share our experiences in constructing such a look-alike system in a production setting that is operating at a large scale. With an enormous universe of users and a large number of advertisers, having a system which runs on a distributed computational framework so that the algorithm can be efficiently parallelized is crucial or else the problem is effectively intractable. For computational efficiency reasons, MM2M type of scoring is more practical as it can achieve similar performance as M2M with 1000x fewer computations.

References

[1] J. Shen, S. C. Geyik, and A. Dasdan, "Effective audience extension in online advertising," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: ACM, 2015, pp. 2099–2108. [Online]. Available: http://doi.acm.org/10.1145/2783258.2788603

[2] Q. Ma, M. Wen, Z. Xia, and D. Chen, "A sub-linear, massive-scale look-alike audience extension system," in *Proceedings of the 5th International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, 2016.

[3] D. Chakrabarti, D. Agarwal, and V. Josifovski, "Contextual advertising by combining relevance with click feedback," in *Proceedings of the 17th International Conference on World Wide Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 417–426. [Online]. Available: http://doi.acm.org/10.1145/1367497.1367554

[4] H. Yang, R. Ormandi, H.-Y. Tsao, and Q. Lu, "Estimating rates of rare events through a multidimensional dynamic hierarchical bayesian framework," *Applied Stochastic Models in Business and Industry*, pp. n/a–n/a, 2016, asmb.2150. [Online]. Available: http://dx.doi.org/10.1002/asmb.2150

[5] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 128–131, 2008.

[6] Reuters, "Trusignal & bluekai release new high-value lookalike audiences for prospecting & branding," http://www.reuters.com/article/mn-trusignal-bluekai-idUSnBwTr26ka+a0+BSW20130313, March 2013.

[7] 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*. ACM, 2011, pp. 85–86.

[8] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, ser. MSST '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/MSST.2010.5496972

[9] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[10] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, p. 4, 2008.

[11] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[12] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863103.1863113

[13] M. Armbrust, T. Das, A. Davidson, A. Ghodsi, A. Or, J. Rosen, I. Stoica, P. Wendell, R. Xin, and M. Zaharia, "Scaling spark in the real world: Performance and usability," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1840–1843, Aug. 2015. [Online]. Available: http://dx.doi.org/10.14778/2824032.2824080

[14] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke, "The stratosphere platform for big data analytics," *The VLDB Journal*, vol. 23, no. 6, pp. 939–964, Dec. 2014. [Online]. Available: http://dx.doi.org/10.1007/s00778-014-0357-y

[15] Y. Chen, H. Chen, A. Gorkhali, Y. Lu, Y. Ma, and L. Li, "Big data analytics and big data science: a survey," *Journal of Management Analytics*, vol. 3, no. 1, pp. 1–42, 2016.

[16] Y. Qu, J. Wang, Y. Sun, and H. Holtan, "Systems and methods for generating expanded user segments," February 2014, uS Patent 8,655,695. [Online]. Available: https://www.google.com/patents/US8655695

[17] O. Chum, J. Philbin, A. Zisserman *et al.*, "Near duplicate image detection: min-hash and tf-idf weighting." in *BMVC*, vol. 810, 2008, pp. 812–815.

[18] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2130–2137.

[19] S. Gollapudi and R. Panigrahy, "Exploiting asymmetry in hierarchical topic extraction," in *Proceedings of the 15th ACM international conference on Information and knowledge management*. ACM, 2006, pp. 475–482.

[20] T. Haveliwala, A. Gionis, and P. Indyk, "Scalable techniques for clustering the web," 2000.

[21] J. Buhler, "Efficient large-scale sequence comparison by locality-sensitive hashing," *Bioinformatics*, vol. 17, no. 5, pp. 419–428, 2001.

[22] B. Matei, Y. Shan, H. S. Sawhney, Y. Tan, R. Kumar, D. Huber, and M. Hebert, "Rapid object indexing using locality sensitive hashing and joint 3d-signature space estimation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 7, pp. 1111–1126, 2006.

[23] A. Rajaraman, J. D. Ullman, J. D. Ullman, and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press Cambridge, 2012, vol. 77.

[24] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '02. New York, NY, USA: ACM, 2002, pp. 380–388. [Online]. Available: http://doi.acm.org/10.1145/509907.509965

[4] Conversions can refer to purchases, subscription to advertiser's services, etc.