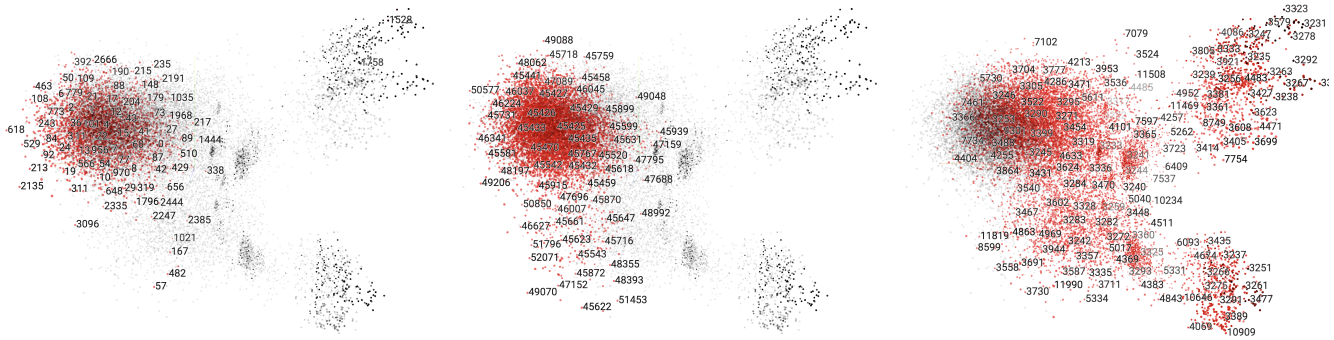
Finding Users Who Act Alike: Transfer Learning for Expanding Advertiser Audiences

Stephanie deWet*
sdewet@pinterest.com
Pinterest
Seattle, WA

Jiafan Ou*
jiafanou@pinterest.com
Pinterest
Seattle, WA



(a) Seed Users (b) Expanded Audience (c) Random Users (aka Background)

Figure 1: Three views of the same embedding space, each with a different group highlighted in red. Taken together, these views show an expanded audience generated based on a seed list of users, taking into account the background distribution of all users.

ABSTRACT

Audience Look-alike Targeting is an online advertising technique in which an advertiser specifies a set of seed customers and tasks the advertising platform with finding an expanded audience of similar users. We will describe a two-stage embedding-based audience expansion model that is deployed in production at Pinterest. For the first stage we trained a global user embedding model on sitewide user activity logs. In the second stage, we use transfer learning and statistical techniques to create lightweight seed list representations in the embedding space for each advertiser. We create a (user, seed list) affinity scoring function that makes use of these lightweight advertiser representations. We describe the end-to-end system that computes and serves this model at scale. Finally, we propose an ensemble approach that combines single-advertiser classifiers with the embedding-based technique. We show offline evaluation and online experiments to prove that the expanded audience generated by the ensemble model has the best results for all seed list sizes.

*Authors contributed equally to this research.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330714

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; **Neural networks**.

KEYWORDS

neural networks; embeddings; transfer learning; lookalike advertising

ACM Reference Format:
Stephanie deWet and Jiafan Ou. 2019. Finding Users Who Act Alike: Transfer Learning for Expanding Advertiser Audiences. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330714

1 INTRODUCTION

Advertisers expand the reach of their products by communicating with potential customers. With the rise of online advertising, advertisers are able to closely specify their target audience using a number of targeting solutions. This includes the ability to show ads to specific demographic groups, people who are searching for specific keywords, or users who have expressed prior interest in specific topics. For each of these types of ads, the advertiser specifies some 'targeting criteria' based on her own understanding of her customers. This approach provides the advertiser more control, but does not leverage the unique knowledge that the advertising platform has about its users.

Audience expansion (or 'look-alike') targeting is a newer technique in which the advertiser provides the ads platform a seed list of her customers, and the platform uses this list to determine a large

expanded audience of similar users. In this technique, the advertiser does not specify any targeting criteria beyond the seed user list and desired audience size. This reduces the number of variables the advertiser needs to control, and also allows the advertiser to leverage the advertising platform's knowledge of its users and their interests.

Building a system at scale that can provide audience expansion for any advertiser presents several different challenges.

- **Variable Seed List Sizes.** Advertiser seed lists can vary drastically in size, from a few hundred users up to tens of millions of users. The model should be able to handle either well.
- **Scalability.** The system needs to be able to scale up to handle hundreds of millions of users and advertisers.
- **Responsiveness.** The system should minimize the time it takes to onboard a new advertiser seed list or a new user.
- **Real-World Performance.** The final output of the system is ads which are served to users. We should be able to measure how well users are matched to advertisers by measuring overall user engagement metrics such as number of clicks, number of ads viewed, etc.
- **Advertiser Data Isolation.** The advertising platform is responsible for ensuring that information from one advertiser's seed list is not used in any way to help other advertisers. In practical terms, this means that seed lists cannot be used as training data in any model shared between advertisers.

1.1 Current work

At Pinterest, we have built an audience expansion product called Act-alikes, which is tasked with finding the set of users who are most similar to a set of seed users. We consider users to be similar if they are likely to interact with the same content on Pinterest.

We train a classifier model for each advertiser's seed list. This approach to the audience expansion task is described by Qu et al. [11] amongst others. The classifier models perform well for seed lists that are sufficiently large in size, but they do not perform well for smaller seed lists. This deficiency motivates us to take a new modeling approach.

Taking a step back, we recognize that the vast majority of content that users see on Pinterest is 'organic' (i.e. not paid content). On Pinterest, paid and organic content are similar in appearance and subject matter. Thus it is reasonable to assume that two users who are interested in the same organic content will also be interested in the same type of paid content. With this assumption in place, we apply the principles of transfer learning to build a model of user similarity based on all of a user's interactions with our entire corpus of content, then apply this model to specific advertisers. More specifically, we break the modeling problem into two parts: (a) building a universal model of user-user similarity based on organic data, to be shared between all advertisers, and (b) representing each advertiser's seed list of users in a way that builds from our understanding of user-user similarity.

This design decision helps us to meet several of the challenges described above. By sharing the universal user-similarity model amongst all advertisers, the second task of building a light-weight model of advertiser seed lists can be done using computationally cheap methods that provide meaningful results even for small seed lists. This gives us scalability and the ability to handle many sizes of seed list.

The literature around word-embeddings (word2vec [8], GloVe [9]) has shown that embeddings can be trained so that dot-products and distances in a low-dimensional embedding space capture similarity and relationships. Therefore, we choose to model users using an embedding approach, with the dot-product between users indicating how similar two users are. We generate the embedding from a collection of user features, not an id. This is crucial because it allows us to immediately compute embeddings for brand new users that the model hasn't seen before. It also allows us to retrain the model infrequently, while updating the users' embedding vectors more frequently as their feature vectors change.

We introduce advertiser information by building a lightweight representation that summarizes the advertiser's seed list users. This is fast to compute, so new advertisers can be brought online quickly. The representation is built on top of a rich embedding model, which means that even if the seed list is small, we are still able to extract meaningful information from it. Each advertisers' representation is only informed by her own data and the user similarity model trained on organic data, thus ensuring that there is no leakage of information between advertisers.

As will be described further in Section 5, the classifier and embedding-based models are complimentary. We combine the two approaches to create an ensemble model that significantly outperforms its components.

1.2 Unique Contributions to Look-alike Advertising

Our approach introduces the following novel contributions to the field of look-alike advertising:

- Modeling users in an embedding space, yielding user embeddings and a user-user similarity metric.
- Using transfer learning to apply a model trained on our full historical user activity logs to specific advertisers.
- Using a lightweight feature selection step to quickly onboard new advertisers.
- Combining an embedding-based model with a single-advertiser classifier approach to create an ensemble model.

Section 2 of this paper discusses related work. Section 3 describes our model in greater detail. Section 4 describes the end to end system and discusses other details of productionizing this model in an industry setting. Section 5 contains results of offline and online experiments. Section 6 contains conclusions and descriptions of future work.

2 RELATED WORK

Audience expansion is used by many modern advertising platforms. Look-alike audiences have become one of the major advertising methods used in the industry. However, there is still little work on this topic published in literature as of today. Classification is a commonly used machine learning approach to identify additional audience members for the advertisers [11]. We developed a classifier-based system, described further in section 3.1.

The classification-based approach targets directly on conversion (desired actions resulting from ad view), so it may ignore seed user characteristics. Shen et al. [12] and Mangalampalli et al. [7] learned the importance of user features and used them to filter candidates. Liu et al. [3] performed similarity-based audience expansion, but this was mostly based on certain attributes or categories, not the users themselves. Ma et al. [6] proposed a two-phase technique consisting of representing users as a feature vector to reduce candidates, and then learning feature importance or a classifier for each advertiser. Ma et al. [5] further proposed scoring potential audience members by seed similarity and feature importance. LSH [1] is normally applied to reduce the computation cost associated with detecting similar users [4–6].

Many approaches have the downside that a different model is trained for each advertiser, solely using seed list data and raw user features. If the size of the seed list is small, the model will not be trained on much data and it probably will not perform well. Additionally, managing models for every advertiser or campaign is expensive to scale up.

Embeddings models represent documents in a vector space with a meaningful distance metric. Work on neural embeddings learned in an unsupervised way include word2vec ([8]) and GloVe ([9]). StarSpace ([14]) proposed a general-purpose approach to embed different kinds of documents in the same space, in such a way that comparisons are meaningful. As far as we know, embedding models have not been applied to the problem of audience expansion.

In the following section we propose a transfer learning ([10]) based approach, which will learn a user embedding based on a large amount of data, and use statistical information to describe the seed lists and expand the audience. We will demonstrate in experiments in section 5 that this method outperforms a classifier-based baseline for small seeds, and that an ensemble of embedding- and classifier-based models performs best for all seed list sizes.

3 ENSEMBLE ACT-ALIKE MODEL

In this section we present the component models of our ensemble act-alike model. Section 3.1 describes a model consisting of advertiser-specific classifiers. In Sections 3.2 and 3.3, we present a novel model which combines a learned user embedding and approximate retrieval techniques and is the main focus of this work.

3.1 Classifier-based Approach

Our baseline model uses a separate logistic regression-based classifier for each advertiser. Let S be the set of seed list users. We label the $|S|$ seed list users as positive examples, and randomly select $|S|$ non-seed users to use as negative examples, so that there are the same number of examples from each class. Each user has a feature vector comprised of continuous features and one-hot representations of discrete features. We then train a logistic-regression based classifier for each seed list. We score every user with the classifier, sort, and take the n highest-scoring users for the advertiser's expanded audience.

This is an expensive training process when it is scaled up to the number of advertisers we typically see in industry. It also requires complicated verification to ensure that the models are running well. The classifier performance is highly dependent on the size of the seed list: this model performs much better for large seed lists than small ones, as we will see in Section 5.

In the remainder of Section 3, we will describe an alternative approach which does not require per-advertiser training. In Section 4, we will describe an ensemble model which combines the two approaches.

3.2 Global User Embedding Model

Pinterest is a visual discovery engine which allows users to search or browse for 'Pins'. Pins are pieces of content on Pinterest, each consisting of an image, link, text description, and topic labels. Users can interact with Pins by saving, clicking, viewing more details, etc. At Pinterest, we have extensive historical logs for each user containing information about the Pins they have seen and interacted with. We will use this information to build a universal user model.

We are inspired by the StarSpace technique ([14]) to learn a neural model which embeds users and Pin topics into the same vector space. The model is designed to learn that users are similar to topics they have engaged with in the past. It should also learn that two users who engage with similar topics are similar.

Note that this dataset and model are very different from the advertiser seed list dataset that is the end goal. There are a few reasons that we decided to try this approach.

- Advertiser seed lists cannot be shared. If we directly use advertiser seed list data for training, we will have to learn a separate model for each advertiser.
- Data quantity. Pinterest's dataset consists of more than 250M monthly users who have collectively saved over 175B Pins. Directly using these historical logs allows our model access to a vast amount of information about users' interests. Users may not have viewed content directly from a specific advertiser in the past, but using an embedding model allows us to infer how much they will like an ad, based on similarity with other content they have seen and enjoyed.
- Similarity in the embedding space is tied to engagement. The embedding model is trained to learn that users are similar to the topics they engage with. This definition of similarity is closely aligned with most advertisers' goal of convincing the user to click on or view an ad.

3.2.1 User Representation. Users are represented by a set of discrete and continuous features. Examples of discrete features are user demographic attributes, and user interested topics, etc. Each discrete feature is drawn from a fixed-length dictionary of possible values. These values are mapped to a set of continuous integer id values, for which we will learn an embedding. Examples of continuous features include historical activity rates (such as click-through rate) and raw counts (such as Pin impressions). Their values are normalized and used directly. As noted in [14] representing users as the combination of their features allows new users to be naturally incorporated as they join the platform.

Some discrete features, such as user interests, are of variable length. We apply a pooling layer to aggregate the embeddings of these features. Then we concatenate all of the discrete features' embeddings and the normalized continuous feature values together into one vector. A dense layer with linear activation function is applied to this vector to enable mixing between the different user

Figure 2: Query user followed by the most similar users, found using nearest neighbor search.

features. The output of this dense layer is our user representation, also referred to as the user embedding (u).

3.2.2 Topic Representation. Pinterest maintains a large dictionary of topics which are used to describe content on the site. Each Pin is mapped to a collection of topics from this dictionary, which describe the Pin's image and associated text. If a Pin is associated with multiple topics, we split the Pin's topics up into separate training examples. We featurize each topic with a learned embedding vector that is the same length as the final user embedding. This is referred to as the topic embedding (t).

3.2.3 Training Data. Positive training examples are drawn from the historical activity logs and consist of users and the Pins with which they interacted. Each Pin is represented by a collection of topics, and then split so that there is one training data entry for each user and Pin topic pair. This creates many positive training examples of the form (u, t^+). If a user interacts with a Pin, it indicates some degree of interest in one or more of the Pins' topics. Thus, our training data consists of noisy pairs of users and the topics they are interested in. We batch together a group of n of these positive examples to create one batch update.

Our intuition is that users should be significantly more interested in topics they've engaged with than in other randomly selected topics, so counter-examples are drawn from the background distribution of all topics. We use a k-negative sampling strategy ([4]) to draw a set of k topics from a log-uniform distribution of all topics for each batch update. For each positive training example, the positive topic t^+ should be more similar to the user u than any of the k background topics t^-. So this gives us a set of k (u, t^-) negative examples to compare against each of the n positive training examples.

Training the model requires comparing users and topics. We implement the loss used in StarSpace ([14]), choosing the inner product as our similarity function and a margin ranking loss. We weight each of the positive and negative pairs equally, and average their hinge losses across the batch. The model minimizes the following loss function:

$$Loss = \frac{1}{k * n} \sum_{i=1}^{n} \sum_{j=1}^{k} \max(0, 1 - (\langle u_i, t_k^+ \rangle - \langle u_i, t_k^- \rangle)) \quad (1)$$

Figure 2 demonstrates the model's output when trained on the Pinterest dataset. It shows a query user and her nearest neighbors in the embedding space. The query user and all of her nearest neighbors are female English-speakers from the US who are interested in weddings. All of these users have also created 'boards' (collections of saved content) dedicated specifically to weddings. The model has successfully ensured that these users are nearby in embedding space.

3.2.4 Stopping criteria. The model is trained on multiple CPUs using the Adagrad optimizer. Although the model is trained to find similarity between a user and a piece of content, it will be applied to the transfer problem of user-to-advertiser similarity. While training the model we periodically evaluated its performance on the transfer

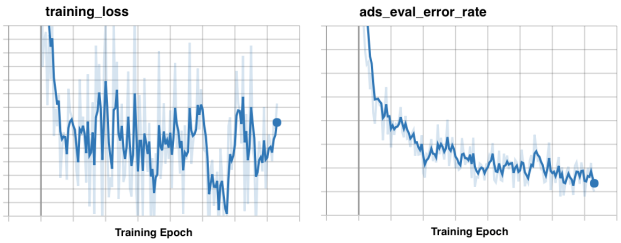
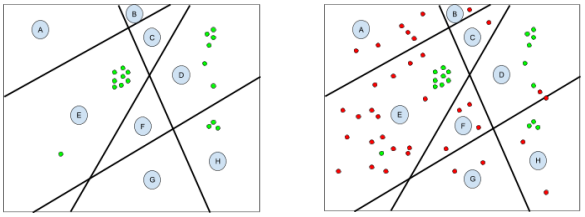


Figure 3: Tensorboard output of model training.



(a) Example for simple voting **(b) Example for density voting**

Figure 4: Example partitioned user space with seed list users (green) and non-seed list users (red)

task (without training on these tasks). We used this transfer task evaluation as a stopping criterion.

In formulating the transfer task evaluation dataset, we selected a seed list at random. We first split the entire set of Pinterest users into two groups - seed users and non-seed users. We reserve 90% of the seed users in a set of known seed users, K. We pair the remaining seed users with non-seed users to build test pairs which consist of one held-out seed user u_s and one non-seed user u_n each. For each evaluation pair, we try to predict which user came from the seed by computing each user u's similarity to the entire set of known seed users. We will denote this similarity as $S_K(u)$:

$$S_K(u) = \sum_{k \in K} \langle u, k \rangle \tag{2}$$

The evaluation succeeds for the example if $S_K(u_s) > S_K(u_n)$. The total evaluation error rate on the transfer task is given by:

$$\text{ads\_eval\_error\_rate} = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 1 & \text{if } S_K(u_{s,i}) < S_K(u_{n,i}) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The tensorboard output of model training is shown in Figure 3. From this output, we see that although the training loss is very noisy, ads_eval_error_rate decreases smoothly. We monitor this transfer task error rate and stop training when it is no longer decreasing.

3.2.5 Model Retraining. We deliberately select user features from a set of features whose dictionary of possible values do not change quickly. For example, the user interest feature is drawn from a dictionary that occasionally adds new terms but is mostly static. User embeddings are recomputed regularly, using the most recent version of the embedding model. We expect the set of all users' embeddings to drift slowly as the Pinterest userbase evolves, but we do not expect them to change rapidly on a day-to-day basis. Thus, we retrain the model on a relatively long cadence to capture this drift in user behavior and interests.

3.3 Embedding-Based User-Advertiser Scoring Model

Each advertiser provides a seed list which maps to a set of Pinterest users. In our embedding space, the advertiser is represented by a set of user embedding vectors. Advertisers can appeal to multiple types of users - for example department stores sell many categories of goods. We expect these advertisers' seed lists to correspond to multiple clusters of user vectors in the embedding space. For these advertisers the simplest possible algorithm - averaging all of the seed users' embeddings ($s = \sum_{i=1}^{|S|} u_i/|S|$) - is not sufficient. This algorithm would find users in the centroid of seed user clusters, who may not be similar to any seed users.

We also expect all advertisers to be affected by a 'background distribution' of users. This is the ambient distribution of all Pinterest users in the embedding space. The distribution will naturally have some higher density regions, since some user demographics and interests have larger representation on Pinterest than others. Our website contains more of some types of content than others, and of course we expect users' on-site behaviors to reflect this. In the embedding space this means that the entire sitewide population of users will not be evenly distributed. Instead, we expect to see clusters of users who behave in similar ways on our site.

The model's goal is to find users who are similar to advertisers' seed lists. Intuitively, this task translates into locating regions in the embedding space where the advertisers' seed users are more densely clustered, after taking the background distribution of users into account.

3.3.1 Locality Sensitive Hashing. We base our algorithm on the random projections method of Locality Sensitive Hashing (LSH) ([1]). Following this technique, we randomly select n random hyperplanes of the learned user embedding space, each defined by the unit vector v_i. These hyperplanes partition the space into 2^n disjoint regions, with each user falling into exactly one region. Two users u_a and u_b would fall into the same region if and only if $\text{sign}(v_i * u_a) = \text{sign}(v_i * u_b)$ for all i. This partitioning does not take the vector magnitudes into account. We would intuitively expect that if the angle between u_a and u_b is small, then the users are more likely to fall into the same region. And indeed, by integrating over all possible hyperplanes, it can be shown that the probability the two users fall into the same region is: $1 - \frac{\theta(u_a, u_b)}{\pi}$ where $\theta(u_a, u_b)$ is the angle between the two user vectors.

3.3.2 Voting Score. After partitioning the embedding space, each user u is assigned to a region r_u. We can now represent an advertiser seed list s by user voting: For each of the 2^n disjoint regions, we count the number of seed users in each region, $c_{s,i}$. This gives us a representation for advertiser seed list s of $(c_{s,1}, c_{s,2}, ..., c_{s,2^n})$ where $c_{s,i} \in [0, |S|]$ and $|S|$ is the number of users in seed list s. We assign each user a score equal to the number of seed users in her region, i.e. $\text{score}(u, s) = c_{s,r_u}$. In this representation, regions with a large

count are more popular. If the user falls into a popular region, she is more likely to be selected in the expansion. An example partitioned user space is shown in Figure 4a. Using the voting score method, the highest scoring regions are E, D, then H.

Table 1: Scores for users by a region. In Simple Voting, the best regions are E, then D, then H. By contrast, in Density Voting, the best regions are D, then H, then E.

Region	Voting Score	Density Score $(\alpha, \beta = 0)$	Density Score $(\alpha = 1, \beta = 4)$
A	0	0	0.25
B	0	0	0.25
C	0	0	0.25
D	6	**0.86**	**0.63**
E	**9**	0.33	0.32
F	0	0	0.25
G	0	0	0.25
H	3	0.5	0.4

3.3.3 Density Score. The voting procedure ignores the distribution of non-seed users in the embedding space. In the real world, some user behaviors and interests are more popular than others - for example, there could be many more users who interact with recipes than with Applied Data Science papers. Our implementation of LSH with random projections did not balance the number of users in each region. Thus we need to take the background distribution of all Pinterest users into account.

We count the number of total users in each region, $c_{b,i}$. We compute a region density score $d_s(r_i)$ for seed list s and region r_i in the following manner:

$$d_s(r_i) = \frac{c_{s,i} + \alpha}{c_{b,i} + \beta} \tag{4}$$

where α and β are smoothing factors and $\alpha \leq \beta$. Figure 4b shows the background users for each region in our example space, and Table 1 shows the scores for each region, computed with both methods. If $\alpha, \beta = 0$, after factoring in the background distribution, regions D and H will get higher scores than region E, because they have fewer non-seed users. In Table 1 we also demonstrate other values of α and β, which help to smooth out the scores for regions where we have fewer examples.

We use the dense vector comprised of all region scores for an advertiser as the compact representation of the advertiser's seed list. We set a region's score to be 0 when there are too few seed users or background users in that region.

Finally, we repeat the entire process of determining random projections and voting m times to mitigate the variance brought in by any particular set of projections. The final seed list representation is a tuple of $m * 2^n$ density scores:

$$s = (d_s(r_1), d_s(r_2), ..., d_s(r_{m*2^n})) \tag{5}$$

For a user u who falls into LSH projection regions $R_u = (t_1, t_2, ..., t_m)$ where $t_i \in [(i-1) * (2^n + 1), i * 2^n]$, the user's affinity score for seed list s is given by:

$$\text{AffinityScore}(u, s) = \frac{1}{m} \sum_{r \in R_u} d_s(r) \tag{6}$$

Intuitively, a user who is very similar to many of advertiser's seed users will have large region scores for most of the k partitionings of the space. A user who is not in a cluster of seed users in the embedding space will not have a high region score for most partitionings.

An example of this procedure in action on a real seed list is displayed in Figure 1. This figure displays a seed list of users, the background distribution of users, and the expanded audience generated as a result. We see that the seed list has a large cluster of users on the far left of this image, and that the expanded audience is also localized in this area.

Note that this approach does not require any expensive advertiser-specific learning steps. Starting with the user-embedding space trained on historical user behavior, we are able to build a representation of the advertisers' seed list without any advertiser-specific feature selection or model training. Additionally, many of the steps in the previous procedure are not re-computed for each advertiser. The background distribution is computed once and shared between all advertisers. The random projections are re-selected only when the model is retrained. Thus, this method should be able to scale up to a large number of advertisers.

4 END-TO-END SYSTEM

In this section we will focus on the end-to-end production system for scoring and serving the model results. The system consists of online ads serving (top half of Figure 5) and an offline data pipeline (bottom half of Figure 5). We run the offline pipeline regularly to generate top users for advertisers and serve them online.

4.1 User Embedding and Seed Representation

A regular pipeline computes the users' latest features and infers their embeddings using the learned user embedding model (described in section 3.2). Since users are fully featurized, we do not need to retrain the embedding model often.

We take the seed users' embeddings to calculate the seed representation by equation (5). We store the representation as a dictionary of (region id, score), leaving out all regions whose score was set to zero because there were too few seed users or background users for us to be confident in the score.

4.2 Audience Scoring

Not all registered Pinterest users actively use the website, so it would be wasteful to score our model against all registered users. We limit our pool of eligible users to users who have recently visited Pinterest. We then compute an affinity score for each (user, seed) pair, using Equation (6). The final expanded audience consists of the p% highest scoring users for each seed (where p% is the advertiser's requested audience size, given as a percent of eligible users).

Model scoring can be broken down into many different tasks of scoring (user, seed list) pairs so that we can use MapReduce jobs to accomplish the work above. In practice, both the seed list representations and user models are very large. In the model scoring phase, we need to score each user against all of the seed lists. We can

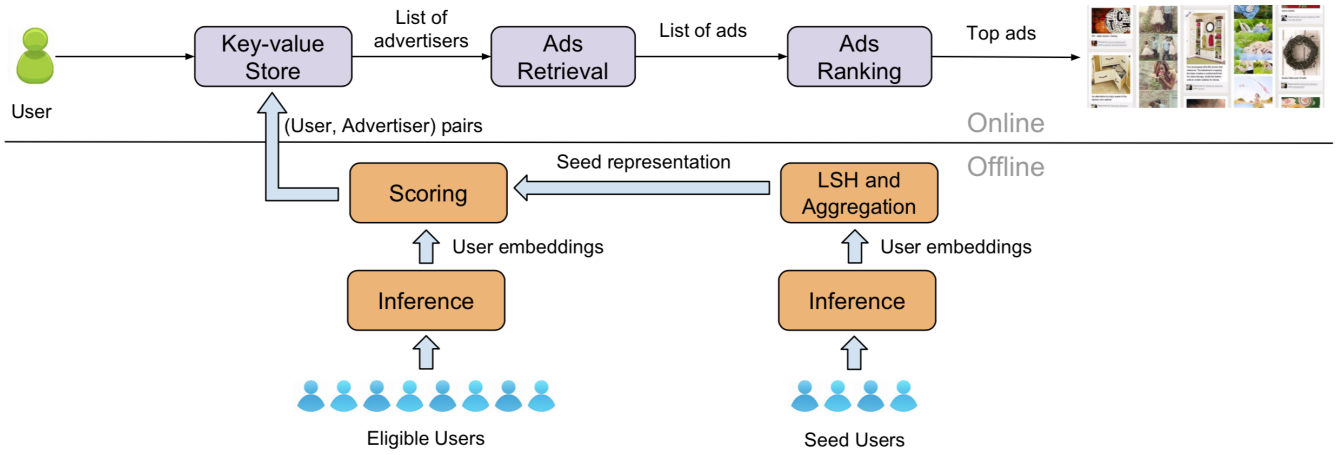


Figure 5: System diagram, consisting of an offline pipeline and online serving system. The offline pipeline computes user embeddings, advertiser representations and expanded audiences. The online serving system maps user requests to candidate ads based on the user's expanded audiences.

split this phase into $|U||T|$ distinct tasks (where $|U|$ is the number of eligible users and $|T|$ is the number of seed lists). We make use of the MapReduce distributed data framework [2] to split this work across a fleet of machines. We distribute the work of taking the cross product of these two datasets using the classic fragment-and-replicate join technique (discussed in context of modern hardware in [13]). The number of records to score scales up linearly with the number of users and advertisers, and finding the top $p\%$ of users for each advertiser scales linearly with the number of users, since we are using a bucket sorting approach. The MapReduce framework can easily handle this sort of linear scaling, so we do not expect to run into any scalability issues in the future.

4.2.1 Sorting. In order to get the highest scoring users for each seed, we have a few options: (i) Sort all scores, then determine the p-percentile cutoff. However, since the total number of eligible users is huge, and normal sorting takes O(NlogN), this would be slow. (ii) Sample the scores, then sort the sample before determining the p-percentile cutoff. However, sampling doesn't do a good job for all advertisers and we need O(N) to do the sampling. (iii) Bucket sort the scores. This method scales the float score to integer values and uses integer buckets to do the sorting in O(N). We chose to use bucket sorting, since it is the fastest.

4.2.2 Blending with Classifier Method. In the experiments detailed in section 5, we find the embedding-based approach performs better than the classifier-based approach for advertisers with small seed lists, but performs slightly worse for advertisers with large seed lists. We also find that the audiences selected by the two approaches vary quite a bit. In our production system we blend the audiences from both sources.

In production, we retrain the advertiser classifiers regularly to take into account updated user features and changes to the seed list, then regenerate the audiences for each advertiser. Similarly, we also recompute user embeddings on the same cadence for the embedding-based models, and regenerate advertiser audiences using the latest user embeddings and seed lists.

First we select the intersection of the two methods' audiences. Then we use a round-robin mechanism to blend the top-scoring candidates from the classifier-based model and the embedding-based model, stopping when we reach the desired audience size. This allows the best candidates of each model type to rise to the top. We select the blending ratio of the two sources depending on seed list size - we select more candidates from the embedding-based model for advertisers with smaller seed lists, and fewer embedding-based candidates for advertisers with larger seed lists.

Results in Section 5 clearly show that this blending technique performs better than either model alone, resulting in significant improvements in offline and online metrics of interest.

4.3 Ads Serving

At this point, we have generated a user list for each advertiser. In order to facilitate online ads serving, we convert the <advertiser, user list> tuple to a <user, advertiser list> tuple, and push this dataset to a key-value store. When a user visits Pinterest, we look up the list of candidate advertisers by user ID. We then fetch the ads associated with these advertisers and merge with the ad candidates generated by other retrieval methods. Finally, we rank all the ads, send the most relevant to a second-price auction, and select the winning ads. The ad Pins are blended with organic Pins and displayed to the users.

5 EXPERIMENTS

We measure the models' performance in offline (section 5.1) and online (section 5.2) experiments.

5.1 Offline Evaluation

We use precision and recall to evaluate how well the models reproduce advertiser seed lists. For the purpose of this evaluation

we consider the seed lists from advertisers as ground truth, but in reality advertiser seed lists are noisy and not all seed list users are representative.

We hold out 10% of the advertiser's seed users (H_s) and sample an equal number of holdout users from the set of non-seed users (H_n). This ensures that we have a balanced number of positive and negative examples. We compute the user-advertiser scoring model using the remaining 90% of seed list users. We then apply this model to score all eligible users, and sort by affinity score. The model must perform well for a range of advertiser-desired audience sizes, so we measure precision and recall at multiple thresholds, counting each held out seed user who scores above the threshold as a success.

$$\text{Precision@N} = \frac{|H_s \cap \{\text{Top N Results}\}|}{|(H_s \cup H_n) \cap \{\text{Top N Results}\}|} \quad (7)$$

$$\text{Recall@N} = \frac{|H_s \cap \{\text{Top N Results}\}|}{|H_s|} \quad (8)$$

The absolute values of precision and recall for our models are proprietary, so all numbers are given relative to a baseline of the classifier model's precision and recall scores. Table 2 shows the models' average precision and recall values for a set of selected advertisers. The embedding model significantly improves on the classifier baseline for both precision and recall, and the blended model improves these metrics even further. Precision and recall are improved at several values of N, indicating that both embedding and blended models improve results for a wide range of advertiser-desired audience sizes. In practice, we focus on improving precision@N, since the seed lists are noisy and many seed users can be outliers.

Table 2: Precision (P) and Recall (R) Relative to Classifier Model

Model	ΔP@100K	ΔP@1M	ΔP@5M
Embedding	+9.67%	+9.55%	+7.31%
Blending	+11.16%	+10.55%	+8.70%
Model	ΔR@100K	ΔR@1M	ΔR@5M
Embedding	+32.71%	+24.69%	+18.41%
Blending	+33.09%	+31.99%	+26.55%

We further slice the precision values by seed list size. Figure 6 shows that the embedding and blending models perform substantially better than the classifier model for smaller seed lists. For larger seed lists, compared to the classifier the embedding model performs significantly worse while the blending model performs nearly as well. Seed list size does not correlate to advertiser size or purchasing power, so it is important to model all sizes of seed lists well.

5.2 Online Evaluation

We used an online A/B experiment to further evaluate the models' performance. We allocated traffic by user ID across all treatments. Our control group was the classifier-based audience expansion model and we had two treatment groups: the embedding-based

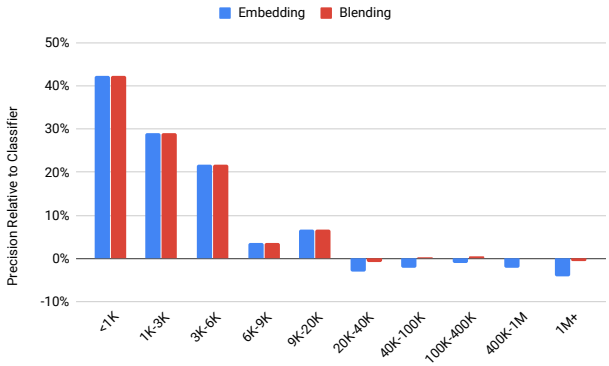


Figure 6: Model Precision Relative to Classifier by Seed Size

models, and the blended model. We evaluated all three models' performance on CPC (Cost-per-click) ads, an important format for which the advertiser's goal is user clicks and our goal is to achieve a high click rate. Our evaluation metric was click-through rate (CTR), a commonly used industry metric.

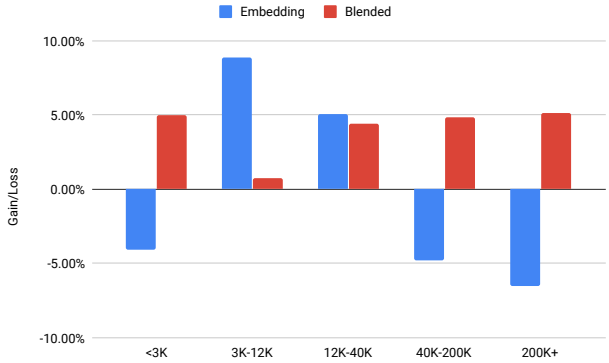


Figure 7: Online CTR by Seed Size for CPC Act-alike Ads (Baseline: Classifier Model)

The results of our online experiment, broken down by seed size, are shown in Figure 7. We utilized only five buckets to minimize dilution of sparse click data. Note that this data is noisy by nature because the ads that are eligible to be shown to users are constantly changing as advertisers change their campaign settings. The embedding model improved CTR relative to the classifier model for two of the five buckets, while the blended model improved CTR for all five buckets. We also aggregated the results across all seed sizes, as shown in table 3. Overall, the embedding-only model decreased CTR compared to the classifier model, but the blended model improved CTR.

The blended classifier and embedding-based act-alike model was shipped into production at Pinterest.

Table 3: Online A/B Test Results (Baseline: Classifier Model)

Treatments	Δ CTR
Embedding Only	-4.1%
Blended	+2.1%

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have described an end-to-end industrial-scale ads look-alike system built around a universal user embedding model and advertiser seed list representations. We used a transfer learning approach to make use of our extensive historical user logs. This approach allowed us to maximize the amount we learned from each user on the seed list, enabling us to model small seed lists substantially better than in classifier-based techniques. This approach also allowed the embedding model to nearly match the precision of the classifier-based techniques for large seed sizes - without the expense of learning advertiser-specific models. Finally, we described how to blend data from classifier- and embedding-based approaches to improve offline and online performance for all seed list sizes.

In the future, we will continue to improve the user embedding model. We intend to incorporate more transfer knowledge by applying multi-task training to the model. We also intend to explore a hybrid approach which will learn a classifier model on top of the user embeddings. We will also explore learning advertiser-specific similarity functions instead of using the dot product function to determine the user-advertiser similarity for all advertisers. Finally, we will continue to add more user features to the model.

Outside of the embedding model, there are also many aspects for us to improve in the advertiser representation and final audience expansion, such as using a more sophisticated LSH partitioning which is aware of region density, aggregating seed list characteristics, improved blending with the classifier model, etc.

Another future direction is to convert the current end-to-end system into a real-time candidate generator. We can compute user embeddings on-the-fly when a user visits the site, use LSH to find the most promising seed-list candidates, and compare the user's score to these seed lists' score thresholds. This will allow us to take the users' most recent actions into consideration, and prevent us from wasting offline computation on non-visiting users.

ACKNOWLEDGMENTS

We want to thank Jian Huang and Jacob Gao from the Pinterest ads team. They actively helped with the experiments and analysis described in this paper, and contributed many ideas including the data blending. They were also closely invovled in paper revision. We also want to thank Mukund Narasimhan for discussions around approximate nearest neighbor techniques, Jinfeng Zhuang for discussions of paper structure and editing, and Yu Liu for encouraging us to formally publish this work.

REFERENCES

[1] Alexandr Andoni and Piotr Indyk. 2008. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM* 51, 1 (Jan. 2008), 117–122. https://doi.org/10.1145/1327452.1327494

[2] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113. https://doi.org/10.1145/1327452.1327492

[3] Haishan Liu, David Pardoe, Kun Liu, Manoj Thakur, Frank Cao, and Chongzhe Li. 2016. Audience Expansion for Online Social Network Advertising. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 165–174. https://doi.org/10.1145/2939672.2939680

[4] Haiyan Luo, Datong Chen, Zhen Xia, Shiyong Cheng, Yi Mao, Shu Zhang, Jiayi Wen, Xiaojiang Cheng, and Herve Marcellini. 2016. Methods and systems for near real-time lookalike audience expansion in ads targeting. Patent No. US20170330239A1, Filed May 13th., 2016, Issued Nov. 16th., 2017.

[5] Qiang Ma, Eeshan Wagh, Jiayi Wen, Zhen Xia, Robert Ormandi, and Datong Chen. 2016. Score Look-Alike Audiences. In *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. 647–654.

[6] Qiang Ma, Musen Wen, Zhen Xia, and Datong Chen. 2016. 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*.

[7] Ashish Mangalampalli, Adwait Ratnaparkhi, Andrew O. Hatch, Abraham Bagherjeiran, Rajesh Parekh, and Vikram Pudi. 2011. 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)*. 85–86.

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). arXiv:1301.3781 http://arxiv.org/abs/1301.3781

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[10] C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman, and F. Provost. 2014. Machine learning for targeted display advertising: transfer learning in action. *Machine Learning* 95, 1 (01 Apr 2014), 103–127. https://doi.org/10.1007/s10994-013-5375-2

[11] Yan Qu, Jing Wang, Yang Sun, and Hans Marius Holtan. 2014. Systems and methods for generating expanded user segments. Patent No. US8655695B1, Filed May 7th., 2010, Issued Feb. 18th., 2014.

[12] Jianqiang Shen, Sahin Cem Geyik, and Ali Dasdan. 2015. Effective Audience Extension in Online Advertising. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 2099–2108. https://doi.org/10.1145/2783258.2788603

[13] Vassilis Stoumpos and Alex Delis. 2009. Fragment and replicate algorithms for non-equi-join evaluation on Smart Disks. In *Autonomous Decentralized Systems, 2009. ISADS'09. International Symposium on*. IEEE, 1–8.

[14] Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2017. StarSpace: Embed All The Things! *CoRR* abs/1709.03856 (2017). arXiv:1709.03856 http://arxiv.org/abs/1709.03856