Introduction

Campaigns are an integral part of Microsoft Advertising. An advertiser leverages our ad platform to create campaigns with certain objectives in mind such as:

1. Creating brand awareness.
2. Driving sales by providing deals and discounts.
3. Boosting sales of a product targeting a specific geographic location or population.

Given the importance of this entity in understanding and analyzing the entire space, it important that we have a way to represent these campaigns in low dimensional dense vector form. Such a representation must encode information about different dimensions in a campaign: ad copy knowledge, location, target, budget, etc. The problem is that learning such a representation is not easy since we have no explicit labels available at campaign level since users never really interact *directly* with a campaign. Hence, we have to rely on indirect signals to learn this representation. This very reason also makes the evaluation of representations difficult. This work is one of the first attempts to learn such a representation using implicit signals, without any explicit labels, and running experiments to evaluate the quality of such embeddings, and proposing a deep learning based model to generalize to novel campaigns during inference time. Our main contributions are as follows:

1. We discuss the data collection strategy for learning campaign embeddings.
2. We develop propose two techniques to learn embeddings for long running campaigns for which we have sufficient statistics.
3. We discuss various evaluation techniques to measure the quality of learned embedding - quantitative and qualitative both.
4. As future work, we propose a deep learning-based encoder to generalize to novel campaigns during inference.

Methodology

At first we will identify the data sources, discuss collection strategy, and then deep dive into the methods used to generate campaign similarities, which is eventually used to generate campaign embeddings.

Dataset Details

We collect ad-query *co-occurrence* statistics for a month. To generate campaign embeddings we choose one month of data (3 - 30 September, 2019), one month of only *impressed* data (1 - 28 October, 2019) for training, and a week's data, following the training dates as our validation data (29 October - 4 November, 2019) to test whether the embeddings learned using training data can generalize to this dataset. We perform basic filtering like removing aggregators (based on an internally maintained list DarkSieve) and very noisy ads (frequency < 10), from our data. We have a total of 650244 unique campaigns in our dataset and hundreds of millions of ad traffic defined at user request level. We leverage this dataset to calculate a campaign pair similarity in the following ways:

1. Method 1: Directly modeling campaign pair similarity using Sorensen similarity.

2. Method 2: Modeling ad pair similarity using Sorensen similarity and then using kernel to model campaign pair similarity.

All these combinations give us various models to work with as will be discussed in the further sections.

Similarity Generation Using Sorensen Similarity Index

Sorensen similarity index is a statistic that is used to define similarity between two samples. Let's assume we have two ads A and B from campaigns P and Q and the set of queries they co-occurred with $Q\_\{A\}$ and $Q\_\{B\}$ respectively, giving us the union query set Q. Thus, we calculate an ad pair (campaign pair) similarity $SS\_\{A, B\}$ ($KS(P, Q)$) using ad-query (campaign-query) co-occurrence probability, $p_{A,q}$ and $p_{B,q}$ respectively.

$$\mathrm{SS_{AB}}=2^* \left(\sum_{q\in Q} \min(p_{A,q},p_{B,q})\right)/\left(\sum_{q\in Q} (p_{A,q}+p_{B,q})\right)$$

This similarity is normalized between [0, 1], where 0 signifies no similarity or competitiveness between an ad pair and 1 signifies high similarity or competitiveness. We use the campaign version of this as our baseline.

Similarity Generation Using Kernels

We can use kernels to estimate a similarity between two distributions. In our case, we assume that a campaign is a distribution over ads transformed to a high dimensional space. Thus, we can use any valid kernel to calculate this similarity. Now as per kernel trick, we do not need to know how to represent ads in this high dimensional space rather we just need to know how to define the inner product between the pair to be able to calculate this similarity of the two distributions (i.e. using Kernel Mean Embeddings). Given two campaigns P and Q such that they are distributions over ads in high dimension and have m and n number of ads respectively. We can define the similarity between these two distributions using a $kernel$ as,

$$KS_{P,Q} \;=\; Kernel(SS_P, SS_Q)$$

When $Kernel$ is a Cosine kernel then the similarity is,

$$\sum_{A\in P}^{m}\sum_{B\in Q}^{n} SS_{AB} \Big/ \sqrt{\sum_{A\in P}^{m}\sum_{A\in Q}^{m} SS_{AA} * \sum_{B\in P}^{n}\sum_{B\in Q}^{n} SS_{BB}}$$

, and when $Kernel$ is a Radial Basic Function (RBF) kernel then similarity is,

$$e^{-\frac{(\sum_{A\in P}^{m}\sum_{A\in Q}^{m} SS_{AA} - 2\sum_{A\in P}^{m}\sum_{B\in Q}^{n} SS_{AB} + \sum_{B\in P}^{n}\sum_{B\in Q}^{n} SS_{BB})}{2\sigma^2}}$$

, where σ is a hyperparameter, which we tune according to the similarity value distribution. We do a sweep from [10^{-3} to 10^{3}].

Embedding Generation from Similarity

Now that we have calculated similarities between all campaign pairs, we use these values to generate vector representation for a campaign. We create a d dimensional vector, where each d is indexed by a campaign and is equal to the total number of campaigns in the dataset. The value at each dimension is the campaign's similarity or competitiveness with the indexed campaign (value we calculated in the previous section). As obvious, these vectors are rather high dimensional hence for computational purposes we do dimensional reduction using random matrix projection. In hypothesize that projecting the original vectors to a lower dimension would not be too lossy preserve most of the information as the vectors are essentially sparse.

Experiments & Results

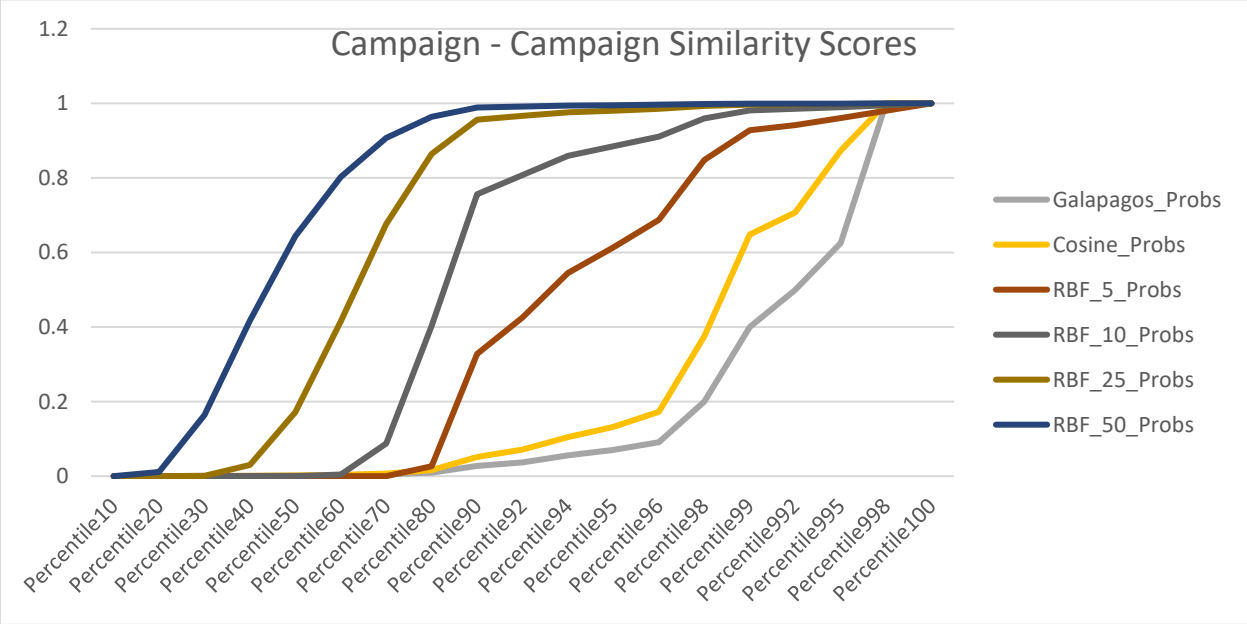
In this section, we will discuss the validation dataset in detail, quantitative and qualitative tasks that we use to evaluate the embeddings, and results.

1. Validation Set Details

In the validation set, we have a total of 416104 campaigns for which we have campaign embeddings (campaigns churn with time hence the featurization set differs from validation set). The average campaign size is 36 while average impression count is 101. Most of the campaigns have rather small number of ads (< 10) though there is a small number of campaigns that do have ads in a high range (number of ads > 10000).

2. Campaign Similarity Distribution

We plot the all pair campaign similarity values to analyze its distribution. From



3. Cluster Analysis

For ease of analysis, we randomly select around 10000 campaign pairs to do cluster analysis; we first start with 100 seed campaign pairs and collect their neighbors for three consecutive levels. We use our campaign embeddings after dimensionality reduction (again through random matrix projection) to do K-means clustering as shown in table below. We use probability-based RBF with sigma 25 for this clustering and set the number of clusters to 100. From each cluster, we randomly sample campaigns and few ad copies within these campaigns for our qualitative analysis. We can observe that there are clear clusters that developed and Ad copies within these clusters are coherent and on topic.

Cluster Topic	Sampled Campaign Ids	Sampled Ad Copies
Job Search	358147244 329329734 378749331	Online Income - Authentic Opportunity - Build an income;Online Income - Authentic Opportunity - Build an income;

		$17-$46/hr Jobs In Arcadia CA - Immediate Hire (All Positions);$17-$46/hr Jobs In Arcadia CA - Immediate Hire (All Positions); $2,600 Guaranteed With Uber - Uber Driver Official Site;Drive Uber-Earn At Least $2600 - Uber Official Site;
Health Remedy	343656873 268385093 369525167	The best hip and joint support - for cats and dogs.;The best hip and joint support - for cats and dogs.; Depression Treatment - Add-On Rx;Help For Adult MDD - Learn More; Compression Sleeves, Socks & Other Garments;Customer Service - Lymphedema Products;
Automotive parts & Automobile	322669462 379774029 369277673	{KeyWord:Best Used Car Deals} - September 2019 Hot Deals;{KeyWord:Tecumseh Parts} - Up to 40% Off - September 2019 Hot Deals; {KeyWord:Honda Service Coupons} - Exclusive Online Deals - Up To 50% Off;{KeyWord:Deals} - {KeyWord:Used Trailers For Sale} - Best Discounts 2019; {KeyWord:} - SUV Sales Near You - {Keyword:Deals!};{KeyWord:} - SUV Sales Near You - {Keyword:Deals!};
Clothing & Apparel	383276807 356702344 268457283	{Keyword: Women Hoodies} - Up To 65% Off - Satisfaction Guarantee;Fashion {Keyword: Women Hoodies} - Up To 65% Off - Different Color Sweatshirts; Shop Muumuu Dresses - The Vermont Country StoreÂ®;Shop Sundresses - The Vermont Country StoreÂ®; {KeyWord:Womens Sweatpants} - Money Saving Deals;Embroidered Patches - Deals Of The Day;

We also compare clusters produced above to those produced using direct Galapagos scores. We observe that directly modeling campaigns, give us more fragmented clusters (scattered and small sized). For example:

campaigns like 329216610, 329068533, 329192675 are all about `Job Search' and are in one cluster using kernel method but are in three small sized clusters in Galapagos. Thus, the clusters generated using our proposed model (RBF) are more coherent than the baseline model.

4. Single Feature Experiment for Click Prediction Task

To further evaluate the usefulness of these embeddings and potential information that they encode, we use them for Click Prediction task (Binary classification) using FastRank model. We used various embedding dimensions for the task as is shown in the table below. Single feature AUC for random features is 0.5 as they have no potential information to help in prediction. RBF kernel with σ as 25 gives the best performance and is robust w.r.t hyperparameters (projected dimension is 128 here). In the table the deciles correspond to campaign ad impression volume and are in the decreasing order. AUC for RBF is much higher than AUC for Galapagos and when CampaignId is used as feature. We see gains in pretty much all the deciles (maximum for the lower volume decile) with below marginal loss in two slices.

Slice	**AUC_Galapagos**	**AUC_RBF_25**	**AUC_Diff**	**AUC_CampaignId**
ALL	0.821074054	**0.824102837**	0.00302878	0.64533174
AdImpressionDecile=10	0.698218045	0.721310824	0.02309278	0.546920448
AdImpressionDecile=9	0.760783986	0.7667667	0.00598271	0.543959434
AdImpressionDecile=8	0.783264131	0.787541348	0.00427722	0.555812741
AdImpressionDecile=7	0.808492899	0.810526576	0.00203368	0.574586315
AdImpressionDecile=6	0.816788984	0.818751149	0.00196217	0.581388514
AdImpressionDecile=5	0.833994579	0.835459813	0.00146523	0.593844267
AdImpressionDecile=4	0.825930206	0.825130272	-0.0007999	0.636613892
AdImpressionDecile=3	0.841799213	0.841452837	-0.0003464	0.677174284
AdImpressionDecile=2	0.853212585	0.853611503	0.00039892	0.754193547
AdImpressionDecile=1	0.912241638	0.912423551	0.00018191	0.868571122

We can interpret that the embeddings are a good predictor of AUC and encode useful information more than ad copy, description, and feature specific COECs as the AUC is higher than any of these AUCs. Though there is a trade-off between embedding size and performance since the smaller the projected dimension is the more lossy the embeddings are.

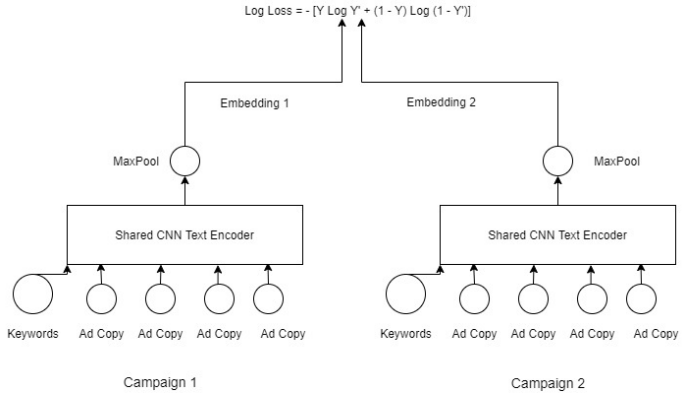
Future Work

One limitation of the approaches described by far is that the method is *transductive*, and hence will not be able to give representation for a campaign not seen in the training dataset. One way is overcome this limitation is by training a model to encode a campaign into low dimensional dense vector representation.

Embedding Encoder Model

We plan to train a Siamese CNN based text encoder model using SentencePiece tokens with cosine similarity (or modified versions that have achieved better performances) with Log loss using back-propagation. This Training will preserve the similarity notion using available feature set and will lead to an *inductive* model can generalize to cold-start campaigns with negligible or no history.

We plan on using Ad copy and keyword tokens as input to this model to predict the similarity between campaigns. The encoder model once trained can be used during inference time to generate dense semantic embeddings for campaigns. Sketch below shows a prototype of our model that we are currently developing.



Conclusion

In this work, we discussed data collection strategy to learn campaign embeddings for long running campaigns with enough history as well as proposed a deep learning based approach, which can generalize to novel campaigns seen only during inference time. We evaluate these embeddings on quantitative and qualitative tasks and establish their usefulness. These embeddings give us coherent meaningful clusters and improve the AUC on the rare campaigns slices.