Extreme Multi-Label Classification for Ad Targeting using Factorization Machines

Martin Pavlovski
mpavlovski@yahooinc.com
Yahoo Research
San Jose, CA, USA

Srinath Ravindran
rsrinath@yahooinc.com
Yahoo Research
San Jose, CA, USA

Djordje Gligorijevic*
dgligorijevic@ebay.com
eBay
San Jose, CA, USA

Shubham Agrawal
shubhama@yahooinc.com
Yahoo Research
San Jose, CA, USA

Ivan Stojkovic
ivans@yahooinc.com
Yahoo Research
San Jose, CA, USA

Nelson Segura-Nunez
nelson.segura-nunez@yahooinc.com
Yahoo Inc.
San Jose, CA, USA

Jelena Gligorijevic
jelenas@yahooinc.com
Yahoo Research
San Jose, CA, USA

ABSTRACT

Applications involving Extreme Multi-Label Classification (XMLC) face several practical challenges with respect to scale, model size and prediction latency, while maintaining satisfactory predictive accuracy. In this paper, we propose a Multi-Label Factorization Machine (MLFM) model, which addresses some of the challenges in XMLC problems. We use behavioral ad targeting as a case study to illustrate the benefits of the MLFM model. Predicting user qualifications for targeting segments plays a major role in both personalization and real-time bidding. Considering the large number of segments and the prediction time requirements of real-world production systems, building scalable models is often difficult and computationally burdensome. To cope with these challenges, we (1) reformulate the problem of assigning users to segments as a multi-label classification (XMLC) problem, and (2) leverage the benefits of the conventional FM model and generalize its capacity to joint prediction across a large number of targeting segments. We have shown that the MLFM model is both effective and computationally efficient compared to several baseline models on publicly available datasets in addition to the targeting use case.

CCS CONCEPTS

• **Computing methodologies → Factorization methods**; **Supervised learning by classification**; • **Information systems → Display advertising**; *Recommender systems.*

*This work was done when the author was at Yahoo.

KEYWORDS

factorization machines; extreme multi-label classification; user modeling; ad targeting

ACM Reference Format:
Martin Pavlovski, Srinath Ravindran, Djordje Gligorijevic, Shubham Agrawal, Ivan Stojkovic, Nelson Segura-Nunez, and Jelena Gligorijevic. 2023. Extreme Multi-Label Classification for Ad Targeting using Factorization Machines. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599822

1 INTRODUCTION

An increasing number of web-based applications such as content recommendation, personalization, information retrieval, shopping and advertising, utilize the power of machine learning to improve user experience. Many of these applications require assigning L target labels ($\mathbf{y} \in \{0,1\}^L$) for a given data point simultaneously, referred to as **Multi-Label Classification**.

Typically, one or more binary classification models are used, each responsible for predictions on their respective labels. Alternatively, a single model can be used to learn and infer the predictions across all labels. Though using multiple binary classifiers, typically in a one-vs-all (OVA) setting, is more prevalent, such a classification scheme has certain limitations in terms of scale, accuracy and maintainability, as we shall discuss shortly. These limitations are magnified in applications that deal with a large number of classes, referred to as **Extreme Multi-Label Classification** (XMLC, XMC, or XC) tasks [3, 4, 28, 51]. However, many XMLC methods have their limitations too.

Factorization Machines (FM) [42] have shown to perform well on tasks involving complex relationships in data, with low latency. We propose a Multi-Label Factorization Machine (MLFM) (see Section 3) to leverage the benefits of conventional FMs to address some of the challenges in extreme multi-label classification. We use behavioral ad targeting as a case study. We also demonstrate the effectiveness of our approach on several public benchmark datasets (Section 4).

Behavioral Ad Targeting [13, 32, 53], also known as interest targeting or audience targeting, groups users into **segments** based on the users' behaviors or interactions such as pages visited, searches performed, and links clicked. Along with other mechanisms like demographic and geographic targeting, behavioral targeting is used by advertisers for customizing display advertisements so as to reach appropriate users (also referred to as customers or audiences). For each incoming ad request, an ad server identifies the correct set of segments corresponding to the user behind the request. This set of segments is then used to select a set of potential advertisements, the best among which will be displayed to the user.

A single user can belong to multiple segments at the same time. For instance, a user can belong to segments corresponding to topics such as owning a pet, applying for a loan, pursuing a part-time degree, and taking their family on a vacation to a theme park. As a result, the number of segments can be large, even up to a few thousand, depending on the advertising company.

Challenges: Ad targeting and other web applications pose the following challenges, and our model, MLFM, is designed to address them. These challenges are not independent and often influence the design decisions when building an end-to-end machine learning system.

Large Number of Segments. Building a scalable model to handle a large number of segments is demanding. Using multiple binary classifiers to address this problem requires generating an individual dataset and tuning hyperparameters for each of the L models, whereas a single multi-label model can be excessively large and computationally expensive. An ideal solution should be scalable without compromising prediction accuracy.

Low Prediction Latency. Prediction latency can affect batch predictions, near real-time and real-time applications. Lower latency ensures timely generation of predictions in both batch and near real-time use cases. Many large-scale real-time prediction systems rely on linear models such as logistic regression in order to meet service-level agreements (SLAs) on latency, which are typically in milliseconds. More complex models can be computationally expensive, both in space utilization and in time complexity.

Relationship Among Segments. Utilizing multiple binary classifiers, one for each label, or even using one-vs-all (OVA) classifiers assumes that both the labels (or segments), and the data points can be treated independently. The labels can either be related explicitly (in case of interest hierarchies [18, 54]), or implicitly, when they have latent relationships [11, 49]. Further, these between-label relationships and the similarity among the data points make the selection of negative examples more complex when using multiple binary classifiers.

(Near) Real-time Prediction. Most approaches in the industry rely on offline batch predictions, which are based on historical user behavior data. Such models are often trained periodically, using features or user activities available up to the day of training, and an offline scoring procedure assigns the labels for each user. However, users' interests are dynamic and even a single page visit or purchase can be indicative of their subsequent behavior. As a result, a real-time or a near-real-time scoring procedure can leverage a user's latest activities to assign an updated interest segment to that user. User history is often limited either because a user is new to the network, or due to various privacy rules including the General Data Protection Regulation (GDPR) and browser/device restrictions. In such cases, (near) real-time prediction would help. Namely, real-time targeting is used for demographic and interest prediction, and recently, contextual targeting is gaining traction across the industry. However, accuracy of prediction at scale remains a challenge.

Contributions: We summarize our contributions as follows:

- We generalize the applicability of FM models to extreme multi-label classification (XMLC) problems, particularly to ad targeting problems involving a large number of targeting segments, which are inherently XMLC problems.
- We propose an alternative lightweight formulation of the multi-label FM model that allows for capturing pairwise feature and field interactions (which would conventionally require quadratic time complexity) in asymptotically *linear time* w.r.t. the number of features (or fields in the case of multi-field categorical data). Since less operations are required to project each feature/field onto a single label, a larger number of labels can be processed.
- We conduct extensive experiments to demonstrate the effectiveness and computational efficiency of MLFM in comparison with several multi-label classification baselines on both publicly available benchmarks and proprietary targeting datasets.

2 BACKGROUND

Considering the large number of targeting segments, the problem of assigning multiple segments to a user's record can also be approached from an extreme multi-label classification perspective. The objective of extreme multi-label classification (XMLC) is to build classifiers capable of automatically assigning a data point the most relevant subset of labels from an extremely large set of possible class labels [4]. To that end, a plethora of XMLC approaches have been proposed over the years, that can be categorized into three main categories: one-vs-all classifiers, tree-based methods and embedding-based methods. Although one-vs-all classifiers [2, 34, 55, 56] are quite common in the literature, most of them ignore feature interactions and learn feature weights w.r.t. each label separately without sharing any weights among the labels. Tree-based methods [1, 19, 20, 38–40], on the other hand, partition the feature and/or label space instead of explicitly capturing feature interactions and their associations with the labels. Finally, label embedding-based methods [5, 24, 33, 41, 50, 52] utilize label sparsity to learn compressed label embeddings and thus reduce the inference time in the compressed space. However, the predicted labels would also need to be projected back into the original label space at an additional cost.

In this work, we focus on approaches from the first two categories as such approaches are directly applicable to our time-critical targeting use case for which embedding-based methods may not be well suited considering their computational complexity.

Furthermore, our work focuses specifically on multi-label tabular numerical/categorical data, for which Factorization Machine (FM) models are particularly suitable. XMLC approaches have been applied to other data types, such as text, in various domains including document tagging [6], product recommendation [8, 58], natural language modeling [21], among others. In Appendix A (Section A.2), we discuss the main aspects in which our work conceptually differs from such approaches.

Table 1: Model equations of first and second-order models.

Model	Model equation	Notation
Logistic Regression (LR) [16, Chap. 4.4, p. 119–128]	$w_0 + \sum_{i=1}^{D} w_i x_i$ (1)	w_0 – bias term; w_i – weight of feature x_i, $\forall i = 1, \ldots, D$.
Degree-2 Polynomial Mapping (Poly2) [9]	$w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} w_{ij} x_i x_j$ (2)	w_{ij} – weight of interaction between features x_i and x_j.
Factorization Machine (FM) [42]	$w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$ (3)	$\mathbf{v}_i, \mathbf{v}_j$ – embeddings for features i and j; $\langle \cdot, \cdot \rangle$ – dot product operator.
Field-aware Factorization Machine (FFM) [22, 23]	$w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} \langle \mathbf{v}_{iF(j)}, \mathbf{v}_{jF(i)} \rangle x_i x_j$ (4)	$\mathbf{v}_{iF(j)}$ – embedding used for x_i to interact with any x_j from field $F(j) \neq F(i)$.
Field-weighted Factorization Machine (FwFM) [37]	$w_0 + \sum_{i=1}^{D} \langle \mathbf{v}_i, \mathbf{w}_{F(i)} \rangle x_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} r_{F(i)F(j)} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$ (5)	$r_{F(i)F(j)}$ – between-field interaction weight; $\mathbf{w}_{F(i)}$ – embedding vector for field $F(i)$.

2.1 Problem Formulation

Consider a dataset with $\hat{D}$ different fields $\{F_1, F_2, \ldots, F_{\hat{D}}\}$ such that a feature f_i belongs to one and only one field $F(i)$. Let $\mathbf{x}$ be a one-hot encoded feature vector $[x_1, \ldots, x_D]$ from that dataset. Only one feature x_i in $\mathbf{x}$ can be active (i.e. present, or non-zero) per field $F(i)$.

For example, in ad targeting, a field F_j = "country" could take values $f_i \in$ {AU, BR, CA, FR, GR, US, ...}. Further, $\mathbf{x}$ is associated with a vector of binary labels $\mathbf{y} \in \{0, 1\}^L$, where $y_l = 1$ if label l is active for $\mathbf{x}$ and $y_l = 0$ otherwise. Note that multiple labels can be active for $\mathbf{x}$. For instance, in ad targeting, each label y_l indicates whether or not the user behind an *ad request* $\mathbf{x}$ qualified for a *targeting segment l*. The objective is to learn a function $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^L$ that maps an example $\mathbf{x}$ to a probability vector $[P(y_1|\mathbf{x}), \ldots, P(y_L|\mathbf{x})]$.

2.2 Preliminaries

Let us focus on a special case of the multi-label classification problem described in Section 2.1 when $L = 1$. This boils down to the *binary classification* problem of modeling the probability for a single class label y, given an example $\mathbf{x}$, as

$$P(y|\mathbf{x}) = \sigma(\phi(\mathbf{x})) = \frac{1}{1 + e^{-\phi(\mathbf{x})}}, \tag{6}$$

where $\phi : \mathbb{R}^D \rightarrow \mathbb{R}$. The following describes several approaches to estimating Eq. (6) that are related to and predecessors of the model presented in this work.

Arguably among the most conventional approaches to addressing the binary classification problem would be to build a Logistic Regression (LR) model that learns a weight for each feature of $\mathbf{x}$ and takes a linear combination of the features and their associated weights (refer to Eq. (1)).

Although simple and lightweight, a linear classification model such as LR has a limited modeling capacity since it ignores any interactions between individual features that might be relevant to predicting the class label y. Thus, one can include additional so-called *cross features* by considering all pairs of the original features, and apply a Degree-2 Polynomial Mapping (Poly2) [9] to learn a weight for each feature pair as shown in Eq. (2). However, this automatically introduces substantial computational overhead, particularly when the number of features is large. For example, in the use case presented in this work where 200,000 features are used, weights for billions of feature pairs would need to be learnt.

As a response to the limitation of Poly2, the FM model [42] (refer to Eq. (3)) was designed to learn high-quality estimates of feature interaction strengths by factorizing the interaction weights as $w_{ij} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle$ while requiring less memory and computational time. Also, FMs have outperformed Poly2 in various applications involving sparse categorical data, as suggested in [37].

To capture the behavior of a feature when it interacts with features from other fields different than its own field, the original FM has been extended to a Field-aware Factorization Machine (FFM) [23]. This is accounted for in FFMs (Eq. (4)) by learning a separate embedding vector, $\mathbf{v}_{iF(j)}$, for each $(x_i, F(j))$ pair such that $\mathbf{v}_{iF(j)}$ is used to calculate the strength of the interaction between the feature x_i and any other feature x_j from field $F(j) \neq F(i)$.

As stressed in [37], despite the improvements of FFMs over conventional FMs, the large number of model parameters makes them inapplicable in real-world production systems. Therefore, to reduce the complexity of FFMs, Pan et al. [37] proposed to model field-level interactions $r_{F(i)F(j)}$ as presented in Eq. (5). The interaction weights of different field pairs eliminate the need for learning $\hat{D} - 1$ different embedding vectors $\mathbf{v}_{iF(j)}$ for each feature i, thus making the Field-weighted Factorization Machine (FwFM) a competitive and memory-efficient extension of FFM.

Since FwFM was initially designed for binary classification and applied to click-through rate prediction in [37], the same authors have generalized the FwFM formulation in [36] to multi-task settings. Nevertheless, they applied their model to predicting *four different types of conversions* by treating each conversion type as a separate task. Moreover, in their setting, only one conversion type can be active per data point, thus making the four tasks *mutually exclusive*; which is characteristic of multi-class as opposed to multi-label problems where the class labels are *mutually non-exclusive*.

In Section 3, we demonstrate that the generalization of FwFM can be adapted to problems involving a large number of mutually non-exclusive class labels (e.g., thousands of labels where multiple labels can be active per data point). In addition, we propose an alternative lightweight formulation that can further reduce the model's computational time, which can be of considerable importance in time-critical settings such as those of the targeting use case presented in this work.

For the reader's reference, the model equations of the afore-described approaches are provided in Table 1.

3 MULTI-LABEL FACTORIZATION MACHINE

In the following, we provide a formal description of the Multi-Label Factorization Machine (MLFM) model aimed to address the problem formulated in Section 2.1, and propose an alternative formulation that allows for reducing the model's original time complexity.

Feature Embedding. First, a feature embedding lookup is created in which every feature, representing a binary random variable f_i from a vocabulary $S_{feat} = \{f_1, f_2, \ldots, f_D\}$, is assigned an embedding $\mathbf{v}_i \in \mathbb{R}^M$ such that $M \ll D$. The values of the feature embeddings are initialized to random uniform values and a mapping $g : S_{feat} \to \mathbb{R}^M$ is used to retrieve the embedding $\mathbf{v}_i = g(f_i)$ of the feature f_i.

Multi-Label Modeling. In order to handle a large number of features, we solely consider second-degree feature interactions. The decision function $\phi(\mathbf{x})$ of a second-degree MLFM is defined as:

$$\mathbf{w}_0 + \left[\sum_{i=1}^{D} w_i^l x_i\right]_{l=1}^{L} + \left[\sum_{i=1}^{D}\sum_{j=i+1}^{D} r_{F(i)F(j)}^l \langle \mathbf{v}_i, \mathbf{v}_j\rangle x_i x_j\right]_{l=1}^{L}, \quad (7)$$

where $\mathbf{w}_0 \in \mathbb{R}^L$ is a bias vector, $\mathbf{W} = [w_i^l]_{D\times L}$ are the feature weights over L class labels; $F(i)$ and $F(j)$ denote the fields that features x_i and x_j belong to, respectively, thus $\mathbf{R} = [r_{F(i)F(j)}^l]_{\hat{D}\times\hat{D}\times L}$; and $\mathbf{V} = [v_{im}]_{D\times M}$ is the feature embedding matrix shared among all labels. $\langle \mathbf{v}_i, \mathbf{v}_j\rangle$ represents the dot product between the embeddings of two features, for each $i, j = 1, \ldots, D$. This extends the capacity of the linear formulation given by the first two terms in Eq. (7) and allows for modeling between-feature as well as between-field interactions. Also, note that instead of using a separate parameter for each *feature interaction* w.r.t. each label (not to be confused with the $r_{F(i)F(j)}^l$ parameters that model *field interactions*), the feature interactions are modeled by factorizing the interaction strengths $w_{ij}, \forall i, j = 1, \ldots, D$ (shared among all labels). This is one of the central advantages of FMs which aids in obtaining high-quality estimates of feature interaction strengths even when dealing with considerably sparse features, as in our targeting use case.

Parameter Learning. Given a labeled dataset $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, the categorical cross-entropy loss for the n^{th} data point is calculated as

$$\ell(\mathbf{x}_n, \mathbf{y}_n) = -\frac{1}{L}\left(\sum_{l=1}^{L} y_{nl}(\log(p_{nl})) + (1-y_{nl})(1-\log(p_{nl}))\right), \quad (8)$$

where $p_{nl} = P(y_{nl}|\mathbf{x}_n) = 1/(1+e^{-\phi^l(\mathbf{x}_n)})$. Note that a sigmoid function is used to obtain the class probabilities instead of a softmax function since a data point might be assigned multiple labels. Finally, starting from randomly initialized parameter values, the optimal model parameters $\mathbf{w}_0, \mathbf{W}, \mathbf{R}, \mathbf{V}$ are determined by minimizing the loss calculated over all data points $\frac{1}{N}\sum_{n=1}^{N} \ell(\mathbf{x}_n, \mathbf{y}_n)$.

Space and Time Complexity. The MLFM model has a space complexity of $O(L+LD+L\hat{D}(\hat{D}-1)/2+DM)$ and a time complexity of $O(D^2M + D^2L)$, where D is the number of features, $\hat{D}$ is the number of fields (or number of active features), M is the feature embedding dimension, and L is the number of labels.

Lightweight Formulation. As one may notice, the interaction term in Eq. (7) is quadratic with respect to D. In the following, we propose an alternative formulation of Eq. (7) in which we decompose the between-feature interaction term such that it can be computed in linear time with respect to the number of features D. First, we factorize the field interactions $r_{F(i)F(j)}^l$ as follows:

$$\begin{aligned}
&\sum_{i=1}^{D}\sum_{j=i+1}^{D} \langle \mathbf{u}_{F(i)}^l, \mathbf{u}_{F(j)}^l\rangle \langle \mathbf{v}_i, \mathbf{v}_j\rangle x_i x_j \\
&= \sum_{i=1}^{D}\sum_{j=i+1}^{D}\left(\sum_{h=1}^{H} u_{F(i)h}^l u_{F(j)h}^l\right)\left(\sum_{m=1}^{M} v_{im} v_{jm}\right) x_i x_j \\
&= \sum_{i=1}^{D}\sum_{j=i+1}^{D}\left(\sum_{h=1}^{H}\sum_{m=1}^{M}\left(u_{F(i)h}^l u_{F(j)h}^l\right)\left(v_{im} v_{jm}\right)\right) x_i x_j \\
&= \sum_{i=1}^{D}\sum_{j=i+1}^{D}\left(\sum_{h=1}^{H}\sum_{m=1}^{M}\left(u_{F(i)h}^l v_{im}\right)\left(u_{F(j)h}^l v_{jm}\right)\right) x_i x_j,
\end{aligned} \quad (9)$$

where $\mathbf{u}_{F(i)}^l, \mathbf{u}_{F(j)}^l \in \mathbb{R}^H$. For simplicity, let us organize the latent feature and field factors into two $H\times M$ matrices, $\mathbf{Q}_i^l = [u_{F(i)h}^l v_{im}]_{h=1,m=1}^{H,M}$ and $\mathbf{Q}_j^l = [u_{F(j)h}^l v_{jm}]_{h=1,m=1}^{H,M}$. Now, the above expression can be rewritten as

$$\sum_{i=1}^{D}\sum_{j=i+1}^{D} \langle \mathbf{Q}_i^l, \mathbf{Q}_j^l\rangle_{\text{F}} x_i x_j = \sum_{i=1}^{D}\sum_{j=i+1}^{D} \langle \mathbf{q}_i^l, \mathbf{q}_j^l\rangle x_i x_j, \quad (10)$$

where the operator $\langle\cdot,\cdot\rangle_{\text{F}}$ computes the Frobenius inner product between two matrices; and $\mathbf{q}_i^l = \text{vec}(\mathbf{Q}_i^l)$, $\mathbf{q}_j^l = \text{vec}(\mathbf{Q}_j^l)$.

Following [42, Lemma 3.1], we further expand Eq. (10) as follows:

$$\begin{aligned}
&\frac{1}{2}\left(\sum_{i=1}^{D}\sum_{j=1}^{D}\langle \mathbf{q}_i^l, \mathbf{q}_j^l\rangle x_i x_j - \sum_{i=1}^{D}\langle \mathbf{q}_i^l, \mathbf{q}_i^l\rangle x_i x_i\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{k=1}^{H*M} q_{ik}^l q_{jk}^l x_i x_j - \sum_{i=1}^{D}\sum_{k=1}^{H*M} q_{ik}^l q_{ik}^l x_i x_i\right) \\
&= \frac{1}{2}\sum_{k=1}^{H*M}\left(\left(\sum_{i=1}^{D} q_{ik}^l x_i\right)\left(\sum_{j=1}^{D} q_{jk}^l x_j\right) - \sum_{i=1}^{D}(q_{ik}^l)^2 x_i^2\right) \\
&= \frac{1}{2}\sum_{k=1}^{H*M}\left(\left(\sum_{i=1}^{D} q_{ik}^l x_i\right)^2 - \sum_{i=1}^{D}(q_{ik}^l x_i)^2\right).
\end{aligned} \quad (11)$$

The above equation is linear in terms of the number of features D and the dimensions of the feature and field embeddings (M and H), that is $O(DMH)$. Considering all L labels, the complexity becomes $O(LDMH)$. Substituting the reformulated expression from Eq. (11)

into the original formulation (Eq. (7)) yields

$$\phi(\mathbf{x}) = \left[\sum_{i=0}^{D} w_i^l x_i\right]_{l=1}^{L} + \left[\frac{1}{2}\sum_{k=1}^{H*M}\left(\left(\sum_{i=1}^{D} q_{ik}^l x_i\right)^2 - \sum_{i=1}^{D}(q_{ik}^l x_i)^2\right)\right]_{l=1}^{L}. \tag{12}$$

Note that, in the above formulation, H is of the same order of magnitude as M or lower and, in practice, H and M can be chosen such that $H, M \ll D$. Moreover, for sparse multi-field categorical data, where typically $\hat{D} \ll D$, the total complexity $O(DL + LDMH)$ can be further reduced to $O(\hat{D}L + L\hat{D}MH) \approx O(L\hat{D}MH)$.

For more details on the implementation of MLFM, the reader is referred to Section A.1.

4 EXPERIMENTS

The MLFM model is aimed to provide a satisfactory classification performance with the lowest inference time possible and a reasonable memory footprint. To that end, we first compared MLFM with baseline models widely used in XMLC problems on standard benchmark datasets. The findings from the experiments discussed in Section 4.3 suggest that MLFM provides the best trade-off between classification performance and inference time when compared to both one-vs-all and XMLC baseline models. Consequently, we apply MLFM to behavioral targeting datasets (Section 4.4).

4.1 Baselines

Conventional OVA Classifiers. The MLFM model was compared against three One-vs-All (OVA) classifiers: (1) an OVA scheme that leverages a separate logistic regression (LR) model [16, Chap. 4.4, p. 119–128] for each label, (2) an OVA-LinSVM model having Support Vector Machines (SVMs) [17, 27, 43] with linear kernels as its base classifiers, and (3) an OVA variant of Multi-Layer Perceptron (MLP).
XMLC Approaches. Tree ensemble-based methods for extreme multi-label classification (XMLC) were also considered as additional baselines. Such models include: FastXML [40], PfastreXML [19] and Parabel [39].

4.2 Experimental Setup

MLFM was trained on batches of 32 data points each, for 10 epochs, using AdamW [29] (a variant of Adam [25] with decoupled weight decay) with a learning rate of 0.01. The baseline models were run with their respective default hyperparameter settings; OVA-MLP was trained with 10 hidden dimensions. Throughout all experiments, the feature embedding dimension M was set to 10. We evaluated MLFM under various values of M, ranging from 10 up to 100, and did not observe a significant change in its predictive performance. Higher values of M impacted the performance on rare classes only, positively in some cases and adversely in others (refer to Section 4.4.3 for more details).

MLFM and the linear OVA models were implemented in Python 3 (with additional usage of PyXCLib[1]) and PyTorch 1.6.0 with CUDA toolkit 10.1, and were run on a machine with 640 GB of memory, 48 CPUs and one NVIDIA Tesla V100 GPU. The inference procedure of MLFM was implemented in Java using the EJML library [2]. For the XMLC ensemble baselines, their original C++ implementations from [4] were used, some of which are multi-threaded.

[1]https://github.com/kunaldahiya/pyxclib
[2]http://ejml.org

The multi-label classification performances of MLFM and the baseline models (listed in Section 4.1) were measured with respect to each label using the well-established performance indicator of area under the ROC curve (AUC) [15, 31]. To obtain a measure for the overall performance of the models, each model's AUCs were (1) macro-averaged over all labels and (2) summarized in a stratified manner similar to [36]:

$$\text{StratifiedAUC} = \frac{\sum_{l=1}^{L} N_l \cdot \text{AUC}_l}{\sum_{l=1}^{L} N_l}, \tag{13}$$

where the AUC value measured for label l is weighted by the label's frequency of positive examples, $N_l = \sum_{n=1}^{N} y_{nl}$.

Note that metrics such as nDCG@k and Precision@k, which are commonly used in the XMLC community, were not considered in this work since we approach the problem formulated in Section 2.1 from a standard multi-label classification perspective rather than a ranking perspective.

4.3 Experiments on Public Datasets

4.3.1 Data Description. The classification performances of MLFM and the baseline models were evaluated on three publicly available benchmark datasets from different domains, of different sizes and characterized by different levels of feature and label sparsity. A brief description of each dataset is provided in the following.
MediaMill Challenge Dataset. As a part of the multimedia indexing challenge posed in [46], static video frames were extracted from the multimedia archive of the TRECVID benchmark [44, 45]. For the purposes of further indexing, a total of 43,907 frames (or single camera shots) were manually assigned labels representing concepts related to program categories, setting, people, objects, activities, events, and graphics. A visual feature extraction based on color-texture histograms was applied to generate a 120-dimensional representation of each key frame. Given such a representation of a key frame, the task is to select only the concepts relevant to the frame from a lexicon of 101 predefined concepts.
Reuters Corpus Volume I. This archive consists of roughly 800,000 manually categorized newswire stories made available by Reuters, Ltd. Each newswire article is represented by 47,236 bag-of-words features corresponding to stemmed unique terms; and tagged with multiple category codes from a set of 2456 possible codes related to topics, industries or regions. That being said, the classification task is to categorize the articles into their relevant categories.
EURLex. The EURLex dataset represents a collection of 51,000 legal documents about European Union law. Each document is represented by a term-frequency vector encoding 200,000 unique tokens. Moreover, the documents are of different types such as treaties, legislations, case-law and legislative proposals; and are annotated using different orthogonal categorization schemes [30]. Considered the most important among the schemes is EuroVoc, a topic hierarchy with almost 4271 categories regarding different aspects of European law. Hence, given a document, the classification task in focus is to classify a document into its corresponding EuroVoc concept categories.

Table 2: Public datasets' statistics.

Dataset	Features D	Labels L	Train	Test	Records/ label	Labels/ record
MediaMill	120	101	30,993	12,914	1902.15	4.38
RCV1	47,236	2,456	623,847	155,962	1218.56	4.79
EURLex	200,000	4,271	45,000	6,000	60.57	5.07

For the public datasets described above, we used their predefined training and test sets from [4]. The descriptive statistics of each dataset are provided in Table 2.

4.3.2 Multi-Label Classification Performance. The test classification performances of all models were evaluated in terms of macro-average and stratified AUC. The results are summarized in Table 3.

It can be observed that the XMLC tree-based methods (FastXML, PfastreXML and Parabel) generally outperform the OVA classifiers with respect to both macro-average and stratified AUC, and across all three datasets. The only exception to this observation is the performance of OVA-MLP on the EURLex dataset where OVA-MLP is more accurate than FastXML and Parabel; yet its performance is still surpassed by that of PfastreXML which is consistently the best-performing method among the XMLC baselines.

Nevertheless, MLFM exhibits either better or at least comparable performance to those of the OVA baselines as well as the XMLC methods which are particularly designed to deal with large label spaces. More specifically, on the MediaMill dataset, MLFM yields lifts in macro AUC (starting from 1% up to 33.7%) as well as in stratified AUC (ranging between 1.7% and 33%). A similar behavior is observed on the EURLex dataset on which MLFM attains improvements of 3.3%-18.9% in macro AUC and corresponding stratified AUC improvements of 1.1%-23.7%; despite the drastically larger feature and label space. As for RCV1, MLFM demonstrates runner-up performance, standing next to PfastreXML in terms of macro AUC, with negligible difference to PfastreXML's stratified AUC.

4.3.3 Computational Efficiency. The inference run times of the models in terms of CPU and GPU are summarized in Table 3.

As one would expect, the ***CPU*** run times of all models increase with the complexity of the datasets. On CPU-based hardware, MLFM achieves either lower or comparable run time to other baselines. As the feature and label spaces become larger, the tree-based XMLC methods are either comparable to (e.g., RCV1) or prevail over (e.g., EURLex) the other models. Their implementations support multi-threading, which gives them an advantage over the other models. It must also be noted that the three XMLC baselines (FastXML, PfastreXML and Parabel) were implemented in C++, whereas the other baselines were implemented in Python. Despite the implementation differences, MLFM and OVA-LR have lower latencies than PfastreXML and Parabel on MediaMill and RCV1. Also, in spite of the slower run time, particularly on EURLex, it should be noted that MLFM achieves better classification performance than the tree-based methods.

The implementations of only three models (OVA-LR, OVA-MLP and MLFM) support ***GPU*** execution. When run on a GPU, the latencies of the three models are comparable. However, their run times are orders of magnitude better than those of the multi-threaded XMLC baselines.

In summary, based on the observations from Sections 4.3.2 and 4.3.3, the MLFM model seems to provide a satisfactory trade-off between classification performance and computational efficiency.

4.4 Experiments on Behavioral Targeting

In behavioral/audience targeting, users are assigned to a set of segments based on their behavior on the web. A segment definition specifies the criteria identifying the users for that particular segment. The criteria, which is aligned with an ad campaign objective, can either be provided by an advertiser or automatically inferred from users' online behavior. In the former case, the advertiser can provide keywords that correspond to user behavior, or define page visits or purchases to identify users who are likely to visit a particular page or purchase a certain product. In the latter case, the users are assigned to an interest category as defined by a taxonomy [18].

In this work, we demonstrate the performance of our model on both types of audience targeting[3]. In Section 4.4.1, we describe the application of our model to conversion-based targeting. A conversion event, or an acquisition, is any interaction or activity performed by a user that is defined by an advertiser as the objective of a campaign, or as being valuable to the business. Next, in Section 4.4.2, we describe our model's application to interest-based targeting, where a user is assigned to hierarchically structured interest categories. While conversion-based targeting is the primary focus of our work, we also applied our work to interest-based targeting, even though our model was not initially designed for handling hierarchical labels.

Real-time and near-real-time targeting are of considerable importance as they allow estimation of a user's behavior based on their most recent set of activities, given that the user's interest or affinity can change over time. Further, when a user's history is limited, predicting the user's conversion behavior or their interests can provide a more enriched user profile for the real-time bidding process, which identifies the best advertisement for the user.

4.4.1 Conversion-Based Targeting. For model training purposes, ad requests were sampled over a period of one week. The ad requests were collected such that for each request an impression was registered upon displaying an ad. A similar dataset for the subsequent three days was collected and used as a holdout (test) set. This resulted in training and test datasets containing ~950 million and ~160 million records, respectively.

For each record, the following fields were extracted: webpage top-level domain, webpage subdomain, webpage URL, user location, i.e. city and country, user's local time, device type, browser type, make and model of mobile device, mobile app name, and webpage category. These fields were one-hot encoded, resulting in millions of binary features. Thus, the features' values were ordered by their frequencies and the top 200,000 were selected. An additional 'other' category was created to cover the less frequent features. The lists of conversion-based segments that a user qualified for were used to label the records.

[3]This research was conducted in accordance with Yahoo's Privacy Policy and respect for applicable user privacy controls.

Table 3: Classification performance and average inference run times (in milliseconds) measured on the publicly available datasets. Bold and italic formatting indicate best and runner-up performance, respectively.

Model	MediaMill				RCV1				EURLex			
	AUC		Inference time		AUC		Inference time		AUC		Inference time	
	Macro	Stratified	CPU	GPU	Macro	Stratified	CPU	GPU	Macro	Stratified	CPU	GPU
OVA-LR	0.6582	0.6689	0.0110	0.0053	0.6197	0.9466	0.2612	0.0104	0.7900	0.9168	6.6999	0.0133
OVA-SVM	0.5090	0.4944	0.0834	–	0.7697	0.7490	5.3943	–	0.7723	0.7474	67.2408	–
OVA-MLP	0.6141	0.7130	0.0443	0.0085	0.9020	0.9618	0.3896	0.0176	0.8967	0.9703	6.7850	0.0237
FastXML	0.6789	0.7946	0.1071	–	0.6906	0.9485	0.2992	–	0.8359	0.9448	0.9486	–
PfastreXML	*0.8354*	*0.8081*	0.4330	–	**0.9354**	**0.9873**	1.8260	–	*0.9282*	*0.9734*	2.8259	–
Parabel	0.7963	0.8024	0.0536	–	0.8439	0.9623	0.7467	–	0.8687	0.9558	2.3686	–
MLFM	**0.8456**	**0.8248**	0.0200	0.0113	*0.9179*	*0.9808*	0.3955	0.0192	**0.9613**	**0.9847**	7.9049	0.0298

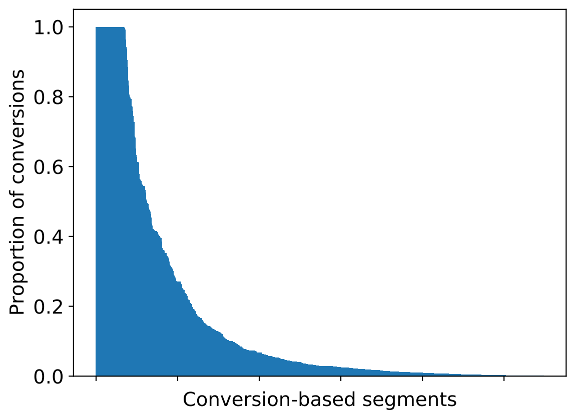


Figure 1: Distribution of labels (conversions) in the conversion-based targeting dataset.

Note that segments with a relatively small number of registered conversions across all records were omitted from the training process, resulting in 1098 segments deemed eligible for training. The conversion frequencies for the selected segments are visualized in Figure 1.

Offline Performance Evaluation. The models trained on the weekly ad request data were used to predict the set of segments that the users qualified for. The predictive performances of the models were measured for each individual segment. Their overall performances across all of the segments are summarized in Table 4. Due to the large size of the dataset, the training of the OVA-MLP model took an extensive amount of time, which makes it not well suited for the time-critical targeting application at hand.

First, it can be observed that the XMLC tree-based methods outperform OVA-LinSVM by a considerable margin. They are quite comparable to OVA-LR with respect to macro-average AUC. Nevertheless, in terms of stratified AUC (weighed by the conversion frequencies of the segments), the XMLC baselines (particularly PfastreXML and Parabel) outperform both OVA methods. When the conversion frequencies are not accounted for in the evaluation, Parabel yields a macro AUC superior to the OVA models and the other XMLC baselines. This is due to Parabel's ability to sidestep the class imbalance problem by partitioning the segments into balanced groups of records.

Table 4: Offline predictive performance, summarized over 1098 conversion-based targeting segments.

Model	Macro Test AUC	Strat. Test AUC
OVA-LR	0.7928	0.7477
OVA-SVM	0.7707	0.7028
FastXML	0.8011	0.7528
PfastreXML	0.8065	0.7804
Parabel	0.8239	0.7892
MLFM	**0.8421**	**0.8006**

MLFM outperforms all OVA and XMLC baselines, yielding lifts of 1.8%-7.1% in macro AUC and 1.1%-9.8% in stratified AUC. As the XMLC methods partition or sample the features, they fail to capture the feature interactions informative to estimating the conversion probabilities for different segments. The linear models ignore such interactions altogether due to their limited modeling capacity. This shows the importance of MLFM's capability to leverage interactions between features in a joint multi-segment conversion prediction scheme.

When applied to (near) real-time targeting, a segment qualification model would need to be computationally efficient to ensure timely generation of predictions. Consequently, for the purposes of the following experiments, we compare MLFM to the most lightweight model, OVA-LR. Based on the results on the benchmark datasets shown in Table 3, the other baselines would not typically meet the response time criteria and were thus deemed inapplicable to this use case.

The overall metrics analyzed in the previous experiment are easy-to-follow and allow for convenient comparison of the models. However, superior macro AUC or stratified AUC of one model over another does not necessarily indicate better performance on the majority of segments, as the former metric is sensitive to very large or very low per-segment AUC values, while the latter inherently favors conversion-prevalent segments. In Figure 2, we show a detailed comparison of the per-segment AUC values obtained by MLFM against those obtained by OVA-LR.

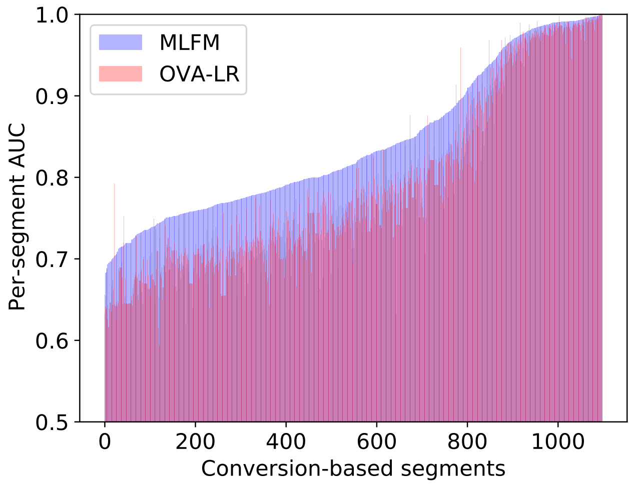


Figure 2: Head-to-head comparison of MLFM and OVA-LR. The heights of the light blue bars represent the per-segment AUCs obtained by MLFM, while the light red bars represent those obtained by OVA-LR. The segments on the x-axis are ordered by MLFM's per-segment AUCs.

We can initially observe that improvements are introduced on the majority of segments as it is visible that the blue bars in Figure 2 generally surpass the red ones. This is the case for 1069 out of the 1098 conversion-based segments, which translates into slightly more than 97% of all segments. Finally, larger performance gaps are attained for segments on which OVA-LR typically manifests lower AUCs, as such segments have fewer training examples. This behavior can be attributed to the ability of MLFM to jointly estimate parameters across all segments as opposed to OVA-LR that learns an LR model for each segment separately.

Performance on Real Traffic. We compared the performance of OVA-LR and MLFM on real traffic for a period of one week. Conversions are direct indicators of the performance of the models, since they capture the users' interest and affinity. Moreover, the ad campaigns that use these segments are optimized for conversions. With that in mind, we compared the number of conversions attributed to the OVA-LR and MLFM models. Overall, MLFM attained a 5.32% lift in the number of conversions. Out of the 1098 segments, OVA-LR had more conversions only on nine segments by about 0.19%.

4.4.2 Interest-Based Targeting. To further inspect the effectiveness of multi-label modeling for targeting segments, MLFM was additionally applied to a secondary real-world use case. As opposed to the use case presented in Section 4.4.1, this use case concerns the assignment of a user to multiple, hierarchical segments based on the user's interests. We refer to these targeting segments as *interest-based* or simply *interest segments*.

Both the training and test datasets were labeled with their corresponding interest categories declared for the users. The interest categories were derived from the users' affinity to each category. Similar to the conversion-based targeting application, each record was described by 200,000 features representing one-hot encodings

Table 5: Offline predictive performance, summarized over all 425 interest-based targeting segments.

Model	Macro Test AUC	Strat. Test AUC
OVA-LR	0.6865	0.6048
MLFM	0.8522	0.8519

Table 6: Inference time per record (in milliseconds).

Model	Inference time (ms)
OVA-LR	0.067447
MLFM (original)	0.159827
MLFM (lightweight)	0.101286

of the fields and was labeled with interest categories corresponding to 425 interest-based segments (the distribution of the segment qualifications is provided in Section A.3). MLFM and OVA-LR were trained on 180 million of the records, while 45 million records were used to assess their predictive performances. The prediction AUCs of both models were then measured for each individual interest segment; their overall performances across all of the segments are summarized in Table 5.

Table 5 indicates that MLFM outperforms OVA-LR by a substantial margin. More specifically, MLFM introduces an overall improvement of 16.6% in macro-average AUC and 24.7% in stratified AUC (weighed by the number of qualified records for each of the segments). These findings suggest that the MLFM model is capable of achieving reasonable performance on hierarchically organized segments as well, even on rare classes as indicated by the improved stratified AUC. This can be attributed to MLFM's capability to take advantage of feature interactions in the dataset, as there is a large overlap between parent and child segments. We observed that the model converged within 1 epoch, which also might be a consequence of the hierarchical relationships and overlap among the segments.

4.4.3 Runtime Considerations. This section encompasses aspects that should be considered when running MLFM, such as inference time, memory consumption and sensitivity to hyperparameters.

Inference Analysis. The efficiency of the models was evaluated by computing the run times of their inference (i.e. prediction) procedures on the conversion-based targeting dataset. For the MLFM model variants, we have set both M and H to 10. The measured inference times were averaged over all records and reported in terms of milliseconds (ms).

From Table 6, it can be observed that OVA-LR exhibits the lowest latency, which is to be expected due to its low model complexity. The MLFM model, using its original formulation (Eq. (7)) is slower than OVA-LR. However, when the lightweight formation from Eq. (12) is used for inference, MLFM's latency drops considerably. The reason is that the asymptotic complexity of MLFM becomes linear in terms of the number of fields $\hat{D}$ (see Section 3).

Memory Requirements. When it comes to memory consumption, all parameters of MLFM are organized into four parameter sets in memory. However, recall that MLFM can essentially be

Table 7: Memory requirements for OVA-LR and MLFM.

Model	Parameters	Dimensions	Example	Size
OVA-LR	Feature weights & Bias terms	$(D+1) \times L$	200K × 1098	~**3.9 GB**
MLFM	Feature weights	$D \times L$	200K × 1098	3.9 GB
	Bias terms	$1 \times L$	1 × 1098	20.4 KB
	Feature embeddings	$D \times M$	200K × 10	48.7 MB
	Interaction weights	$\hat{D}^2 \times L$	324 × 1098	6.3 MB
	Grand total			~**4.00 GB**

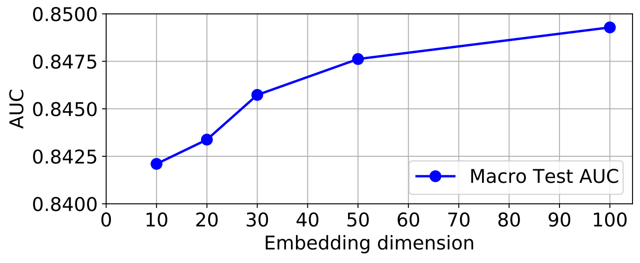
thought of as a joint extension of per-segment models with additional modeling of feature interactions. That being said, OVA-LR already consumes the same memory as MLFM's linear projection weights and bias terms (totaling slightly over 3.9 GB), hence MLFM requires only several tens of additional megabytes (or around 2.5% of additional memory) to store the remaining feature embeddings and field interaction weights.

Hyperparameter Sensitivity. The sensitivity of MLFM to the selection of the feature embedding dimension, being MLFM's central hyperparameter, was also analyzed. For this purpose, MLFM was trained using different embedding dimensions, starting from 10, through 20, 30, 50, up to 100. In each case, MLFM's multi-label classification performance as well as its memory consumption were measured.

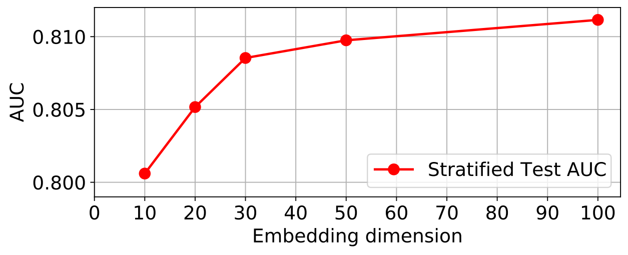
From Figures 3a and 3b, it can be initially observed that the predictive performance increases with the feature embedding dimension. This observation holds across the great majority of segments as indicated by the change in macro-average AUC. On the other hand, the slight increase in stratified AUC suggests that higher values of M impacted only the performance on rare classes (segments with less prevalent training examples), positively in some cases and adversely in others. However, despite the relative and rather incremental increase, there does not seem to be a significant change in the AUC metrics.

More precisely, using a larger embedding dimension will result in only a slight performance improvement (~1% or less) at the cost of the inevitable increase in model complexity, and thus in latency. This suggests that selecting a larger embedding dimension would not be necessary, which is also the main reason behind the selection of $M = 10$ in all of the previous experiments.

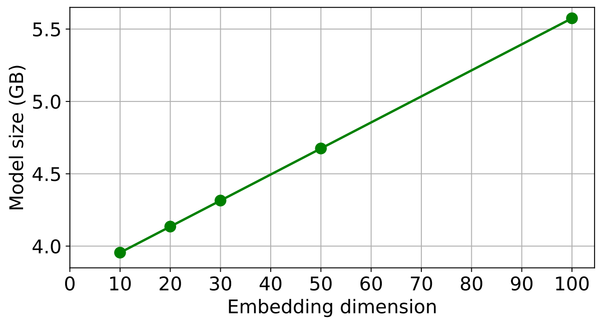
Further, Figure 3c shows that the required memory to store MLFM's parameters increases linearly with the embedding dimension since, apart from the feature embeddings of size $D \times M$, the other model parameters are not effected by changes in M (recall Table 7). Nevertheless, as it was previously observed that a dimension as low as 10 should be sufficient for attaining satisfactory classification performance, there is no need to further expand the model size.



(a)



(b)



(c)

Figure 3: Effect of different embedding dimensions on MLFM's classification performance on the conversion-based targeting dataset in terms of (a) macro-average and (b) stratified AUC; as well as the corresponding impact on (c) the memory required for storing MLFM's parameters.

5 CONCLUSION

In this paper, we presented an approach, Multi-Label Factorization Machine (MLFM), that extends the Factorization Machine (FM) model in order to handle extreme multi-label classification tasks with relative ease. The proposed MLFM model is capable of handling a large number of labels without compromising prediction accuracy. We used behavioral ad targeting as a case study, and presented the results of applying our model to assign users to targeting segments. We also demonstrated the performance of our model on benchmark datasets in comparison with several baseline models. As a part of our future work, among other open problems in XMLC, we aim to extend MLFM to use cases involving label hierarchies (as briefly discussed in Section 4.4.2) and further mitigate the ubiquitous challenge of modeling label distribution tails for rare-class prediction.

ACKNOWLEDGMENTS

We would like to thank Krzysztof Dembczynski, Narayan Bhamidipati, Yifan Hu and Jimmy Yang for their valuable suggestions.

REFERENCES

[1] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*. 13–24.

[2] Rohit Babbar and Bernhard Schölkopf. 2017. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the tenth ACM international conference on web search and data mining*. 721–729.

[3] Samy Bengio, Krzysztof Dembczynski, Thorsten Joachims, Marius Kloft, and Manik Varma. 2019. Extreme classification (dagstuhl seminar 18291). In *Dagstuhl Reports*, Vol. 8. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[4] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. 2016. The extreme classification repository: Multi-label datasets and code. http://manikvarma.org/downloads/XC/XMLRepository.html

[5] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems* 28 (2015).

[6] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. Large-scale multi-label text classification on EU legislation. *arXiv preprint arXiv:1906.02192* (2019).

[7] Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, Japinder Singh, and Inderjit S. Dhillon. 2021. Extreme Multi-Label Learning for Semantic Matching in Product Search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2643–2651.

[8] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. 2020. Taming pretrained transformers for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3163–3171.

[9] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. 2010. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research* 11, 4 (2010).

[10] Chen Chen, Haobo Wang, Weiwei Liu, Xingyuan Zhao, Tianlei Hu, and Gang Chen. 2019. Two-stage label embedding via neural factorization machine for multi-label classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3304–3311.

[11] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. 2012. On label dependence and loss minimization in multi-label classification. *Machine Learning* 88, 1 (2012), 5–45.

[12] Wei Deng, Junwei Pan, Tian Zhou, Deguang Kong, Aaron Flores, and Guang Lin. 2021. Deeplight: Deep lightweight feature interactions for accelerating ctr predictions in ad serving. In *Proceedings of the 14th ACM international conference on Web search and data mining*. 922–930.

[13] Google. 2023. Google Ad Targeting. https://support.google.com/google-ads/answer/1704368

[14] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 1725–1731.

[15] David J Hand and Robert J Till. 2001. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine learning* 45, 2 (2001), 171–186.

[16] Trevor Hastie, Robert Tibshirani, and Jerome H Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer.

[17] Chih-Wei Hsu and Chih-Jen Lin. 2002. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks* 13, 2 (2002), 415–425.

[18] IAB. 2020. IAB Taxonomy. Retrieved February 1, 2023 from https://iabtechlab.com/standards/audience-taxonomy/

[19] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 935–944.

[20] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hullermeier. 2016. Extreme f-measure maximization using sparse probability estimates. In *International conference on machine learning*. PMLR, 1435–1444.

[21] Yacine Jernite, Anna Choromanska, and David Sontag. 2017. Simultaneous learning of trees and representations for extreme classification and density estimation. In *International Conference on Machine Learning*. PMLR, 1665–1674.

[22] Yuchin Juan, Damien Lefortier, and Olivier Chapelle. 2017. Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 680–688.

[23] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM conference on recommender systems*. 43–50.

[24] Nikos Karampatziakis and Paul Mineiro. 2015. Scalable multilabel prediction via randomized methods. *arXiv preprint arXiv:1502.02710* (2015).

[25] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[26] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1754–1763.

[27] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C Weng. 2007. A note on Platt's probabilistic outputs for support vector machines. *Machine learning* 68, 3 (2007), 267–276.

[28] Weiwei Liu, Haobo Wang, Xiaobo Shen, and Ivor Tsang. 2021. The emerging trends of multi-label learning. *IEEE transactions on pattern analysis and machine intelligence* (2021).

[29] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[30] Eneldo Loza Mencía and Johannes Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 50–65.

[31] Donna Katzman McClish. 1989. Analyzing a portion of the ROC curve. *Medical decision making* 9, 3 (1989), 190–195.

[32] Meta. 2023. Facebook Ad Targeting. https://www.facebook.com/business/ads/ad-targeting

[33] Paul Mineiro and Nikos Karampatziakis. 2015. Fast label embeddings for extremely large output spaces. *arXiv preprint arXiv:1503.08873* (2015).

[34] Alexandru Niculescu-Mizil and Ehsan Abbasnejad. 2017. Label filters for large scale multilabel classification. In *Artificial intelligence and statistics*. PMLR, 1448–1457.

[35] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2876–2885.

[36] Junwei Pan, Yizhi Mao, Alfonso Lobos Ruiz, Yu Sun, and Aaron Flores. 2019. Predicting different types of conversions with multi-task learning in online advertising. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2689–2697.

[37] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In *Proceedings of the 2018 World Wide Web Conference*. 1349–1357.

[38] Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 441–449.

[39] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*. 993–1002.

[40] Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 263–272.

[41] Piyush Rai, Changwei Hu, Ricardo Henao, and Lawrence Carin. 2015. Large-scale bayesian multi-label learning via topic-based label embeddings. *Advances in neural information processing systems* 28 (2015).

[42] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[43] Ryan Rifkin and Aldebaro Klautau. 2004. In defense of one-vs-all classification. *The Journal of Machine Learning Research* 5 (2004), 101–141.

[44] Alan F Smeaton. 2005. Large scale evaluations of multimedia information retrieval: The TRECVid experience. In *International Conference on Image and Video Retrieval*. Springer, 11–17.

[45] Alan F Smeaton, Paul Over, and Wessel Kraaij. 2004. TRECVID: Evaluating the effectiveness of information retrieval tasks on digital video. In *Proceedings of the 12th annual ACM international conference on Multimedia*. 652–655.

[46] Cees GM Snoek, Marcel Worring, Jan C Van Gemert, Jan-Mark Geusebroek, and Arnold WM Smeulders. 2006. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th ACM international conference on Multimedia*. 421–430.

[47] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[48] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-matrixed factorization machines for recommender systems. In *Proceedings of the Web Conference 2021*. 2828–2837.

[49] Piotr Szymański, Tomasz Kajdanowicz, and Nitesh Chawla. 2018. LNEMLC: Label network embeddings for multi-label classification. *arXiv preprint arXiv:1812.02956* (2018).

[50] Yukihiro Tagami. 2017. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining.* 455–464.

[51] Willem Waegeman, Krzysztof Dembczyński, and Eyke Hüllermeier. 2019. Multi-target prediction: a unifying view on problems and methods. *Data Mining and Knowledge Discovery* 33, 2 (2019), 293–324.

[52] Chang Xu, Dacheng Tao, and Chao Xu. 2016. Robust extreme multi-label learning. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* 1275–1284.

[53] Yahoo. 2023. Yahoo Ad Targeting. https://www.advertising.yahooinc.com/our-dsp/target

[54] Eric Ye, Xiao Bai, Neil O'Hare, Eliyar Asgarieh, Kapil Thadani, Francisco Perez-Sorrosal, and Sujyothi Adiga. 2022. Multilingual Taxonomic Web Page Classification for Contextual Targeting at Yahoo. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 4372–4380.

[55] Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. 2017. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 545–553.

[56] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. 2016. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International conference on machine learning.* PMLR, 3069–3077.

[57] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in Neural Information Processing Systems* 32 (2019).

[58] Hsiang-Fu Yu, Jiong Zhang, Wei-Cheng Chang, Jyun-Yu Jiang, Wei Li, and Cho-Jui Hsieh. 2022. Pecos: Prediction for enormous and correlated output spaces. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 4848–4849.

[59] Jiong Zhang, Wei-Cheng Chang, Hsiang-Fu Yu, and Inderjit Dhillon. 2021. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *Advances in Neural Information Processing Systems* 34 (2021), 7267–7280.

A SUPPLEMENTAL MATERIAL

A.1 Implementation Details

For the benefit of facilitating reproducibility, we provide a pseudocode for MLFM's training and inference procedures in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 MLFM's Training Framework

```
Input:
    Training set $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ (transformed to LibSVM format)
    Set of fields $\{F_1, F_2, \ldots, F_{\hat{D}}\}$
    Number of training epochs $T$
    Batch size $B$
    Embedding dimension $M$
Training procedure:
 1: Randomly initialize the model parameters $\mathbf{\Theta} = (\mathbf{w}_0, \mathbf{W}, \mathbf{R}, \mathbf{V})$
 2: Build a feature vocabulary $S_{feat} = \{f_1, f_2, \ldots, f_D\}$
 3: Create a feature weight lookup $\omega : S_{feat} \rightarrow \mathbb{R}^L$
 4: Create a feature embedding lookup $g : S_{feat} \rightarrow \mathbb{R}^M$
 5: for $t = 1, \ldots, T * B$ do
 6:     $(\mathbf{X}, \mathbf{Y}) \leftarrow get\_next\_batch\left(\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N\right)$
 7:     $\mathbf{\Phi} \leftarrow MLFM\_inference(\mathbf{X}, \mathbf{\Theta}, \omega, g)$   // Based either on
        // Alg. 2 or its lightweight counterpart (Eq. (12))
 8:     $\mathbf{P} \leftarrow sigmoid(\mathbf{\Phi})$   // Refer to Eq. (6)
 9:     $loss \leftarrow \ell(\mathbf{P}, \mathbf{Y})$   // Refer to Eq. (8)
10:     Update model parameters $\mathbf{\Theta}$ based on the current $loss$
Output: Return the optimized $\mathbf{\Theta}^*$
```

Algorithm 2 Inference with MLFM

```
Input:
    Batch $\mathbf{X}$ of $B$ data points
    Model parameters: $\mathbf{\Theta} = (\mathbf{w}_0, \mathbf{W}, \mathbf{R}, \mathbf{V})$
    Map $\omega$ that retrieves a feature's weights w.r.t. all labels from $\mathbf{W}$
    Map $g$ that retrieves a given feature's embedding from $\mathbf{V}$
Procedure:
    // Calculate the linear projection term
 1: $\hat{\mathbf{W}} \leftarrow [\underbrace{\omega(x_{nj})}_{1\times L}]_{n=1,j=1}^{B,\hat{D}}$   // size: $B \times \hat{D} \times L$
 2: $\hat{\mathbf{W}} \leftarrow [\sum_{j=1}^{\hat{D}} \hat{w}_{njl}]_{n=1,l=1}^{B,L}$   // size: $B \times L$
 3: $\mathbf{\Phi}^{lin} \leftarrow \hat{\mathbf{W}} + \underbrace{[\mathbf{w}_0, \ldots, \mathbf{w}_0]}_{\text{repeat } B \text{ times}}^\top$   // size: $B \times L$
    // Calculate the feature interaction term
 4: $\hat{\mathbf{V}}_n \leftarrow [\underbrace{g(x_{nj})}_{1\times M}]_{j=1}^{\hat{D}}, \forall n = 1, \ldots, B$   // $B$ matrices of size: $\hat{D}\times M$
 5: $\hat{\mathbf{W}}^{inter} \leftarrow [\hat{\mathbf{V}}_n \hat{\mathbf{V}}_n^\top]_{n=1}^B$   // size: $B \times \hat{D} \times \hat{D}$
 6: $\mathbf{\Phi}^{inter} \leftarrow [\sum_{j=1}^{\hat{D}} \sum_{k=j+1}^{\hat{D}} \hat{w}_{njk}^{inter} r_{jk}^l]_{n=1,l=1}^{B,L}$   // size: $B \times L$
Output: Return $\mathbf{\Phi}^{lin} + \mathbf{\Phi}^{inter}$
```

Lightweight Inference. The inference procedure described in Algorithm 2 is based on the standard MLFM formulation (Eq. (7)). In terms of model parameters, the only difference for the lightweight formulation would be the use of the field embedding matrix $\mathbf{U}$ of size $\hat{D} \times H \times L$ in place of the field interaction weight matrix $\mathbf{R} = [r_{F(i)F(j)}^l]_{\hat{D}\times\hat{D}\times L}$. Since the linear projection term in the lightweight formulation (Eq. (12)) remains the same as the one in the standard formulation (Eq. (7)), Lines 1-3 in Algorithm 2 would remain unchanged.

Only the interaction term (Lines 4-6) would need to be replaced by calculating Eq. (11) for each data point $\mathbf{x}_n$ in the batch $\mathbf{X}$. In this regard, two additional optimizations can be applied from an implementation perspective, as described in the following.

First, recall that the $\mathbf{Q}_i^l = [u_{F(i)h}^l v_{im}]_{h=1,m=1}^{H,M}$ matrices are calculated using the field and feature embeddings in $\mathbf{U}$ and $\mathbf{V}$, respectively. Since, upon training, the optimal $\mathbf{U}^*$ and $\mathbf{V}^*$ will not change, the $\mathbf{Q}_i^l$ matrices for the most frequent features can be precomputed and organized in a lookup table into memory. That being provided, during inference, the values q_{ik}^l can be retrieved in constant time from the corresponding $\mathbf{Q}_i^l$ for the highly prevalent features and included in the sum from Eq. (11).

Second, irrespective of whether the inference is applied to a single data point or a batch of data, the linear term and the interaction term can be computed independently. Consequently, these tasks can be parallelized.

Note that the same training framework (Algorithm 1) can be used with the proposed lightweight formulation of MLFM.

A.2 Other Related Approaches

In this section, we would like to address two specific aspects of the related work: (1) recent advances in FM modeling, and (2) extreme classification approaches such as Extreme Multi-label Text Classification (XMTC).

A.2.1 Recent FM Approaches. Various deep FM models [12, 14, 26] have been introduced to jointly learn explicit and implicit (higher-order) feature interactions. Other approaches make effective use of popular deep learning paradigms for the same purpose. For example, AutoInt [47] utilizes attention mechanisms and residual networks into a multi-head self-attentive architecture to automatically learn high-order feature interactions. Further, the FM2 approach [48] was designed to model field interactions and the authors devised a unified framework of the factorization machine model family, in which other FM variants can be derived as special cases. Almost all of the aforementioned approaches were originally proposed for tasks such as click-through rate (CTR) prediction in the realm of recommender systems and online advertising. However, they cannot be applied to XMLC problems, at least not directly.

In contrast, Neural FMs (NFMs) [10] have been explored for conventional multi-label classification in relatively small label spaces. The framework incorporates several hidden layers, applied on top of two separate FMs, to encode both the features and labels, as well as another set of hidden layers to reconstruct the labels from their embeddings. Lastly, the two Neural FMs in the encoding stage are based on the original FM formulation that does not account for different data fields.

As a result of being rather sophisticated, the afore discussed deep FM variants were not considered in our work as they require additional parameterization that will in turn impact memory, storage, and latency, which are the primary challenges in our use case (described in Section 1).

A.2.2 XMTC Approaches. Although beyond the scope of our work, it is worth mentioning that there is another active area of research that concerns XMC problems based on textual input [7, 8, 35, 57–59], also referred to as Extreme Multi-label Text Classification (XMTC) problems.

As the name suggests, XMTC approaches are designed for applications where the input is unstructured text, and often employ state-of-the-art architectures such as bi-directional LSTMs and Transformers. Consequently, these approaches utilize the power of GPUs for both model training and inference in order to achieve reasonable speed up. However, the cost of such infrastructure is not warranted in all applications, and some applications rely on lightweight models to meet the latency requirements in production. Moreover, it is not trivial to adapt such approaches for tabular data.

Further, some of these approaches were built to utilize an explicit label hierarchy or correlations in the output space. This is not always the case in most recommendation and ad targeting applications, where such hierarchies may have to be inferred from the data, or may be entirely absent.

As a result, such approaches are not directly applicable to the use cases we focus on in this paper. However, we provide a performance comparison with PECOS [58] as a representative of the family of recent XMTC approaches. More precisely, we considered the XR-Linear instantiation of PECOS as a baseline since it can take tabular numerical data as input; the other instantiations require the presence of textual input. The comparison was conducted on the same public benchmark datasets discussed in Section 4.3.

Note that we have used a tabular (bag-of-words) feature representation for the datasets, i.e. the same representation with which they were originally provided in [4]. The results are summarized in Table 8. Also, as a reminder, we considered standard classification metrics such as AUC instead of metrics such as nDCG@k and Precision@k since we approach the problems described in this paper from a standard multi-label classification perspective rather than a ranking perspective.

Table 8: Classification performance in terms of AUC obtained by the XR-Linear instantiation of PECOS and MLFM on the publicly available datasets.

Model	**MediaMill**		**RCV1**		**EURLex**	
	Macro	Stratified	Macro	Stratified	Macro	Stratified
XR-Linear	0.6584	0.7867	0.8138	0.9375	0.6450	0.9031
MLFM	0.8456	0.8248	0.9179	0.9808	0.9613	0.9847

Table 8 suggests that MLFM obtains lifts in classification performance relative to that of XR-Linear. Nevertheless, it should be noted that XR-Linear is primarily designed and intended for ranking, not necessarily for conventional XMLC problems, hence its lower performance in terms of standard classification metrics (such as macro-average and stratified AUC) which are considered the main performance indicators for our problem setting.

A.3 Additional Details on the Interest-Based Targeting Dataset

In Figure 4, we provide the label distribution for the interest-based targeting dataset.

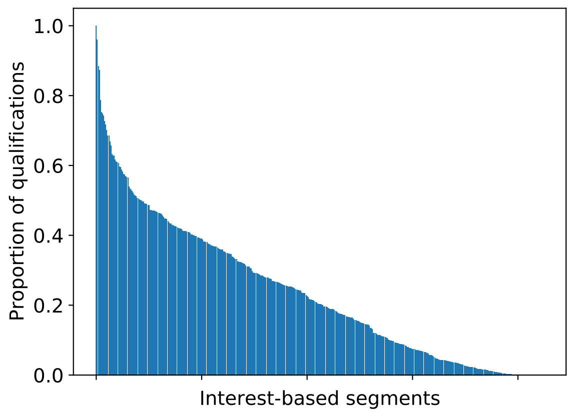


Figure 4: Distribution of labels (segment qualifications) in the interest-based targeting dataset.