Leveraging Group-Level Signals for Robust Many-Domain Generalization

First Author$^{1[0000-1111-2222-3333]}$, Second Author$^{2,3[1111-2222-3333-4444]}$, and Third Author$^{3[2222--3333-4444-5555]}$

No Institute Given

Abstract. Many-domain generalization poses a significant challenge for machine learning models, as real-world data often comprises a vast number of domains in both training and unseen test sets. Traditional approaches that focus on a limited number of domains fail to capture this complexity, leading to suboptimal performance and poor generalization. To address this issue, we propose a novel group-wise reweighting strategy that leverages diverse group-level features, such as label entropy, representation statistics, and gradient properties. By determining the relative importance of groups during training based on these features, our approach effectively tackles the limitations of existing methods that rely solely on group error. Our results demonstrate substantial improvements in worst-group and tail performance on out-of-sample test data across multiple datasets, highlighting the efficacy of our selective upweighting strategy. To further enhance performance, we adopt a learning-to-rank framework that integrates multiple group features. While this approach achieves significant gains over empirical risk minimization, the challenges in consistently outperforming individual features underscore the inherent difficulties in achieving transferable robustness amid the varying group characteristics and distributions across datasets. SHAP analysis confirms the heterogeneous importance of different features across datasets, emphasizing the need for adaptive strategies. Our work presents a promising avenue for developing robust models in the challenging many-domain generalization setting, paving the way for future research in this area.

Keywords: Many Domain Generalization · Group Features · Group Ranking and Reweighting

1 Introduction and Related Work

The increasing prevalence of datasets containing observations from a vast number of distinct groups or domains poses a significant challenge in the real-world applications of machine learning. Examples of such data include users in online review systems [14], patients in healthcare settings [35], and IoT sensors deployed in a city to collect air quality data [18]. In these scenarios, users, patients, and sensors constitute the different data domains or groups, respectively[1]. The classification tasks involving these types of data are particularly challenging because

[1] We use the terms groups and domains interchangeably in this paper.

there are often unseen or out-of-distribution (OOD) domains at test time, meaning the model must generalize to new domains that were not encountered during training. Researchers have attempted to frame this problem as domain generalization (DG), where the goal is to learn a model that can perform well on previously unseen domains, such as new users or patients [32].

Current DG methods can be broadly grouped into three main categories. The first set of approaches, called *domain matching techniques*, focus on achieving invariance at different levels, including feature invariance [31,23,10,11], classifier invariance [3,1,36], and style augmentation [25,13]. These techniques aim to learn domain-invariant representations or classifiers to facilitate generalization to new domains. The second class of methods includes *reweighting strategies*, which can be further divided into two categories: 1) reweighting individual examples based on example errors [19,5,8] or gradients [27], and 2) reweighting groups based on group features [29,3]. These methods assign different weights to training examples or groups to emphasize the importance of certain samples or domains during training. However, these methods may overlook valuable group-level information and overly rely on singular group error metrics. The third class of methods, called *meta-learning methods* [17,4,35], address DG by simulating train/test domain shifts during episodic training, fostering a model's ability to adapt to new domains with minimal data.

Despite the progress made by these DG approaches, they have primarily focused on scenarios involving a limited number of training and/or test domains. However, real-world applications often involve numerous domains, ranging from hundreds to millions. For instance, consider the task of online review rating classification [26,22], where users have a wide range of demographics, purchasing habits, and product preferences. These diverse characteristics significantly influence users' rating patterns, rendering the traditional approach of categorizing them into a limited number of domains ineffective. Even though some existing approaches don't limit the number of users (domains), scaling them to many users is challenging. For example, for domain matching methods [31,23,3], performing the matching process for every pair of users is computationally expensive. Even if the matching is done within each batch, it may not be effective because users with fewer reviews are less likely to be present in each batch than users with more reviews, leading to suboptimal invariance learning.

Besides that, the emergence of numerous new users (domains) at test time, unseen during training, poses an even bigger challenge for domain generalization (DG) methods. Traditional DG approaches, which typically target only a few test domains, acknowledge that test users may exhibit substantially different rating behaviors compared to the training set due to temporal shifts. However, the presence of numerous test users with diverse and dissimilar preferences within the test set itself, arising from personal biases, further complicates the generalization task. Capturing such differences is more manageable when dealing with a limited number of unseen test domains. For instance, a large shopping promotion during the test period may lead to a homogenization of users' rating behavior, simplifying the DG task compared to the complex task of modeling individual user behavior separately across a large number of domains. However, some DG approaches [2,28,8] assume that the test distribution lies within the convex hull of the training domain distributions, which may not hold true in the many-domain setting. These assumptions limit the applicability of such methods in real-world scenarios where the test distributions can be highly diverse and may not be adequately represented by the training data. Furthermore, in this setting, reporting only the average domain performance is insufficient, and it becomes crucial to employ metrics that cover a wider range of domain performances, especially focusing on poorly performing domains. The discrepancy between the theoretical assumptions of traditional DG and the practical realities of many-domain scenarios necessitates the development of new approaches capable of efficiently handling a large number of domains while also considering the potential presence of minority groups.

In this work, we focus on the DG framework while exploring the nuanced landscape of DG within the context of many-domain scenarios, formally defining the **many-domain generalization** problem that has been rarely explored in the literature. We propose treating each individual entity, such as a user in the review system mentioned above, as a unique domain to capture the intricate nuances in their behavior. This breaks away from the conventional DG methods and embraces datasets with numerous training domains, emulating the reality of having data from a diverse user base. We also delve into the setting of numerous unseen domains at test time. In this setting, inspired by [14], we prioritize tail performance metrics, such as the worst domain's performance or the 10th percentile performance, over average metrics. This focus ensures the model's robustness for the most challenging domains, such as those with highly specific preferences or inconsistent rating patterns. To navigate this complexity, we introduce the concept of 'group' as an interchangeable term with 'domain' under this challenging setting, emphasizing the diversity of user segments. We believe that using 'group' to describe a domain in the new setting to distinguish it from the traditional setting more accurately reflects the nature of the data, particularly when discussing group features. In the above example, a group contains all reviews from a single user, and a dataset contains a multitude of such groups.

This ambitious setting not only challenges the limits of existing DG methodologies but also motivates us to develop a method that explicitly incorporates group information. Hence, we propose a novel approach for group-wise reweighting by leveraging diverse group features, such as group label entropy, group gradient statistics, and group size, among others introduced in §3.1. By doing so, we aim to achieve enhanced generalization across numerous groups. Our results demonstrate that reweighting based on the ranking of features other than group error yields significantly better or comparable performance across datasets, with features like group label entropy and group representation statistics achieving improvements of over 10% in worst-group performance on multiple datasets. Next, we adopt a Learning to Rank (LTR) framework to explore the potential of incorporating multiple features jointly for group-wise reweighting. While using multiple features together via LTR can significantly improve upon Empirical Risk Minimization (ERM), it does not necessarily outperform using each feature individually, likely due to the varying distributions of different features across datasets and their differing impacts on group heterogeneity. SHapley Additive exPlanations (SHAP) analysis confirms the varying importance of different features across datasets, highlighting the challenges in achieving transferable robustness. In summary, our main contributions include:

– Define and explore the challenging but practically relevant setting of many-domain generalization.
– Propose a novel group-wise reweighting method using diverse group features for many-domain generalization.
– Leveraging the learning-to-rank (LTR) framework to develop a new method to incorporate multiple group features jointly for reweighting for many-domain generalization.
– Uncovering heterogeneity of group features across different datasets via SHAP analysis.

Our findings highlight the importance of considering various group features beyond group error for many-domain generalization. We hope this research opens up new avenues for future work in domain generalization, emphasizing the need for adaptive strategies to capture the heterogeneous importance of group-level features across domains.

2 Problem Formulation

While many methods excel at traditional domain generalization tasks, they perform poorly on the more challenging task of **generalization to many-domains** [15]. The many-domain generalization problem requires the following four conditions:

(1) The training set includes numerous domains (or groups) (often ≥ 100), instead of only a handful of domains;
(2a) The test set also has numerous domains (or groups), instead of only one test domain as in most domain generalization tasks;
(2b) As a result of (2a), from a "fairness" perspective, the focus extends beyond average performance to include tail performance, such as the worst test domain's performance and the 10th percentile performance;
(3) The test domains are out-of-distribution (OOD) unseen domains, instead of the same domains as training as in the subpopulation shift setting.

The **many-domain generalization problem** can be formally defined as follows: We consider the standard supervised learning setup of classifying an input $x \in \mathcal{X}$ as a label $y \in \mathcal{Y}$. The training data comprises groups from a set $\mathcal{G}_{train}$, where each group $g \in \mathcal{G}_{train}$ consists of n_g data points from a probability distribution $P_g(\mathcal{X}, \mathcal{Y})$. Similarly, in the test set $\mathcal{D}_{test}$, there is a different set of groups $\mathcal{G}_{test}$. The groups in $\mathcal{G}_{test}$ are unseen in the training set $\mathcal{G}_{train}$, i.e., $\mathcal{G}_{test} \cap \mathcal{G}_{train} = \emptyset$. It is important to emphasize that the number of groups in the training set $\mathcal{D}_{train}$ and in the test set $\mathcal{D}_{test}$ are numerous. For example, there are at least hundreds of groups for $|\mathcal{G}_{train}|$ and dozens for $|\mathcal{G}_{test}|$. The goal is to learn a model $f_\theta : \mathcal{X} \times \mathcal{G} \rightarrow \mathcal{Y}$ parameterized by $\theta \in \Theta$ so that the optimal model achieves strong average and tail performances on the test set. Many real-world applications face the challenge of many-domain generalization, a problem that holds great significance despite receiving less attention in academia due to its inherent complexity.

Another related framing of the grouped data setting is the "subpopulation shift" problem, which focuses on the performance of the tail (worst) groups [15]. Subpopulation shift occurs when a model trained on unrepresentative data underperforms on certain subgroups, leading to biased outcomes. This framework usually assumes that the domains forming the training and test sets are the same, but they differ greatly in their proportional makeup [6,33]. Addressing subpopulation shifts requires methods that can effectively capture the causal features for the seen domains [3], while generalization for many domains calls for more flexible and adaptive approaches that can handle the diversity and novelty of unseen domains.

The many-domain generalization setting contrasts with the traditional Domain Generalization problem, which satisfies condition (3), violates condition (2b), and is agnostic to conditions (1) and (2a). Subpopulation shift problems violate conditions (2a) and (3) and are agnostic to condition (1). Although the worst test domain performance is considered during evaluation (2b), the problem is simplified to the task of generalizing well across a limited number of seen domains.

Condition	(1) Numerous groups in training set	(2a) Numerous groups in test set	(2b) Focus on tail performance	(3)OOD unseen test groups
"Standard" Domain Generalization	× (usually a handful)	× (usually one)	×	✓
Sub-population Shift	× (usually a handful)	× (usually a handful)	✓	×
Ours	✓	✓	✓	✓

Table 1: Many-Domain Gneralization Problem Setting Summarized

3 Methods

As mentioned earlier, previous approaches to domain generalization have primarily focused on optimizing group error with the hope that it generalizes to test groups. While improving test group performance is the ultimate goal, focusing solely on group error during the training stage may not necessarily lead to better out-of-distribution (OOD) group performance, particularly in the challenging setting of many-domain generalization. Recognizing this limitation, we decided to explore alternative features that could potentially improve the worst-group performance in this demanding context. In the next subsection, we introduce the group features that we consider, which include *group error, group representation standard deviation, group gradient mean, group gradient standard deviation, group size, group label entropy, and group label and error mutual* *information.* Broadly, these features can be categorized based on their dependence on the model and the training process. Some features, such as group size and label entropy, are model-independent and can be computed directly from the dataset without requiring a trained model. Other features, like group error and representation standard deviation, are model-dependent, as they rely on the predictions or learned representations of a trained model. Lastly, features such as group gradient mean and standard deviation are training-dependent, as they are computed using the gradients obtained during the model's training process. Subsequently, in §3.2, we discuss our approach to ranking and reweighting groups based on individual features. Then, in §3.3, we present our learning-to-rank (LTR) approach, which leverages multiple features simultaneously to rank and reweight groups, aiming to capture their combined informative value. By exploring these diverse group-level features, which encompass model-independent, model-dependent, and training-dependent characteristics, and developing novel ranking and reweighting strategies, we aim to provide a more comprehensive and effective approach to tackling the many-domain generalization problem, moving beyond the limitations of relying solely on group error optimization.

3.1 Different Group Features for Many-Domain Generalization

As described in Section §2, assume a group g from the training set $\mathcal{D}_{train}$ contains a set of instances $S_g = [(x_1, y_1), (x_2, y_2), \dots, (x_{n_g}, y_{n_g})]$. A model f is trained on $\mathcal{D}$ and gives the prediction labels to instances of S_g as $[\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{n_g}]$. Additionally, f learns a representation function $\phi : x \mapsto \mathbb{R}^M$ for x_i to predict its label where M is the dimension of the representation.
Group Error: Group error is the most commonly considered feature in domain generalization. The intuition is that groups with poorer performance during training should be assigned higher weights to encourage the model to focus on them and improve their performance. Formally, the group error is defined as $\varepsilon_g = \sum_{i=1}^{n_g} \mathbb{1}\{\hat{y}_i \neq y_i\}$, where $\mathbb{1}\{\cdot\}$ is the indicator function that equals 1 when the predicted label $\hat{y}_i$ does not match the true label y_i for an instance i in group g, and 0 otherwise. By summing up these indicators, we obtain the total number of misclassified instances within the group. This group error metric serves as a straightforward and effective measure of a group's performance, allowing us to identify and prioritize groups that require more attention during the training process. Incorporating group error as a feature in our reweighting scheme enables the model to adaptively focus on challenging groups and improve its generalization ability across different domains.
Group Representation Standard Deviation: Inspired by Krueger et al. [16], we define $d_i^\phi = \|\phi(x_i) - \bar{\phi}_g\|_2$, where $\bar{\phi}_g = \frac{1}{n_g}\sum_{i=1}^{n_g} \phi(x_i)$ is the average representation of all instances in the group. This measures the distance of an instance's representation to the central representation of the group. The group representation standard deviation is then expressed as $\sigma_g^\phi = \sqrt{\frac{1}{n_g}\sum_{i=1}^{n_g}(d_i^\phi - \bar{d}_g^\phi)^2}$, where $\bar{d}_g^\phi = \frac{1}{n_g}\sum_{i=1}^{n_g} d_i^\phi$ is the average distance. σ_g^ϕ reflects the homogeneity of a group, with a higher value indicating greater diversity within the group's representations. This measure can be used to assess the consistency of instances within

a group and potentially identify groups that may be more challenging for the model to learn from. By incorporating the group representation standard deviation as a feature in our reweighting scheme, we aim to prioritize groups with higher homogeneity, as they may be more informative for improving the model's generalization performance across different domains.

Group Gradient Mean & Standard Deviation: Recent work by Ren et al. [27] demonstrates that the gradient contains valuable information about group robustness. In our approach, we compute the gradient of the loss for each instance with respect to the last prediction layer $W \in \mathbb{R}^{M\times T}$ of the model f: $\nabla\mathcal{L}_i = \text{flatten}\left(\frac{\partial\mathcal{L}(f(x_i),y_i)}{\partial W}\right)$, where T is the number of distinct output labels and $\mathcal{L}$ is the loss function. Since the original gradient is also in $\mathbb{R}^{M\times T}$, we flatten the gradient array to $\mathbb{R}^{MT\times 1}$. We then compute the distance between each gradient in a group and the group's mean gradient: $d_i^{\nabla l} = \|\nabla\mathcal{L}_i - \nabla\bar{\mathcal{L}}_g\|_2$, where $\nabla\bar{\mathcal{L}}_g = \frac{1}{n_g}\sum_{i=1}^{n_g}\nabla l_i$ is the group gradient mean. The Group Gradient Standard Deviation is then calculated as: $\sigma_g^{\nabla\mathcal{L}} = \sqrt{\frac{1}{n_g}\sum_{i=1}^{n_g}(d_i^{\nabla\mathcal{L}} - \bar{d}_g^{\nabla\mathcal{L}})^2}$, where $\bar{d}_g^{\nabla\mathcal{L}} = \frac{1}{n_g}\sum_{i=1}^{n_g} d_i^{\nabla\mathcal{L}}$ is the average distance. To reduce the dimensionality of the high-dimensional gradient mean, we apply Principal Component Analysis (PCA) to the raw gradient mean $\nabla\bar{\mathcal{L}}_g$, obtaining a two-dimensional vector $\in \mathbb{R}^2$. We use the absolute values of these two dimensions as our final gradient mean feature, which reflect the relative position of a group's average gradient compared to other groups and allow for flexible transfer across datasets. These gradient-based features enable us to assess the homogeneity and distinctiveness of each group, guiding the reweighting process to improve worst-group performance in the challenging many-domain generalization setting.

Group Size: The most natural way to improve group robustness is by balancing the size of groups in a dataset [28]. The group size should be informative of the importance of groups in the training time. Formally, it is the total number of instances n_g as defined above.

Group Label Entropy: The group label entropy, defined as $H_g(Y) = -\mathbb{E}[-\log_2(P_g(Y))]$ where $P_g(Y)$ represents the label distribution within group g, is expected to reflect the spurious correlation between group features and labels, which can cause the model to suffer from distribution shifts [24,9]. Unlike previous studies on spurious correlation that define groups using binary and known or synthetic spurious features [28,19,29], in many-domain settings, spurious correlations can be much more numerous and high-dimensional, making explicit definition nearly impossible. We propose using group label entropy as a simple metric to capture the spuriousness of a group, with the intuition that groups with higher label entropy may suffer less from spurious correlations due to a more diverse label distribution suggesting a weaker dependence between group features and labels. Incorporating group label entropy as a feature in our reweighting scheme allows us to prioritize groups with higher label entropy, potentially improving the model's ability to handle distribution shifts without explicitly defining or identifying spurious correlations.

Group Label & Error Mutual Information: Although label entropy reflects spurious correlation, it does not necessarily lead to poor group performance, as the model may learn to disentangle group features and labels independently. We aim to give more importance to groups that have both strong spurious correlations and poor training performance. To achieve this, we use the mutual information of label and error for a group g: $I_g(Y;E) = H_g(Y) - H_g(Y|E)$, where $H_g(Y|E)$ is the conditional entropy of label given error. The conditional entropy is equal to $H_g(Y,E) - H_g(E)$, where $H_g(Y,E)$ is the entropy of the joint distribution of label and error for group g, and $H_g(E)$ is the entropy of the error distribution for g. $I_g(Y;E)$ measures the dependence between error and labels. If group g has a high $I_g(Y;E)$, the model may be unfair to this group, suggesting that upweighting the group during training could be beneficial. By incorporating this mutual information metric as a feature in our reweighting scheme, we can prioritize groups that exhibit both strong spurious correlations and poor performance, potentially improving the model's fairness and generalization ability.

3.2 Ranking and Reweighting Based on Individual Features

Inspired by recent work on group distributional robustness [19,28], we propose a method that performs iterative upweighting of training examples to uniformly compare all features. Our approach iteratively ranks groups by their features and upweights poorly performing groups. The key idea is to upweight training samples from groups with higher feature value rankings, assigning them differential importance based on their rank. The ranking direction of groups according to a feature is determined by its correlation with error. Groups with higher errors are considered more important for reweighting. If a feature positively correlates with error, groups with higher values in this feature are ranked higher.

To map rankings into actual weights for each group, we use a log reverse rank weighting scheme inspired by the Discounted Cumulative Gain (DCG) [12]. Given a group g that ranks at the i^{th} percentile according to the learned rank, its corresponding weight is determined by

$$w_g = \max\left(\frac{\log C}{\log i}, 1\right), \tag{1}$$

where C represents a predefined cutoff threshold. The cutoff ensures that any group ranked above the C^{th} percentile is assigned a weight of 1. This weighting mechanism provides a gradual and logarithmic discount in weight as the group is ranked higher, reflecting a group's relative importance. The cutoff threshold C allows for the saturation of weights beyond a certain rank, ensuring that lower-ranked groups receive equal and default weighting.

Group size and label entropy do not change with training, so reweighting is only applied once. The ranking of groups according to other features is reevaluated after each training epoch and reweighted accordingly. Regardless of the features, group weights are always decided by w_g as described in Equation 1. The first epoch follows standard Empirical Risk Minimization (ERM) without reweighting. We provide a proof of convergence for our reweighting algorithm based on group error ranking in the Appendix (Section §10).

3.3 Combining Features Together: A Learning-to-rank (LTR) Approach

Combining individual features that independently improve performance may further enhance overall performance. However, ranking multiple features directly is not possible in the same way as ranking a single feature. Instead, learning to rank (LTR) can be used to learn the ranking of multiple features. In the traditional LTR task, the input consists of a set of queries Q, a set of documents D, and a set of relevance labels R associated with each query-document pair. For each query $q \in Q$ and document $d \in D$, there exists a relevance label $r \in R$ indicating the document's relevance to the query. The objective is to learn a ranking function $f : Q \times D \to \mathbb{R}$ that, given a new query and a set of documents, produces a ranking score for each document, with higher scores denoting greater relevance.

In our context, let $\mathcal{G}$ represent a set of groups in a dataset $\mathcal{D}$, where each group $g \in \mathcal{G}$ is analogous to a document D in the traditional LTR scenario. A 'query' Q is an assignment of groups: $A : \mathcal{G} \mapsto \mathcal{D}_{train}, \mathcal{D}_{test}$, assigning each group to either the training or test set. For a set of groups $\mathcal{G}$ under assignment A, each group is associated with a relevance label r denoting group importance. We can achieve better tail performance in the test dataset by reweighting groups during training using a function of group importance $h(r)$. The training objective becomes $\mathcal{L}_{weighted} = \sum_{g \in \mathcal{G}} h(r)\mathcal{L}(g)$, where $\mathcal{L}(g)$ is the loss of group g. The goal is to optimize test performance by emphasizing tail groups and reducing reliance on top-performing ones during training. By adopting an LTR approach, we can effectively learn the ranking of groups based on multiple features, enabling us to assign appropriate weights to each group during training. This allows us to focus on the most important groups for improving tail performance in the test dataset, potentially leading to better generalization across domains.

Our objective is to robustly learn an LTR model from a source dataset $\mathcal{D}$ that maps groups to a ranking of relevance r under an assignment A. This model can then be applied to reweight groups in other datasets for better generalization. Unlike traditional domain adaptation, we expect the LTR model to learn to rank groups by capturing useful group features jointly, enabling it to rank groups in new datasets. To achieve this, we need to construct LTR training data from the source dataset, including group features and their associated rankings. In the following sections, we introduce the designed group features, the pseudo-ranking of groups for training, and the LTR training procedure once features and rankings are obtained.

Let $\mathcal{D}$ be our source dataset where each instance x_i is associated with a specific group g_i. Our goal is to ensure no overlap of groups between training and testing datasets while introducing distribution shifts for evaluation purposes. The steps to construct the LTR training data are as follows:

Step 1: Group-wise Splitting. The original dataset $\mathcal{D}$ is group-wise split into a training set $\mathcal{D}_{\text{train}}$ and a testing set $\mathcal{D}_{\text{test}}$ so that no group is shared between $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$. As mentioned below, we used existing data that is already split and ensured the distribution shift from training to test, as there are large performance gaps between in-domain and out-of-domain data.

Step 2: Generating Training Subsets. To create K distinct distribution shifts from training to test, we sample K different subsets of groups from $\mathcal{D}_{\text{train}}$. Let the set of all groups in $\mathcal{D}_{\text{train}}$ be $\mathcal{G}_{\text{train}}$ and all groups in $\mathcal{D}_{\text{test}}$ be $\mathcal{G}_{\text{test}}$. For each $k \in 1, 2, ..., K$, we form a subset of groups $\mathcal{G}^{(k)}_{\text{train}}$ by randomly sampling from $\mathcal{G}_{\text{train}}$ without replacement.

Each training subset $\mathcal{D}^{(k)}_{\text{train}}$ then includes instances of all $g_i \in \mathcal{G}^{(k)}_{\text{train}}$. Hence, each $\mathcal{D}^{(k)}_{\text{train}}$ is a distinct collection of groups, ensuring different distribution shifts relative to $\mathcal{D}_{\text{test}}$. Additionally, we ensure that the number of groups in each subset $\mathcal{G}^{(k)}_{\text{train}}$ is consistent across all K subsets. Here, one pair of $\mathcal{G}^{(k)}_{\text{train}}$ and $\mathcal{G}_{\text{test}}$ corresponds to one assignment of groups $A^{(k)}$ in the set of groups $\mathcal{G}^{(k)} = \mathcal{G}^{(k)}_{\text{train}} \cup \mathcal{G}_{\text{test}}$. For all assignments, we keep the test data the same ($\mathcal{G}_{\text{test}}$) as the goal is to create enough different distribution shifts from training to test that bring different LTR training samples.

Step 3: Getting LTR Training Features. For each subsampled training set $\mathcal{D}^{(k)}_{\text{train}}$, we first train a model f_k on $\mathcal{D}^{(k)}_{\text{train}}$. Then, for each group g in $\mathcal{D}^{(k)}_{\text{train}}$, we evaluate f_k to obtain the seven features for LTR training mentioned in §3.1: $x^{(k)}_g = (\varepsilon^{(k)}_g, \sigma^{\phi(k)}_g, \nabla\bar{\mathcal{L}}^{(k)}_g, \sigma^{\nabla\mathcal{L}(k)}_g, n^{(k)}_g, H^{(k)}_g(Y), I^{(k)}_g(Y;E))$. Let $X^{(k)} = \{x^{(k)}_g | g \in \mathcal{G}^{(k)}_{\text{train}}\}$.

Step 4: Get LTR Pseudo Rank. Our objective in this step is to estimate the relative importance of each group in the subsampled training sets to build training instances for the LTR model. To this end, we create a pseudo rank of the groups based on their prediction errors when a model is trained on $\mathcal{D}_{\text{test}}$:

1. Train a model f_{test} on the test dataset $\mathcal{D}_{\text{test}}$ from Step 1.
2. For each subsampled training set $\mathcal{D}^{(k)}_{\text{train}}$, evaluate f_{test} on it to obtain group-wise errors.
3. Rank all groups in $\mathcal{D}^{(k)}_{\text{train}}$ based on their group errors, with groups having higher errors ranked higher. This ranking produces a pseudo rank, $R^{(k)}$, which serves as an approximation of the group's importance during training.

Given a source dataset $\mathcal{D}_{\text{source}}$, a set of subsampled groups $\mathcal{G}^{(k)}$ obtained via the steps mentioned in §3.3, group assignments $A^{(k)}$, surrogate rankings $R^{(k)}$, and training features $X^{(k)}$, we can train an LTR model for many-domain generalization in the same way as a traditional LTR model for information retrieval. We use ListNet [7] as our LTR model, which is one of the pioneering approaches in learning-to-rank that considers the relation across documents by using list-wise loss instead of pair-wise loss. Once we train the LTR model, we apply it to the training of a new dataset $\mathcal{D}_{\text{target}}$ to rank the importance of new groups in the dataset. We then reweight the new groups according to the learned ranking using the inverse log equation described in Equation 1. During the training of the new dataset, the ranking of groups is reevaluated from the LTR model after each epoch and then reweighted accordingly.

4 Experiments and Results

4.1 Datasets and Experiment Setup

We use three real-world text classification datasets that have been commonly used in prior literature: AMAZON-WILDS [14], Yelp Open Dataset[2], and two variations of CivilComments [14] (CivilComments1 and CivilComments2). The prediction task for Amazon and Yelp datasets is to classify the review text into its corresponding 1-to-5-star rating. For both CivilComments datasets, the task is the binary classification of the toxicity of online posts. Each review or post is associated with a *user* group. The OOD validation and test sets comprise reviews/posts from different sets of users (groups) than the training set. For more preprocessing and descriptive dataset details, please refer to the Appendix 7.

We use Yelp, CivilComments1 and CivilComments2 as the *source dataset* for LTR training and then apply it to the *target dataset* Yelp, Civilcomments1, and Amazon to evaluate LTR models trained from the source. The target datasets are also used to train and evaluate reweighting based on individual features. We don't use Amazon for LTR training due to the relatively large size of the dataset which precluded us from finishing model training in a reasonable amount of time.

We compare our methods against commonly used baseline methods **Empirical Risk Minimization (ERM)**, **Group DRO** [28], and **Just-Train-Twice (JTT)** [19]. We fine-tuned DistilBERT[30] as the base learner for the text classification. The neural ranking model is ListNet[7]. For more details of the experiment setup, please refer to the Appendix 8.

4.2 Results: Using Individual Features

Datasets	**Error**	**Gradient Std.**	**LE**	**Rep. Std.**
Yelp	62.5/$\underline{52.0}$/21.0	62.8/$\underline{52.0}$/19.0	61.9/51.0/23.0	61.9/$\underline{52.0}$/**28.0**
CivilComments1	77.9/64.3/53.8	79.6/70.2/53.8	**80.5**/72.1/**69.2**	79.6/70.8/53.8
Amazon	70.2/$\underline{54.7}$/**17.3**	70.5/53.3/13.3	71.2/$\underline{54.7}$/13.3	71.7/53.3/8.0
MI	**Group Size**	**ERM**	**JTT**	**GroupDRO**
62.8/51.2/23.0	$\underline{63.0}$/$\underline{52.0}$/23.0	$\underline{63.0}$/$\underline{52.0}$/18.0	61.7/51.0/19.0	59.2/49.0/27.0
79.9/66.5/61.5	79.3/$\underline{72.7}$/53.8	80.1/$\underline{72.7}$/53.8	78.4/57.9/53.8	79.3/69.0/61.1
71.4/$\underline{54.7}$/12.0	70.1/53.3/16.0	**71.9**/53.3/12.0	71.6/53.3/9.3	70.0/53.3/8.0

Table 2: Performance of ranking and reweighting according to each feature. 'MI' is the group label & error mutual information feature. 'LE' is the group label entropy feature. 'Rep. Std.' is the group representation standard deviation feature. The results are in the format of (average accuracy/10th percentile accuracy/worst group accuracy). The best performance of each metric is highlighted in bold, and the tied best performance is emphasized with an underline.

The reweighting results according to the ranking of each feature are shown in Table 2, presented in the format (average accuracy/10th percentile accuracy/worst group accuracy). As can be seen, the performance of the worst group

[2] https://www.yelp.com/dataset/

after reweighting using different features is generally better than that of ERM on all three datasets. Using group representation standard deviation (Rep. Std.) on Yelp resulted in a 10% improvement in worst-group performance, while group label entropy (LE) on CivilComments1 led to a substantial 15.4% enhancement. On Amazon, using group error achieved a 5.3% increase in worst-group performance compared to ERM. ERM performs well in average group performance, as expected, while JTT does not show consistent improvements and can even harm performance in some cases. GroupDRO improves worst-group performance in Yelp and CivilComments1 but significantly degrades average performance. Notably, features other than group error can yield equivalent or better performance in certain scenarios, highlighting the potential of exploring diverse group-level features for reweighting in many-domain generalization.

4.3 Results: Combining multiple features via Learning-to-rank (LTR)

Table 3 presents the results of LTR methods and other baseline methods. Our methods are denoted as 'LTR (source dataset),' where the source dataset that is used to train the LTR model is shown in the parenthesis. As can be seen, all LTR methods consistently outperform or are comparable to ERM across all three datasets. LTR(Yelp) shows the best results, improving the worst-group accuracy by 10% on Yelp and 7.3% on CivilComments1. However, combining multiple features does not significantly outperform the use of a single feature, and in some cases, it is even less effective than using just one. This could be due to two reasons: (1) different features contribute differently to performance for the same dataset (as seen in Table 2), and (2) the contribution of the same feature varies across different datasets. The distribution of these features in the source dataset used to train the LTR model may differ from that of the test dataset, leading to suboptimal transfer. Similar conclusions are also reflected in the SHAP analysis in Section §5. Effectively using these individually useful features represents a potential direction for future research.

Datasets	**LTR (Yelp)**	**LTR (C1)**	**LTR (C2)**	**ERM**	**JTT**	**GroupDRO**
Yelp	61.8/51.0/**28.0**	62.5/53.0/19.0	**63.1**/53.0/23.0	63.0/52.0/18.0	61.7/51.0/19.0	59.2/49.0/27.0
CivilComments1	79.8/70.2/**61.5**	**80.4**/72.3/53.8	79.5/69.7/58.3	80.1/**72.7**/53.8	78.4/57.9/53.8	79.3/69.0/61.1
Amazon	70.7/**54.7**/13.3	70.8/53.3/13.3	70.6/53.3/12.0	**71.9**/53.3/12.0	71.6/53.3/9.3	70.0/53.3/8.0

Table 3: Performance of reweighting groups using LTR. Source datasets for training LTR are shown in parenthesis, e.g., Yelp, CivilComments1 (C1), CivilComments2 (C2). The results are in the format of (average accuracy/10th percentile accuracy/worst group accuracy). The best performance of each metric is highlighted in bold, and the tied best performance is emphasized with an underline.

5 Analysis of Feature Importance by SHAP

Next, in order to understand the LTR results, we employ SHapley Additive exPlanations (SHAP) [21] to quantify the importance of each feature in predicting rankings. SHAP is a game-theoretic approach that explains the output of any machine learning model by providing a comparable measurement of the importance of each feature. By utilizing SHAP, we aim to validate the selection of predictive features and identify the importance of features across different source datasets.

For each source dataset used to train the LTR model, we randomly subsample 100 instances to initialize a SHAP Kernel Explainer. We then apply the explainer with the trained LTR model on another randomly sampled set of 2,000 instances to compute the SHAP values of each feature. These SHAP values provide insights into the relative contribution of each feature to the model's output, allowing us to assess the significance of individual features in the context of the specific source dataset.

This analysis helps us understand the role of different group-level features in determining the ranking of groups and their importance for reweighting. By comparing the SHAP values across different source datasets, we can identify patterns and variations in feature importance, which can inform the development of more effective and robust reweighting strategies for many-domain generalization.

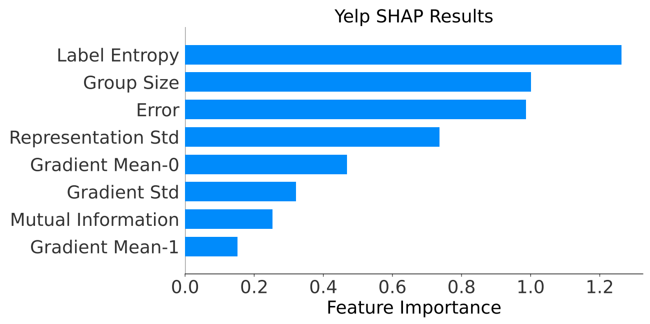


Fig. 1: SHAP feature importance of the Yelp dataset. All features positively contribute to the prediction of the ranking of groups. The label entropy is the most significant feature in all datasets which highlights the challenge of spurious features in the many-domain settings.

Figure 1 shows the SHAP result when the Yelp dataset is used as the source training dataset for the LTR model. The results for the other two datasets are presented in Appendix 9. In the case of Yelp, we observe that all features positively contribute to the prediction of the ranking, which further validates the use of features other than group error for reweighting. Group label entropy is the most important feature across all three datasets, aligning with the performance results in Table 2. As mentioned in Section §3.1, label entropy reflects the spurious correlation between group features and labels, implying that spurious correlations remain the main challenge in the many-domain settings of distribution shift. The significance of both group size and error features validates traditional methods for tackling distribution shifts, such as upsampling and boosting.

When comparing across different datasets, it is evident that the importance of features varies. For example, the contribution of error is the lowest in Civil-

Comments1, whereas it is comparatively higher in Yelp. This observation helps explain why the same feature shows different levels of contribution to performance across various datasets, as seen in Table 2, and why it is challenging to transfer an LTR model trained on one dataset to another test dataset when integrating all features, as shown in Table 3. These findings highlight the need for adaptive strategies that can effectively capture and leverage the varying importance of group-level features across different domains to achieve robust and transferable performance in many-domain generalization settings.

6 Conclusion

In this paper, we have explored group-wise reweighting strategies for robust and generalizable machine learning models on the challenging yet practical setting of many-domain generalization. We propose using alternative group features beyond the traditionally utilized group error to determine the relative importance of groups for reweighting, and our results demonstrate that features like group label entropy and group representation statistics can achieve over 10% improvements on multiple datasets' worst-group performance compared to standard Empirical Risk Minimization. To further investigate the potential benefits of incorporating multiple predictive group features jointly, we adopted a learning-to-rank (LTR) framework, but found that using multiple features via LTR does not necessarily outperform the use of each feature individually, likely due to the varying distributions and differing impacts on group heterogeneity of features across datasets, posing challenges for transferable learning. Our SHAP analysis provides supporting evidence by showcasing the varying significance of features across datasets. We hope our exploration of group features and algorithmic paradigms provides a promising path forward for many-domain generalization and future work will study how to learn the joint application of multiple features to achieve cross-dataset robustness.

References

1. Ahuja, K., Shanmugam, K., Varshney, K., Dhurandhar, A.: Invariant risk minimization games. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 145–155. PMLR (13–18 Jul 2020), https://proceedings.mlr.press/v119/ahuja20a.html
2. Albuquerque, I., Monteiro, J., Darvishi, M., Falk, T.H., Mitliagkas, I.: Generalizing to unseen domains via distribution matching. arXiv preprint arXiv:1911.00804 (2019)
3. Arjovsky, M., Bottou, L., Gulrajani, I., Lopez-Paz, D.: Invariant risk minimization. arXiv preprint arXiv:1907.02893 (2019)
4. Balaji, Y., Sankaranarayanan, S., Chellappa, R.: Metareg: Towards domain generalization using meta-regularization. Advances in neural information processing systems **31** (2018)
5. Ben-Tal, A., Den Hertog, D., De Waegenaere, A., Melenberg, B., Rennen, G.: Robust solutions of optimization problems affected by uncertain probabilities. Management Science **59**(2), 341–357 (2013)

6. Borkan, D., Dixon, L., Sorensen, J., Thain, N., Vasserman, L.: Nuanced metrics for measuring unintended bias with real data for text classification. CoRR **abs/1903.04561** (2019), `http://arxiv.org/abs/1903.04561`
7. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th international conference on Machine learning. pp. 129–136 (2007)
8. Duchi, J.C., Glynn, P.W., Namkoong, H.: Statistics of robust optimization: A generalized empirical likelihood approach. Mathematics of Operations Research **46**(3), 946–969 (2021)
9. Geirhos, R., Jacobsen, J.H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., Wichmann, F.A.: Shortcut learning in deep neural networks. Nature Machine Intelligence **2**(11), 665–673 (2020)
10. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D., Li, W.: Deep reconstruction-classification networks for unsupervised domain adaptation. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. pp. 597–613. Springer (2016)
11. Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., Smola, A.: A kernel method for the two-sample-problem. Advances in neural information processing systems **19** (2006)
12. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS) **20**(4), 422–446 (2002)
13. Kang, J., Lee, S., Kim, N., Kwak, S.: Style neophile: Constantly seeking novel styles for domain generalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7130–7140 (2022)
14. Koh, P.W., Sagawa, S., Marklund, H., Xie, S.M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R.L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B.A., Haque, I.S., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., Liang, P.: WILDS: A benchmark of in-the-wild distribution shifts. In: International Conference on Machine Learning (ICML) (2021)
15. Koh, P.W., Sagawa, S., Marklund, H., Xie, S.M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R.L., Gao, I., et al.: Wilds: A benchmark of in-the-wild distribution shifts. In: International Conference on Machine Learning. pp. 5637–5664. PMLR (2021)
16. Krueger, D., Caballero, E., Jacobsen, J.H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., Courville, A.: Out-of-distribution generalization via risk extrapolation (rex). In: International Conference on Machine Learning. pp. 5815–5826. PMLR (2021)
17. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.: Learning to generalize: Meta-learning for domain generalization. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
18. Liang, X., Li, S., Zhang, S., Huang, H., Chen, S.X.: Pm2. 5 data reliability, consistency, and air quality assessment in five chinese cities. Journal of Geophysical Research: Atmospheres **121**(17), 10–220 (2016)
19. Liu, E.Z., Haghgoo, B., Chen, A.S., Raghunathan, A., Koh, P.W., Sagawa, S., Liang, P., Finn, C.: Just train twice: Improving group robustness without training group information. In: International Conference on Machine Learning. pp. 6781–6792. PMLR (2021)
20. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
21. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)

22. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. pp. 142–150. Association for Computational Linguistics, Portland, Oregon, USA (June 2011), http://www.aclweb.org/anthology/P11-1015
23. Muandet, K., Balduzzi, D., Schölkopf, B.: Domain generalization via invariant feature representation. In: Dasgupta, S., McAllester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 28, pp. 10–18. PMLR, Atlanta, Georgia, USA (17–19 Jun 2013), https://proceedings.mlr.press/v28/muandet13.html
24. Nagarajan, V., Andreassen, A., Neyshabur, B.: Understanding the failure modes of out-of-distribution generalization. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=fSTD6NFIW_b
25. Nam, H., Lee, H., Park, J., Yoon, W., Yoo, D.: Reducing domain gap by reducing style bias. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8690–8699 (2021)
26. Ni, J., Li, J., McAuley, J.: Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (2019)
27. Ren, M., Zeng, W., Yang, B., Urtasun, R.: Learning to reweight examples for robust deep learning. In: International conference on machine learning. pp. 4334–4343. PMLR (2018)
28. Sagawa*, S., Koh*, P.W., Hashimoto, T.B., Liang, P.: Distributionally robust neural networks. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=ryxGuJrFvS
29. Sagawa, S., Raghunathan, A., Koh, P.W., Liang, P.: An investigation of why overparameterization exacerbates spurious correlations. In: International Conference on Machine Learning. pp. 8346–8356. PMLR (2020)
30. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)
31. Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. In: AAAI (2016)
32. Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W., Yu, P.: Generalizing to unseen domains: A survey on domain generalization. IEEE Transactions on Knowledge and Data Engineering (2022)
33. Williams, A., Nangia, N., Bowman, S.R.: A broad-coverage challenge corpus for sentence understanding through inference. arXiv preprint arXiv:1704.05426 (2017)
34. Xiao, R., Ge, Y., Jiang, R., Yan, Y.: A unified framework for rank-based loss minimization. arXiv preprint arXiv:2310.17237 (2023)
35. Yang, C., Westover, M.B., Sun, J.: Manydg: Many-domain generalization for healthcare applications. In: The 11th International Conference on Learning Representations, ICLR 2023 (2023)
36. Yao, H., Wang, Y., Li, S., Zhang, L., Liang, W., Zou, J., Finn, C.: Improving out-of-distribution robustness via selective augmentation. In: International Conference on Machine Learning. pp. 25407–25437. PMLR (2022)

Appendix

7 Datasets

AMAZON-WILDS: Collected as part of the WILDS dataset suite [14], the Amazon-WILDS dataset involves predicting star ratings from users' reviews of Amazon products. The training set has $245,502$ reviews from 1252 users (at least 75 reviews per user). The ID validation set consists of 46,950 reviews from 626 of the 1252 users in the training set. The ID test set is the same size as the ID validation set and contains reviews from the remaining 626 users from the training dataset. Finally, the OOD validation and the OOD test sets each have $100,050$ reviews (75 reviews per user) from $1,334$ new users.
Yelp Business Reviews: WILDS dataset suite [14] contains a modified version of the Yelp Open Dataset; however, there's no accuracy drop from their ID set to OOD set. Thus, we modify it by clustering at the user level in a similar fashion as we did for the IMDB dataset. We set k=6 and select the two farthest clusters as the OOD and ID sets to have a significant out-of-distribution performance drop. The training set comprises $64,931$ reviews from 500 users. The ID validation and test sets consist of $20,070$ and $21,083$ reviews from 333 out of the 666 users in the training set, respectively. Finally, the OOD validation and test sets include $52,200$ and $52,300$ reviews from 522 and 523 unseen users, respectively.
CivilComments: We rebuild the CivilComments dataeset [14] into two variations which also follow the most challenging many-domain generalization setting we mentioned above. In the original dataset, each instance is associated with an 8-dimensional binary vector where each component corresponds to whether the comment mentions one of the 8 demographic identities: male, female, LGBTQ, Christian, Muslim, other religions, Black, and White. Then the author reports the data performance under each demographic. Here, in the first variation (CivilComments1), we rebuild the data where instances were grouped by their unique binary vector combinations. Thus, comments with identical demographic mentions were categorized together. Ideally, there could be 2^8=256 groups but we filtered out the groups with less than 10 instances. We get 137 groups and randomly select 22 groups as the validation data, 20 groups as the test data, and the remaining 95 groups as training data.

For all the datasets we see so far, there is no instance overlap between groups. In real-world contexts, individual identities are multifaceted and seldom exist in isolation. Here, in the second variation (CivilComments2), we rebuild the data where instances sharing mentions of at least three common identities are considered to belong to the same group. As a result, groups can have overlaps. For instance, a group mentioning Male, LGBTQ, and Christian would have overlaps with a group mentioning Male, LGBTQ, and White. The overlap instances will mention Male, LGBTQ, White, and Christian. This overlapping nature allows for a more fluid understanding of the group features we designed above. Ideally, there will be $\binom{8}{3}$=56 groups but we filtered out groups with less than 10 instances. We get 55 groups in total and treat them as training groups. As there will be overlaps in the training groups and test groups if we further split the data, we don't use it as the target evaluation dataset as it falls out of the many-domain generalization setting. We will merely use it as a source dataset for LTR model training.

8 Experiment Setup

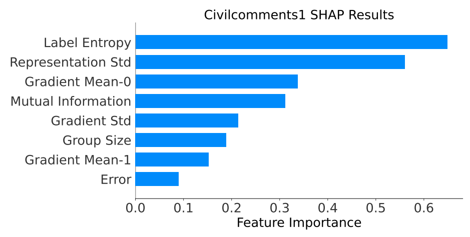
ERM is the standard training method that trains models to minimize the average training loss. Group DRO uses distributionally robust optimization to explicitly minimize the loss on the worst-case domain (or group) during training. JTT has shown superior performance over Group DRO and its variations on several challenging benchmark applications. JTT involves a two-stage training approach which first trains a standard ERM model for several epochs and then trains a second model that upweights the training examples that the first model has misclassified. We finetuned the base-uncased DistilBERT [30] model as our base learner for all task training methods. The number of epochs is 5 with early stopping; the batch size is 16; the learning rate is 1×10^{-5} for AdamW optimizer; L2-regularization strength is 0.01. For the ranking and reweighting algorithm, we use each feature mentioned in §3.1 to rank except gradient mean as it is multi-dimensional. We performed a grid search to tune the cutoff hyperparameter $C \in [10, 20, 50, 100]$ for reweighting. For **JTT**, the number of epoch in the first stage $T \in \{0, 1, 2\}$ and the reweighting factor $\lambda \in \{2, 3, 5\}$. For **Group DRO** we fixed the step size as 0.01 following the best practice reported in [15].

For the neural ranking model ListNet, we simplify it to just one linear layer for better interpretability features. The learning rate is 1×10^{-2} for AdamW optimizer [20] with a weight decay of 1×10^{-2}. The batch size is 128 and the number of epochs is 100. To get the training features for ListNet training, we first finetuned the entire source dataset $\mathcal{D}_{source}$ again on a base-uncased Distilbert. Then, we fine-tuned only the last layer of the model for each subsampled training set $\mathcal{D}^{(k)}_{train}$ for one epoch. Other hyperparameters are the same as the model training mentioned above. The number of training subsets for all three datasets is 500, which means we have 500 ranking instances for the training of ListNet for each dataset. All the experiments are run on the NVIDIA GEFORCE RTX 2080 Ti using the PyTorch Framework.

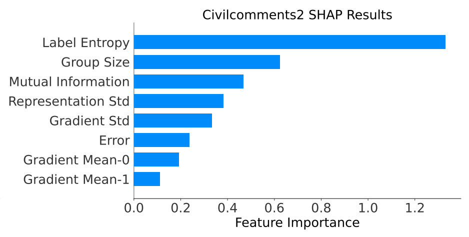
9 SHAP results

10 Convergence of the Algorithm

We can show the convergence of our reweighting method when ranking by group loss by drawing upon the methodology used by [34], who establish convergence for rank-based loss minimization. Our objective is compatible with their framework by creating a sequence of ordered groups. The objective function is then defined as a weighted sum of these group losses, where the weights are calculated as the product of group size and reweighting factors assigned based on group rank.



(a) SHAP feature importance of the Civilcomments1 dataset.



(b) SHAP feature importance of the Civilcomments2 dataset.

Fig. 2: Comparative SHAP feature importance for Civilcomments datasets.

For our proposed algorithm, the objective is

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} w_i n_{[i]} Y_{[i]},$$

where the $\boldsymbol{\theta}$ denotes the model's parameters, and N is the total number of groups in the training set. Y_i represents the average training loss of the i^{th} group for $i \in \{1, ..., n\}$. Let $Y_{[1]} \leq ... \leq Y_{[n]}$ be the order statistics. w_i is the weight corresponding to the i^{th} smallest group and $n_{[i]}$ is the number of instances within group $g_{[i]}$.

When group sizes are the same $n_{[1]} = n_{[2]} = \ldots = n_{[N]} =$ c, we have

$$\mathcal{L}(\boldsymbol{\theta}) = c\left(\sum_{i=1}^{N} w_i Y_{[i]}\right) \tag{2}$$

whose convergence has been proved by [34] under the alternating direction multiplier method (ADMM) optimization.

However, when the group sizes differ, the proof of convergence cannot be directly applied. The main challenge comes from the fact that $w_i * n_{[i]}$ varies with group size since $n_{[i]}$ (the number of instances in a group) depends on $g_{[i]}$. To address this, we employ a simple modification of our approach: Rather than assigning weights to each group, we split the dataset into the same-size quantiles and then assign the weights to each quantile accordingly. This can be better understood through Figure 3, where the groups are arranged in order of their group loss. In this arrangement, $g_{[N]}$ represents the group with the poorest performance, followed by $g_{[N-1]}$ as the second poorest, and so forth. Each triangle represents all instances of the corresponding groups. As the number of instances in each quantile (Q_i) is the same and groups may have different sizes, instances of some groups will be split into different quantiles (e.g. black and yellow part for $g_{[N]}$). The instances that fall in the same quantile (with the same color shown in Figure 3) will be upweighted by the same factor w_q according to Equation **??**.

Then the new objective should be:

$$\mathcal{L}(\boldsymbol{\theta}) = n\left(\sum_{q=1}^{100} w_q \tilde{Y}_q\right) \tag{3}$$

where $n = \frac{\sum_{i=1}^{N} n_i}{100}$ is the number of instances in each quantile which is a constant. $\tilde{Y}_q$ is the average loss of quantile q. Then as long as the $\tilde{Y}_q$ is sorted, the convergence can be reached again according to [34] under ADMM optimization. This is feasible when we keep the losses of each part of the group across boundaries the same (e.g. if the black and yellow parts for group $g_{[N]}$ shown in Figure 3 have the same loss and so on for each group then $\tilde{Y}_q$ is sorted.) Given a set of instance losses of a group, deciding whether it can be partitioned into two (or multiple) subsets such that the sum of the numbers in each set is the same is naturally a partition problem (which is NP-Hard).

We side-step solving this intractable problem by assigning weights at the group level by taking a weighted average over the reweighting factors of parts of the group across boundaries. For example, assume there are x instances in $g_{[N]}$ that fall into Q_{100}(black) and y instances that fall into Q_{99}(yellow), then the reweighting factor for the group is $\frac{x*w_{100}+y*w_{99}}{x+y}$. Generally, if a group $g_{[i]}$ spans across quantiles $Q_k, Q_{k-1}, \ldots, Q_{k-j}$, and the number of instances in each quantile is $n_{Q_k}, \ldots, n_{Q_{k-j}}$, then the weight for $g_{[i]}$ will be:

$$w_i' = \frac{\sum_{m=0}^{j} w_{k-m} n_{Q_{k-m}}}{n_{[i]}} \tag{4}$$

where $\sum_{m=0}^{j} n_{Q_{k-m}} = n_{[i]}$ the total number of instances of the group. And for any $m \neq 0$ and $m \neq j$, $n_{Q_{k-m}} = n$ should be the number of instances in each quantile. Even though knowing the real split of the group across quantiles is NP-hard, $n_{Q_k}, \ldots, n_{Q_{k-j}}$ is known since the size of the group and the number of instances in a quantile is known. (e.g., In Figure 3, if we know the number of instances in a quantile $n = 1000$ and further assume the group size of $g_{[N]}$ is $n_{[N]} = 1050$, then there should be 1000 instances in the black part of $g_{[N]}$ and 50 instances in the yellow part of $g_{[N]}$ and so on for all the groups.) Then, the new objective function can be written as:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \left(\sum_{i=1}^{N} w_i' n_{[i]} Y_{[i]}\right) \tag{5}$$

We can prove that the reweighted loss of a group under objective 3 is the same as the reweighted loss of a group under objective 5. Without loss of generality, we can assume that a group $g_{[i]}$ should be split into two quantiles under the ideal objective 3 as $g_{[N]}$ in Figure 3. Assume the sets of instances in two quantiles are $\mathcal{X}$ and $\mathcal{T}$ correspondingly. The losses of instances in $\mathcal{X}$ are $\{X_1, ..., X_\alpha\}$ and the losses of instances in $\mathcal{T}$ are $\{T_1, ..., T_\beta\}$ where α and β are the sizes of two parts and $\alpha + \beta = n_{[i]}$. Then under the ideal split so that each part has the same average loss, we have $\frac{\sum_{i=1}^{\alpha} X_i}{\alpha} = \frac{\sum_{j=1}^{\beta} T_j}{\beta}$. The average loss of the group $Y_{[i]} = \frac{(\sum_{i=1}^{\alpha} X_i + \sum_{j=1}^{\beta} T_j)}{n_{[i]}}$. Under the objective 3, the reweighted loss of this group is then

$$\begin{aligned}
& w_\alpha \sum_{i=1}^{\alpha} X_i + w_\beta \sum_{j=1}^{\beta} T_j \\
=& \frac{1}{\alpha}(\alpha w_\alpha \sum_{i=1}^{\alpha} X_i + \alpha w_\beta \sum_{j=1}^{\beta} T_j) \\
=& \frac{1}{\alpha}(\alpha w_\alpha \sum_{i=1}^{\alpha} X_i + \beta w_\beta \sum_{i=1}^{\alpha} X_i) \\
=& \frac{\sum_{i=1}^{\alpha} X_i}{\alpha}(\alpha w_\alpha + \beta w_\beta) \\
=& \frac{(\alpha+\beta)\sum_{i=1}^{\alpha} X_i}{(\alpha+\beta)\alpha}(\alpha w_\alpha + \beta w_\beta) \\
=& \frac{(\alpha w_\alpha + \beta w_\beta)}{\alpha+\beta}(1+\frac{\beta}{\alpha})\sum_{i=1}^{\alpha} X_i \\
=& w'_g(1+\frac{\beta}{\alpha})\sum_{i=1}^{\alpha} X_i \text{ according to Equation 4} \\
=& w'_g(\sum_{i=1}^{\alpha} X_i + \sum_{j=1}^{\beta} T_j) \text{ according to the even loss split} \\
=& w'_g n_{[i]} Y_{[i]}
\end{aligned} \tag{6}$$

We have shown that the reweighted loss of a group in objective 3 is the same as the reweighted loss of this group in objective 5 using the weighted reweighting factors in Equation 4. When a group is split into more than two quantiles, the proof can be applied recursively to adjacent quantiles as illustrated above. Hence, one can implement Objective 12 in practice without solving the NP-hard partition problem. Again, as mentioned earlier, the convergence of objective 3 has been proven by [34] under ADMM optimization.

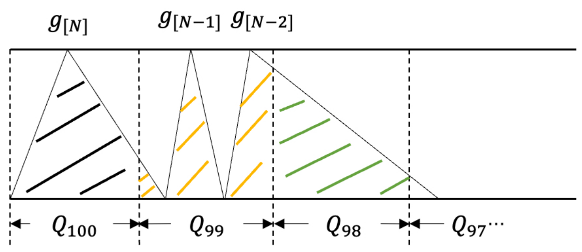


Fig. 3: Figure showing the division of instances into quantiles in the scenario in which groups differ in size. Suppose groups are ranked based on their loss, with $g_{[N]}$ representing the group with the worst performance and $g_{[N-1]}$ as the second worst, continuing in this order. Further, each triangle represents all instances belonging to its respective group. Given that each quantile (Q_i) contains an equal number of instances and the groups vary in size, instances from the same group may be distributed across different quantiles. This is illustrated by the division of $g_{[N]}$ into both black and yellow segments. Instances within the same quantile, even if they originate from different groups, are assigned the same upweighting factor which is determined by the equation **??**.