RANK4CLASS: EXAMINING MULTICLASS CLASSIFICATION THROUGH THE LENS OF LEARNING TO RANK

Anonymous authors
Paper under double-blind review

ABSTRACT

Multiclass classification (MCC) is a classical machine learning problem which aims to classify each instance into one of a predefined set of classes. Given an instance, a classification model computes a score for each class, all of which are then used to sort the classes. The performance of a classification model is usually measured by Top-K Accuracy (e.g., $K = 1$ or 5). In this paper, we examine MCC through the lens of learning to rank (LTR) in the deep learning setting. By viewing MCC as to rank classes for an instance, we first argue that ranking metrics from the information retrieval literature, such as Normalized Discounted Cumulative Gain (NDCG), can be more informative than the existing Top-K metrics in evaluating the performance of classification models, especially for real-world user-facing applications. We further demonstrate that the most popular MCC architecture in deep learning can be mathematically formulated as a LTR pipeline equivalently, with a specific set of choices in terms of ranking model architecture and loss function. Based on these observations, we propose several techniques, stemmed from the rich LTR literature, to improve the MCC performance. Comprehensive empirical results on both text and image classification tasks, with diverse datasets and backbone models (e.g., BERT for text classification and ResNet for image classification) show the value of our proposed framework.

1 INTRODUCTION

Multiclass classification (MCC) is the problem of classifying instances into one of a predefined set of classes (Hastie et al., 2001). It is one of the most fundamental machine learning problems that has broad applications in many fields such as natural language processing (Sun et al., 2019) and computer vision (He et al., 2016). For example, deciding the category of a news article or the subject of an image is formulated as an MCC problem.

Different classification models for MCC have been proposed in the past, ranging from the linear models to nonlinear decision trees and neural models (Aly, 2005). In the modern deep learning era, numerous works approach the MCC problem with deep neural networks. While there are significant advances in neural architectures in different fields, dominating MCC methods share the same recipe: an input instance, being it a feature vector, an image, or a sentence, is fed into a neural model and scored against a predefined set of classes. The model is then trained by using a loss function, typically the softmax cross entropy loss, between the labels and scores over all classes (Goodfellow et al., 2016). During inference, the candidate classes are sorted after an instance is scored against them. Metrics such as Top-1 Accuracy (also known as simply "classification accuracy") or Top-5 Accuracy (the percentage of test instances whose correct class label is in the top 5 predicted classes) are used to compare different classification models. Following this recipe, most efforts in the literature focus on designing more powerful neural network architectures for representation learning (He et al., 2016; Krizhevsky et al., 2012; Huang et al., 2017; Dosovitskiy et al., 2021; Minaee et al., 2021). Few work has studied a formulation different from the recipe above so far.

Different from existing works, we seek for a different formulation for MCC by examining it through the lens of learning to rank (LTR), a rich research field stemmed from information retrieval (Liu, 2009). As shown in the paper, the benefits of such a formulation are that it allows us to better evaluate model performances by borrowing more informative ranking metrics, train models with ranking losses that are closely connected with ranking metrics, and use flexible ranking architectures for MCC in deep learning.

We first show that the classical MCC problem can be treated as a ranking problem. As hinted above in our description of MCC, dominating neural MCC methods score a given instance on a predefined set of classes and sort the classes during inference. This is equivalent to a LTR setting where a set of items (i.e., classes) are scored and ranked given a query (i.e., the input instance). In fact, a Top-K Accuracy measure itself can be viewed as a ranking metric, similar to Precision@K in the information retrieval literature. However, other ranking metrics such as Normalized Discounted Cumulative Gain (NDCG) are commonly used in LTR because they are more informative than Precision@K to reflect the utility of a model in real-world user-facing applications. They can be borrowed to enrich the MCC evaluation metrics but have not been adopted widely.

For modeling, we show a general "equivalent view" for MCC from ranking perspectives, and that the classical MCC model architecture can be transformed to an equivalent ranking architecture. This view will give us insight into the design choices of dominating MCC models and their limitations. We then propose several approaches to further improve MCC performance in the LTR setting. In this paper, we mainly focus on two aspects: loss functions and model architectures. For loss functions, one major research focus in LTR is designing ranking losses specific to certain metrics so there is a rich literature to leverage. For model architecture, we show that the vast MCC literature focuses on only one component in the ranking architecture, the input instance encoding. With the LTR view, we can enhance the modeling capacity of other components as well. In this paper, we specifically study the effect of enhancing the interactions between instances and classes, a setup similar to the one commonly used in information retrieval for queries and items (Li & Xu, 2014).

We report experimental results on a variety of MCC tasks, including different datasets and backbone models for different modalities (image and text). Examples include BERT (Devlin et al., 2019) for text classification on GoEmotions (Demszky et al., 2020) and ResNet (He et al., 2016) for image classification on ImageNet (Krizhevsky et al., 2012). Results show that the proposed methods outperform or perform competitively with the baselines in all settings. We expect that the promising results can encourage the community to further examine MCC from LTR perspectives.

By making the connection between LTR and MCC in deep learning, our contributions can be summarized as follows:

- We show that ranking metrics can be more informative than the Top-K Accuracy metrics and thus propose to also use ranking metrics for MCC evaluations.
- We show that the commonly used neural architecture in MCC is equivalent to a specific ranking architecture with limited interactions between instances and classes.
- We propose to use ranking losses for MCC and also other ranking architectures that enable richer interactions between instances and classes for MCC.
- We conduct experiments on both image and text classification tasks and show the usefulness of ranking losses and ranking architectures in MCC.

The paper is organized as follows. We give an in-depth analysis of classification and ranking metrics in Section 2. In Section 3, we mathematically show an equivalent view of MCC from the ranking perspectives and examine its design choices. In Section 4, we describe alternatives on the selection of loss functions and interaction architectures that can potentially improve MCC performance and evaluate them on various classification tasks in Section 5. We discuss related work in Section 6. We conclude the paper and discuss future directions in Section 7.

2 RANKING METRICS FOR MCC

2.1 METRICS FOR CLASSIFICATION

The basic binary classification problem classifies examples into positive and negative classes. MCC extends binary classification to more than 2 classes. Going from binary to multi-class is not trivial for evaluations. Binary classification metrics are usually *class-oriented*. For example, the metrics such as AUC and Accuracy are based on measures such as true positives (TP) and false negatives (FN) that are computed with respect to positive and negative classes (Sokolova & Lapalme, 2009). These metrics are not directly used in MCC when there are more than two classes. In contrast,

motor scooter leopard

Model 1 (top 5 classes)

go-kart	leopard
motor scooter	jaguar
bicycle	cheetah
golfcart	snow leopard
moped	egyption cat

Top-1 Accuracy: 0.5
Top-5 Accuracy: 1.0
NDCG@5: **0.82**

Model 2 (top 5 classes)

motor scooter	snow leopard
golfcart	puma
bumper car	cheetah
go-kart	leopard
moped	tiger

Top-1 Accuracy: 0.5
Top-5 Accuracy: 1.0
NDCG@5: **0.72**

Figure 1: Here we show an simulated example of evaluating image classification performance of two models with Top-1/5 Accuracy and NDCG@5. The Top-1/5 Accuracy of two models are exactly the same, and can not differentiate the two models. While NDCG@5 can successfully tell that Model 1 has a better classification performance than Model 2.

MCC metrics are usually *instance-oriented*. The commonly used metrics are the Top-K Accuracy metrics, popularized by the ImageNet competition (Russakovsky et al., 2015). Some earlier works like Crammer & Singer (2002) defined the "empirical error" when working on multi-class SVM algorithm, which is equivalent to Top-1 Accuracy.

2.2 Relations to Ranking Metrics

The Top-K Accuracy metrics can be thought as a type of ranking metrics. In fact, it is very close to the Precision@K ranking metrics: Top-K Accuracy is the same as $\min(1, \text{K} \cdot \text{Precision@K})$ and Top-1 Accuracy is the same as Precision@1. Besides Precision@K, there are other commonly used ranking metrics such as Normalized Discounted Cumulative Gains (NDCG) and Mean Reciprocal Rank (MRR) (Järvelin & Kekäläinen, 2002). To the best of our knowledge, these ranking metrics are not commonly used for MCC evaluations but can be more informative in evaluating MCC tasks.

We use NDCG as an example. Let $\mathbf{y}$ be the relevance labels of all items and y_i be the one for the i-th item, the calculation of NDCG is as follows:

$$NDCG(\pi_s, \mathbf{y}) = \frac{DCG(\pi_s, \mathbf{y})}{DCG(\pi_*, \mathbf{y})}, \text{and } DCG(\pi, \mathbf{y}) = \sum_{i=1}^{n} \frac{2^{y_i} - 1}{\log_2(1 + \pi(i))}, \tag{1}$$

where π_s is a ranked list induced by sorting the items based on model outputs, π_* is the ideal ranked list sorted by $\mathbf{y}$, n is the total number of items and $\pi(i)$ is the rank of the i-the item. In practice, NDCG@K, the Top-K version of NDCG, is commonly used and defined by summing over only the top-K items in π_s and π_*. When there is a single $y_i = 1$ in $\mathbf{y}$, it's easy to show that NDCG@1 is equivalent to Precision@1 and Top-1 Accuracy.

By treating classes as items and define y_i as whether an instance belongs to class i, the NDCG metric can be used to evaluate MCC models. In Figure 1, we simulate an image MCC application in a user-facing interface. In particular, there are two models to be compared. We can see that NDCG@5 is more informative to distinguish the difference between the results from two models, where both Top-1 Accuracy and Top-5 Accuracy metrics cannot. The reason is that there is a position discounting function in NDCG, while Top-K metrics impose a simple hard cut at position K. Therefore, NDCG@5 can capture more information when comparing different models. In the rest of this paper, we choose NDCG@5 as our representatives for ranking metrics and further demonstrate its advantage in MCC evaluations in Section 5.

3 MCC from Ranking Perspectives

3.1 Classical MCC Model Architectures

The general model architecture for MCC based on deep neural networks (DNN) is composed of three parts: the input instance to be classified, an encoder to extract latent representations of the input, and a classification layer for generating scores on candidate classes, as illustrated in Figure 2 **Left**. We

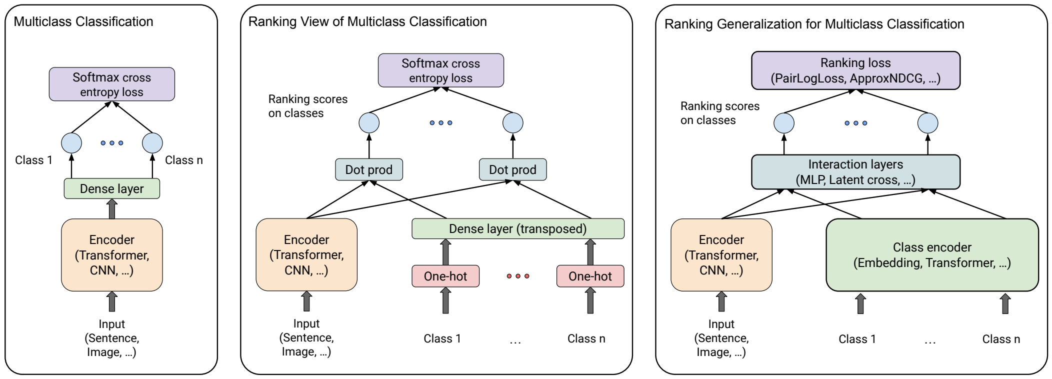

Figure 2: Classical MCC models from LTR perspectives. **Left**: the typical DNN structure for MCC; **Middle**: the equivalent neural ranking structure; **Right**: The ranking generalization for MCC which allows the exploration of different class encoders, interaction layers, and loss functions.

use $\mathbf{x}$ to represent the input instance such as a textual sentence or an image. The encoder can have different structure designs based on the modality of the input, such as transformers (Vaswani et al., 2017) to encode textual sentences; and convolutional neural networks (CNN) (Krizhevsky et al., 2012) to encode images. The encoder can be represented as a function $\mathcal{H}(\cdot)$ that maps $\mathbf{x}$ to a d-dimensional embedding vector $\mathbf{h} = \mathcal{H}(\mathbf{x})$. The classification layer is in most cases a dense layer with weight matrix $W \in \mathbb{R}^{n\times(d+1)}$. Then the classification scores are calculated by $\mathbf{s} = W\mathbf{h}'$, where $\mathbf{h}' := [\mathbf{h}, 1]$ with an added dimension of value 1 for the bias, and $\mathbf{s}$ is a n-dimensional vector for n classes. The score for the i-th class is $s_i = \mathbf{e}_i^\top W\mathbf{h}'$, where $\mathbf{e}_i$ is an n-dimensional one-hot vector with the i-th dimension being 1.

For training neural MCC models, the softmax cross entropy loss is used by default in almost all prior work (Goodfellow et al., 2016; Zhang et al., 2021). However, whether it is the most suitable loss for optimizing the evaluation metrics of interest is not carefully studied. Very recently, empirical results show that the mean squared error (MSE) can sometimes outperform softmax cross entropy loss in MCC tasks, but rescaling is needed in certain tasks (Hui & Belkin, 2021).

3.2 An Equivalent View from Neural Ranking Models

LTR learns a ranking model to score and sort a set of items (e.g., documents, news, etc.) based on their relevance to a given query. Neural ranking models (Guo et al., 2020) adopt DNNs to match the query and items with their latent representations.

We show the equivalent view of a classical MCC model as a neural ranking model in Figure 2 **Middle**. In particular, we treat the input instance $\mathbf{x}$ as the query, and the n candidate classes as input items to be ranked by the neural ranking model. The same as in the classical MCC model, we apply the encoder $\mathcal{H}(\cdot)$ to $\mathbf{x}$ to get a d-dimensional embedding $\mathbf{h} = \mathcal{H}(\mathbf{x})$, and $\mathbf{h}' = [\mathbf{h}, 1]$. For representing the classes, we use one-hot vector $\mathbf{e}_i$ for the i-th class and obtain its embedding $\mathbf{c}_i = W^\top\mathbf{e}_i$, where W is the weight matrix from the classification layer of the classical MCC model. Finally, the ranking score of the i-th class can be calculated by $s'_i \equiv \mathbf{c}_i^\top\mathbf{h}' = \mathbf{e}_i^\top W\mathbf{h}' = s_i$. In this way, we show the classical MCC model based on DNNs can be transformed to an equivalent neural ranking model when they are both trained by the softmax cross entropy loss.

4 Ranking Architectures for MCC

Here we discuss the possible new designs of different components inspired by the ranking view of MCC. We call the new framework ranking for multiclass classification, or *Rank4Class* in short.

As discussed in Section 1, existing work mostly focus on the instance encoder. From the ranking perspective, however, we can see that there are several other promising directions to improve MCC performance. First, existing work mainly use softmax cross entropy loss, while there are many advanced ranking losses that can be leveraged. Second, the interaction between the instance and class embeddings is simply a dot product, while richer interaction patterns can be explored. Third, while existing work focus on instance encoder, the class encoder is a simple linear projection that uses one-hot encoding of classes as input, where more powerful class encoders with richer inputs (e.g., class metadata) can be deployed. We illustrate the potential improvements in Figure 2 **Right**. In this paper, we focus on ranking losses and interaction patterns, and leave the rest for future work.

4.1 RANKING LOSSES FOR MCC

Although softmax cross entropy loss (SoftmaxCE) is widely adopted for training MCC models, how it can optimize the evaluation metrics is not clear. We have shown that ranking metrics are more suitable for evaluating MCC tasks, where different ranking losses should be considered since they can be better choices to directly optimize certain ranking metrics, such as NDCG. Next, we first discuss the soundness of SoftmaxCE in MCC tasks with respect to ranking metrics. Then we present two other ranking losses as examples, i.e., pairwise logistic loss (PairLogLoss) and approximate NDCG loss (ApproxNDCG).

We denote the ground-truth label for the i-th class of an instance by $y_i, i \in \{1, \ldots, n\}$, where only the correct class has label 1 and the rest classes have label 0. To compute the SoftmaxCE, the classification scores $s_i, i \in \{1, \ldots, n\}$ produced by the model are first projected to the probability simplex $\mathbf{p}$ by softmax activation as $p_i = \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}}$, then the cross entropy loss is defined as

$$\ell_{ce}(\mathbf{y}, \mathbf{p}) = -\sum_{i=1}^{n} y_i \log p_i.$$

Intuitively the SoftmaxCE is promoting the correct class against all other classes since only the term with $y_i = 1$ is counted in the loss. Therefore, it can be viewed as a listwise ranking loss, which aims to rank the class with label 1 above all other classes. In fact, Bruch et al. (2019) has shown that the SoftmaxCE is a bound on MRR and NDCG, which explains its promising performance in MCC tasks.

There is rich literature of developing ranking losses for optimizing ranking metrics (Burges et al., 2005; 2006; Burges, 2010). Such ranking losses are usually directly derived from the target ranking metric, and bound the metric by approximation techniques (Qin et al., 2010). One of the most historical and popularly used ranking losses is the PairLogLoss.

$$\ell_{pl}(\mathbf{y}, \mathbf{s}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{I}_{y_i > y_j} \log(1 + e^{-\sigma(s_i - s_j)}), \tag{2}$$

where $\mathbb{I}$ is the indicator and σ is a hyper-parameter. The PairLogLoss is proved to be able to minimize the rank of the relevant item (Wang et al., 2018).

Another popular ranking loss is the ApproxNDCG (Qin et al., 2010), which directly optimizes the NDCG metric in Eq 1. The rank of an item i can be computed as $\pi_s(i) = 1 + \sum_{j \neq i} \mathbb{I}_{s_i < s_j}$, where the indicator $\mathbb{I}_{s<t}$ is discrete but can be approximated by a sigmoid function to be smooth:

$$\mathbb{I}_{s<t} = \mathbb{I}_{t-s>0} \approx \frac{1}{1 + e^{-\alpha(t-s)}}, \tag{3}$$

where $\alpha > 0$ is a parameter to control how tightly the indicator is approximated.

To summarize, the popular SoftmaxCE can be viewed as a ranking loss that bounds a specific ranking metric. Other ranking losses can also be valuable. We further investigate the empirical use of different ranking losses for MCC in Section 5.

4.2 ENHANCING INSTANCE-CLASS INTERACTIONS

After obtaining the embeddings $\mathbf{h}$ for the input instance and $\mathbf{c}_i$ for class i, it is also important to design the matching mechnism for producing the score s_i between the embeddings of the instance and the class. For this purpose, we can add different interactions between $\mathbf{h}$ and $\mathbf{c}_i$ rather than simple dot-product for producing the score, $s_i = \mathcal{I}(\mathbf{h}, \mathbf{c}_i)$,. $\mathcal{I}(\cdot, \cdot)$ is a function to represent the interaction between instance and class embeddings which produces a ranking score. In this paper, we enhance the interaction by the following two patterns:

Table 1: Statistics of datasets used in experiments.

Dataset	#classes	Train	Validation	Test
GoEmotions	28	36.3K	4.5K	4.6K
MIND	18	78.8K	25.7K	25.9K
ImageNet	1000	1.28M	50K	-
CIFAR-10	10	50K	-	10K

- LC+MLP: We first apply the latent cross (LC) operation (Beutel et al., 2018) on $\mathbf{h}, \mathbf{c}_i$ and then follow with a multilayer perceptron (MLP) to get the score s_i.
- Concat+MLP: We first concatenate (Concat) the two embeddings $\mathbf{h}, \mathbf{c}_i$ and then follow with an MLP to compute s_i.

5 EXPERIMENTS

In this section, we study the Rank4Class framework on several datasets and instance encoders in comparison to classical MCC models. We consider both text classification and image classification tasks for evaluation. The datasets we used are summarized in Table 1. For text classification, we include GoEmotions (Demszky et al., 2020) and MIND (Wu et al., 2020). The GoEmotions dataset contains instances with multiple labels. Since we focus on single-label MCC tasks in this paper, we filter out instances with more than one label. We adopt ELECTRA (Clark et al., 2020) and BERT (Devlin et al., 2019) as text encoders in the experiments. For image classification, we use ImageNet (Russakovsky et al., 2015) and CIFAR-10 (Krizhevsky & Hinton, 2009), which are two most popularly studied datasets for image classification. We adopt ResNet50 (He et al., 2016) as image encoder for ImageNet and VGG16 (Simonyan & Zisserman, 2014) for CIFAR-10. More details of the use of datasets and configurations of instance encoders are given in Appendix A.

For text classification datasets with both validation and test sets provided, we tune hyper-parameters on the validation set and report results on the test set. For ImageNet and CIFAR-10, we tune hyper-parameters and report the performance on the validation and test set respectively, which is the norm in the literature. More details on experimental settings such as data processing and hyper-parameter tuning are included in Appendix B. We use Top-1 Accuracy (equivalent to NDCG@1), Top-5 Accuracy, and NDCG@5 for evaluation of the MCC performance.

In Section 5.1, we introduce different configurations of the Rank4Class framework with respect to loss functions and interaction patterns, and summarize its overall performance in MCC tasks. In Section 5.2, we study different ranking losses in optimizing MCC tasks with respect to different evaluation metrics. In Section 5.3, we examine the effect of different interaction patterns between the instance and class embeddings with fixed loss functions. We discuss the application of different metrics in evaluating MCC performance in both sections according to the results.

5.1 OVERALL PERFORMANCE OF RANK4CLASS

Besides SoftmaxCE, PairLogLoss, and ApproxNDCG discussed in Section 4.1, we also include Gumbel-ApproxNDCG (Bruch et al., 2020) loss and mean squared error (MSE) in experiments. We study different combinations between these five losses and the two interaction patterns introduced in Section 4.2 in the Rank4Class framework. In Table 2, we report the best performance of different configurations for Rank4Class under each metric in comparison to the baseline MCC models. As shown in the table, Rank4Class can improve the MCC performance in virtually all tasks through specific combinations of losses and interaction patterns. This shows the increased headroom from Rank4Class on MCC tasks on different evaluation metrics by adding more flexible design options in different components from LTR perspectives. The complete results of all combinations of losses and interactions are included in Appendix C, where we use $*$ to mark the combinations that achieve the best performance in each task under each metric.

Table 2: Results from classical MCC models and Rank4Class. Bold font indicates the best value in each row.

Dataset	Encoder	Metrics	Classical MCC	Rank4Class
GoEmotions	BERT	Top-1 Accuracy	0.5900	**0.5935**
		Top-5 Accuracy	0.8665	**0.8815**
		NDCG@5	0.7390	**0.7462**
	ELECTRA	Top-1 Accuracy	0.6155	**0.6264**
		Top-5 Accuracy	0.8955	**0.9135**
		NDCG@5	0.7696	**0.7830**
MIND	BERT	Top-1 Accuracy	0.6910	**0.6980**
		Top-5 Accuracy	0.9385	**0.9475**
		NDCG@5	0.8278	**0.8356**
	ELECTRA	Top-1 Accuracy	0.7291	**0.7360**
		Top-5 Accuracy	0.9575	**0.9630**
		NDCG@5	0.8573	**0.8613**
ImageNet	ResNet50	Top-1 Accuracy	0.7626	**0.7642**
		Top-5 Accuracy	0.9310	**0.9325**
		NDCG@5	0.8574	**0.8585**
CIFAR-10	VGG16	Top-1 Accuracy	0.9344	**0.9360**
		Top-5 Accuracy	0.9980	**0.9985**
		NDCG@5	0.9719	**0.9723**

5.2 Effect of Ranking Losses

In this section, we study the use of ranking losses for optimizing MCC performance. In particular, we use PairLogLoss and ApproxNDCG as two most representative ranking losses in comparison to the SoftmaxCE. The results on other losses can be find in Appendix C. We only vary the loss function in the base Rank4Class structure in Figure 2 (Middle), so the “SoftmaxCE” method is the baseline that is equivalent to classical MCC models.

The results are shown in Table 3 and 4 for text and image classification tasks respectively. Overall, PairLogLoss or ApproxNDCG can outperform SoftmaxCE in nearly all tasks and metrics except for the Top-1 Accuracy on GoEmotions with BERT and MIND with BERT. Besides, PairLogLoss is generally good at Top-5 Accuracy than SoftmaxCE and ApproxNDCG, achieving the best Top-5 Accuracy in all tasks. ApproxNDCG performs well on both Top-1 Accuracy and NDCG@5 (achieves the top in four out of six tasks), which shows its effectiveness in directly optimizing NDCG metrics. Moreover, we see that the improvements from PairLogLoss and ApproxNDCG are more significant on text classification tasks than that on image classification tasks. Our observation is similar to that in (Hui & Belkin, 2021), which hypothesized the reason being that popular image encoders are all heavily tuned with the SoftmaxCE.

Finally, we observe that different metrics are not always consistent in evaluating MCC tasks. For example, on GoEmotions with ELECTRA, PairLogLoss performs better than ApproxNDCG on Top-5 Accuracy, while ApproxNDCG outperforms PairLogLoss on NDCG@5. This means that the PairLogLoss tends to rank the correct class in top 5 positions in more instances than ApproxNDCG, but ApproxNDCG can put the correct class relatively higher in the rankings. Furthermore, on MIND with BERT, Top-5 Accuracy can not tell if PairLogLoss or ApproxNDCG is better. But NDCG@5 can successfully differentiate them since it also takes in absolute rank of the correct class in top 5 positions into account and thus is more informative.

5.3 Effect of Interactions Between Instance and Classes

In this section, we study the effect of different interaction patterns between instance and class embeddings for producing ranking scores. Specifically, we use the two interaction patterns in section 4.2 with 2-layer MLPs.

Table 3: Results on text classification tasks trained with different losses.

Dataset	Encoder	Metrics	SoftmaxCE	PairLogLoss	ApproxNDCG
GoEmotions	BERT	Top-1 Accuracy	**0.5900**	0.5821	0.5858
		Top-5 Accuracy	0.8665	**0.8800**	0.8700
		NDCG@5	0.7390	**0.7427**	0.7402
	ELECTRA	Top-1 Accuracy	0.6155	0.6022	**0.6233**
		Top-5 Accuracy	0.8955	**0.9120**	0.9075
		NDCG@5	0.7696	0.7717	**0.7803**
MIND	BERT	Top-1 Accuracy	**0.6910**	0.6879	0.6903
		Top-5 Accuracy	0.9385	**0.9475**	**0.9475**
		NDCG@5	0.8278	0.8325	**0.8336**
	ELECTRA	Top-1 Accuracy	0.7291	0.7228	**0.7335**
		Top-5 Accuracy	0.9575	**0.9625**	0.9570
		NDCG@5	0.8573	0.8574	**0.8589**

Table 4: Results on image classification tasks trained with different losses.

Dataset	Encoder	Metrics	SoftmaxCE	PairLogLoss	ApproxNDCG
ImageNet	ResNet50	Top-1 Accuracy	0.7626	0.7636	**0.7638**
		Top-5 Accuracy	0.9310	**0.9320**	0.9315
		NDCG@5	0.8574	**0.8582**	0.8580
CIFAR-10	VGG16	Top-1 Accuracy	0.9344	0.9357	**0.9359**
		Top-5 Accuracy	0.9980	**0.9985**	**0.9985**
		NDCG@5	0.9719	0.9721	**0.9723**

We compare these two types of interactions with the dot-product baseline between the instance and class embeddings by fixing the loss function. In particular, we use ApproxNDCG in text classification tasks and SoftmaxCE in image classification tasks. The study of different interactions on other losses are included in Appendix C. The results are shown in Table 5 and 6 for text and image classification tasks. The results demonstrate that the two added interactions can outperform or achieve competitive performance than simple dot-product in all tasks and metrics. This shows the effectiveness of adding enhanced interactions based on the Rank4Class framework. In particular, latent-cross embedding tends to perform better than the concatenation of instance and class embeddings, achieving top performance in four out of six on Top-1 Accuracy and five out of six tasks on NDCG@5. Again, we see that different evaluation metrics are not always consistent with each other, and NDCG@5 can be more informative than Top-5 Accuracy, as observed in Section 5.2.

6 Related Work

Modern deep neural networks for MCC converge to the same recipe: given an input instance, a neural encoder is learned to output scores for a set of classes. The vast research literature focus on developing more effective encoders in diverse domains, such as computer vision (He et al., 2016; Krizhevsky et al., 2012; Huang et al., 2017; Dosovitskiy et al., 2021), natural language processing (Sun et al., 2019; Minaee et al., 2021), and automatic speech recognition (Moritz et al., 2019), among others. The softmax cross entropy loss is virtually the dominant loss function discussed in these papers. Only very recently, Hui & Belkin (2021) study the mean squared error as another loss for MCC. Our work is orthogonal to the extensive research on neural encoders in that we provide a new formulation from the LTR perspective. Such a perspective inspires more diverse loss functions and model architectures that can model interactions between inputs and classes more effectively.

Learning to rank (LTR) is a long-established interdisciplinary research area at the intersection of machine learning and information retrieval (Liu, 2009). Neural rankers are dominating in ranking virtually all modalities recently, including text ranking (Lin et al., 2020), image retrieval (Gordo

Table 5: Results on text classification tasks of different interactions trained with ApproxNDCG.

Dataset	Encoder	Metrics	dot product	LC+MLP	Concat+MLP
GoEmotions	BERT	Top-1 Accuracy	0.5858	**0.5935**	0.5908
		Top-5 Accuracy	0.8700	0.8760	**0.8780**
		NDCG@5	0.7402	**0.7462**	0.7461
	ELECTRA	Top-1 Accuracy	**0.6233**	**0.6233**	0.6220
		Top-5 Accuracy	0.9075	**0.9135**	0.9085
		NDCG@5	0.7803	**0.7830**	0.7797
MIND	BERT	Top-1 Accuracy	0.6903	0.6919	**0.6923**
		Top-5 Accuracy	0.9475	0.9475	**0.9495**
		NDCG@5	0.8336	**0.8343**	0.8335
	ELECTRA	Top-1 Accuracy	0.7335	0.7326	**0.7360**
		Top-5 Accuracy	0.9570	**0.9630**	0.9570
		NDCG@5	0.8589	**0.8613**	0.8605

Table 6: Results on image classification tasks of different interactions trained with SoftmaxCE.

Dataset	Encoder	Metrics	dot product	LC+MLP	Concat+MLP
ImageNet	ResNet50	Top-1 Accuracy	0.7626	**0.7638**	0.7634
		Top-5 Accuracy	0.9310	0.9320	**0.9325**
		NDCG@5	0.8574	**0.8584**	0.8583
CIFAR-10	VGG16	Top-1 Accuracy	0.9344	**0.9360**	0.9357
		Top-5 Accuracy	**0.9980**	**0.9980**	**0.9980**
		NDCG@5	0.9719	0.9722	**0.9723**

et al., 2016), and tabular data ranking (Qin et al., 2021). Many LTR papers focus on more effective loss functions (Qin et al., 2010; Bruch et al., 2020) to rank items with respect to a query. The focus of this paper is to introduce new techniques stemmed from LTR to solve MCC problems.

Multi-label classification (MLC) differs from MCC in that there are more than one labels for each instance. MLC is generally treated as a different problem from classical MCC: the number of labels assigned to an instance could be arbitrary and one research focus is to decide the threshold to cutoff the prediction list. Its recently popular sub-field, extreme multi-label classification (Zhang et al., 2018), also has a different and specific setting (e.g., the number of labels is huge) and research focus (e.g., efficiency to rank in the huge label space). LTR techniques have been proposed for MLC in the past (Yang & Gopal, 2012). While the perspective is similar to ours, the work is not in the deep learning setting and thus does not have the equivalence view of MCC in LTR. Our work makes the connection and also proposes extensions for MCC in the deep learning setting.

7 CONCLUSION

In this paper we examine the classical MCC problem through the lens of learning to rank. Such a perspective brings benefits to MCC from three aspects: ranking metrics, ranking losses, and ranking architectures. We first show that ranking metrics can be more informative for MCC evaluations. Then, in the deep learning setting, we show an equivalent view of MCC in LTR setting. Such a connection provides new perspectives for MCC with respect to loss functions and model architectures. We studies these new formulations of MCC on various datasets and observe promising results.

Our work opens up several research directions. First, the new ranking architectures allow to take more class information such as class metadata into account and it is interesting to study how this additional information can improve MCC. Second, it is also possible to apply the proposed framework to binary classification. Third, classes are usually not independent and our framework can incorporate the relationship between classes into the MCC through attentions, which is worth studying.

REFERENCES

Mohamed Aly. Survey on multiclass classification methods. *Neural Netw*, 19:1–9, 2005.

Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pp. 75–78, 2019.

Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. A stochastic treatment of learning to rank scoring functions. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 61–69, 2020.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 89–96, 2005.

Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems*, 2006.

Christopher J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical report, Microsoft Research, 2010.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, March 2002.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. GoEmotions: A Dataset of Fine-Grained Emotions. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2021.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *European Conference on Computer Vision*, pp. 241–257. Springer, 2016.

Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, 57(6):102067, 2020.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. 2001.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross entropy in classification tasks. In *International Conference on Learning Representations*, 2021.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, pp. 422–446, October 2002.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, pp. 1097–1105, 2012.

Hang Li and Jun Xu. Semantic matching in search. *Foundations and Trends in Information Retrieval*, 7(5):343–469, 2014.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. Pretrained transformers for text ranking: BERT and beyond. *arXiv preprint arXiv:2010.06467*, 2020.

Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning–based text classification: A comprehensive review. *ACM Computing Surveys (CSUR)*, 54(3):1–40, 2021.

Niko Moritz, Takaaki Hori, and Jonathan Le Roux. Triggered attention for end-to-end speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5666–5670, 2019.

Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2970–2978, 2019.

Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, 13(4):375–397, 2010.

Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. Are neural rankers still outperformed by gradient boosted decision trees? In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437, 2009.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pp. 194–206. Springer, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1313–1322, 2018.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. MIND: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Yiming Yang and Siddharth Gopal. Multilabel classification with meta-level features in a learning-to-rank framework. *Machine Learning*, 88(1-2):47–68, 2012.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. Deep extreme multi-label learning. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pp. 100–107, 2018.

A Datasets and Encoders

A.1 Datasets

- **GoEmotions** (Demszky et al., 2020) is the largest manually annotated dataset for fine-grained emotion classification. It contains 58k English Reddit comments, labeled with 27 emotion categories and a default "Neutral" category. On top of the raw data, the authors also provided a version that only includes comments with two or more raters agreeing on at least one label, split into train/test/validation sets. In our experiments, we filter out comments with more than 1 emotion labels in all three sets.
- **MIND** (Wu et al., 2020) is a large-scale dataset for news recommendation. MIND contains about 130k English news articles. Every news article contains rich textual content including title, abstract, body, category and entities. We use the concatenation of title and abstract as the content of the news and use the category of the news as the classification label. We split all news instances into train/test/validation sets by roughly 60/20/20 for experiments.
- **ImageNet** (Krizhevsky et al., 2012) is an image dataset with around 1.28 million images in the training set and 50k images in the validation set. Each image is labeled by one of 1,000 classes. The images are cropped and resized to $224 \times 224 \times 3$ pixels as the input following the preprocessing in `https://github.com/tensorflow/models/tree/master/official/vision/image_classification/resnet`. We follow the common practices of using ImageNet and report the results on the validation set.
- **CIFAR-10** (Krizhevsky & Hinton, 2009) contains 50k training images and 10k test images. There are 10 classes and each class has 6k images. All images have the same size of $32 \times 32 \times 3$.

A.2 Instance Encoders

- **BERT** (Vaswani et al., 2017) is a model based on transformers pretrained on a large corpus of English data. We use the BERT-Base-uncased model from `https://github.com/google-research/bert` for finetuning in our experiments. The maximum sequence length is set to 32 for GoEmotions and 128 for MIND.
- **ELECTRA** (Clark et al., 2020) is another pretrained transformer model. We use the ELECTRA-Base-uncased model from `https://github.com/google-research/electra` in our experiments. The maximum sequence length is set in the same way as BERT.

- **ResNet50** (He et al., 2016) is a 50 layers deep convolutional neural network with residual connections. We use the implementation of ResNet50 from the tensorflow official models at `https://github.com/tensorflow/models/tree/master/official/vision/image_classification/resnet`. We adopt the pretrained network on ImageNet and finetune it in our Rank4Class framework.
- **VGG16** (Simonyan & Zisserman, 2014) is a convolutional neural network with 16 layers. We use the implementation of VGG16 from `https://github.com/geifmany/cifar-vgg` with input size 32×32 in our experiments on CIFAR-10.

B EXPERIMENTAL SETTINGS

We provide the implementation and hyper-parameter tuning details for reproducibility. For text classification tasks, we tokenize the raw sentences into word ids based on BERT vocabulary, and create the input masks and segment ids following standard BERT input formats. For MIND, we concatenate the title and abstract of each news by adding a "[SEP]" token between them. ELECTRA uses exactly the same input formats as BERT. The data processing of image datasets follow the standard methods as given in the original papers. We implement the Rank4Class pipeline based on TF-ranking (Pasumarthi et al., 2019). We adopt TF-ranking's implementation of SoftmaxCE, MSE and all the ranking losses.

For instance encoders, we use the default hyper-parameters suggested in the original implementations listed in Appendix A without further tuning. We use the same number of hidden units in the two layers of MLP for interactions, and tune the number of hidden units in {64, 128, 256, 512}. For all experiments, we use Adam (Kingma & Ba, 2014) and Adagrad (Duchi et al., 2011) as optimizers for training models. We tune the initial learning rate for all experiments in the range of $1e^{-7}$ to 0.1 with a multiplicative step size of 3. Adam has a slight edge over Adagrad in most experiments and the best learning rate for Adam are $3e^{-6}$ and $1e^{-5}$ for most experiments. We use a batch size of 32 for text classification tasks, and a batch size of 64 for image classification tasks. For all configurations of the models and datasets, we train the model for 100,000 steps in text classification tasks, and 50 epochs in image classification tasks. We pick the best checkpoint on the validation set (if provided) for evaluation.

C RESULTS ON DIFFERENT COMBINATIONS OF RANKING LOSSES AND INTERACTION PATTERNS

Here we provide the experiment results on different combinations of loss functions and interaction patterns. Besides PairLogLoss and ApproxNDCG included in Section 5, we also include results from Gumbel-ApproxNDCG and MSE here. Note that we use the rescaled MSE as suggested in (Hui & Belkin, 2021) on ImageNet dataset, since there is only 1 correct class in 1,000 classes, and regular MSE performs poorly on such imbalanced data. In particular, as the number of combinations is large, we display the results with respect to each dataset and instance encoder in each table for better visualization and comparison. Then in each table, we group the results from different combinations of loss functions and interaction patterns according to the three evaluation metrics. We underline the top 3 combinations (ties are included) under each metric, and use $*$ to mark the best. The results of all datasets and instance encoders are shown in Tables 7 - 12.

Firstly, the conclusions on loss functions, interactions and evaluation metrics from the tables are the same as those in Sections 5.2 and 5.3. The Gumbel-ApproxNDCG performs well when more complicated interactions such as latent cross and concatenating embeddings are applied. MSE achieves the best top-1 accuracy in GoEmotions with BERT and CIFAR-10 with VGG16. It indicates that MSE can be effective in optimizing top-1 metrics in certain tasks, which aligns with the observation in Hui & Belkin (2021). However, MSE is not suitable for direct application in problems with highly imbalanced correct and incorrect classes as on the ImageNet classification task. A few rescaling techniques with hyper-parameters need to be applied for MSE to perform properly, which also increase the burden of hyper-parameters tuning. Besides, adding different combinations of losses and interaction patterns can always improve the performance, as indicated by $*$ under each metric. Note that we did not further tune the hyper-parameters of instance encoders for different architectures of Rank4Class. It is possible that the hyper-parameters of encoders are more suitable for the

Table 7: Results on GoEmotions with BERT as text encoder.

GoEmotions + BERT						
Metric	Interaction	SoftmaxCE	PairLogLoss	ApproxNDCG	Gumbel ApproxNDCG	MSE
Top-1 Accuracy	dot product	0.5900	0.5821	0.5858	0.5889	0.5880
	LC+MLP	0.5865	0.5802	0.5935*	0.5930	0.5869
	Concat+MLP	0.5898	0.5845	0.5908	0.5841	0.5837
Top-5 Accuracy	dot product	0.8665	0.8800	0.8700	0.8620	0.8500
	LC+MLP	0.8715	0.8775	0.8760	0.8715	0.8570
	Concat+MLP	0.8760	0.8815*	0.8780	0.8710	0.8615
NDCG@5	dot product	0.7390	0.7427	0.7402	0.7371	0.7300
	LC+MLP	0.7411	0.7422	0.7462*	0.7453	0.7350
	Concat+MLP	0.7436	0.7419	0.7461	0.7405	0.7355

Table 8: Results on GoEmotions with ELECTRA as text encoder.

GoEmotions + ELECTRA						
Metric	Interaction	SoftmaxCE	PairLogLoss	ApproxNDCG	Gumbel ApproxNDCG	MSE
Top-1 Accuracy	dot product	0.6155	0.6022	0.6233	0.6137	0.6203
	LC+MLP	0.6131	0.6087	0.6233	0.6155	0.6264*
	Concat+MLP	0.6100	0.6052	0.6220	0.6194	0.6185
Top-5 Accuracy	dot product	0.8955	0.9120	0.9075	0.8935	0.8970
	LC+MLP	0.9070	0.9110	0.9135*	0.9000	0.8985
	Concat+MLP	0.9025	0.9130	0.9085	0.9010	0.8960
NDCG@5	dot product	0.7696	0.7717	0.7803	0.7672	0.7737
	LC+MLP	0.7745	0.7749	0.7830*	0.7741	0.7769
	Concat+MLP	0.7706	0.7747	0.7797	0.7735	0.7724

classical MCC models, and fine-tuning may further boost the performance of Rank4Class. Last but not least, we see that NDCG is more informative in evaluating MCC performance than Top-5 Accuracy which creates many ties. For example, In Table 9 for MIND with BERT, Top-5 Accuracy fails to find the best performing model, while NDCG@5 can successfully differentiate them.

Table 9: Results on MIND with BERT as text encoder.

MIND + BERT						
Metric	Interaction	SoftmaxCE	PairLogLoss	ApproxNDCG	Gumbel ApproxNDCG	MSE
Top-1 Accuracy	dot product	0.6910	0.6879	0.6903	0.6912	0.6929
	LC+MLP	0.6980*	0.6949	0.6919	0.6921	0.6944
	Concat+MLP	0.6949	0.6871	0.6923	0.6908	0.6929
Top-5 Accuracy	dot product	0.9385	0.9475*	0.9475*	0.9425	0.9305
	LC+MLP	0.9465	0.9470	0.9475*	0.9455	0.9285
	Concat+MLP	0.9445	0.9475*	0.9495	0.9435	0.9270
NDCG@5	dot product	0.8278	0.8325	0.8336	0.8311	0.8259
	LC+MLP	0.8356*	0.8354	0.8343	0.8328	0.8248
	Concat+MLP	0.8331	0.8327	0.8335	0.8316	0.8237

Table 10: Results on MIND with ELECTRA as text encoder.

MIND + ELECTRA						
Metric	Interaction	SoftmaxCE	PairLogLoss	ApproxNDCG	Gumbel ApproxNDCG	MSE
Top-1 Accuracy	dot product	0.7291	0.7228	0.7335	0.7323	0.7301
	LC+MLP	0.7311	0.7221	0.7326	0.7345	0.7308
	Concat+MLP	0.7322	0.7247	0.7360*	0.7345	0.7335
Top-5 Accuracy	dot product	0.9575	0.9625	0.9570	0.9575	0.9465
	LC+MLP	0.9596	0.9625	0.9630*	0.9610	0.9480
	Concat+MLP	0.9575	0.9630*	0.9570	0.9605	0.9445
NDCG@5	dot product	0.8573	0.8574	0.8589	0.8589	0.8526
	LC+MLP	0.8567	0.8571	0.8613*	0.8609	0.8531
	Concat+MLP	0.8581	0.8585	0.8605	0.8599	0.8525

Table 11: Results on ImageNet with ResNet50 as image encoder.

ImageNet + ResNet50						
Metric	Interaction	SoftmaxCE	PairLogLoss	ApproxNDCG	Gumbel ApproxNDCG	MSE (rescaled)
Top-1 Accuracy	dot product	0.7626	0.7636	0.7638	0.7635	0.7354
	LC+MLP	0.7638	0.7630	0.7635	0.7640	0.7582
	Concat+MLP	0.7634	0.7621	0.7639	0.7642*	0.7617
Top-5 Accuracy	dot product	0.9310	0.9320	0.9315	0.9320	0.9105
	LC+MLP	0.9320	0.9325*	0.9320	0.9320	0.9310
	Concat+MLP	0.9325*	0.9325*	0.9320	0.9320	0.9320
NDCG@5	dot product	0.8574	0.8582	0.8580	0.8582	0.8324
	LC+MLP	0.8584	0.8580	0.8582	0.8584	0.8544
	Concat+MLP	0.8583	0.8577	0.8582	0.8585*	0.8572

Table 12: Results on CIFAR-10 with VGG16 as image encoder.

CIFAR-10 + VGG16						
Metric	Interaction	SoftmaxCE	PairLogLoss	ApproxNDCG	Gumbel ApproxNDCG	MSE
Top-1 Accuracy	dot product	0.9344	0.9357	0.9359	0.9355	0.9352
	LC+MLP	0.9360*	0.9358	0.9354	0.9360*	0.9358
	Concat+MLP	0.9357	0.9356	0.9356	0.9353	0.9360*
Top-5 Accuracy	dot product	0.9980	0.9985*	0.9985*	0.9980	0.9985*
	LC+MLP	0.9980	0.9985*	0.9975	0.9980	0.9975
	Concat+MLP	0.9980	0.9980	0.9975	0.9985	0.9965
NDCG@5	dot product	0.9719	0.9721	0.9723*	0.9721	0.9716
	LC+MLP	0.9722	0.9721	0.9717	0.9717	0.9714
	Concat+MLP	0.9723*	0.9721	0.9719	0.9722	0.9716