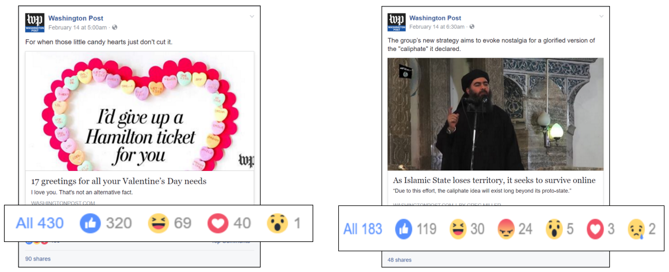
Predicting Emotions of Facebook Posts as the Imbalanced Label Ranking Problem

Anonymous
Anonymous

Anonymous
Anonymous


Washington Post
February 14 at 5:00am ·
For when those little candy hearts just don't cut it.
I'd give up a Hamilton ticket for you
17 greetings for all your Valentine's Day needs
I love you. That's not an alternative fact.
All 430 👍 320 😆 69 ❤ 40 😮 1
90 shares

Washington Post
February 14 at 6:30am ·
The group's new strategy aims to evoke nostalgia for a glorified version of the "caliphate" it declared.
As Islamic State loses territory, it seeks to survive online
"Due to this effort, the caliphate idea will exist long beyond its proto-state."
All 183 👍 119 😆 30 😡 24 😮 5 ❤ 3 😢 2
48 shares

(a) (b)

Fig. 1: Two Washington Post Facebook posts with different emoticon reactions @ www.facebook.com/washingtonpost/

***Abstract*—We investigate how to accurately predict latent emotions of Facebook posts by modeling it as the *label ranking* problem, a type of preference learning problem in AI. It learns to associate instances with different rankings over a predefined label set. Unlike conventional label ranking data, however, social emotion mining data such as Facebook posts is severely imbalanced, rendering existing label ranking solutions ineffective. Toward this challenge, we propose: (1) a novel measure, named as G-mean-rank, that is robust toward imbalanced label ranking data; and (2) a novel decision tree based label ranking algorithm, named as DTPG. Through empirical validation using four real datasets, we demonstrate the superiorities of our proposals over two popular label ranking measures and four competing label ranking algorithms.**

Acknowledgment

The authors would like to thank...

I. Introduction

It has become increasingly important for businesses to better understand their users and leverage the learned knowledge to their advantage. One popular method for such a goal is to mine users' digital footprints to unearth latent "emotions" toward particular products or services. In this paper, we focus on such a problem, known as *social emotion mining* [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], to uncover latent emotions of social media posts, documents, or users. As a concrete example of social emotion mining, consider Facebook posts, examples shown in Fig. 1, to which users can select one of six emoticons {like, love, haha, wow, sad, angry} to express their emotions.

In this setting, a natural question is if one can accurately predict the resultant emotions of a given post or not. The problem can be framed in roughly three ways: (1) predict single dominating emoticon out of six (i.e., multiclass classification); (2) predict the ranking among six emoticons (i.e., label ranking) according to number of clicks; and (3) predict accumulated numbers of clicks of six emoticons (i.e., regression). We argue that neither the classification nor the regression framework is suitable for the given problem. First, it is hardly convincing that single out of six emoticons can contrast different latent emotions across posts. For example, two posts in Fig. 1 have very different contents and contexts but share the dominating emoticon of "like." Second, as the target space of the regression framework is very large (i.e., $\mathcal{N}^d$, where $\mathcal{N}$ is natural number set and d is the size of the label set), solving the problem precisely becomes challenging, yet knowing the exact number of emoticons generates little extra information than knowing the ranking among emoticons. Therefore, we chose the second framework and model the problem as the "label ranking" problem. The label ranking framework reduces the target space from the regression framework, but still retains considerable expressive power–i.e., $d!$ different rankings per post. For example, two posts in Fig. 1 have different rankings of emotions such as [like, haha, love, wow] and [like, haha, angry, wow, love, sad], respectively.

The label ranking problem asks if one can learn a model to annotate an instance with a ranking over a finite set of predefined labels. Label ranking can be seen as a specific type of the preference learning problem [**?**] in AI. Previous works on label ranking in general assume that data is balanced. However, in the case of social emotion mining, data from different community might have different tendency in selecting labels, which results in imbalanced ranking data. For example, ordinary Facebook users (i.e., posters) tend to post happy stories more frequently and their friends (i.e., readers) are more willing to give positive feedback such as "like" or "haha". Therefore, the ranking distribution is highly biased toward positive labels than negative ones. In label ranking, further, predicting lower ranks may reveal more informative and rewarding insights. Conventional label ranking methods, however, might be overwhelmed by positive labels and fail to reveal rare rankings with more angry reactions than love, for instance.

Toward this challenge of "imbalance" in the label ranking problem, we make two contributions: (1) We first show the inadequacy of popular performance measures in label ranking (e.g., *Kendall tau correlation* [**?**]) to handle the imbalanced data and propose an improved measure, named as G-mean-rank, that automatically assigns higher penalty for labels seldom highly ranked, avoiding the majority rankings to cover the information conveyed by minority ones. We intuitively and experimentally demonstrate that G-mean-rank is more meaningful than existing ones in label ranking; and (2) We develop the Decision Tree with Point-wise Gini index (DTPG) method, a decision tree based label ranking algorithm that is robust in imbalanced data without any re-sampling or costs as hyper-parameters. It achieves competitive results in terms of existing performance measures and significant improvement over the state-of-the-art algorithms in terms of (more meaningful) G-mean-rank.

II. RELATED WORKS

There are three classes of label ranking methods. First, label-wise methods [?], [?], [?], [?], [?] treat label ranking as regression problems for the relevant score of each label or position of ranking. Second, pair-wise methods [?], [?], [?], [?], [?], [?], [?], [?], [?] decompose label ranking problem to binary classification problem for each pair of labels and then aggregating pairwise results into rankings. Third, rank-level methods employ different ranking distance measures to directly predict rankings without decomposing, such as Mallows model [?] based methods [?], [?] and weighted distance model [?] based methods [?], [?]. Our proposed solution, DTPG, belongs to the third class.

Previous label ranking works [?], [?], [?], [?], [?], [?], [?], [?] typically evaluate preformance using ranking distance measure such as *Kendall tau correlation* [?] or Spearman's rank correlation [?], which are known to be ineffective in handling imbalanced data. On the other hand, social emotion mining works [?], [?], [?], [?] typically measure performance using metrics from information retrieval community, such as $ACC@k$ and $nDCG@k$ [?], emphasizing the intuition that higher ranked positions are more informative. Similarly, some rank modeling works in statistics [?], [?], [?] weight the distance between ranks to model such bias. Note that the bias here is rewarding heterogeneity of different ranking positions, rather than label-wise bias considered in this work. Hence, in imbalanced data, those performance measures are not good enough.

Imbalanced data problem has been previously investigated under the classification framework [?]. In terms of performance measure, the inadequacy of a simple but popular metric, accuracy, in handling imbalanced cases is widely shown by works [?], [?], [?], [?], [?]. Therefore, to address the shortcomings of the measure accuracy, previous works have proposed alternatives such as F-measure, ROC curves [?], [?] and G-Mean [?] as performance measurement for imbalanced cases. We extend the notion of G-Mean for multi-class classification to the problem of label ranking and propose our own G-mean-rank in this work.

Other popular methods to address imbalance data learning problem in classification framework include random sampling [?], [?] and cost-sensitive methods [?], [?], [?]. Both methods try to first obtain balanced data from original imbalanced data so that the problem is reduced to the balanced classification. However, these methods often involve tricky hyper-parameter tuning, especially in multi-class classification [?], which will become even more severe issue in label ranking framework. In contrast, the DTPG method proposed in this work is free of hyper-parameters related to data imbalance.

III. PRELIMINARIES

A. The Label Ranking Problem

Label ranking can be seen as an extension of conventional classification. Each instance $\mathbf{x}$ from an instance space X with dimension m, the number of features, is associated with a ranking vector $\phi(\mathbf{x})$, over a finite set of predefined labels $\mathcal{Y} = \{y_1, y_2, ..., y_d\}$ of size d. $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_d(\mathbf{x}))$, where $\phi_i(\mathbf{x}) \in \mathcal{Y}$ and $\phi_i(\mathbf{x}) \neq \phi_j(\mathbf{x}), \forall i \neq j$. $\phi_i = y_l$ indicates that label y_l ranks on position i. For annotation consistence, a ranking position vector is defined as $\pi(\mathbf{x}) = (\pi_{y_1}(\mathbf{x}), \pi_{y_2}(\mathbf{x}), ..., \pi_{y_d}(\mathbf{x}))$, where $y_i \in \mathcal{Y}$ and $\pi_{y_i} \in \{1, 2, ..., d\}$, which means that label y_i ranks on position $\pi_{y_i}(\mathbf{x})$ for instance $\mathbf{x}$. For example, a ranking $\phi = (y_1, y_4, y_3, y_2)$, can also be written as $\pi = (1, 4, 3, 2)$. Since the relation between ϕ and π is bijective, below we refer to both as ranking. A complete ranking is a permutation of $\mathcal{Y}$. Hence $\phi(\mathbf{x}) \in \Omega_d$ for complete ranking, where Ω_d denotes the class of permutations of $\mathcal{Y}$ of size d. As usual, in addition, $\hat{\phi}$ denotes a predicted ranking.

In real-world applications, observed rankings are not necessarily complete. For instance, Fig. 1(a) has an incomplete ranking of four emotions such as [like, haha, love, wow] while Fig. 1(b) has a complete ranking of six emoticons such as [like, haha, angry, wow, love, sad], respectively. In social emotion mining context, there are posts without any vote on certain emoticon labels, which implies that those labels are not relevant at all to the corresponding posts. Then, those irrelevant labels should not appear in normal rankings. Hence in this work, ranking $\phi(\mathbf{x})$ is allowed to be incomplete, but only in the lowest positions, that is, $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_l(\mathbf{x}), -1, ..., -1)$, indicating that only labels in $\{\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_l(\mathbf{x})\} \subset \mathcal{Y}$ are relevant to instance $\mathbf{x}$ while other labels are irrelevant to $\mathbf{x}$. Then $\pi_{y_i}(\mathbf{x}) = +\infty$, if y_i is not in $\phi(\mathbf{x})$. In general, $\phi(\mathbf{x}) \in \Omega_d^* = \cup_{l=1}^{d}(\Omega_d^l \circ \{-1\}^{d-l})$, where Ω_d^l denotes the class of permutation of $\mathcal{Y}$ taken l at a time and $\circ$ is concatenation operation.

We call this setting as *tail abstention*, where "tail" indicates that those irrelevant labels are at an extra tail position as irrelevant, and "abstention" [?], [?] means that there is no preference relation between those irrelevant labels. The symbols used in this work are summarized in Table. I. The goal of label ranking is to learn a mapping f: $\mathcal{X} \rightarrow \Omega_d^*$ based on training data, where each instance is with a feature vector $\mathbf{x}$ and an associated ranking ϕ. The label ranking problem is formally defined as follows.

Problem 1 (Label Ranking): For an instance with feature vector $\mathbf{x}$, predict ranking $\phi(\mathbf{x}) = f(\mathbf{x})$, where $\phi(\mathbf{x}) \in \Omega_d^*$.

TABLE I: symbol reference

Symbol	Description
$\mathcal{X}$	feature space
$\mathcal{Y}$	label set
m	dimension of the feature space
d	size of $\mathcal{Y}$
Ω_d	rankings with size of label set d
Ω_d^*	Ω_d enlarged with tail abstention
$\mathbf{x}$	feature vector
ϕ	ranking vector
π	ranking position vector
ϕ_i	label ranks on position i in ϕ
π_y	the position on which label y ranks in π

B. Imbalance in Label Ranking

In social emotion mining context, imbalance in data refers to the characteristics of data where documents with some emotional reactions are rarer than those with others. In the context of label ranking, it means that instances with some rankings are rarer than those with others. In the classification setting, imbalance in data is the skewed distribution of data among different classes. However, in the label ranking setting, imbalance is *not* that obvious. A naive choice is treating different rankings as different classes and the problem reduces to a classification problem. However, classification framework ignores the fact that different rankings are not independent or equal-interval with each other. Instead, therefore, here imbalance is defined based on pairwise comparisons.

Given any pair of labels $\{y_i, y_j\}$, $y_i, y_j \in \mathcal{Y}$, and an instance ν, a pairwise comparison function is defined as:

$$I_\nu(y_i, y_j) = \begin{cases} 1, & \text{if } \pi_{y_i} < \pi_{y_j} \text{ for instance } \nu \\ 0, & \text{otherwise.}^1 \end{cases} \tag{1}$$

Then, for each label pair y_i, y_j, imbalance in data set $\mathcal{D}$ can be seen as the difference between $\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j)$ and $\sum_{\nu \in \mathcal{D}} I_\nu(y_j, y_i)$. Since a ranking consists of pairwise comparisons of all pairs, a single-value imbalance measure for label ranking, $IMBA-rank$, of data set $\mathcal{D}$ is defined as:

$$IMBA\text{−}rank(\mathcal{D}) = \frac{1}{2} \sum_{i,j=1,\, i \neq j}^{d} |log(\frac{\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j) + 1}{\sum_{\nu \in \mathcal{D}} I_\nu(y_j, y_i) + 1})|, \tag{2}$$

where the extra $+1$ terms in both numerator and denominator avoid the zero in equation and the absolute operator ensures the imbalance of different ordered pairs of labels not to cancel each other. Then, when data is perfectly balanced, $IMBA - rank$ should be 0. The more imbalance the data has, the larger $IMBA - rank$ gets.

Take the Facebook scenario in Sec. I as an example. In a data set of posts generated by ordinary users, posts with positive feedback are dominating. In this intuitively imbalanced data set, posts with emoticon "love" ranking higher than "angry" are far more frequent than those with opposite ranking. On the other hand, according to eq. 2, the term of pair of labels $\{y_i, y_j\} = \{\text{“}love\text{”}, \text{“}angry\text{”}\}$ contributes to large $IMBA-rank$, which by our definition, indicates imbalance. Hence our definition of imbalance is consistent with what is required in real world applications.

[1]In the tail abstention case, when only one of the two labels is in $\phi(\mathbf{x}_\nu)$, then that label is considered a rank higher than the other; when neither of them is in $\phi(\mathbf{x}_\nu)$, then they are not comparable, so that $I_\nu(y_i, y_j) = 0$

TABLE II: confusion matrix of two-class classification

		True classes p	True classes n
Hypothesis	Y	TP(True Positive)	FP(False Positive)
	N	FN(False Negative)	TN(True Negative)
column sum		p_c	n_c

IV. Imbalance-Aware Performance Measure

A. Previous Measures for Label Ranking

In this section, some popular and commonly used performance measures are described.

One of the most popular performance measures in label ranking community is *Kendall's tau correlation* [?], which measures the ratio of ratio of pairwise rank inversions of labels between two rankings. The correlation tau for two rankings $\{\pi, \hat{\pi}\}$ is formally defined as

$$tau = \frac{C(\pi, \hat{\pi}) - D(\pi, \hat{\pi})}{C(\pi, \hat{\pi}) + D(\pi, \hat{\pi})}, \tag{3}$$

where $D(\pi, \hat{\pi}) = \#\{(i,j) | i < j, \pi_{y_i} > \pi_{y_j} \wedge \hat{\pi}_{y_i} < \hat{\pi}_{y_j}\}$ and $C(\pi, \hat{\pi}) = \#\{(i,j) | i < j, \pi_{y_i} > \pi_{y_j} \wedge \hat{\pi}_{y_i} > \hat{\pi}_{y_j}\}$ denote the number of discordant and consistent-ordered pairs of labels between two rankings, respectively.

To emphasize the importance of higher positions in ranking, previous works on social emotion mining usually use $ACC@k$ as performance measure. The $ACC@k$ of an instance is defined as

$$ACC@k(\phi, \hat{\phi}) = I(\phi_i = \hat{\phi}_i \vee \phi_i = -1 \vee \hat{\phi}_i = -1 | \forall i \in \{1, 2, ..., k\}), \tag{4}$$

where $I()$ is the indicator function and $ACC@k(\phi, \hat{\phi}) = 1$ only when the first k positions of two rankings are consistent, noting that necessary modification concerning tail abstention is made compared with common definition.

A more flexible measure is $nDCG@k$ (Normalized Discounted Cumulative Gain at position k) [?] which is frequently used in information retrieval when multiple levels of importance of different ranking positions are considered. However, the importance level, or relevance score, is case-specific. Therefore, $nDCG@k$ is not compared in this work.

B. Imbalance Problem in Classification

As the definition of imbalance in the label ranking setting is based on decomposed pairwise comparisons, the performance measure problem of imbalance data in the classification setting will shed some light on that in label ranking.

In the basic two-class classification problem, all classification results can be shown in a confusion matrix, as Table II. The column labeled by p (resp. n) contains instances in positive (resp. negative) class as ground-truth; the row labeled by Y (resp. N) is the results predicted to be positive (resp.

negative). Typical performance measure used in balanced data is $Accuracy = \frac{TP+TN}{p_c+n_c}$, calculating the percentage of correctly predicted instances in the whole data set regardless of what class they belong to. Problem occurs when data is imbalanced, which means, without loss of generality, $p_c \gg n_c$. In terms of $Accuracy$, it is much more rewarding to enlarge top left cell (TP) than right bottom cell (TN) in the confusion matrix, which are both important for a good classifier. In an imagined scenario, if $p_c : n_c = 95 : 5$, then a naive classifier that classifies all instances as positive would achieve 95% $Accuracy$. This classifier is obviously ineffective in practice.

A straightforward solution is to combine classification performance in different columns in a more balanced way. First, performance in each column can be calculated as $recall_p = \frac{TP}{p_c}$ or $recall_n = \frac{TN}{n_c}$, where in terms of two-class classification, $recall_p$ is usually called $recall$ and $recall_n$ is known as $specificity$. Then, these two performance results can be combined regardless of class distribution of data. To assign more penalty to biased performance in different columns, $G-mean$ uses geometric mean of two results, defined as $G-mean = \sqrt[2]{recall_p * recall_n}$. It is obvious that $G-mean$ will not be high if performance in either column is bad.

C. *Imbalance Problem in Label Ranking*

Here we show that previous measures in label ranking suffer from imbalanced data problem similar to imbalance problem in classification.

First, *Kendall's tau correlation*, or simply tau, as a performance measure, is based on pairwise comparisons. Concerning only one pair of labels, the problem reduces to classification and tau measures the percentage of correctly predicted instances in the whole data set regardless of different orders of that pair, which is exactly the same as accuracy. According to the definition of imbalance in label ranking, which is based on pairwise imbalance, using tau as the performance measure, the instances with rare ordered pairs of labels, usually more informative, become cheap to ignore.

In terms of $ACC@k$, specific position has to be considered to illustrate the problem. Considering any position r in rankings such that $r <= k$, to assign a label $y \in \mathcal{Y}$ on r is a multi-class classification problem. According to the definition of imbalance in label ranking, given a pair of labels $\{y_i, y_j\} \subset \mathcal{Y}$, assuming y_i dominates y_j in the data set, that is, $\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j) \gg \sum_{\nu \in \mathcal{D}} I_\nu(y_j, y_i)$, then generally there are more instances with $\phi_r = y_i$ than those with $\phi_r = y_j$if r is a higher position, and vice versa. Hence, the multi-class classification problem is imbalanced. In addition, since $ACC@k$ in terms of that position only considers the percentage of correctly classified labels, it suffers from the same problem as $Accuracy$ in classification in imbalanced data set. Similar problem can be shown in other position-weighted label ranking performance measures, such as $nDCG@k$ and weighted distance [**?**].

For a better illustration, consider a toy data set as an example. Here the label set $\mathcal{Y} = \{y_i | i \in \{1, 2, 3, 4\}\}$ with

TABLE III: A quadrant table

	classification	label ranking
balanced	$Accuracy$, ...	$ACC@k$, tau, ...
imbalanced	$G-mean$, ...	?

$d = 4$. The dataset contains 100 instances where 90 of them are associated with rank $\boldsymbol{\phi^9} = (y_1, y_2, y_3, y_4)$ while the others are associated with $\boldsymbol{\phi^1} = (y_1, y_4, y_2, y_3)$. Then, a naive label ranking algorithm predicts all instances to be with rank $\hat{\phi} = \boldsymbol{\phi^9}$ for this toy data set. Then, the performance of the naive algorithm is $tau \approx 93\%$ and $ACC@k = 90\%$ if $k = 2$, which is high performance compared with $tau = 100\%$ and $ACC@k = 100\%$ for a perfect algorithm.

In real world applications, the naive label ranking algorithm is useless and such a high performance result is misleading. For example, in social emotion mining, if the toy data set is the collection of posts of a celebrity on Facebook, with labels $(y_1, y_2, y_3, y_4) = (like, love, haha, angry)$, then the post with minority ϕ^1 ranking of labels tells a totally distinct story compared with usually positive posts. Hence certainly a label ranking algorithm should be able to point the post out from others based on post features.

Therefore, we view that an appropriate performance measure in the imbalanced label ranking problem, just as $G-mean$ in the imbalanced classification problem, is currently lacking, as illustrated in Table. III

D. *Our Proposal: G-mean-rank*

In the classification setting, the key to solve the imbalance problem on performance measure is to separate the calculation of classification performance in different classes (different columns in the confusion matrix Table. II). Similarly, in the label ranking setting, the importance of such separation is illustrated by tau and $ACC@k$ as counter-examples. However, as previously argued, rankings themselves are not appropriate to be treated as classes, ranking as a whole is not a good separation unit. Another possible choice is to decompose label ranking into label-wise classification problems, which for each label, considers different positions it ranks as classes. However, as different positions in ranking are considered different in importance, such decomposition does not match the intuition about rankings.

The only remaining but natural choice is, then, to calculate performance in different decomposed ordered pairwise comparison classes separately. For each ordered pair of labels (y_i, y_j), the label ranking problem reduces to the classification problem where class of instance ν is defined as Positive if $I_\nu(y_i, y_j) = 1$ or Negative if $I_\nu(y_j, y_i) = 1$. Note that if two labels are both missing, the instance is excluded for this ordered pair. Since the Negative class of ordered pair (y_i, y_j) is the same as Positive class of (y_j, y_i), only positive class for each ordered pair is considered. Then, the performance in terms of each ordered pair is simply performance in classification setting. Similar to classification in Sec. IV-B, the recall for ordered pair (y_i, y_j) in data set $\mathcal{D}$ is defined as

$$recall_{\mathcal{D}}(y_i, y_j) = \frac{\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j) I_{\hat{\nu}}(y_i, y_j) + 1}{\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j) + 2}, \quad (5)$$

where $I_{\hat{\nu}}(y_i, y_j)$ is the pairwise comparison function for predicted ranking for instance ν, and the extra $+1$ term in numerator and $+2$ term in denominator are smooth terms.

The final step is to combine them into one quantity. Similar to classification, geometric mean is used and *G-mean-rank* GMR for data set $\mathcal{D}$ is defined as

$$GMR_{\mathcal{D}} = \sqrt[P]{\prod_{i \neq j}^{d} recall_{\mathcal{D}}(y_i, y_j) I(\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j) > 0)}, \quad (6)$$

where $I(\sum_{\nu \in \mathcal{D}} I_\nu(y_i, y_j) > 0)$ factor excludes ordered pairs that do not exist in $\mathcal{D}$ and $P = \sum_{\nu \in \mathcal{D}} I(I_\nu(y_i, y_j) > 0)$ is the number of ordered pairs involved. For each pair of labels, GMR is the same as $G-mean$ for two-class classification. Hence according to the definition of imbalance in label ranking, GMR is insensitive to imbalanced label ranking data.

In the toy data set used in Sec. IV-C, the predicted rankings of the naive label ranking algorithm is considered. There are six ordered pairs (y_1, y_2), (y_1, y_3), (y_1, y_4), (y_2, y_3), (y_2, y_4) and (y_3, y_4) that are perfectly predicted, and the remaining two (y_4, y_2) and (y_4, y_3) that are not well predicted. Then GMR for this naive algorithm in the toy data set is $GMR \approx 53\%$, which is not high compared with 97% for a perfect algorithm. Hence GMR rightfully gives sufficient penalty to a naive algorithm which is not supposed to perform well.

E. Comparing Different Measures

In general, it is challenging to compare different measures to demonstrate one's superiority over the other. As our focus of study is to handle imbalanced data set, we use the notion of the *robustness* of measures to determine one's superiority. Intuitively, the robustness is how well a given performance measure can capture the performance of an algorithm in handling imbalanced data. Here, the performances of different measures are captured by the difference of its values when applied to a naive algorithm vs. a state-of-the-art algorithm. Although ideally it should be compared with a perfect oracle algorithm, using the state-of-the-art is more valuable in practice. To avoid the case where a state-of-the-art algorithm may have bad results for certain performance measures, the highest values achieved by all algorithms are used (see more details in Sec. VI-B).

V. Imbalance-Aware Prediction Algorithm

A. Four Baseline Algorithms

Typical social emotion mining data contains natural numbers of votes for different labels. Though as argued before, label ranking is the most appropriate framework for social emotion mining, it coerces the original target space, which might loses some information about data. Hence the first baseline algorithm we choose is multi-class logistic regression (**LogR**), which takes use of all information about data. LogR takes the number of votes for different labels for each instance as input for target and predict the probability distribution of labels by maximum likelihood estimation. To compare with label ranking models, labels are ranked according to their predicted probability. The probability distribution model for label $y \in \mathcal{Y}$ is

$$P(y|\mathbf{x}) = \frac{exp(\beta_y \cdot \mathbf{x})}{Z(\mathbf{x})}, \quad (7)$$

where β_y is $(m+1)$-dimension parameter vector for label y, $\mathbf{x}$ is a feature vector with additional constant 1, and $Z(\mathbf{x})$ is the normalization constant. The parameters $\{\beta_y | y \in \mathcal{Y}\}$ are estimated by maximizing the likelihood of training set $\mathcal{D}$, as

$$\mathcal{L}(D; \beta) = \prod_{\nu \in \mathcal{D}} \prod_{y \in \mathcal{Y}} P(y|\mathbf{x}_\nu)^{N_{\nu y}}, \quad (8)$$

where $N_{\nu y}$ is the number of votes for label y in instance ν.

We also compared our algorithm with two state-of-the-art label ranking algorithms. One is ranking by pairwise comparison (**RPC**) [**?**]. It predicts pairwise order for each pair of labels using logistic regression and then combines them into ranking output. There is only one minor modification. In case of tail abstention, both missing label pairs indicate that they are not preferred to each other, as abstention means. Therefore, missing label pairs are assigned a small weight α, and counted as one preference relation for each order. The weight is picked empirically, $\alpha = 1/64$ used in this work, and results do not appear sensitive to the weight for a range of α values. The other algorithm is the label-wise decomposition (**LWD**) [**?**], which predicts position probability distribution of each label and then combines them to minimize expected Spearman's footrule [**?**]. Note that both label ranking algorithms represent the state-of-the-arts of two categories– i.e., pairwise decomposition and label-wise decomposition [**?**]. As to rank-level, a typical label ranking tree (LRT) algorithm may be too slow. We theoretically and experimentally compare the running time of our algorithm against LRT.

Finally, as to the naive algorithm described in Sec. IV-E, we use a simple heuristic one, named as **NAIVE**, that assigns the most common ranking in the training set to all instances in the test set regardless of their feature values. Because the data we use is rankings converted from numbers of votes for different labels, the output ranking of the NAIVE algorithm is set as the ranking of labels according to the accumulated votes in the data set.

B. Our Proposal: Decision tree with Point-wise Gini-index (DTPG)

Decision tree is one of the most popular methods in data mining [**?**], where tree based models can be used for both classification and regression problems. In a decision tree, each node T, regardless of being a leaf or internal node, represents a part of instance space $\mathcal{X}_T \subseteq \mathcal{X}$. In addition, the corresponding part in the target space $\Omega_T \subseteq \Omega$ is expected to be a small neighborhood around a point, or more generally, a subspace, spanned by the mapping from $\mathcal{X}_T$, corresponding to a constant or local model. The goal of recursive partitioning is to ensure that the descendant nodes have a smaller neighborhood compared with what the antecedent nodes have.

In the label ranking problem, the target space Ω_d^* is the collection of all rankings of size d, together with rankings with tail abstention. It is a space with a non-trivial topological structure rather than just independent discrete points. As a result, the label ranking problem should be considered as a regression problem. However, simple regression models might not work, because a general class of continuous mappings from Euclidean space to Ω_d^* is difficult to find. Decision tree model involves only local interpolation, averaging over a neighborhood in ranking space and local distance measure. In addition, decision tree is known to be insensitive to scaling in feature space. Hence it rules out the extrapolation of dominating rank by continuity, which is assumed by most methods. As long as the partition of the feature space is good enough, measured by the size of neighborhood in the target space that it is mapped to, rare instances in the target space can be figured out. Therefore, decision tree is a good model for handling the imbalanced label ranking problem.

Tree Induction. To learn a decision tree, a general algorithm begins with all instances in the root node. Then, it partitions the training data recursively, by one-dimension splits according to the comparison between thresholds and a feature value. The decision tree in this work is a binary tree.

The threshold and the feature for each split are selected by an exhausted search so that the sizes of the neighborhoods in the target space, estimated by training data in the resultant child nodes, become the smallest. Hence, the key design issue is to select an estimator of the size of the neighborhood in the target space. Note that in the label ranking problem, the size is not well defined in the first place. Therefore, it leaves much freedom in designing the estimator, or in other words, the impurity of the instances in a node. One intuition about the impurity of a set of rankings is the impurity of labels on each ranking position. Because labels are not related each other, for a given position, it is natural to adopt the impurity measurement in classification tree to measure the impurity of labels. Here, we choose the popular Gini index, and the Gini index of a tree node T for position i is defined as:

$$Gini_i(T) = \sum_{y \in \mathcal{Y}} \frac{(n_i(T) - n_{iy}(T))n_{iy}(T)}{n_i(T)^2}\,, \tag{9}$$

where $n_{iy}(T) = \sum_{\nu \in T} I(\phi_i(\mathbf{x}_\nu) = y)$ is the number of instances with label y ranking on position i and $n_i(T) = \sum_{y \in \mathcal{Y}} n_{iy}(T)$ denotes the number of instances with any label ranking on position i. Note that in tail abstention case, $n_i(T)$ can be different from $|T|$, the number of instances in T. Then the impurity for rankings of the node T can be measured by weighted sum of Gini index for each position, that is,

$$Gini(T) = \sum_{i=1}^{d} \frac{n_i(T)Gini_i(T)}{|T|}, \tag{10}$$

which is called point-wise Gini index. It is weighted sum rather than simple average because positions with missing labels contains less information so that should be considered less important. As a criterion for split, we use the weighted average of the child node impurity, the same with that in regression tree. For parent node T and potential child nodes T^- and T^+, the split criterion is defined as

$$criterion = |T|^{-1}(|T^+|Gini(T^+) + |T^-|Gini(T^-)). \tag{11}$$

Current stopping criterion is rather simple. The partitioning stops when no further partitioning is possible, that is, when there is no partitioning whose $criterion$ is smaller than $Gini(T)$ for current node T. As for pruning, we use Gini(T) replacing accuracy in classification tree, since GMR for a single node of tree is meaningless. The cost-complexity coefficient for pruning is determined by cross validation in training set with the goal to minimize GMR [**?**]. However, experiments in imbalanced data show that the performance is almost the same with that without pruning and the cost-complexity coefficient is always small, indicating that cost in training set, which is GMR here, is much more important than the complexity of the model. This can be interpreted as the risk of over-fitting is overwhelmed by the risk of missing rare rankings in imbalance case. Therefore, we do not post-prune the trees.

Consistence with Ranking Theory. This point-wise Gini index is consistent with our intuition about the purity of a set of rankings. To show that, we have to assume a measurement of the size of a neighborhood $\Omega(T)$ around a point in Ω_d^*, noting that the center ranking π_0 is unknown. Mallows model [**?**] is a popular assumption of probability model of rankings, using the annotation from [**?**], defined as

$$P(\pi|\theta, \pi^0) = \frac{exp(-\theta D(\pi, \pi^0))}{\psi(\theta)}, \tag{12}$$

where $\psi(\theta) = \sum_{\pi \in \Omega_d} exp(-\theta D(\pi, \pi^0))$ is normalization constant, θ the spread parameter, π^0 the center ranking and $D(,)$ the distance between two rankings, which is the number of discordant pairs between two rankings. Assuming the rankings in T are independently generated according to Mallows model, the spreading parameter θ measures the size of $\Omega(T)$. As Mallows model only considers complete raking, here we restrict to complete rankings. Under independence assumption, the expectation of point-wise Gini index for node T is

$$E(Gini(T)) = E(\sum_{i}^{d} \frac{n_i(T)Gini_i(T)}{|T|}) = \sum_{i}^{d} E(Gini_i(T)), \tag{13}$$

where $n_i(T) = |T|$ without tail abstention and for simplicity, ignoring T,

$$E(Gini_i) = \frac{(n_i - 1)}{n_i}(1 - \sum_{y \in \mathcal{Y}} P(\phi_i = y)^2). \tag{14}$$

According to Mallows model, without loss of generality, we assume the center ranking $\phi_i^0 = y_i, \forall i \in \{1, 2, ..., d\}$. Then ranking ϕ with $\phi_i = y_j$, is with probability $P(\phi) = \frac{\mathcal{O}(exp(-|i-j|\theta))}{\psi(\theta)}$. Therefore, for large enough θ, $P(\phi_i = y_j) =$

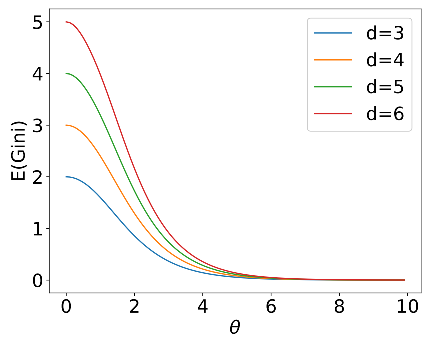


Fig. 2: E(Gini) versus θ

$\mathcal{O}(exp(-|i-j|\theta))$. Hence, when θ is larger, which is when the spread of Mallows model is smaller, then the probabilities $P(\phi_i = y_j)$ over $y_j \in \mathcal{Y}$ are more skewer toward smaller $|i-j|$. Therefore according to eq. 14, $E(Gini_i)$ is smaller, so is $E(Gini(T))$, as illustrated in Fig. 2 with different sizes of label set in the limitation of $n_i \to \infty$. Therefore, $Gini$ is a good estimator of the impurity of rankings in a node.

Time Complexity. It is worthwhile to note one trick used in the decision tree induction algorithm. In the exhausted search for the best threshold of a given feature for partitioning a node T, the instances should be sorted by the feature values. Afterward, the threshold search is to calculate the split criterion for potential partitions with thresholds along the sorted list. If the calculation of split criterion for each potential partition takes $o(|T|)$ steps on average, such sorting is helpful.

Algorithm 1 Best Split Threshold

```
Input: Instance set $\mathcal{D}$, feature $x$
Output: $threshold$, $criterion$
1:  sort $\mathcal{D}$ according to value of $x$
2:  $criterion \leftarrow +\infty$, $threshold \leftarrow Null$
3:  calculate $\{n_i(T^{\pm}), n_{iy}(T^{\pm}) | \forall i, y\}$ with $T^- = \emptyset$ and $T^+ = \mathcal{D}$
4:  for each instance $\nu$ in sorted $\mathcal{D}$ do
5:      $T^- \leftarrow T^- \cup \{\nu\}$, $T^+ \leftarrow T^+ \setminus \{\nu\}$
6:      update $\{n_i(T^{\pm}), n_{iy}(T^{\pm}) | \forall i, y\}$ with $T^{\pm}$
7:      calculate $criterion\_temp$ with $\{n_i(T^{\pm}), n_{iy}(T^{\pm}) | \forall i, y\}$
8:      if $criterion\_temp < criterion$ then
9:          $criterion \leftarrow criterion\_temp$
10:         $threshold \leftarrow x_\nu$
11:     end if
12: end for
13: output $threshold$ and $criterion$
```

The procedure of searching best split threshold for a given feature in DTPG is shown Algorithm. 1. For each potential partition, In $step.5$, the update involves the deletion (addition) of one ranking in each potential child node, which takes $\Theta(d^2)$ steps. In $step.6$, the calculation takes one scan over $\mathcal{D}$, taking $\Theta(|T|d^2)$ steps. Therefore, on average, the calculation of split criterion for each potential partition takes $\Theta(d^2)$ steps, typically much smaller than $|T|$. Hence, in DTPG, the running time for each partition is $\Theta(m|T|(log|T| + d^2))$.

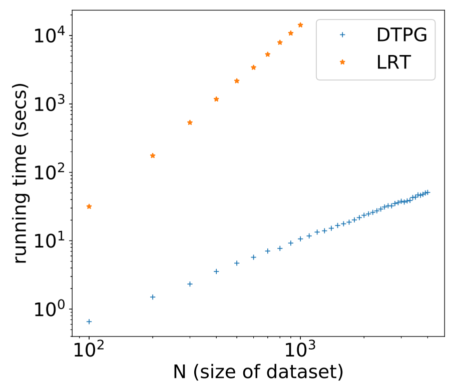


Fig. 3: Running time in a log-log plot

In contrast, LRT [?] takes $\Omega(|T|)$ steps in calculating split criterion for each potential partition. Therefore, the running time for each partition is $\Omega(m|T|^2 d^2)$. The running times of two methods applied to data of different sizes are shown in Fig. 3. Note that LRT algorithm is much slower than our proposed DTPG, when applied in large data set.

Prediction. In tree induction, we only care the size of the neighborhood in target space, but ignore the neighborhood center. In prediction, given an instance in test set, as it propagates to a leaf node, a ranking output should be assigned to the node.

There might be more than one rankings in a leaf node. Hence the problem is also called ranking aggregation. As for the DTPG algorithm in this work, the stopping criterion is so strict that the rankings in a leaf are pure enough. The only problem to consider is instances with tail abstention. There are two cases. First, if there are instances with tail abstention in the leaf node, but all labels appear in at least one rankings in the node, then the potential positions for those missing labels in rankings with tail abstention are considered to be similar to rankings with those labels. The idea is to choose more informative output. Second, if some labels are missing for all rankings in the node, then the output ranking should also keep those labels missing. This is different from the prediction method used in [?], where missing labels are considered as missing from random positions which is not consistent with the practical meaning of missing labels here.

Here for DTPG, we use a position-wise ranking aggregation method. From highest to lowest ranking position, given a position, it assigns the label that has not been assigned and appears most frequently at that position, to each position. When there is no such label for a position, it resorts to label distributions of other positions, from highest to lowest and does the same. If such label cannot be found for a position, which is only the case when all labels existing in the ranking set have been assigned, it leaves the position i empty, or $\phi_i = -1$.

VI. Empirical Validation

We attempt to validate if: (1) our proposed G-mean-rank is superior to two popular label ranking measures; and (2) our proposed DTPG outperforms competing four label ranking algorithms.

A. Datasets and Set-Up

In this work, we use emoticon clicks data of Facebook posts. For each post, there are six emoticon labels, {"like", "love", "haha", "wow", "sad", "angry"}. Each user (i.e., reader) can select one of the six labels for each post. For evaluating LogR, we use the number of votes for labels per post as the input. To obtain ranking, for each post, the labels are sorted according to their number of votes. There are four data sets: (1) public posts from random ordinary users, denoted as ROU (Random Ordinary Users); (2) New York Times (NYT)[2] posts; (3) the Wall Street Journal (WSJ)[3] posts; and (4) the Washington Post (WaPo)[4] posts. We have crawled all four sets of posts in 2016 after Facebook introduced six emoticons.

In a social emotion mining problem, the choice of effective features drawn from posts is critical. As we want to focus on the evaluation of our two proposals for the imbalanced label ranking problem (instead of finding effective features), however, we avoid more sophisticated features (e.g., user related or network structure based), and instead use fundamental textual features, extracted via AlchemyLanguage API [**?**]. For posts in ROU, only posts with text is included, and the document emotion of the text given by AlchemyLanguage is used as features. For posts in other three sets, if there is a link to external original full news, the document emotion of the full news is used as feature, and otherwise, only the text in posts is used. The returned document emotion from AlchemyLanguage consists of $[0, 1]$ scores, for five emotion dimensions, "anger", "joy", "fear", "sadness" and "disgust". The scores measure the amplitude of each emotion conveyed by the text. Then the four data sets are with the same feature and target format, that is, $\mathcal{Y} = \{\text{“}like\text{”}, \text{“}love\text{”}, \text{“}haha\text{”}, \text{“}wow\text{”}, \text{“}sad\text{”}, \text{“}angry\text{”}\}$ with $d = 6$ and $m = 5$ dimensional feature space.

The details of four data sets are shown in Table IV. Comparing ROU and the other three sets, the $IMBA-rank$ of ROU is much higher. This is due to the fact that readers of the posts from ordinary users are usually their friends, who tend to give positive feedback, $\{\text{“}like\text{”}, \text{“}love\text{”}, \text{“}haha\text{”}\}$ rather than negative one, $\{\text{“}sad\text{”}, \text{“}angry\text{”}\}$ All our datasets are significantly imbalanced in that "*like*" or other positive labels are more frequent. This is partially due to the interface limitation such that users have to hover their mouse over the *Like* button to be able to select other emoticons. To illustrate imbalance in label ranking more clearly, we also show the pairwise comparison matrix of ROU, as Table V, where the number in each entry (y_i, y_j) counts the number of posts with y_i being higher ranked than y_j. For instance, there are only 158 posts where "*angry*" is ranked higher than "*love*" compared with $3,987$ in contrast.

[2]www.facebook.com/nytimes/

[3]www.facebook.com/wsj/

[4]www.facebook.com/washingtonpost/

TABLE IV: Summary of four data sets

	ROU	NYT	WSJ	WaPo
# posts	$17,394$	$4,684$	$7,464$	$6,117$
$IMBA-rank$	3.03	1.94	2.96	2.14
# votes				
#"*like*"	$833,931$	$7,988,184$	$2,438,664$	$3,811,318$
#"*love*"	$14,386$	$577,780$	$105,003$	$221,827$
#"*haha*"	$3,281$	$433,866$	$129,709$	$248,211$
#"*wow*"	$2,610$	$327,862$	$83,543$	$178,952$
#"*sad*"	$2,430$	$786,071$	$70,039$	$332,367$
#"*angry*"	678	$1,066,545$	$92,504$	$548,515$

TABLE V: Pairwise comparison matrix of ROU

	"*like*"	"*love*"	"*haha*"	"*wow*"	"*sad*"	"*angry*"
"*like*"	–	$17,140$	$17,119$	$17,195$	$17,192$	$17,206$
"*love*"	79	–	$3,796$	$3,747$	$3,965$	$3,987$
"*haha*"	77	$1,089$	–	$1,241$	$1,382$	$1,383$
"*wow*"	29	749	$1,223$	–	$1,375$	$1,380$
"*sad*"	48	331	367	360	–	384
"*angry*"	23	158	188	181	207	–

B. Results and Discussion

Evaluating Measures. To show that one measure is better than the other, we first pick an algorithm A that is supposed to *not* work well for the imbalanced label ranking problem as well as a more advanced algorithm, B, that should outperform A. Then, a good measure should be able to capture reasonable performance improvement when algorithms B is compared against A. Based on this intuition, we use NAIVE in Sec. V-A as an example of A, and LogR, RPC, LWD, and DTPG as an example of B.

We compared three performance measures, tau, $ACC@k$ and GMR. For $ACC@k$, k is set as 3 to mimic the behavior of Facebook, where only top-3 emoticons of posts are shown by default. In Table VI, the improvement of best results of the other four algorithms (i.e., LogR, RPC, LWD, and DTPG) over NAIVE results in terms of three measures are shown. On average, GMR is able to capture more significant improvement than both tau and $ACC@3$ do. Concerning each data set, we see that $ACC@3$ only gets high improvement, 670.3% in NYT data set, but moderate improvement in the other three data sets. However, GMR captures high improvement in all data sets except ROU (but still substantially better than tau and $ACC@3$). This is due to the fact that posts in ROU tend

TABLE VI: Improvement of best over NAIVE results

Improvement (%)	ROU	NYT	WSJ	WaPo	average
$ACC@3$	1.6	670.3	19.9	134.7	206.6
tau	2.3	59.0	7.9	11.7	20.0
GMR	**125.5**	**754.8**	**506.1**	**521.6**	**477.0**

TABLE VII: Summary of results

		ROU	NYT	WSJ	WaPo
$ACC@3$	NAIVE	0.837	0.053	0.171	0.098
	LogR	0.843	0.068	0.152	0.108
	RPC	**0.850**	0.185	0.191	0.182
	LWD	0.849	0.201	**0.205**	**0.230**
	DTPG	0.814	**0.409**	0.203	0.176
tau	NAIVE	0.932	0.399	0.567	0.503
	LogR	0.934	0.428	0.580	0.520
	RPC	0.933	0.497	0.595	0.541
	LWD	0.937	0.499	0.603	**0.562**
	DTPG	**0.954**	**0.634**	**0.612**	0.554
GMR	NAIVE	0.152	0.080	0.088	0.077
	LogR	0.29	0.307	0.50	0.340
	RPC	0.24	0.429	0.345	0.358
	LWD	0.295	0.433	0.289	0.390
	DTPG	**0.343**	**0.68**	**0.534**	**0.48**

to be too short for AlchemyLanguage to extract meaningful emotions, which influence the performance of all algorithms except NAIVE. In conclusion, we believe that GMR is better suited than $ACC@3$ and tau to handle imbalanced label ranking data.

Evaluating Algorithms. The complete results are shown in Table VII. The table contains 5-fold cross-validation experimental results of all five algorithms, NAIVE, LogR, RPC, LWD and DTPG, applied on four data sets, evaluated by three measures, tau, $ACC@3$ and GMR. Table VII shows that DTPG achieves significantly better performance in all four data sets at the significance level of 0.05 in terms of GMR. In terms of $ACC@3$ and tau, DTPG gets competitive performance compared with RPC and LWD. As illustrated above, GMR is more meaningful than the other two measures in the imbalanced data set. Therefore, the imbalance-aware DTPG outperforms all other algorithms. Both LogR and LWD are label-wise and RPC is pairwise label ranking algorithm. In the imbalanced label ranking data, label-wise method suffers from skewed distribution of positions where labels rank and pairwise methods suffer from imbalanced distribution of different ordered pairs. As a result, they would miss important but rare instances in the imbalanced data which could be more informative. Previously popular measures, such as $ACC@k$ and tau, for label ranking are not designed to be able to tell that problem.

Implication. What does it mean for DTPG to achieve the huge improvement in GMR? For instance, DTPG outperforms the best performing competing algorithm RPC by 57%. According to the definition of GMR in eq. 6, this means that for each emotional reaction pattern, or ordered pair of labels, there are on average 57.0% more posts, missed by RPC. Note that for majority patterns, the $recall$ scores are already high, so that for minority ones, such improvement becomes even much bigger than the average. For example, among the $1,089$ posts with more votes for "$haha$" than for "$love$" in ROU, there are 234[5] successfully extracted by DTPG, but only 60, 41 and 15 extracted by LogR, RPC and LWD, respectively.

VII. Conclusion

In this work, we investigate the imbalanced label ranking problem, motivated by the needs to predict latent emotions in Facebook posts. To overcome the challenges caused from imbalanced data, we first propose an improved measure, named as G-mean-rank, which is robust toward imbalanced label ranking data. Both synthetic and experimental analysis show the superiority of G-mean-rank over common measures such as *Kendall's tau correlation* and $ACC@k$. Then, we also propose a novel decision tree based label ranking algorithm, named as DTPG, and empirically validate its superiority over four competing label ranking algorithms.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[5]Here, the number is the sum of the number of posts extracted by each trial in 5-fold cross validation.