Cost-Sensitive Learning to Rank

Ryan McBride, Ke Wang
Simon Fraser University, BC, Canada
rom2@sfu.ca and wangk@cs.sfu.ca

Zhouyang Ren, Wenyuan Li
Chongqing University, Chongqing, China
rzhouyang@gmail.com and wenyuan.li@ieee.org

Abstract

We formulate the *Cost-Sensitive Learning to Rank* problem of learning to prioritize limited resources to mitigate the most costly outcomes. We develop improved ranking models to solve this problem, as verified by experiments in diverse domains such as forest fire prevention, crime prevention, and preventing storm caused outages in electrical networks.

1. Introduction

Motivation

Learning to Rank originated in information retrieval (IR) and considers learning a ranking model for instances (such as documents, movies, or web pages) according to their relevance to a query. The training data is a list of pairs (x, y) for each query, with y as the instance's relevance to the query (e.g., from zero, no relevance, to five, very relevant) and x as instance features. An effective ranking model aims to rank relevant instances correctly based on x for a future query. Learning to Rank research thus primarily focuses on devising more useful IR based metrics (Wang et al. 2016), improving performance on search engine data (Qin et al. 2010), or developing solutions in new IR fields (e.g., relevant image retrieval (Wang et al. 2014)).

However, we argue that many non-IR applications in industry could benefit from similar ranking models: companies can learn to prioritize limited resources to events with a high y, the numeric cost/impact. One such problem is with our industrial partner BC Hydro: storm caused power outages result in severe social peril and millions of dollars in economic losses every year (Insurance Journal 2017)(Bukaty 2013)(CBC News 2015)(Lawrence Berkeley National Laboratory 2004). A power company could rank potential outages for the next storm so high damage outages can be prevented. We define such problems as cost-sensitive ranking, problems of maximizing a company's cost-saving via a ranking model:

Definition 1 (Cost-Sensitive Learning to Rank) *Consider a training set of m "lists", $\mathcal{T} = \{L_i\}_{i=1}^{m}$. The ith list $L_i = (X_i, Y_i)$ contains $X_i = \{x_{i,1}, \cdots, x_{i,n_i}\}$ and $Y_i = \{y_{i,1}, \cdots, y_{i,n_i}\}$ that describe the features x and the cost/impact y of each instance in the list. The goal is to build a ranking model that can be used to rank future lists with instances that have known x but unknown y to mitigate damage or attain profit. Per the standard Learning to Rank assumptions, future lists are assumed sampled from the same underlying distribution as $\mathcal{T}$ (e.g., each $L_i = (X_i, Y_i)$ is i.i.d. and $(x_{i,j}, y_{i,j})$ are also i.i.d. within each L_i).*

Figure 1: An example storm-caused power outage in Illinois from interactions between weather, cable placement, and surrounding vegetation. Ranking cities or geographical areas by their risk of large outages in the same storm is an example of our proposed Cost-Sensitive Ranking problem. Image credit to Wikipedia user Robert Lawton.

Figure 2: A training set consisting of instances of electricity supplying networks exposed to storms. Note that 1 m/s is approximately 2.2 miles per hour or 3.6 kilometres per hour.

Storm ID	Electrical Network In	Cable Length	Wind Speed	Impact (Y)
$Storm_1$	City 1	5 km	14 m/s	10000 customers
$Storm_1$	City 2	4 km	15 m/s	100 customers
$Storm_1$	City 3	3 km	16 m/s	0 customers
$Storm_2$	Rural 1	3 km	13 m/s	100 customers
$Storm_2$	Rural 2	4 km	12 m/s	1 customer
$Storm_2$	Rural 3	5 km	10 m/s	0 customers

For example, the set $\mathcal{T}$ in Figure 2 contains two lists cor-

responding to electrical networks exposed to two storms. An instance is a network hit by a storm with x of (City, Cable Length, Wind Speed) and y as the impact of the outage in terms of the number of customers losing power. Similar to IR research, only the instances in the same list are ranked or compared to each other.

We argue that this ranking problem applies to many non-IR domains. After all, risk management is the identification and prioritization of risks, where risks can come from various sources: uncertainty in financial markets, legal liabilities, credit risk, accidents, natural causes, and disasters. By ranking which instances in a list have the highest risk, cost-savings or profit can be attained. For example, every summer wildfires destroy hundreds of homes due to dry weather conditions, and a list of geographical regions can be ranked by the y of the cost/damage of fires (e.g., destroyed homes or evacuated population size); in business, a charity organization wants to rank a list of potential donors/buyers by a y of donation/purchase amounts so this profit can be accessed; in finance, an insurance or a loan agency wants to rank applicants by risk factors or credit scores. In all the above cases, a ranked order is important because there are only limited resources or positions available for a relevant set of instances. Cost-Sensitive ranking in such domains is of interest.

We also develop new ranking evaluators and algorithms due to inadequacies in current ranking literature. One solution to Definition 1 is applying existing IR based Learning to Rank solutions by mapping an instance to a document and replacing the relevance y with the numeric impact y. In Section 2, we will explain two issues of such an approach. First, they ignore each list's importance; ranking well in $Storm_1$'s list in Figure 2 is far more important than ranking well in $Storm_2$ due to a hundred times more customers losing power. However, existing models can incorrectly consider both storm lists as equally important. Secondly, many methods assume domain or problem properties that may not apply to these cost-sensitive domains; one assumption is that y is IR's graded relevance that is exponentially increasing, incorrectly implying that ranking an instance with a $y = 51$ customer impact first ($2^{51} - 1$ utility) can be roughly twice as valuable than ranking a $y = 50$ customer impact first ($2^{50} - 1$ utility). Such models distort model value and result in poor performance, as verified in experiments.

Contributions

We summarize our contributions as follows:

- Section 2: We argue that existing Learning to Rank and Machine Learning paradigms fail to capture key requirements of Cost-Sensitive Learning to Rank.
- Section 3: We develop appropriate Cost-Sensitive Ranking problems of ensuring that a large portion of the total impact/cost is captured by a ranking model.
- Section 4: We adapt existing ranking model paradigms to solve such problems.
- Section 5: We run experiments to validate the benefits of our new solutions on both proprietary and public data sets.

2. Related Work

In this section, we argue that existing Machine Learning paradigms from Regression and Learning to Rank fail to address Definition 1's goal of ranking to identify many high cost events. Table 1 lists some notation.

Table 1: Notation Summary

$\mathcal{T} = \{L_i\}_{i=1}^m$	A training set of multiple lists.
$L_i = (X_i, Y_i)$	A list of multiple instances/items with features X and impact Y.
$X_i = \{x_{i,1}, .., x_{i,n_i}\}$	$x_{i,j}$ is features of the jth item in L_i.
$Y_i = \{y_{i,1}, .., y_{i,n_i}\}$	$y_{i,j}$ is the impact/cost for $x_{i,j}$.
$\pi_i(j)$	A model π's position of the jth item in L_i (e.g., $\pi_i(j) = 1$ if ranked first).

Regression

Regression is a modeling paradigm built to minimize the deviation between the known risk y and the model's predicted risk $\bar{y}$, e.g., the RMSE (root mean square error) of $\sqrt{\frac{\sum(y-\bar{y})^2}{m}}$. In risk management, regression is widely used to estimate instances' risk; for example, the majority of AI research in outage prevention uses regression to predict the y magnitude of power loss (Guikema et al. 2014)(Kankanala, Das, and Pahwa 2014)(Nateghi, Guikema, and Quiring 2014)(Sullivan et al. 2010)(Wanik et al. 2015). Considering this success, these models could rank instances from highest predicted risk to lowest predicted risk. Furthermore, arguments for applying Learning to Rank over Regression can require IR-only assumptions (e.g., (Cao et al. 2007)'s query-based argument). However, effective regression models may still not rank well: consider the performance of two models on Figure 2's $Storm_1$ where $y_{1,1} = 10000$ customers, $y_{1,2} = 100$ customers, and $y_{1,3} = 0$. The first model, Model 1, predicts $\overline{y_{1,1}} = 4999$, $\overline{y_{1,2}} = 5000$, and $\overline{y_{1,3}} = 5001$, achieving a RMSE of 2868.0. This model misranks every instance so the smallest customer outage is ranked first. The second model, Model 2, predicts $\overline{y_{1,1}} = 100000$, $\overline{y_{1,2}} = 5000$, and $\overline{y_{1,3}} = -5000$ with an RMSE of 300090.6. Even though Model 2 has an over hundred times worse RMSE than Model 1, Model 2 perfectly ranks the instances. Another example is Isotonic regression, which imposes a constraint that the predicted order is consistent with the known order in the training data. This hard constraint ignores and may conflict with a company's cost-sensitive ranking goal.

Learning to Rank

We first discuss the most popular IR based ranking metric, the Normalized Discounted Cumulative Gain ($NDCG$), then generalize. Information retrieval's training set is a set of lists where each list L_i is associated with a query q_i, a document set D_i, and the relevances $Y_i = \{y_{i,1}, .., y_{i,n_i}\}$, where $y_{i,j}$ is the graded relevance level of the jth document in the document list for q_i. Typically $y_{i,j}$ is either binary (1, relevant, or 0, irrelevant) or zero out of five scale indicating the level of relevance. The $DCG@k$ (Discounted Cumulative Gain at k) (Liu et al. 2009) measures the goodness of a ranking model π at position k defined by

$$DCG@k(\pi, L_i) = \sum_{j:\pi_i(j)\leq k} G_i(j) \cdot \eta(\pi_i(j)). \quad (1)$$

$G_i()$ is the gain function and $\eta()$ is a discounting function, and $\pi_i(j)$ is the ranking position given by π of the jth instance in L_i. The $DCG@k$ represents the cumulative gain of returning the top k ranked documents with discounts on the positions; $G_i(j) = 2^{y_{i,j}} - 1$ and $\eta(\pi_i(j)) = 1/log_2(\pi_i(j) + 1)$ are widely used to convey that users' benefit of accessing the instance exponentially increases with higher relevance and the satisfaction of accessing information logarithmically decreases when the ranked position, $\pi_i(j)$, increases.

To derive the $NDCG@k$, first this $DCG@k$ is normalized to range between 0% and 100%:

$$NDCG@k(\pi, L_i) = \frac{1}{G_{max,i}(k)} \sum_{j:\pi_i(j)\leq k} G_i(j) \cdot \eta(\pi_i(j)). \quad (2)$$

where $G_{max,i}(k)$ is the $DCG@k$ for a perfect ranking model π. For a collection $\mathcal{T} = \{(q_i, L_i), Y_i\}_{i=1}^m$ corresponding to m queries $q_1, \cdots, q_m$, the overall $NDCG@k$ is the average $NDCG@k$ of all queries:

$$NDCG@k(\pi, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{L_i \in \mathcal{T}} NDCG@k(\pi_i, L_i) \quad (3)$$

One solution for Cost-Sensitive Learning to Rank is to map each list of instances into a query then apply IR models. For example, the ith storm can be mapped to an ith query, the jth network/instance's properties to the document properties of the jth instance in D_i, and the impact $y_{i,j}$ can be mapped to a relevance level. However, we argue that the $NDCG@k$ and other metrics can misjudge model utility due to significant differences between the information retrieval domain and common cost-sensitive ranking domains via two issues:

Issue 1: Cost-Insensitive List Importance. Let us consider the two lists (as in, two storms) in Figure 2 and how the $NDCG@k$ incorrectly judges performance in each list:

Example 1 *Suppose that a ranking model, called the π_{Cable} model, ranks the outages by the Cable Length feature, the $NDCG@2$ of this model is computed by*

$$NDCG@2(\pi_{Cable}, Storm_1) = \frac{(2^{10,000} - 1) + \frac{(2^{100}-1)}{log_2 3}}{(2^{10,000} - 1) + \frac{(2^{100}-1)}{log_2 3}} = 100\%$$

$$NDCG@2(\pi_{Cable}, Storm_2) = \frac{(2^{0} - 1) + \frac{(2^{1}-1)}{log_2 3}}{(2^{100} - 1) + \frac{(2^{1}-1)}{log_2 3}} \approx 0\%$$

$$NDCG@2(\pi_{Cable}, \mathcal{T}) = (100\% + 0\%)/2 = 50\%$$

Suppose that another ranking model, called π_{Wind}, ranks the outages by wind speed. It is easy to see $NDCG@2(\pi_{Wind}, \mathcal{T}) = 50\%$. This model has the same $NDCG@2$ score as the first model. This is not intuitively correct: the first model ranks $Storm_1$ correctly but ranks $Storm_2$ incorrectly, whereas the second model ranks $Storm_2$ correctly but ranks $Storm_1$ incorrectly; however, ranking $Storm_1$ correctly is far more important than ranking $Storm_2$ correctly because of larger impacts in $Storm_1$. The $NDCG@k$ fails to capture this difference by taking the average of the normalized score of each list.

Issue 2: Single List Cost-Insensitivity Furthermore, the $NDCG@k$ fails to capture performance even for a single list. Recall that $y_{i,j}$ in IR represents the graded relevance level, and $G_i(j) = 2^{y_{i,j}} - 1$ models the importance of of such relevances. For real valued impacts $y_{i,j}$ related to industrial quantities, such as the number of customers losing power, $2^{y_{i,j}} - 1$ does not capture the importance of impact:

Example 2 *Consider a list L_i with $Y_i = \{70, 0, 0, 50, 50, 50\}$ where y represents the number of customers losing power and there are two models with $k = 3$, $\pi_1 =<70, 0, 0>$ (e.g., the instance with $y = 70$ customers is ranked first) and $\pi_2 =<50, 50, 50>$. Intuitively, π_2 predicts more outage impacts than π_1. However, $DCG@3(\pi_1) = (2^{70} - 1)$ and $DCG@3(\pi_2) = (2^{50} - 1)(1 + \frac{1}{log_2(2)} + \frac{1}{log_2(3)}) = (2^{50} - 1) \cdot 2.63$, suggesting that π_1 is roughly four hundred thousand times better than π_2. The same holds for $NDCG@3$, which is the $DCG@3$ divided by the constant $G_{max,i}(3)$. Therefore, the DCG and $NDCG$ fail to capture a natural notion of cost such as the number of customers losing power. In addition, the $NDCG@k$ is unable to adapt to which ranking is most useful to a company based on y's domain context and cost.*

Other ranking approaches also suffer from Issue 1 or 2. Many evaluators are 'normalized" akin to the $NDCG@k$ so each list contributes equally to performance, implying Issue 1. This includes the *Mean Average Precision* (Liu et al. 2009), the average *Kendall Tau Correlation Coefficient* (Liu et al. 2009), or the many adaptions of the $NDCG@k$ (e.g., (Bian et al. 2010)'s approach of different queries relying on different $NDCG@k$ settings). For Issue 2, current ranking metrics similarly ignore essential properties of cost-sensitive ranking. For example, *Positional Metrics* only depend on the ranking position $\pi_i(j)$ (e.g., the κ-$NDCG@k$ (Niu et al. 2012)), *Pairwise Metrics* (Liu et al. 2009) only consider if pairs are ranked correctly and not the gap between the $y_{i,j}$ values within pairs (linked to *Classification for Imbalanced Data* (Maimon and Rokach 2005)(Liu et al. 2009) for binary $y_{i,j}$), and *Multiple Instance Ranking* (Bergeron et al. 2008) is a problem of predicting which instance has the highest $y_{i,j}$ value in a list (e.g., the store that sells the most products in a region); all these methods ignore the exact $y_{i,j}$ impacts of an outage/event and how it relates to cost. For this reason, Multiple Instance Ranking and the κ-$NDCG@k$ both incorrectly prefer Example 2's worst model, π_1, and equally prefer π_{Cable} and π_{Wind} in Example 1, for example.

Similar arguments apply to other ranking methods that are not built with a notion of cost-saving interpretable to a company. For example, Wang et al.'s work measures how consistently a ranking model outperforms IR ranking baselines (Wang et al. 2016), which does not apply for non-IR domains. (Xu et al. 2006) associates a misranking cost for every pair of IR's graded relevances, which is IR specific and appears disconnected from risk management domains where

costs savings arise from real-world resource deployment to a small portion of ranked items.

3. Cost-Sensitive Ranking Definitions

We address these issues via two new cost-sensitive ranking evaluators and problem definitions: the Cost-Sensitive Learning to Rank problem, that solves Issue 1 and Issue 2, and the Cost-Reweighted Learning to Rank problem, that solves only Issue 1.

To fix Issue 2, we set a cost-sensitive ranking evaluator $\mathcal{R}$ for a single list assuming domain-specific properties:

- $Cost(x_{i,j}, y_{i,j})$: this is the cost saved or gained from allocating resources to the jth instance in the list L_i given its features x and y impact. The specification of $Cost$ is domain specific; in the outage problem, BC Hydro sets $Cost$ as proportional to the number of customers that lose power (e.g., $Cost() = y_{i,j}$).
- k: we assume a user can only act on some top-k instances in a ranked list due to resource limitations.
- $Pr(\pi_i(j))$: this is the Bernoulli probability of a company acting on the jth instance in the list L_i according to the rank given by the model π, i.e., $\pi_i(j)$, as derived from a user's resource allocation policies. For the outage ranking problem, BC Hydro preferred the linear decreasing probability $Pr(\pi_i(j)) = max(1 - \frac{\pi_i(j)-1}{k}, 0)$ over other options. This function achieves a probability of 100% for the first ranked instance, when $\pi_i(j) = 1$, then linearly decreases until a ranking position of $k + 1$, with a 0% probability.

This defines the Cost-Sensitive ranking evaluator $\mathcal{R}@k$ for a given ranking model π and a list L_i:

$$\mathcal{R}@k(\pi, L_i) = \frac{\sum_{j:\pi_i(j)\leq k} Cost(x_{i,j}, y_{i,j}) \cdot Pr(\pi_i(j))}{Ideal\mathcal{R}@k(L_i)} \tag{4}$$

where $Ideal\mathcal{R}@k(L_i)$ is the highest possible numerator for the perfect ranking model.

Intuitively, $\mathcal{R}@k(\pi, L_i)$ is the percentage of expected gain or cost saved by acting on the top-k ranked list produced by π. Note that Eq. (2)'s $NDCG@k$ is in fact $\mathcal{R}$ with cost set to the gain function of $G_i(j) = 2^{y_{i,j}} - 1$ and the probability set to the discounting function $\eta(\pi_i(j)) = 1/log_2(\pi_i(j) + 1)$. Nonetheless, $Cost$ and Pr are interpretable in non-information retrieval domains and so is $\mathcal{R}$, as a cost-saving percentage. This fixes Issue 2:

Example 3 *With the outage domain's $\mathcal{R}@k$ settings above ($Cost(x_{i,j}, y_{i,j}) = y_{i,j}$ and $Pr(\pi_i(j)) = max(0, 1 - \frac{\pi_i(j)-1}{k})$), consider the two ranking models $\pi_1 =< 70, 0, 0 >$ and $\pi_2 =< 50, 50, 50 >$ of the list with $Y_i = \{70, 50, 50, 50, 0, 0, 0\}$ again. $Ideal\mathcal{R}@3$ is the constant $70 + 0.66 \cdot 50 + 0.33 \cdot 50 = 120$. The $\mathcal{R}@3$ of π_1 is $\frac{70}{120}$ and the $\mathcal{R}@3$ of π_2 is $\frac{50(1+0.66+0.33)}{120} = \frac{100}{120}$. Thus $\mathcal{R}@3$ correctly favors π_2 over π_1.*

Similar examples apply to other problems: the appropriate cost-sensitive $\mathcal{R}$ prefers the more cost effective model.

With a suitable $\mathcal{R}$, we define the cost-sensitive evaluator $\mathcal{R}_{CS}$ for a set of lists $\mathcal{T}$. To address Issue 1, we weight $\mathcal{R}(\pi, L_i)$ by the cost-saving of the perfect model for L_i:

$$\mathcal{R}_{CS}@k(\pi, \mathcal{T}) = \frac{\sum_{L_i\in\mathcal{T}} Ideal\mathcal{R}@k(L_i) \cdot \mathcal{R}@k(\pi, L_i)}{\sum_{L_i\in\mathcal{T}} Ideal\mathcal{R}@k(L_i)} \tag{5}$$

The numerator is the model π's expected cost saved summed over every list while the denominator is the best achievable cost saved; in contrast to the multiple list $NDCG@k$ and other approaches, a ranking model's performance in each list is correctly proportional to the total cost that could be saved by ranking within that list. This fixes Issue 1:

Example 4 *In Example 1, the $NDCG@2$ suffers from equally favoring π_{Cable} and π_{Wind} model though the former has a larger impact on cost saving than the latter. However, the $\mathcal{R}_{CS}@2$ correctly favors π_{Cable} that has a larger impact/cost because the contribution of each list's $Ideal\mathcal{R}@k(L_i)$, 10050 for $Storm_1$ and 100.5 for $Storm_2$, reflects each list's potential cost-saving:*

$$\mathcal{R}@2(\pi_{Cable}, Storm_1) = \frac{10000 \cdot 1 + 100 \cdot 0.5}{10050} = 100\%$$
$$\mathcal{R}@2(\pi_{Cable}, Storm_2) = \frac{0 + 1 \cdot 0.5}{100.5} = 0.5\%$$
$$\mathcal{R}_{CS}@2(\pi_{Cable}, \mathcal{T}) = \frac{10050 \cdot 1 + 100.5 \cdot 0.014}{10050 + 100.5} = 98.5\%$$

$$\mathcal{R}@2(\pi_{Wind}, Storm_1) = \frac{0 + 100 \cdot 0.5}{10050} = 0.5\%$$
$$\mathcal{R}@2(\pi_{Wind}, Storm_2) = \frac{100 \cdot 1 + 1 \cdot 0.5}{100.5} = 100\%$$
$$\mathcal{R}_{CS}@2(\pi_{Wind}, \mathcal{T}) = \frac{10050 \cdot 0.005 + 100.5 \cdot 1}{10050 + 100.5} = 1.5\%$$

We then use $\mathcal{R}_{CS}$ to fully define the Cost-Sensitive Learning to Rank problem:

Definition 2 (Cost-Sensitive Learning to Rank (Final)) *Given a training set $\mathcal{T}_{Training} = \{L_i\}_{i=1}^m$, where $L_i = (X_i, Y_i)$, X_i as the features of instances, and $Y_i = \{y_{i,1}, \cdots, y_{i,n_i}\}$ as numeric-valued impacts where $y_{i,j} \geq 0$, find a ranking model π that maximizes $\mathcal{R}_{CS}@k(\pi, \mathcal{T})$ on future $\mathcal{T}$ drawn from the same underlying distribution as $\mathcal{T}_{Training}$.*

In general, the future data $\mathcal{T}$ is not available, so a common practice is to reserve some lists of $\mathcal{T}_{Training}$, denoted by $\mathcal{T}_{Testing}$, to simulate the future data, and find a ranking model with high cost savings for $\mathcal{T}_{Testing}$.

We also develop a weaker cost-sensitive ranking problem, Cost-Reweighted Learning to Rank, that only fixes Issue 1 but not Issue 2. This formulation will be leveraged by our Cost-Reweighted algorithm in Section 4 to adapt existing ranking models to be more cost-sensitive. The Cost-Reweighted objective $\mathcal{R}_{CR}@k$ is similar to R_{CS} except the model π is evaluated with a different ranking evaluator $\mathcal{R}'@k(\pi, L_i)$:

$$\mathcal{R}_{CR}@k(\pi, \mathcal{T}) = \sum_{L_i \in \mathcal{T}} \frac{Ideal\mathcal{R}@k(L_i) \cdot \mathcal{R}'@k(\pi, L_i)}{\sum_{L_i \in \mathcal{T}} Ideal\mathcal{R}@k(L_i)} \quad (6)$$

For example, $\mathcal{R}'$ could be the single-list $NDCG@k$ and $\mathcal{R}@k(\pi, L_i)$ could be the outage domain appropriate $\mathcal{R}@k$ motivated earlier. Under this setting, the "weight" $\frac{Ideal\mathcal{R}@k(L_i)}{\sum_{L_i \in \mathcal{T}} Ideal\mathcal{R}@k(L_i)}$ is the cost proportion of L_i, the percentage of total cost saving achievable in L_i (e.g., for $k = 2$, $Storm_1$'s weight is 99.5% while $Storm_2$'s is 0.5%). Even if $\mathcal{R}'@2$ does not solve Issue 2 in Example 1, the reweighted evaluator $\mathcal{R}_{CR}$ would nonetheless better judge performance; for example, the $\mathcal{R}_{CR}@2$ of π_{Cable} is 99.0% and π_{Wind}'s is 0.1%, reflecting a power company's preferences.

The Cost-Reweighted Learning to Rank problem with the $\mathcal{R}_{CR}$ objective is:

Definition 3 (Cost-Reweighted Learning to Rank) *Given the training set $\mathcal{T}$ in Definition 2, we find a ranking model π that maximizes $\mathcal{R}_{CR}@k(\pi, \mathcal{T})$ on future $\mathcal{T}$ drawn from the same underlying distribution as $\mathcal{T}_{Training}$.*

4. Algorithms

We next propose solutions to the Cost-Sensitive (CS) and the Cost-Reweighted (CR) Learning to Rank problems.

Cost-Sensitive Model Search

We can exactly optimize the Cost-Sensitive Ranking problem by adapting LambdaMART (Burges 2010), Coordinate Ascent (Friedman 2001), and AdaRank (Xu and Li 2007) into Cost-Sensitive MART (CS-MART), Cost Sensitive Coordinate Ascent (CS-C. Ascent), and Cost-Sensitive AdaRank (CS-AdaRank). This is done by replacing the model's original objective with our cost-sensitive goal $\mathcal{R}_{CS}$.

Cost-Sensitive MART. The original LambdaMART is a gradient ascent method that guides model search by the following difference of the $NDCG@k$ between the original model π (with parameters θ) and a new model $\pi(\theta_{i,j})$:

$$\begin{aligned} Z_{i,j} =& NDCG@k(\pi(\theta_{i,j}), \mathcal{T}_{Training}) \\ & - NDCG@k(\pi(\theta), \mathcal{T}_{Training}) \end{aligned}$$

Intuitively, $\pi(\theta_{i,j})$ specifies the hypothetical model obtained from the model specified by θ by swapping the positions of the ith ranked document with the jth ranked document. We adapt LambdaMART by replacing $NDCG@k$ with our cost-sensitive $\mathcal{R}_{CS}$ in $Z_{i,j}$.

Cost-Sensitive Coordinate Ascent. Coordinate Ascent starts with a random model parameter vector, θ, and suggests a new model parameter vector, θ_{new}, and computes the $NDCG@k$ improvement

$$\begin{aligned} \delta(\theta_{new}, \theta, \mathcal{T}_{Training}) =& NDCG@k(\pi(\theta_{new}), \mathcal{T}_{Training}) \\ & - NDCG@k(\pi(\theta), \mathcal{T}_{Training}) \end{aligned}$$

If the improvement is positive and above a given threshold, it sets θ to θ_{new}, otherwise, θ remains the same. This process is then repeated until θ converges. Exact details can be found in (Friedman 2001). We adapt this algorithm by substituting the $NDCG@k$ with the $\mathcal{R}_{CS}$.

Cost-Sensitive AdaRank. AdaRank (Xu and Li 2007) uses boosting to optimize the sum of a per-list evaluator $E(\pi, L_i)$ over multiple lists:

$$AdaRank(\pi, \mathcal{T}_{Training}) = \sum_{L_i \in \mathcal{T}_{Training}} E(\pi, L_i)$$

where E must range between -1 and 1, otherwise, convergence is not guaranteed. Generally, E is set to $NDCG@k(\pi, L_i)$ to maximize the average $NDCG@k$ over multiple lists in Eq. (3); AdaRank thus inherits the $NDCG$'s limitations in Section 2.

For our new solution, CS-AdaRank, we replace the objective E with the per-list objective $\frac{\mathcal{R}@k(\pi, L_i)}{\sum_{L_i \in \mathcal{T}_{Training}} Ideal\mathcal{R}@k(L_i)}$. The new objective is in fact proportional to our $\mathcal{R}_{CS}$ objective.

Cost-Reweighted Model Search

General ranking models cannot be easily adapted to optimize an arbitrary cost-sensitive objective: for example, methods such as ListNet (Cao et al. 2007) are hard-coded to only support the $NDCG@k$'s exponential graded relevance y values and that each list contributes equally to performance. Similarly, ranking methods that adjust a set of models to a new ranking objective are only built for existing metrics and their assumptions (e.g., (Wu et al. 2010)'s Mean Average Precision optimization). Adapting these methods would thus require significant redevelopment.

Instead, we optimize Definition 3's Cost-Reweighted Ranking problem, a weaker cost-sensitive problem that addresses only Issue 1 by preferring models that perform well on lists that contribute the most to cost-saving. Our **Cost-Reweighted** model search optimizes this problem by adapting a class of gradient ascent methods with a per-list loss function $Loss(\pi, L_i)$; this means we can adapt many modern ranking methods, such as ListNet (Cao et al. 2007), LambdaMART (Burges 2010), Coordinate Ascent (Friedman 2001), RankNet (Burges 2010), LambdaRank (Burges 2010), and the many modifications and derivatives of the listed approaches. To explain this adaption, first recall that gradient ascent ranking methods assume:

- The model π's goal is to optimize the average $\mathcal{R}'@k(\pi, L_i)$ over every list in $\mathcal{T}_{Training}$. For example, if $\mathcal{R}'@k = NDCG@k$ (Eq. (2)) then π optimizes the overall $NDCG@k$ (Eq. (3)).
- Since $\mathcal{R}'@k(\pi, L_i)$ does not have a "smooth" gradient, the method uses the substitute function $Loss(\pi, L_i)$ where optimizing $Loss(\pi, L_i)$ optimizes $\mathcal{R}'@k(\pi, L_i)$.
- Over many iterations, π's model parameters are updated via this loss function and the corresponding gradient.

To optimize the Cost-Reweighted $\mathcal{R}_{CR}@k(\pi, \mathcal{T})$, we reweight the $Loss$ for each list by its cost importance or "weight", $\frac{Ideal\mathcal{R}@k(L_i)}{\sum_{L_i \in \mathcal{T}} Ideal\mathcal{R}@k(L_i)}$:

$$Loss_{CR}(L_i) = \frac{Ideal\mathcal{R}@k(L_i)}{\sum_{L_i \in \mathcal{T}} Ideal\mathcal{R}@k(L_i)} \cdot Loss(\pi, L_i)$$

.

The new gradient for this $Loss_{CR}$ is the original $Loss$'s gradient multiplied by the same cost-reweighted constant. By gradient ascent properties with multiplication by a constant, this method trivially optimizes the average $\frac{Ideal\mathcal{R}@k(L_i)}{\sum_{L_i \in \mathcal{T}} Ideal\mathcal{R}@k(L_i)} \cdot \mathcal{R}'@k(\pi, L_i)$ over all L_i, which is proportional to Eq. (6)'s $\mathcal{R}_{CR}$. We will note the simplicity of this adaption: it is easy to find the gradient update code then multiply it by the per-list constant.

In experiments, we apply this Cost-Reweighted adaption to ListNet and RankNet. Both these models use an $\mathcal{R}'@k$ of the $NDCG@k$. In more detail:

Cost-Reweighted ListNet. The original ListNet optimizes the $NDCG@k$ with a loss function based on the cross entropy between the permutation probability of the given ranking model versus the permutation probability of a perfect model. For example, the permutation probability of the jth instance in L_i being the top-ranked instance is $\frac{e^{y_{i,j}}}{\sum_{j'=1}^{n_i} e^{y_{i,j'}}}$ (Cao et al. 2007). By considering this probability for all instances, the authors define a distribution used for the cross-entropy. Cost-Reweighted ListNet replaces this loss function with the reweighted version $Loss_{CR}(L_i)$.

Cost-Reweighted RankNet. RankNet's loss is based on cross-entropy of the probability that each pair of instances in a list are ranked correctly to optimize the $NDCG@k$ (Burges 2010). This cross-entropy loss function is replaced with the cost-reweighted version, $Loss_{CR}(L_i)$.

5. Experiments

Methodology

In experiments, we validate two claims:

- *Claim 1:* Section 4's Cost-Sensitive adapted algorithms outperform their baselines from Learning to Rank literature. In particular, we consider the following pairs of algorithms presented in Section 4: CS-MART and LambdaMART, CS-C. Ascent and Coordinate Ascent, CS-AdaRank and AdaRank, CR-RankNet and RankNet, and CR-ListNet and ListNet.
- *Claim 2:* The best Cost-Sensitive algorithm outperforms the best regression based ranking method. We consider three regression algorithms, Random Forests, MART, and Linear Regression. We will note that Random Forests was the best competitor in some outage-related problems (e.g., (Nateghi, Guikema, and Quiring 2014)).

RankLib (https://sourceforge.net/p/lemur/wiki/RankLib/) is used for all baselines. All algorithms are evaluated by $\mathcal{R}_{CS}@k$ in Eq. (5) with an appropriate cost-sensitive ranking evaluator $\mathcal{R}@k$. Testing uses five-fold cross-validation via LETOR's separation of data with three folds used for training, one for validation, and one for testing (Qin et al. 2010). Each fold has a roughly equal number of lists.

Data Sets and Cost-Setting

We consider two proprietary outage data sets and three public UCI data sets. Each data set is partitioned into lists based on separations of geography (e.g., crimes in the same state) and/or time (forest fire damage in the same month, outage damage in the same storm, or a concrete batch's strength of batches produced on the same day) with a goal of ranking to identify instances with high y values in each list. Attributes and details on each data set are provided in Table 2.

The two outage problems cover storms from 2010 to 2015 in BC Hydro's operating region in British Columbia: *High-Risk Outages*'s data is from areas with a high risk of outages and *Customers Outages*' is from a set of networks that supply many customers in urban areas. There are 85 attributes in these data sets: 59 properties of the network/region (e.g., total cable length in the network, the number of poles, and the percentage of cables underground), 25 weather properties (e.g., average wind speed, maximum wind speed, rainfall, humidity, and the density of nearby vegetation determined from satellite readings) from NOAA's North American Regional Reanalysis readings and NASA's global remote sensing records, and the y impact is the number of customers that lose power from outages in that area.

Table 2: Data Set details (e.g., number of attributes). RankLib does not support missing values or categorical attributes so we removed any attribute with missing values and convert each categorical attribute with C categories into C binary attributes. This modifies *Crime*'s 123 attributes to 103 attributes.

Data Set	Instances	#Att.	# Lists	Impact (Y)
High-Risk Outages	95,849	85	333 Storms	# Customers Losing Power
Customers Outages	122,030	85	302 Storms	# Customers Losing Power
Crime	1994	103	46 States	Crimes in District
Forest Fire	517	12	12 Months	Forest Fire Area
Concrete	1030	8	14 Ages	Concrete Strength

The cost-sensitive evaluator $\mathcal{R}_{CS}@k$ is set to Section 3's outage domain setting, $Cost()$ as a linear cost and $Pr()$ as a linearly decreasing probability. We consider this setting reasonable for the non-outage domains as well; it is domain-appropriate for a company/organization to rank such that the largest portion of crime, forest fire burnt acreage, or strongest concrete are identified. For the two outage data sets, we use ks of 10 and 50, based on domain knowledge on how many networks may be strengthened before a storm given a 24 hour lead time. For the other data sets, we use two ks: a low k (12.5% of the average number of instances in a list) and a more medium k (25% of the average list length).

Detailed Results

Table 3's results validate our claims. Note that CR-ListNet and ListNet have incomplete entries due to numeric calculation issues related to these methods' cross-entropy calculation; as discussed in Section 4, this cross-entropy involves

Table 3: Our new Cost-Sensitive and Cost-Reweighted methods, noted with a diamond, achieve better cost-savings than competitors: the bolded top-3 algorithms in each problem setting (column) are the new methods in all but one data set. The best CS method has statistically significant improvements over competitors (p-value < 0.05) except for in Forest Fires and Concrete.

	$\mathcal{R}_{CS}@k$ for Different Data Sets.									
	High-Risk Outages		Customers Outages		Crime		Forest Fires		Concrete	
Algorithm	Low k (k=10)	Mid k (k=50)	Low k (k=10)	Mid k (k=50)	Low k (k=6)	Mid k (k=11)	Low k (k=6)	Mid k (k=11)	Low k (k=10)	Mid k (k=19)
◊ CS-MART	**11.3%**	**27.2%**	**19.2%**	**25.3%**	**96.9%**	**98.2%**	19.1%	18.3%	**91.0%**	**94.3%**
LambdaMART	8.5%	22.4%	6.0%	12.4%	83.9%	87.1%	9.8%	25.2%	**90.5%**	**93.6%**
◊ CS-C. Ascent	**10.5%**	**26.6%**	**17.0%**	**26.2%**	**97.6%**	**98.2%**	**24.0%**	28.4%	**91.0%**	92.7%
C. Ascent	7.2%	21.6%	10.0%	17.2%	46.8%	44.1%	20.2%	26.8%	88.7%	91.1%
◊ CS-AdaRank	**9.7%**	**25.1%**	**13.8%**	**25.9%**	**97.9%**	**97.2%**	**29.4%**	**38.4%**	84.0%	87.6%
AdaRank	7.3%	23.0%	1.7%	4.7%	17.4%	37.4%	14.9%	22.8%	87.9%	86.3%
◊ CR-RankNet	7.0%	23.2%	10.5%	20.4%	75.9%	79.4%	20.4%	**33.7%**	89.4%	87.6%
RankNet	3.1%	11.8%	5.9%	16.2%	75.1%	77.7%	19.5%	30.8%	84.0%	87.0%
◊ CR-ListNet	-	-	-	-	-	-	**25.5%**	**34.5%**	77.6%	80.0%
ListNet	-	-	-	-	-	-	9.7%	21.6%	75.6%	81.6%
Linear Regression	0.8%	8.1%	5.6%	13.0%	85.7%	87.2%	21.2%	29.6%	90.4%	**93.2%**
Random Forests	8.1%	24.7%	11.9%	20.6%	92.5%	94.3%	21.0%	27.9%	90.0%	92.2%
MART	9.0%	23.2%	11.3%	19.4%	83.6%	85.9%	14.3%	23.6%	83.6%	85.8%

terms such as $\frac{e^{y_{i,j}}}{\sum_{j'} e^{y_{i,j'}}}$ (Cao et al. 2007) where $y_{i,j}$ can be greater than 10,000, resulting in numeric overflow issues.

For *Claim 1*, our Cost-Sensitive Learning to Rank methods overwhelmingly outperform their corresponding Learning to Rank baseline: in 81 of 84 tests the new method outperforms its paired baseline. For example in the Customers Outages data set, CS-MART's $\mathcal{R}_{CS}@k$ objective prioritized ranking well in major storms with high impact outages (e.g, thousands/tens of thousands of customers lose power), especially related to high surrounding vegetation in urban areas. In contrast, LambdaMART's $NDCG@k$ focuses on more frequent minor storms where only tens or hundreds of customers lose power since a good performance in many storms is more important for the $NDCG@k$. Admittedly, in Concrete most algorithm improvements over their baseline are marginal. We find that this problem is relatively easy because concrete strength is most heavily correlated with only a few attributes so most algorithms are able to extract similar models with similar performances.

For *Claim 2*, the bolded results, the top-3 performers, are overwhelmingly Cost-Sensitive or Cost-Reweighted methods. The best Cost-Sensitive method is better than the best baseline, which is statistically significant in Customers Outages, High-Risk Outages, and Crime via a p-value < 0.05. The difference in performance arises from how models leverage attributes, which we explored in Outages Customers where CS-MART achieves 19.2% while Random Forest achieves 11.9%. CS-MART tended towards model splits based on humidity and vegetation, that were heavily correlated with which networks will have high cost failures in the same storm. In contrast, Random Forests' models primarily split on wind speed and rainfall because these attributes results in a higher risk of outages and are linked to a low root mean square error due to this correlation; however, these attributes are not as useful for ranking since most network areas are exposed to the same storm and therefore very similar wind/rainfall conditions. Regression's ignorance of the ranking goal can thus lead to poor performance.

Table 4: Both our contributions are beneficial since a method that only fixes Issue 1 via the Cost-Reweighted adaption is outperformed by the Cost-Sensitive algorithms that fix both issues. Note that AdaRank cannot be Cost-Reweighted because it is not a gradient ascent method.

	$\mathcal{R}_{CS}@k$ for Outage Data Sets			
	High-Risk Outages		Customers Outages	
Algorithm	k=10	k=50	k=10	k=50
CS-MART	**11.3%**	**27.2%**	**19.2%**	**25.3%**
CR-MART	8.7%	25.1%	12.8%	21.9%
CS-C. Ascent	**10.5%**	**26.6%**	**17.0%**	**26.2%**
CR-C.Ascent	8.1%	23.7%	11.2%	19.5%

We also test whether our two fixes in Section 2 and 3 both contribute to performance: in Table 4, our Cost-Sensitive algorithms outperform the Cost-Reweighted version that addresses only Issue 1 (the cost or importance of each list) but does not address Issue 2 (individual list cost-sensitivity) as intended. Similar results apply to the other data sets.

We will stress that the effectiveness of our Cost-Sensitive approach is measured by comparing the new CS or CR algorithm with its original counterpart in each pair; the large variance across different pairs is caused by the relevant benefits of each search strategy.

6. Acknowledgments

Our thanks to Canada's NSERC for PGS-D and CRD funding, BC Hydro's Tom Gutwin and Darcy Dommer, and the National "111" Project of China (Project No. B08036).

7. Conclusion

Experiments validate our Cost-Sensitive Learning to Rank paradigm. Our suite of Cost-Sensitive and Cost-Reweighted solutions outperform existing ranking methods which ignore a company's use case: ranking to guide the prevention of costly damage. This solution is thus useful in diverse risk management domains, such as power outage prevention.

References

Bergeron, C.; Zaretzki, J.; Breneman, C.; and Bennett, K. P. 2008. Multiple instance ranking. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, 48–55. New York, NY, USA: ACM.

Bian, J.; Liu, T.-Y.; Qin, T.; and Zha, H. 2010. Ranking with query-dependent loss for web search. 141–150.

Bukaty, R. F. 2013. Thousands in Maine remain without power, nearly a week after massive ice storm - The Boston Globe.

Burges, C. J. 2010. From ranknet to lambdarank to lambdamart: An overview.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F.; and Li, H. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, 129–136. ACM.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* 1189–1232.

Guikema, S. D.; Nateghi, R.; Quiring, S. M.; Staid, A.; Reilly, A. C.; and Gao, M.-L. 2014. Predicting hurricane power outages to support storm response planning. *Access, IEEE* 2:1364–1373.

2017. Insured losses for Europe's storm Zeus estimated at US$200m: Perils.

Kankanala, P.; Das, S.; and Pahwa, A. 2014. Adaboost: An ensemble learning approach for estimating weather-related outages in distribution systems. *Power Systems, IEEE Transactions on* 29(1):359–367.

Lawrence Berkeley National Laboratory 2004. Understanding the cost of power interruptions to U.S. electricity consumers.

Liu, T.-Y., et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3(3).

Maimon, O., and Rokach, L. 2005. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*.

Nateghi, R.; Guikema, S. D.; and Quiring, S. M. 2014. Forecasting hurricane-induced power outage durations. *Natural Hazards* 74(3):1795–1811.

CBC News 2015. *B.C. storm: 22,000 customers remain without power - British Columbia - CBC News*.

Niu, S.; Guo, J.; Lan, Y.; and Cheng, X. 2012. Top-k learning to rank: labeling, ranking and evaluation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 751–760. ACM.

Qin, T.; Liu, T.-Y.; Xu, J.; and Li, H. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13(4):346–374.

Sullivan, M. J.; Mercurio, M. G.; Schellenberg, J. A.; and Eto, J. H. 2010. How to estimate the value of service reliability improvements. 1–5.

Wang, J.; Song, Y.; Leung, T.; Rosenberg, C.; Wang, J.; Philbin, J.; Chen, B.; and Wu, Y. 2014. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1386–1393.

Wang, X.; Bendersky, M.; Metzler, D.; and Najork, M. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 115–124. ACM.

Wanik, D.; Anagnostou, E.; Hartman, B.; Frediani, M.; and Astitha, M. 2015. Storm outage modeling for an electric distribution network in northeastern usa. *Natural Hazards* 79(2):1359–1384.

Wu, Q.; Burges, C. J.; Svore, K. M.; and Gao, J. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* 13(3):254–270.

Xu, J.; Cao, Y.; Li, H.; and Huang, Y. 2006. Cost-sensitive learning of svm for ranking. In *European conference on machine learning*, 833–840. Springer.

Xu, J., and Li, H. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 391–398. ACM.