Ranking with Slot Constraints

Wentao Guo*
Cornell University
Department of Computer Science
Ithaca, New York, USA
wg247@cornell.edu

Andrew Wang*
Cornell University
Department of Computer Science
Ithaca, New York, USA
azw7@cornell.edu

Bradon Thymes
Cornell University
Department of Computer Science
Ithaca, New York, USA
bmt63@cornell.edu

Thorsten Joachims
Cornell University
Department of Computer Science
Ithaca, New York, USA
tj@cs.cornell.edu

ABSTRACT

Rankings are increasingly used as part of human decision-making processes to most effectively allocate reviewing resources. Many of these processes have complex constraints, and we identify *slot constraints* as a model for a wide range of application problems – from college admission with limited slots for different majors, to composing a stratified cohort of eligible participants in a medical trial. In this paper, we formalize the slot-constrained ranking problem as producing a ranking that maximizes the number of filled slots if candidates are evaluated by a human decision maker for slot eligibility in the order of the ranking. We show that naive adaptations of the Probability Ranking Principle (PRP) can be highly sub-optimal for slot-constrained ranking problems, and we devise a new ranking algorithm, called MatchRank. MatchRank generalizes the PRP, and it subsumes the PRP as a special case when there are no slot constraints. Our theoretical analysis shows that MatchRank has a strong approximation guarantee without any independence assumptions between slots or candidates. Furthermore, we show how MatchRank can be implemented efficiently. Beyond the theoretical guarantees, empirical evaluations show that MatchRank can provide substantial improvements over a range of synthetic and real-world tasks.

CCS CONCEPTS

• **Information systems → Top-k retrieval in databases**.

KEYWORDS

ranking; slot constraints; maximum bipartite matching

ACM Reference Format:
Wentao Guo*, Andrew Wang*, Bradon Thymes, and Thorsten Joachims. 2024. Ranking with Slot Constraints. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3672000

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08…$15.00
https://doi.org/10.1145/3637528.3672000

1 INTRODUCTION

Rankings have become a ubiquitous interface whenever there is a need to focus attention among an otherwise impractically or intractably large number of options. Beyond their conception as an interface for query-based retrieval [e.g., 31], rankings are now widely used in related tasks like recommendation and advertising. In addition, a substantial number of new ranking applications come with new types of constraints, and we identify the notion of *slot constraints* as a frequent requirement. With slot constraints we refer to capacity constraints for different types of relevant candidates. For example, there is only a certain number of slots for each major in a college admissions task; or the cohort of a medical trial may need to fulfill constraints on gender and race to be representative [7]; and in a multi-stage retrieval pipeline we may have assortment constraints [41]. In these applications, the goal is to fill all slots with relevant candidates, and each candidate can have a different probability of relevance for each slot.

For rankings without constraints, the Probability Ranking Principle (PRP) [30] has long been understood to provide the ranking that maximizes the number of relevant candidates that are found in the top-k of the ranking, for any k. However, the PRP does not apply to ranking problems with *slot constraints*, and naive extensions of the PRP can be highly sub-optimal by disparately spending human evaluation effort on candidates for which there are no open slots while ignoring other candidates that are relevant for unfilled slots. Furthermore, ranking with slot constraints is different from both intrinsically and extrinsically diversified ranking [28], since it involves a vector of relevance probabilities for each candidate and it allows us to put explicit constraints on the set of *relevant* results. Conventional diversification methods cannot handle relevance vectors, and they typically focus on the composition of the ranking instead of on the composition of the relevant results (e.g., demographic parity [17]). Furthermore, they typically do not allow the specification of explicit constraints (e.g., [9, 10, 43]).

To remedy this shortcoming, we formalize and address the problem of *ranking with slot constraints* in this paper. In our formulation, a human decision maker can define an arbitrary set of slots (e.g.,

*Both authors contributed equally to this research.

admission slots for each major) that need to be filled with relevant candidates (i.e., qualified students). The ranker then supports the human decision maker in allocating the evaluation effort while leaving the relevance decisions to the human. Specifically, given a relevance model that estimates the relevance probability of each candidate for each slot, the goal is to compute a ranking under which the expected number of filled slots is maximized. This model is very general, as candidates can be qualified for any number of slots, and we can use any probabilistic model with arbitrary dependencies between the slot relevances of all candidates.

Under this model, we show that there is a connection between slot-constrained ranking and bipartite matching, and we derive a general ranking algorithm, called MatchRank, that merely requires efficient sampling from the relevance model. We theoretically analyze MatchRank and show that it provides a strong approximation guarantee. In particular, we prove that any top-k of the ranking computed by MatchRank guarantees an expected number of filled slots that is asymptotically at most $(1-1/e)$ away from the optimal set with high probability. Furthermore, we show how MatchRank can be implemented efficiently to handle ranking problems of substantial size. Finally, we provide an extensive empirical evaluation of MatchRank, showing that it can outperform heuristic baselines by a substantial margin, and perform accurately on a real-world college-admission problem.

2 RELATED WORK

In the following we detail how our new setting of ranking with slot constraints differs from existing research areas.

Search result diversification is a widely studied problem in IR in which one aims to cover multiple intents or aspects of an ambiguous or composite query. Specifically, in extrinsic diversification [28] the goal is to cover all intents of a query to ensure that the user finds at least one relevant result despite the uncertainty about the query intent. This typically leads to coverage-style objectives [9, 42], not matching problems like in slot-constrained ranking. For intrinsic diversification [28], the goal to put together a portfolio of items, but none of the existing methods provides the same flexibility in specifying complex systems of slots with arbitrary dependencies. Another difference of our setting to existing diversification approaches is that we do not need to model similarity between items, either explicitly through predefined aspects [32] or implicitly through similarity metrics [36]. However, one could use our slot-constrained ranking framework as a method for diversification, especially in high-stakes selection problems where practitioners require both full transparency and control over the composition of the set of selected items, and the ability to specify complex real-world matching constraints.

Beyond the standard diversity settings, Agarwal et al. [2] consider how to maximize user engagement while satisfying a minimum impression requirement per search result. However, their target objective is not a matching problem, since serving an item to one user does not exclude the use of such item with another user. Also related but different from our setting is the approach of Dang and Croft [13]. Their "seats" allocation approach is relevant to our slot constraints, *except that we directly treat the slots as a constraint instead of a normalization factor*. Therefore, we use the maximum bipartite matching algorithm [19] for the ranking objective.

Fairness in ranking is also frequently implemented by adding constraints to the ranking, since ranking by predicted merit can lead to both poor representation [4] and suboptimal performance [29] of admitted cohorts. In many cases, these fairness constraints ensure a certain amount of representation from different groups in various positions of the ranking (e.g., demographic parity) [17, 22]. This is critically different from our slot constraints, since slot constraints act on the *relevant* items, not all items. Note that under differential estimation accuracy between groups, merely ensuring representational fairness can still be unfair to the relevant items [16, 33]. Furthermore, slot constraints are different from independent diversity constraints [8], as slot constraints are always mutually exclusive (e.g., college applicants can only be admitted by and matriculate in 1 major of studies).

Matching problems have a wide range of applications in job markets, dating, and resource allocation in online clouds [14, 21, 38]. The typical setting in *stochastic matching* is that each edge in a graph is realized independently with (predicted) probability p [1, 5], which is analogous to the college admission scenario where we only have a calibrated regression model to know the predicted probability of a candidate C being relevant to a slot S. Dickerson et al. [14] consider online bipartite matching to improve the diversity and relevance of search results by maximizing a multilinear objective over the set of matched edges. Ahmed et al. [3] propose a quadratic programming based objective for the diversity of a matching and propose a scalable greedy algorithm to trade off efficiency and diversity. Instead of evaluating an objective on top of matching on a sampled graph, we use the size of the bipartite matching on a sampled graph to derive a ranking of candidates that maximizes the size of the bipartite matching in the true relevance matrix during the human evaluation phase. *We are the first to formulate this ranking problem to maximize the size of the bipartite matching, and this is a core contribution of this paper.*

3 PROBLEM SETTING

Consider that we have c candidates $\mathcal{C} = \{C_1, ..., C_c\}$, and we have $s < c$ slots $\mathcal{S} = \{S_1, ..., S_s\}$ that we need to fill with relevant candidates. Each candidate can be relevant to any number of slots, or no slots at all. We denote whether candidate C_i is relevant to slot S_j via the matrix entry $R_{ij} \in \{0, 1\}$. We use the generic concept of "relevance" to indicate whether a candidate matches a slot. This allows us to model a broad range of selection problems as follows:

Hiring: A company has a number of openings for different roles, with a specific number of interview slots budgeted for each role. Applicants may be qualified for some subset of roles. An applicant is relevant for an interview slot if qualified and interested in the particular opening.

College Admission: Slots correspond to seats in various majors (100 slots for CS, 50 slots for Math, etc.), and in each major a certain number of slots is reserved for low socioeconomic status students. A student is relevant for any slot in a major if both qualified for and interested in that major.

Medical Trial: Researchers need to find qualifying participants for a medical trial among millions of electronic health records.

The trial is designed with a certain number of slots by gender, race and disease severity. Patients are relevant for a slot if they match demographic requirements and disease severity determined by further testing.

In all of these application scenarios our goal is to fill all slots with relevant candidates. Throughout this paper we assume that relevance is binary, but we conjecture that many of our results can be extended to non-binary relevance values.

If the relevance matrix R was fully known, the problem of finding relevant candidates to fill all slots would be solved by the maximum bipartite matching algorithm [19]. In practice, however, we only have uncertain information $P(R)$ about the true relevances. $P(R)$ can be approximated by a probabilistic relevance prediction model learned from data ($\hat{P}(R)$). Furthermore, accurately revealing the true relevance vector $R_i \in \{0,1\}^s$ of any particular candidate C_i for all s slots comes at substantial cost. In the admission and hiring example, assessing relevance requires detailed human review of the application, and in the medical example it requires additional medical testing. We would thus like to avoid evaluating candidates that do not contribute to filling more slots, either because these candidates are not relevant or because we already have identified other relevant candidates for these slots.

To achieve this goal, we would like to compute a ranking σ of candidates so that evaluating the candidates $\sigma[1], ..., \sigma[c]$ from top to bottom maximizes the number of filled slots given the information contained in $P(R)$. Without slot constraints (i.e., for any candidate C_i: $\forall j,k : R_{ij} = R_{ik}$) the ranking problem has a well-known solution that follows from the Probability Ranking Principle (PRP) [30]: simply ranking candidates by their probability of relevance is optimal under most sensible metrics. However, this PRP ranking can be highly sub-optimal under general slot constraints, as the following toy example shows.

Example 1 (Suboptimality of PRP for Ranking with Slot Constraints). *Consider a problem with $c = 1000$ candidates and $s = 10$ slots. Candidates $C_1, ..., C_{500}$ have a probability of relevance of 0.5 for slots $S_1, ..., S_5$, and 0 for the other slots. Analogously, candidates $C_{501}, ..., C_{1000}$ have a probability of relevance of 0.4 for slots $S_6, ..., S_{10}$, and 0 for the other slots. Any heuristic based on sorting candidates by a score computed from their probability of relevance would either produce a ranking equivalent to $C_1, ..., C_{500}, C_{501}, ..., C_{1000}$ or equivalent to $C_{501}, ..., C_{1000}, C_1, ..., C_{500}$. However, either ranking would be highly suboptimal, since one type of slots would not be filled until after reviewing at least 500 candidates.*

Note that this high degree of suboptimality already surfaces in this particularly simplistic example, where there are only two types of slots and candidates are relevant to at most one type of slots. In the more general case, where we can have complex systems of slots where each candidate can have dependent probabilities of being relevant to multiple slots, it is not even clear how to heuristically apply the conventional PRP. This motivates the need for a new algorithm that goes beyond sorting candidates by some heuristic function of relevance, but that explicitly takes the slot constraints into account. In the following we develop the MatchRank algorithm that does not have the inefficiencies of the PRP ranking and that provides provable guarantees on the quality of the ranking for arbitrary

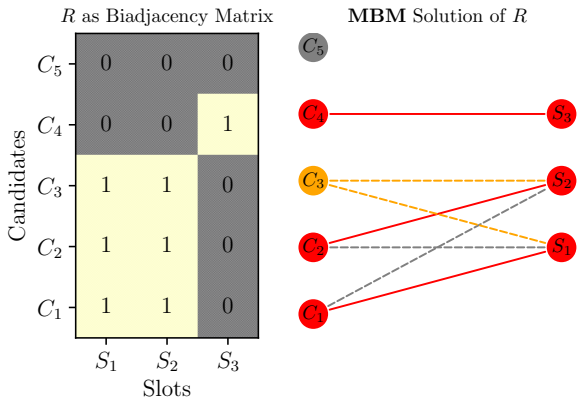


Figure 1: Example showing how MBM computes an optimal assignment of candidates to slots for a known relevance matrix R. In this figure, C_5 is not relevant for any slots. C_1, C_2 and C_4 are relevant and can be matched with available slots. C_3 is relevant for S_1 and S_2, but both are already occupied.

slot constraints and relevance models. To start the derivation, the following begins with a formal definition of the ranking objective.

3.1 Ranking Objective

We formalize the problem of ranking under slot constraints in two steps. We first define the problem of finding a candidate set X_k of a given size k that is optimal. In the second step we show how to construct a nested sequence

$$X_0 \subset X_1 \subset X_2 \subset ... \tag{1}$$

of such candidate sets that naturally forms a ranking σ. In particular, any two consecutive candidate sets X_k and X_{k+1} differ by only one candidate $X_{k+1} \setminus X_k = C_{\sigma[k+1]}$, which corresponds to the element $\sigma[k+1]$ of ranking σ. Note that X_0 is always the empty set.

To evaluate a given candidate set $X \subseteq C$, we use the size of the maximal matching between relevant candidates in X and slots $\mathcal{S}$. This is illustrated in Figure 1, where the candidates in X and the slots in $\mathcal{S}$ form a bipartite graph (right panel). The corresponding submatrix of the relevance matrix R (left panel) defines the biadjacency matrix of the graph, where an edge exists whenever a candidate C_i is relevant for slot S_j. The right panel of Figure 1 shows the (not necessarily unique) maximum bipartite matching $\{C_1 \to S_1, C_2 \to S_2, C_4 \to S_3\}$, which corresponds to the largest number of slots that can be filled with relevant candidates from X. We denote this maximum bipartite matching size as $\mathrm{MBM}(X, \mathcal{S}|R)$.

While $\mathrm{MBM}(X, \mathcal{S}|R)$ gives us the optimal solution for a known relevance matrix R, we need to evaluate candidates sets X under uncertainty about what the correct relevance matrix is. A natural metric for evaluating a candidate set X under $P(R)$ is to measure the expected size of the matching, which corresponds to the expected number of slots that can be filled with candidates from X.

$$M(X) = \mathbb{E}_{R\sim P(R)}\left[\mathrm{MBM}(X, \mathcal{S}|R)\right] \tag{2}$$

$$= \sum_R \mathrm{MBM}(X, \mathcal{S}|R) P(R) \tag{3}$$

When evaluating a ranking σ, we will apply this metric $M(X)$ to each top-k prefix X_k of the ranking σ.

3.2 MatchRank Algorithm

Given the metric $M(X)$ from Equation (3), our goal is to find a ranking σ of the candidates in C so that $M(X_k)$ for any top-k prefix is as large as possible under $P(R)$. We split this goal into three steps. First, we show how to estimate $M(X)$ for any candidate set X and for any $P(R)$ that permits sampling. Second, we show how to construct a single candidate set X_k of size k that has a large value for $M(X_k)$. Third, we show that our construction of the X_k in the previous step naturally produces a ranking.

Estimating $M(X)$. We do not make any structural assumptions on the distribution $P(R)$, and $P(R)$ can contain arbitrary dependencies among the entries. The following is a general method for evaluating a given candidate set X, where we merely require that we can sample relevance matrices R from $P(R)$. With such samples, we can compute Monte-Carlo estimates of $M(X)$ as follows.

Let $\mathcal{R} = [R_1, ..., R_n]$ be n samples of relevance matrices drawn i.i.d. from $P(R)$. The Monte-Carlo estimate of $M(X)$ is

$$\hat{M}(X) = \frac{1}{n}\sum_{i=1}^{n} \text{MBM}(X, \mathcal{S}|R_i). \tag{4}$$

By the weak law of large numbers, $\hat{M}(X)$ converges to $M(X)$ for large n. We will later characterize how the number of samples n affects the algorithm.

Constructing the ranking. Now that we know how to estimate the quality $\hat{M}(X)$ of any particular candidate set X, we can think about finding a candidate set that maximizes $\hat{M}(X)$. However, naively enumerating all subsets $X \subset C$ of size $|X| = k$ and evaluating $\hat{M}(X)$ would not be efficient. Furthermore, it would not be clear how to make sure that the candidate set is nested and forms a ranking.

To avoid this combinatorial enumeration, we instead construct $\hat{M}(X)$ using the following greedy algorithm, which we will prove to enjoy strong approximation guarantees. Since this algorithm adds one candidate in each iteration, it naturally constructs a ranking and we show that the approximation guarantees hold for any top-k of the ranking.

Algorithm 1: MatchRank

Input: candidates C; slots $\mathcal{S}$; sampled relevances $\mathcal{R} = [R_1, ..., R_n]$;
$X_0 \leftarrow \emptyset$; $A \leftarrow C$; $\sigma = []$; $k \leftarrow 1$
while $A \neq \emptyset$ **do**
$C_{\text{best}} \leftarrow \text{argmax}_{C \in A} \frac{1}{n}\sum_{i=1}^{n} \text{MBM}(X_{k-1} \cup \{C\}, \mathcal{S}|R_i)$
$\sigma[k] = C_{\text{best}}$
$X_k \leftarrow X_{k-1} \cup \{C_{\text{best}}\}$; $A \leftarrow A - \{C_{\text{best}}\}$; $k \leftarrow k + 1$
end while
Output: ranking σ

In each iteration k, the MatchRank algorithm finds the candidate C_{best} that most improves $\hat{M}(X_{k-1} \cup \{C_{\text{best}}\})$ for the current candidate set X_{k-1}. It then places C_{best} into position k of the ranking. Furthermore, it adds C_{best} to the current top-k set X_k, and it removes C_{best} from the set of remaining candidates A. These iterations continue until all candidates have been added to the ranking.

If only a top-k ranking is desired, one could also stop early. Note that Algorithm 1 is optimized for clarity, but Section 3.4 presents several efficiency improvements.

3.3 Theoretical Analysis

We now analyze theoretically how effective MatchRank is on constructing a ranking that optimizes the objective $M(X)$ given in Equation (3). We start by stating our main result, which we will then prove subsequently. The main result states that for any k, the top-k candidate set X_k constructed by MatchRank enjoys an approximation guarantee compared to the optimal candidate set

$$X_k^* = \underset{X \subseteq C \wedge |X|=k}{\text{argmax}} \; M(X).$$

Note that these optimal X_k^* may not be nested and may not form a ranking, unlike the X_k constructed by MatchRank.

Theorem 3.1. *The ranking σ produced by MatchRank when given s slots and n Monte-Carlo samples $\mathcal{R} = [R_1, ..., R_n]$ from $P(R)$ enjoys the following approximation guarantee for each top-k set X_k in σ: with probability $1 - \delta$ (where $0 < \delta < 1/2$),*

$$M(X_k) \geq \left(1 - \frac{1}{e}\right) M(X_k^*) - 2s\sqrt{\frac{O(k \ln k) + \ln(2/\delta)}{2n}},$$

where $X_k^* = \text{argmax}_{X \subseteq C \wedge |X|=k} M(X)$ is the optimal set.

The proof of Theorem 3.1 is given in Appendix A. Its main steps are to first show that $\hat{M}(X)$ is monotone submodular, which implies that the greedy nature of MatchRank provides a $(1 - 1/e)$ approximation guarantee for $\hat{M}(X)$. We then show that optimizing $\hat{M}(X)$ provides a solution that is close to optimizing $M(X)$ directly.

3.4 Computational Efficiency of MatchRank and Improvements

The MatchRank algorithm as written in Algorithm 1 is optimized for clarity, but there are a number of improvements that can substantially speed up computation. To motivate these improvements, we first analyze the runtime complexity of Algorithm 1.

For computing the top k positions of the ranking when there are c candidates, s slots, and n Monte-Carlo samples, the greedy maximizer inside Algorithm 1 will evaluate $O(k\,c)$ sets. For each such set, it will find the MBM solutions of all bipartite graphs from $\mathcal{R}$, which takes $O(n\,c\,s\,\sqrt{c+s})$ time per set when using the classic Hopcroft-Karp MBM algorithm [19]. So, a naive implementation will take $O(k\,n\,c^2\,s\,\sqrt{c+s})$ time. However, this implementation is unnecessarily slow.

We can improve the time efficiency of *MatchRank* by following the principle that any unmatched candidate can increase the matching size by at most 1. So, if we are given a candidate set $X \subseteq C$ and an unmatched candidate C, finding the $\text{MBM}(X \cup \{C\}, \mathcal{S}|R)$ can be reduced to determining if there is an augmenting path to the matching of $\text{MBM}(X, \mathcal{S}|R)$ starting from C. If such an augmenting path exists, then we can extend the matching and $\text{MBM}(X \cup \{C\}, \mathcal{S}|R) = \text{MBM}(X, \mathcal{S}|R) + 1$. It no augmenting path exists, then we will know that $\text{MBM}(X \cup \{C\}, \mathcal{S}|R) = \text{MBM}(X, \mathcal{S}|R)$ and the matching remains unchanged. Therefore, it will take $O(c\,s)$ time (a BFS) per each unmatched candidate per ranking step, and we obtain an $O(k\,n\,c\,(c\,s)) = O(k\,n\,c^2\,s)$ algorithm.

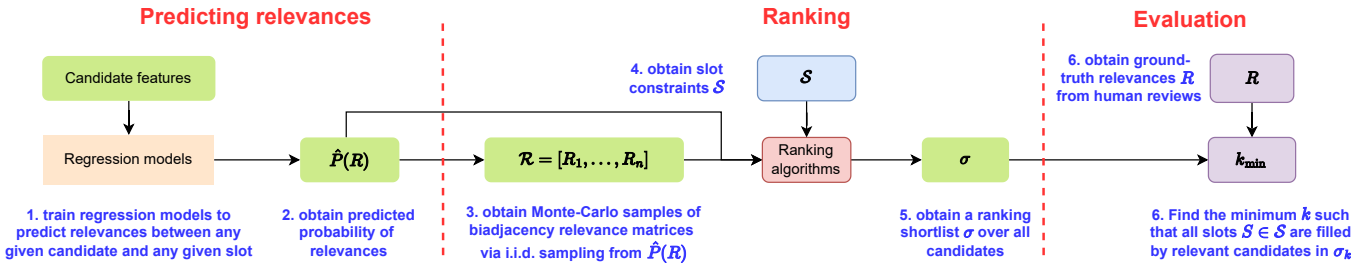


Figure 2: Application and evaluation pipeline used in ranking experiments.

If we consider the typical scenario where $c > s$, we can improve the time complexity *by keeping a list of unmatched slots instead of candidates* for each bipartite graph, and on each ranking step we start from each unmatched slot and follow the BFS to find all augmenting paths that end on unmatched candidates. We need $O(c\,s^2)$ time per ranking step, and in total only $O(k\,n\,c\,s^2)$ time. We also note that on each ranking step each Monte-Carlo estimate R_i is independent and with perfect parallelism, we could eliminate the dependency over n to get $O(k\,c\,s^2)$ time complexity.

A further improvement in runtime can be achieved by exploiting that MatchRank is a greedy algorithm for maximizing a submodular objective. For any such algorithm, we can use lazy evaluation [23] to accelerate the ranking process in practice[1]. Lazy greedy maintains a priority queue of stale marginal gains to reduce unnecessary computation of marginal gains for many examples per step. Since marginal gains can never increase due to the submodularity of the objective, the stale marginal gains provide an upper bound on the improvement. Thus, if a stale marginal gain is not large enough to propose a candidate as a greedy maximizer, then recomputing its marginal gain is not necessary. This is particularly effective in our matching scenarios, since many candidates are not relevant for some slots. This means they will have small marginal gains even in the first step of ranking, and we can significantly reduce computation of their marginal gains during subsequent iterations when using lazy greedy.

Finally, we could replace the greedy algorithm with some approximate version of greedy that is substantially faster, such as stochastic greedy [24] and threshold greedy [6] for monotone submodular maximization problem. We have performed initial evaluations of these methods with promising results, but did not find a need for them for our experiments. Generally, we found the exact greedy algorithm to be tractable for datasets with up to 50,000 candidates.

Note that the $\sqrt{\cdot}$ term approaches zero as we increase the number n of sampled relevance matrices. This means that for large Monte-Carlo samples, the approximation factor approaches $1 - 1/e$, where e is Euler's number.

The monotone submodularity of $\hat{M}(X)$ opens a large arsenal of submodular optimization methods for constructing candidate sets X_k with provable approximation guarantees. We opt for the greedy algorithm, since it naturally constructs a ranking.

[1]We run all of our experiments with lazy greedy.

4 EMPIRICAL EVALUATION

We now evaluate the MatchRank algorithm on three types of data. The first is fully synthetic data, where we can control all aspects of the ranking problems to understand the conditions under which MatchRank improves over baseline heuristics. Second, we evaluate MatchRank on a number of benchmark datasets. And finally, we verify the applicability of MatchRank on a real-world college-admissions problem. We provide software for reproducing the empirical results and to enable followup work on MatchRank[2].

Evaluation Process and Metric. For each problem, our evaluation follows the process depicted in Figure 2. We first use a training set to learn a model — usually a calibrated regression model – which we can use to infer the relevance probabilities $\hat{P}(R)$ for the candidates in the test set. We apply MatchRank and other baseline rankers to rank this test set, which only requires sampling from $\hat{P}(R)$.

To evaluate any ranking σ, we use the following process and metric. For each top-k prefix σ_k of σ, we reveal the ground-truth relevance labels R of these k candidates and compute how many slots can be filled when optimally matching these k candidates to the slots. This is precisely the size of the matching $MBM(\sigma_k, \mathcal{S}|R)$. Our final evaluation metric for σ is the smallest k for which the prefix σ_k fills all $|\mathcal{S}|$ slots with relevant candidates.

$$k_{\min} = \operatorname*{argmin}_{k \in [c]} \{MBM(\sigma_k, \mathcal{S}|R) = |\mathcal{S}|\} \tag{5}$$

This means that $k_{\min}$ is the number of candidates in σ that need to be reviewed before all slots are filled.

Since $k_{\min}$ scales with the total number of slots, we report the normalized $k_{\min}/|\mathcal{S}|$, so that the best possible score is 1.

Baseline Rankers. We compare MatchRank against the following baselines rankers. These methods compute a score for each candidate, and then rank by this score. The baseline rankers differ by how they aggregate the estimated marginal relevance probabilities $\hat{P}(R_{C,S})$ (estimated from the Monte-Carlo samples) for each candidate C across all slots S. The first heuristic is motivated by the soft **AND** rule $\prod_{S \in \mathcal{S}} \hat{P}(R_{C,S})$. The second uses the soft **OR** rule $1 - \prod_{S \in \mathcal{S}} (1 - \hat{P}(R_{C,S}))$. Both the **AND** and the **OR** rule skip probabilities that are zero when computing the product. The third heuristic, called Total Relevance (**TR**) merely sums the relevance probabilities $\sum_{S \in \mathcal{S}} \hat{P}(R_{C,S})$ across all slots. The final heuristic is a normalized version of the total relevances, called **NTR**, that normalizes with the competition for each slot (number of relevant candidates for

[2]https://github.com/GarlGuo/ranking_with_slot_constraints.git

Table 1: Synthetic Datasets: Performance of MatchRank in comparison to the heuristic baselines, reporting mean and standard deviation in the format of "mean_{std}" (for standard error divide by $\sqrt{1000}$) of $k_{\min}/|\mathcal{S}|$ over 1000 random draws of the true relevance matrix R from $P(R)$.

	Default	# Slots Per Group		# Group Memberships		# Samples		$P(R)$	
		30	70	1	3	100	1000	S	L
MatchRank	$\mathbf{1.27}_{0.06}$	$\mathbf{1.26}_{0.07}$	$\mathbf{1.29}_{0.05}$	$\mathbf{2.05}_{0.20}$	$\mathbf{1.12}_{0.03}$	$\mathbf{1.32}_{0.10}$	$\mathbf{1.25}_{0.03}$	$\mathbf{1.52}_{0.13}$	$\mathbf{1.14}_{0.03}$
AND	$5.01_{0.29}$	$6.06_{0.64}$	$4.29_{0.20}$	$11.26_{0.31}$	$2.73_{0.24}$	$5.01_{0.29}$	$5.00_{0.28}$	$5.84_{0.39}$	$4.45_{0.25}$
OR	$4.20_{0.33}$	$4.56_{0.53}$	$3.85_{0.22}$	$11.26_{0.31}$	$2.29_{0.21}$	$4.21_{0.34}$	$4.20_{0.33}$	$5.31_{0.41}$	$3.29_{0.29}$
TR	$4.41_{0.31}$	$5.48_{0.39}$	$4.00_{0.23}$	$10.73_{0.44}$	$2.57_{0.17}$	$4.14_{0.31}$	$4.73_{0.31}$	$5.46_{0.41}$	$3.90_{0.26}$
NTR	$1.35_{0.07}$	$1.45_{0.12}$	$1.30_{0.04}$	$3.69_{0.14}$	$\mathbf{1.12}_{0.02}$	$1.45_{0.07}$	$1.39_{0.09}$	$1.66_{0.13}$	$1.20_{0.05}$
Random	$1.69_{0.11}$	$1.78_{0.18}$	$1.68_{0.02}$	$3.70_{0.40}$	$1.23_{0.04}$	$1.69_{0.11}$	$1.69_{0.11}$	$2.51_{0.25}$	$1.35_{0.07}$

each slot in each sampled relevance matrix) $\sum_{S\in\mathcal{S}} \frac{\hat{P}(R_{C,S})}{\sum_{C'\in C}\hat{P}(R_{C',S})}$. We use the *same* sampled relevance matrices $\mathcal{R} = [R_1, \ldots, R_n]$ as what we use for MatchRank when computing the ranking shortlist for TR and NTR. Note that the conventional diversification methods we discussed in the related work section cannot handle relevance signals that are multivariate vectors to optimize the number of filled slots, and thus an empirical comparison is not sensible.

4.1 Synthetic Datasets

We first focus on a synthetic dataset where we can control the structure of $P(R)$, so that we can investigate all problem dimensions that affect the MatchRank algorithm. Furthermore, we can directly use $P(R)$ instead of $\hat{P}(R)$, which avoids confounding the algorithm's behavior with potential inaccuracies in a learned $\hat{P}(R)$.

Experiment Setup. To construct synthetic $P(R)$, we define g groups (default $g = 10$) and each group has s slots (default $s = 50$). We then create 10,000 candidates, where each candidate is randomly assigned to a groups (default $a = 2$). If candidate C is a member of group $j \in \{1, ..., g\}$, we first sample p from the Gaussian distribution $\mathcal{N}(p_{\text{base}} + 0.03 * j, 0.1)$ with default p_{base} as 0.3, and clip p to the range of (0.0001, 0.9999). We then sample from a Bernoulli distribution with probability p, and the success outcome means C is relevant for all slots associated with group j. If candidate C is not member of group j, then C is not relevant for any slots associated with group j, or formally $P(R_{C,S}) = 0$ if slot S belongs to group j. If not mentioned otherwise, parameters are at their default value.

We draw 200 Monte-Carlo i.i.d. samples $\mathcal{R} = [R_1, ..., R_{200}]$ as input to the ranking algorithms. For evaluation, we draw a ground-truth relevance matrix R and compute $k_{\min}/|\mathcal{S}|$, repeat this evaluation 1000 times, and report the mean and standard deviation.

Comparing MatchRank against the Baselines. The first column of Table 1 shows the performance of MatchRank and the baselines for the default values of the synthetic data generator. MatchRank achieves a performance of 1.27, which means that on average only an additional 27% of candidates need to be reviewed beyond the number of slots. The best heuristic is NTR, which averages 35%. Most heuristics do worse than random, which requires 69%. The reason is that the heuristics systematically miss candidates of some group, such that those slots cannot be filled.

Effects of the Number of Slots Per Group. We now vary the number of slots per group from the default of 50 to 30 and 70. The results for 30 and 70 are in the third main column of Table 1, while the results for 50 are in the default column. All other parameters are at their default values. MatchRank again show stable performance over all three settings and dominates most baselines. Only NTR comes close for larger numbers of slots.

Effects of Number of Group Memberships. We now assign each candidate to be a member of 1, 2 (default) and 3 groups respectively. All other parameters are at their default. The fourth main column of Table 1 shows that the problem becomes easier for all methods, when each candidate can be in more groups. We see that the advantage of MatchRank is greater for harder problems.

Effect of Number of Monte-Carlo Samples. We now investigate how important the number n of Monte-Carlo samples $\mathcal{R}$ is, as we vary n among 100, 200 (default), and 1000. The results can be found in the fifth main column of Table 1. We find that *MatchRank* already performs well for $n = 100$, although having larger samples still improve the results. A larger n may be even more important for $P(R)$ with stronger dependencies between candidates and slots.

Effect of Overall Relevance Level. We now vary the overall probability of relevance. In our model, this is controlled by the parameter p_{base}, and the higher p_{base} the greater the overall probability of relevance. We vary p_{base} between 0.2 (S), 0.3 (default), and 0.4 (L). The results can be found in the last major column of Table 1. As p_{base} increases, the problem of finding relevant candidates to fill the slots becomes easier. Both MatchRank and the baselines benefit from this, but MatchRank maintains a consistent advantage.

Analyzing the Variability of MatchRank. We want to further understand the variability of $k_{\min}/|\mathcal{S}|$ of MatchRank across ground-truth relevance matrices. So we prepare a histogram of $k_{\min}/|\mathcal{S}|$ for MatchRank evaluating on $\mathcal{R}_{\text{test}}$ in default settings as shown in Figure 3. This figure demonstrates that the distribution of $k_{\min}/|\mathcal{S}|$ is slightly right-tailed as the sample mean is higher than the sample median. This could be explained as MatchRank being prone to overestimating candidates' probability of relevances. Therefore, it is likely that some slots are not filled by relevant candidates even when MatchRank believes all slots have been filled. Such phenomenon is illustrated by the fact that there are still 24.9% R samples for which slots have not been filled after $\Delta\mathcal{R}$ becomes 0 (the purple line) in the Figure 3.

Table 2: Real-World Benchmarks: Performance of MatchRank in comparison to heuristic baselines in terms of $k_{\min}/|\mathcal{S}|$. For each trial, we repeat for 3 random seeds and report the mean and standard deviation of $k_{\min}/|\mathcal{S}|$ in the format of "mean$_{\text{std}}$".

Datasets	Medical			Bibtex			Delicious		
Slots Per Label	5	10	15	10	20	30	10	30	50
MatchRank	$\mathbf{2.17}_{0.26}$	$\mathbf{2.00}_{0.10}$	$\mathbf{2.23}_{0.37}$	$\mathbf{2.62}_{0.53}$	$\mathbf{2.33}_{0.51}$	$\mathbf{2.07}_{0.12}$	$\mathbf{1.09}_{0.07}$	$\mathbf{1.05}_{0.01}$	$\mathbf{1.07}_{0.01}$
AND	$4.12_{0.72}$	$3.00_{0.16}$	$2.57_{0.23}$	$7.74_{1.00}$	$4.92_{0.36}$	$4.20_{0.23}$	$1.68_{0.06}$	$1.94_{0.17}$	$1.69_{0.03}$
OR	$4.82_{0.99}$	$3.36_{0.17}$	$2.80_{0.25}$	$6.42_{0.43}$	$4.40_{0.17}$	$3.68_{0.04}$	$2.49_{0.11}$	$2.10_{0.09}$	$2.05_{0.08}$
TR	$4.40_{0.50}$	$3.27_{0.43}$	$2.75_{0.29}$	$6.35_{0.44}$	$4.33_{0.15}$	$3.68_{0.08}$	$2.21_{0.29}$	$2.11_{0.10}$	$1.98_{0.05}$
NTR	$4.83_{0.43}$	$2.87_{0.21}$	$2.08_{0.09}$	$5.96_{0.40}$	$3.35_{0.17}$	$2.48_{0.05}$	$1.52_{0.25}$	$1.52_{0.26}$	$1.50_{0.01}$
Random	$4.56_{0.59}$	$3.95_{0.42}$	$3.49_{0.22}$	$8.10_{0.06}$	$6.48_{0.17}$	$6.41_{0.27}$	$1.37_{0.02}$	$1.42_{0.06}$	$1.39_{0.04}$

Datasets	TMC2007			Mediamill			Bookmarks		
Slots Per Label	30	50	70	10	30	50	10	30	50
MatchRank	$\mathbf{1.24}_{0.09}$	$\mathbf{1.30}_{0.04}$	$\mathbf{1.28}_{0.02}$	$\mathbf{1.03}_{0.00}$	$\mathbf{1.07}_{0.03}$	$\mathbf{1.11}_{0.01}$	$\mathbf{4.66}_{0.29}$	$\mathbf{3.60}_{0.44}$	$\mathbf{3.70}_{0.13}$
AND	$13.68_{1.47}$	$9.71_{1.09}$	$7.98_{1.08}$	$1.84_{0.11}$	$2.07_{0.21}$	$2.06_{0.03}$	$68.60_{1.83}$	$26.95_{4.57}$	$24.81_{0.92}$
OR	$3.18_{0.42}$	$3.31_{0.16}$	$3.59_{0.39}$	$2.77_{0.29}$	$2.66_{0.29}$	$2.22_{0.10}$	$20.12_{1.56}$	$10.95_{0.62}$	$8.93_{0.68}$
TR	$8.71_{0.25}$	$6.26_{0.25}$	$5.52_{0.30}$	$3.12_{0.17}$	$2.27_{0.11}$	$2.11_{0.05}$	$19.63_{2.01}$	$11.64_{0.60}$	$9.23_{0.43}$
NTR	$1.82_{0.09}$	$1.48_{0.08}$	$1.37_{0.07}$	$2.63_{0.24}$	$1.67_{0.03}$	$1.49_{0.04}$	$22.71_{2.40}$	$10.13_{0.73}$	$7.02_{0.23}$
Random	$4.37_{0.05}$	$3.79_{0.05}$	$3.55_{0.12}$	$2.74_{0.48}$	$2.46_{0.31}$	$2.80_{0.14}$	$13.78_{0.69}$	$11.93_{0.50}$	$13.14_{0.57}$

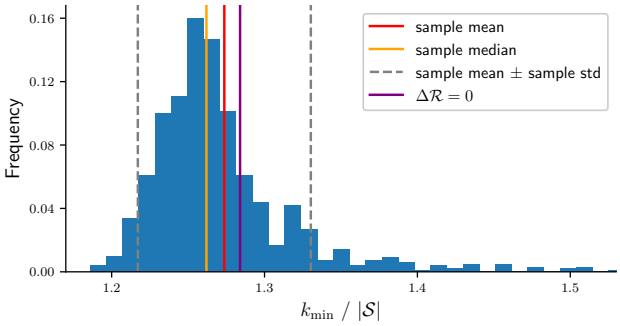


Figure 3: Synthetic Datasets: Histogram of $k_{\min}/|\mathcal{S}|$ for MatchRank under default settings for 1000 draws of the ground-truth relevances from $P(R)$. The vertical lines are the mean, median, and standard deviation of the histogram. $\Delta\mathcal{R} = 0$ is the position in the ranking when all slots in the Monte-Carlo $\mathcal{R}$ samples are filled and the marginal gain of adding another candidate becomes 0.

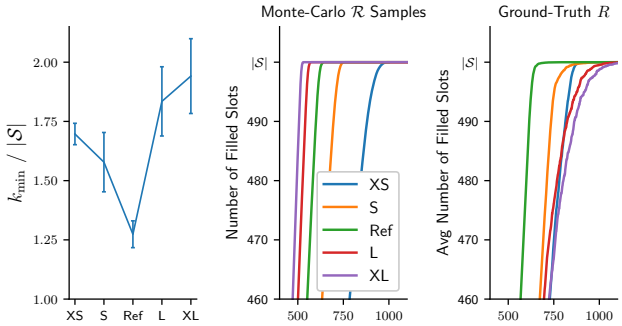


Figure 4: Effect from $\hat{P}(R)$ misspecification, where Ref is the correct model. The left subfigure shows mean and standard deviation of $k_{\min}/|\mathcal{S}|$.

Analyzing the Robustness of MatchRank to $\hat{P}(R)$ Misspecification. We now investigate how robust MatchRank is against an inaccurately learned $\hat{P}(R)$. We draw the ground-truth relevance labels from a model $P(R)$ with $p_{\text{base}} = 0.3$ (Ref), but draw the Monte-Carlo samples from misspecified models $\hat{P}(R)$ with p_{base} set to 0.1 (XS), 0.2 (S), 0.4 (L), and 0.5 (XL). Figure 4 shows that MatchRank performs best for the correctly specified $\hat{P}(R)$ as expected. Among the misspecified $\hat{P}(R)$, we can observe that *MatchRank* is more vulnerable to overestimation of relevance probabilities than underestimation. This can be explained as follows. If MatchRank is erroneously convinced that a slot is filled with high probability, it will not add an alternative candidate for this slot to the ranking. This fact is illustrated in the middle subfigure of Figure 4, where MatchRank fills all slots in the Monte-Carlo $\mathcal{R}$ samples faster as the overall relevance level increases. However, the average number of filled slots using the ground-truth relevances R is not necessarily higher, and in fact both L and XL have fewer filled slots per ranking step than S and XS for R. So it may be advisable to clip $\hat{P}(R)$ to a maximum value that is well below 1 to increase robustness to misspecification.

4.2 Real-World Benchmark Datasets

We now evaluate MatchRank on six benchmark datasets, where we now learn the relevance model $\hat{P}(R)$ from training data. These benchmark datasets are constructed from the multi-label datasets Medical [26], Bibtex [20], Delicious [39], TMC2007 [35], Mediamill [34], and Bookmarks [20] from the Mulan data repository [40]. Our data source is the scikit-multilearn library [37].

Experiment Setup. Each dataset comes with a train/test split. We consider each example in test set as a candidate, and each label as a group. We first select 10 labels that cover different numbers of relevant candidates (the specific label choices are included in Table 4 in Appendix B), and then allocate a certain number of slots for each label. We train $\hat{P}(R)$ on the training set using a calibrated binary logistic regression for each label. For the Medical, Bibtex, Delicious, Mediamill, and Bookmarks dataset, we use Platt scaling method [27] to calibrate the probabilistic predictions while for the TMC2007 dataset, we use isotonic regression. To raise the noise in the relevance prediction to a more challenging and realistic level, we mask 20% of the label in both the train and test set. We then draw 100 Monte-Carlo samples from $\hat{P}(R)$ as input to the ranking algorithm. All rankers are evaluated against the true masked relevance labels in the test set, meaning that a test example matches a slot if it contains the corresponding label.

Results. The result are shown in Table 2. For all benchmark datasets and all numbers of slots per label, MatchRank delivers the best ranking performance in terms of $k_{\min}/|\mathcal{S}|$. The most competitive heuristic ranker is NTR, but the results show that its heuristic can fail on some datasets and provide substantially worse ranking performance than MatchRank (e.g. Bookmarks). These results demonstrate that MatchRank performs robustly over a wide range of datasets where the $\hat{P}(R)$ is learned from training data.

Analysis. To provide additional insights into the behavior of MatchRank in comparison to the heuristic rankers, Figure 5 shows their behavior on each dataset with medium amount of slots per label. On each subfigure, left panels show the average number of filled slots over the Monte-Carlo samples as the shortlist size k grows. Across all datasets, MatchRank finds a close to optimal ranking while all heuristics require a substantially longer shortlist. This indicates that the heuristics optimize a fundamentally wrong objective. Right panels of each subfigure in Figure 5 show the number of filled slots when using the ground-truth labels for matching. These plots generally show a gap between MatchRank and the optimal ranking one can compute from the ground-truth labels. Since MatchRank performs close to optimal on the Monte-Carlo samples, this gap can be attributed to the inherent inaccuracy and uncertainty of $\hat{P}(R)$ in predicting the ground-truth relevance. This suggests that MatchRank performs close to optimal in terms of its optimization performance, and that the remaining suboptimality is largely a result of an imperfect $\hat{P}(R)$.

4.3 College Admission Dataset

To verify the effectiveness of MatchRank under real-world conditions, we consider an anonymized undergraduate admission dataset from a selective US university. The groups in this datasets are majors, and we posit that each major has only a fixed number of slots for admitting qualified students. We consider all majors that admitted at least 50 students, which leaves us with 13 majors and 19421 applicants in the test set. On the admission decisions from the prior year we train a boosted tree model with XGBoost [11] using a logistic loss objective and L2-regularization. This model is used to predict each applicant's probability of being admitted, and we clip the maximum $\hat{P}(R)$ by 0.3. Applicants can indicate their interests in majors, and the probability of admission to a major that the applicant did not indicate interest in is set to zero. This provides us with $\hat{P}(R)$ for all test applicants.

Table 3: College Admission: Performance of MatchRank in comparison to heuristics in terms of $k_{\min}/|\mathcal{S}|$. For each trial, we repeat for 3 random seeds and report the mean and standard deviation of $k_{\min}/|\mathcal{S}|$ in the format of "mean_{std}".

$s_{\max}$	30	50	70	100
MatchRank	$\mathbf{5.74}_{0.26}$	$\mathbf{7.65}_{3.08}$	$\mathbf{7.34}_{2.87}$	$\mathbf{7.52}_{3.07}$
AND	$12.63_{5.98}$	$8.69_{3.48}$	$10.64_{4.45}$	$9.43_{3.16}$
OR	$22.84_{12.13}$	$19.87_{10.14}$	$17.65_{8.64}$	$16.44_{7.81}$
TR	$22.58_{11.07}$	$16.89_{7.58}$	$14.07_{6.03}$	$12.57_{5.02}$
NTR	$7.02_{2.73}$	$8.32_{3.58}$	$8.10_{3.49}$	$7.73_{3.38}$
Random	$30.78_{16.13}$	$23.11_{11.61}$	$19.18_{9.32}$	$17.01_{7.83}$

We create 100 Monte-Carlo sampled relevance matrices from this $\hat{P}(R)$ as input to all ranking algorithms. To set the number of available slots for each member, we use $\min(\lfloor 0.7 * |\text{relevant applicants for this major}| \rfloor, s_{\max})$ to get an interesting relationship between supply and demand, and we vary $s_{\max}$ in the following experiments. We then run our ranking algorithms, and during evaluation we reveal the applicants ground-truth admissions decisions (i.e. the true relevance matrix R) for each candidate. To evaluate, we again find the minimum shortlist size k for each algorithm at which all slots are filled with relevant applicants.

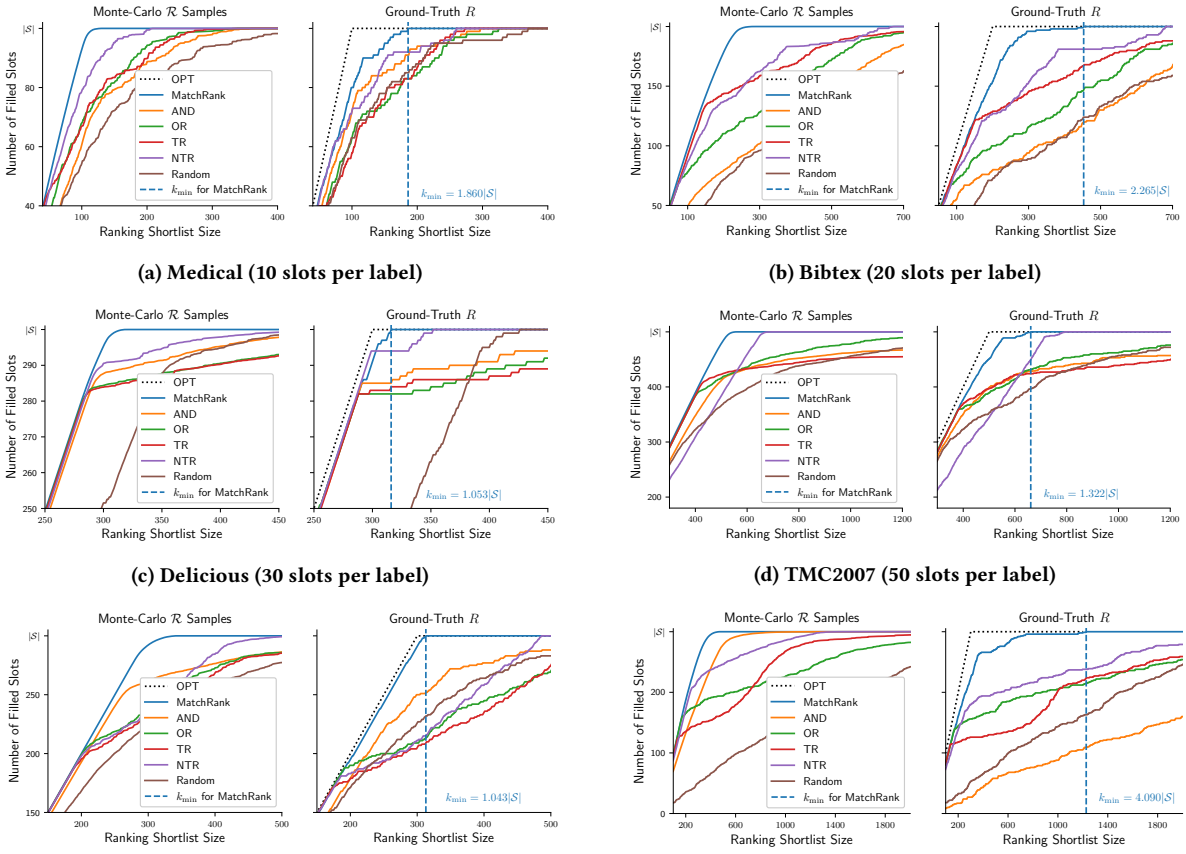
The results are provided in Table 3. This dataset is substantially more challenging for all methods, as the density of relevant candidates is small and we need to find a larger fraction of the relevant candidates to fill all slots. However, even in this challenging setting, we see that MatchRank is more effective than the heuristic baselines. This holds particularly when the number of slots per major is smaller, as the performance gap between MatchRank and other heuristic algorithms becomes larger. For larger $s_{\max}$ the majority of relevant candidates need to be found, such that even a small degree of inaccuracy in $\hat{P}(R)$ can have large impact.

5 EXTENSIONS AND FUTURE WORK

Instead of binary relevances as assumed in this paper, some applications may require real-valued relevances (e.g., star-ratings) where the decision maker aims to maximize the sum of relevances under slot constraints. In this setting, the optimal solution is given by the Maximum Weight Bipartite Matching $MWBM(X, \mathcal{S}|R)$ for the weighted biadjacency matrix R. If the MWBM objective is also monotone submodular[3], we could simply replace the MBM in MatchRankwith the MWBM and provide a similar approximation guarantee for this weighted version of MatchRank.

Another extension is the use of high-dimensional matching instead of the bipartite matching considered in this paper. This could model slot constraints over more than one category. For example, college admission may have slot constraints not only for majors but also for extracurricular teams (e.g., orchestra, athletics). Since slots for majors are orthogonal to the slots for extracurricular teams, this

[3]We find a submodularity proof for maximum generalized flow problem in Fleischer [15], and we would refer readers to that paper as such discussion is beyond the scope of this paper.



(a) Medical (10 slots per label)

(b) Bibtex (20 slots per label)

(c) Delicious (30 slots per label)

(d) TMC2007 (50 slots per label)

(e) Mediamill (30 slots per label)

(f) Bookmarks (30 slots per label)

Figure 5: Real-World Benchmarks: number of filled slots on the Monte-Carlo samples versus the expected number of filled slots w.r.t. the ground-truth relevances R. On the right plot of each subfigure, "OPT" is the optimal ranking if the ground-truth R was known, which achieves $MBM(\sigma^{OPT}_{|\mathcal{S}|}, \mathcal{S}|R) = |\mathcal{S}|$. The dashed blue line represents the $k_{\min}$ result for MatchRank, with the exact ratio over $|\mathcal{S}|$ illustrated nearby. Each subfigure represents a single seed result of the described experiment setting.

corresponds to a three-dimensional matching (3-DM) problem. The 3-DM problem is NP-hard and even APX-hard [18], and the best approximation algorithm so far achieves an error bound of $(4/3 + \epsilon)$ [12]. In general, Hazan et al. [18] show that all d-DM ($d \geq 3$) problems cannot be approximated within a factor of $O(d/\ln d)$ unless P = NP. But even if this matching problem cannot be solved exactly any more, the approximate solutions could give rise to an approximate ranking algorithm analogous to MatchRank.

6 CONCLUSION

We introduce the problem of ranking under slot constraints, which allows practitioners to specify conditions that arise in a wide variety of applications. To solve this ranking problem, we develop the MatchRank algorithm and show that it provides a theoretical guarantee on its ranking performance. A key insight is that the ranking objective can be related to the maximum bipartite matching problem, and that it is monotone submodular. We also show how MatchRank can be implemented efficiently so that it can efficiently handle real-world ranking problems of substantial size. Beyond its theoretical guarantees, MatchRank shows superior ranking performance across extensive experiments compared to several heuristic baselines. This holds across a wide range of datasets and experiment conditions, and MatchRank shows robustness to sample size and misspecified relevance distributions. We conclude that the ability to model complex problems and provide accurate rankings across a wide range of domains, backed by theoretical guarantees, makes the slot constraint framework a promising paradigm for tackling complex real-world ranking problems.

ACKNOWLEDGMENTS

This research was supported in part by NSF Awards IIS-2008139, IIS-2312865, and OAC-2311521. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Marek Adamczyk, Brian Brubach, Fabrizio Grandoni, Karthik A Sankararaman, Aravind Srinivasan, and Pan Xu. 2020. Improved approximation algorithms for stochastic-matching problems. arXiv:2010.08142

[2] Deepak Agarwal, Shaunak Chatterjee, Yang Yang, and Liang Zhang. 2015. Constrained Optimization for Homepage Relevance. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15 Companion)*. Association for Computing Machinery, New York, NY, USA, 375–384. https://doi.org/10.1145/2740908.2745398

[3] Faez Ahmed, John P Dickerson, and Mark Fuge. 2017. Diverse weighted bipartite b-matching. arXiv:1702.07134

[4] Peter Arcidiacono, Michael Lovenheim, and Maria Zhu. 2015. Affirmative action in undergraduate education. *Annu. Rev. Econ.* 7, 1 (2015), 487–518.

[5] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2019. The stochastic matching problem with (very) few queries. *ACM Transactions on Economics and Computation (TEAC)* 7, 3 (2019), 1–19.

[6] Ashwinkumar Badanidiyuru and Jan Vondrák. 2014. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 1497–1514.

[7] Claudia R Baquet, Patricia Commiskey, C Daniel Mullins, and Shiraz I Mishra. 2006. Recruitment and participation in clinical trials: socio-demographic, rural/urban, and health care access predictors. *Cancer detection and prevention* 30, 1 (2006), 24–33.

[8] Robert Bredereck, Piotr Faliszewski, Ayumi Igarashi, Martin Lackner, and Piotr Skowron. 2018. Multiwinner elections with diversity constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[9] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 335–336.

[10] Olivier Chapelle, Shihao Ji, Ciya Liao, Emre Velipasaoglu, Larry Lai, and Su-Lin Wu. 2011. Intent-based diversification of web search results: metrics and algorithms. *Information Retrieval* 14 (2011), 572–592.

[11] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[12] Marek Cygan. 2013. Improved approximation for 3-dimensional matching via bounded pathwidth local search. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 509–518.

[13] Van Dang and W. Bruce Croft. 2012. Diversity by Proportionality: An Election-Based Approach to Search Result Diversification. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Portland, Oregon, USA) *(SIGIR '12)*. Association for Computing Machinery, New York, NY, USA, 65–74. https://doi.org/10.1145/2348283.2348296

[14] John P Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. 2019. Balancing relevance and diversity in online bipartite matching via submodularity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1877–1884.

[15] Lisa Fleischer. 2010. Data center scheduling, generalized flows, and submodularity. In *2010 Proceedings of the Seventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*. SIAM, 56–65.

[16] Nikhil Garg, Hannah Li, and Faidra Monachou. 2021. Dropping Standardized Testing for Admissions Trades Off Information and Access. arXiv:2010.04396 [cs] http://arxiv.org/abs/2010.04396

[17] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. 2019. Fairness-Aware Ranking in Search & Recommendation Systems with Application to LinkedIn Talent Search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 2221–2231. https://doi.org/10.1145/3292500.3330691

[18] Elad Hazan, Shmuel Safra, and Oded Schwartz. 2003. On the complexity of approximating k-dimensional matching. In *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*. Springer, 83–97.

[19] John E Hopcroft and Richard M Karp. 1973. An n^5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing* 2, 4 (1973), 225–231.

[20] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*, Vol. 18. Citeseer, 5.

[21] Samir Khuller, Stephen G Mitchell, and Vijay V Vazirani. 1994. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science* 127, 2 (1994), 255–267.

[22] Jon Kleinberg and Manish Raghavan. 2018. Selection Problems in the Presence of Implicit Bias. arXiv:1801.03533 [cs, stat] http://arxiv.org/abs/1801.03533

[23] Michel Minoux. 1978. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*. Springer, 234–243.

[24] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. 2015. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[25] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14, 1 (1978), 265–294.

[26] John Pestian, Chris Brew, Pawel Matykiewicz, Dj J Hovermale, Neil Johnson, K Bretonnel Cohen, and Wlodzislaw Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Biological, translational, and clinical language processing*. 97–104.

[27] John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* 10, 3 (1999), 61–74.

[28] Filip Radlinski, Paul N. Bennett, Ben Carterette, and Thorsten Joachims. 2009. Redundancy, Diversity and Interdependent Document Relevance. *SIGIR Forum* 43, 2 (dec 2009), 46–52. https://doi.org/10.1145/1670564.1670572

[29] Leonard Ramist, Charles Lewis, and Laura McCamley-Jenkins. 1994. Student Group Differences in Predicting College Grades: Sex, Language, and Ethnic Groups. *ETS Research Report Series* 1994 (1994), 41.

[30] Stephen E Robertson. 1977. The probability ranking principle in IR. *Journal of documentation* 33, 4 (1977), 294–304.

[31] Gerard Salton (Ed.). 1971. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs, NJ.

[32] Rodrygo Santos, Craig Macdonald, and Iadh Ounis. 2010. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*. 881–890. https://doi.org/10.1145/1772690.1772780

[33] A. Singh, D. Kempe, and T. Joachims. 2021. Fairness in Ranking under Uncertainty. In *Neural Information Processing Systems (NeurIPS)*.

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

[35] Ashok N Srivastava and Brett Zane-Ulman. 2005. Discovering recurring anomalies in text reports regarding complex space systems. In *2005 IEEE aerospace conference*. IEEE, 3853–3862.

[36] Zhan Su, Zhicheng Dou, Yutao Zhu, Xubo Qin, and Ji-Rong Wen. 2021. Modeling Intent Graph for Search Result Diversification. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 736–746. https://doi.org/10.1145/3404835.3462872

[37] P. Szymański and T. Kajdanowicz. 2017. A scikit-based Python environment for performing multi-label classification. arXiv:1702.01460 [cs.LG]

[38] Joseph Thekinen and Jitesh H Panchal. 2017. Resource allocation in cloud-based design and manufacturing: A mechanism design approach. *Journal of Manufacturing Systems* 43 (2017), 327–338.

[39] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2008. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, Vol. 21. 53–59.

[40] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2009. Mining multi-label data. *Data mining and knowledge discovery handbook* (2009), 667–685.

[41] Lequn Wang, Thorsten Joachims, and Manuel Gomez Rodriguez. 2022. Improving Screening Processes via Calibrated Subset Selection. arXiv:2202.01147 [cs, stat] http://arxiv.org/abs/2202.01147

[42] Yisong Yue and T. Joachims. 2008. Predicting Diverse Subsets Using Structural SVMs. In *International Conference on Machine Learning (ICML)*. 271–278.

[43] ChengXiang Zhai, William W Cohen, and John Lafferty. 2015. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Acm sigir forum*, Vol. 49. ACM New York, NY, USA, 2–9.

In Section A we will provide a proof of Theorem 3.1. In Section B, we report the selected labels in our real-world benchmark experiments.

A PROOF OF THEOREM 3.1

The first step in proving Theorem 3.1 is to prove that the maximum bipartite matching is monotone submodular.

Lemma A.1. *For any relevance matrix R and any set of slots $\mathcal{S}$, the size of the maximum bipartite matching is monotone in X*

$$\forall X \subseteq C, \forall C \in C : MBM(X \cup \{C\}, \mathcal{S}|R) \geq MBM(X, \mathcal{S}|R) \tag{6}$$

and also submodular in X, which means that $\forall X \subseteq C, \forall C, C' \in C$

$$MBM(X \cup \{C\}, \mathcal{S}|R) - MBM(X, \mathcal{S}|R) \geq MBM(X \cup \{C, C'\}, \mathcal{S}|R) - MBM(X \cup \{C'\}, \mathcal{S}|R).$$

Proof. By a matching $\mathcal{M}$ from X to $\mathcal{S}$, we refer to a set of candidate-slot pairs $\mathcal{M} \subset X \times \mathcal{S}$ such that all pairs are relevant ($R_{cs} = 1$ for all $(c, s) \in \mathcal{M}$) and no candidate or slot appears in more than one pair in $\mathcal{M}$. We call a matching $\mathcal{M}$ from X to $\mathcal{S}$ maximum if $|\mathcal{M}|$ is maximized over all matchings from X to $\mathcal{S}$.

For monotonicity, we know that every single matching from X to $\mathcal{S}$ is also a possible matching from $X \cup \{C\}$ to $\mathcal{S}$, thus the size of the maximum matching from $X \cup \{C\}$ to $\mathcal{S}$ must be at least the size of the maximum matching from X to $\mathcal{S}$. So Equation 6 holds.

For submodularity, if we denote

$$M_0 = \text{MBM}(X, \mathcal{S}|R) \tag{7}$$

$$M_1 = \text{MBM}(X \cup \{C\}, \mathcal{S}|R) \tag{8}$$

$$M_2 = \text{MBM}(X \cup \{C'\}, \mathcal{S}|R) \tag{9}$$

$$M_{12} = \text{MBM}(X \cup \{C, C'\}, \mathcal{S}|R) \tag{10}$$

We want to show that

$$M_1 - M_0 \geq M_{12} - M_2. \tag{11}$$

Let us denote $M_1 - M_0$ as the LHS and $M_{12} - M_2$ as the RHS for Equation (11). Equation (6) says both the LHS and RHS are nonnegative. Additionally, the LHS is upper bounded by 1, as given any matching $\mathcal{M}$ of size M_1 from $X \cup \{C\}$ to $\mathcal{S}$, we can construct a matching $\mathcal{M}'$ from X to $\mathcal{S}$ of size at least $M_1 - 1$ by removing the unique pair involving C from $\mathcal{M}$, if it exists, which shows $M_0 \geq |\mathcal{M}'| \geq M_1 - 1$. By the same reasoning, the RHS is upper bounded by 1, and indeed we also have $0 \leq M_2 - M_0 \leq 1$. Then since both the LHS and RHS can only take on values 0 or 1, in order to prove equation (11) it suffices to show that if the RHS equals 1, then the LHS equals 1.

Assume the RHS equals 1. Then any maximum matching $\mathcal{M}$ from $X \cup \{C, C'\}$ to $\mathcal{S}$ must include a pair (C, S) for some $S \in \mathcal{S}$, or else $\mathcal{M}$ would also be a valid matching from $X \cup \{C'\}$ to $\mathcal{S}$, which would imply that $M_2 \geq |\mathcal{M}| = M_{12}$, violating the assumption. We split on two cases, which as we noted earlier are exhaustive.

Case: $M_2 - M_0 = 0$. Let $\mathcal{M}_0$ be some maximum matchings from X to $\mathcal{S}$. Since $M_2 = M_0$ by assumption, $\mathcal{M}_0$ is also a maximum matching from $X \cup \{C'\}$ to $\mathcal{S}$. Indeed, $\mathcal{M}_0$ is also a matching from $X \cup \{C, C'\}$ to $\mathcal{S}$, but not a maximum matching, as $M_{12} > M_2$ by assumption. Therefore, by Berge's theorem, there must exist an "augmenting path" P consisting of candidate-slot pairs $(c_1, s_1), \ldots, (c_T, s_T) \in (X \cup \{C, C'\}) \times \mathcal{S}$ for all t such that all pairs are relevant, i.e. $R_{c_t, s_t} = 1$ for all t, and the path starts and ends on unmatched edges and alternates between matched and unmatched edges, i.e. $(c_t, s_t) \notin \mathcal{M}_0$ for all t and $(c_t, s_{t-1}) \in \mathcal{M}_0$ for all $t \in \{2, \ldots, T\}$. But then since P contains one more unmatched than matched edge, "applying" P to $\mathcal{M}_0$ via the set difference operation gives a maximum matching from $X \cup \{C, C'\}$ to $\mathcal{S}$, since $\mathcal{M} := (\mathcal{M}_0 \setminus P) \cup (P \setminus \mathcal{M}_0)$ satisfies $|\mathcal{M}| = M_0 + 1 = M_2 + 1 = M_{12}$. But since $\mathcal{M}$ is a maximum matching from $X \cup \{C, C'\}$ to $\mathcal{S}$, we must have that $\mathcal{M}$ contains candidate C as shown earlier, thus it cannot contain C' by the alternating edges property of P as both C and C' are unmatched in $\mathcal{M}_0$. But then $\mathcal{M}$ is a valid matching from $X \cup \{C\}$ to $\mathcal{S}$, hence $M_1 \geq |\mathcal{M}| = M_{12} = M_0 + 1$ as desired.

Case: $M_2 - M_0 = 1$. Let $\mathcal{M}$ be a maximum matching from $X \cup \{C, C'\}$ to $\mathcal{S}$; then we must have $|\mathcal{M}| = M_{12} = M_0 + 2$ by assumption. If candidate C' does not appear in $\mathcal{M}$, then $\mathcal{M}$ is a valid matching from $X \cup \{C\}$ to $\mathcal{S}$, so $M_1 \geq |\mathcal{M}| = M_2 + 1 \geq M_0 + 1$ and we are done. Otherwise, assume $\mathcal{M}$ includes a pair (C', S') for some $S' \in \mathcal{S}$. But then $\mathcal{M}'' := \mathcal{M} \setminus \{(C', S')\}$ is a valid matching from $X \cup \{C\}$ to $\mathcal{S}$ with $|\mathcal{M}''| = M_{12} - 1 = M_0 + 1$, hence $M_1 \geq M_0 + 1$ as desired. □

The monotone submodularity of $\text{MBM}(X, \mathcal{S}|R)$ implies that our estimate $\hat{M}(X)$ is also monotone submodular, since this property is closed under addition.

Lemma A.2. *For any sample of relevance matrices $\mathcal{R} = [R_1, ..., R_n]$ and set of slots $\mathcal{S}$, $\hat{M}(X)$ from Equation (4) is monotone in X*

$$\forall X \subseteq C, \forall C \in C : \hat{M}(X \cup \{C\}) \geq \hat{M}(X)$$

and also submodular in X, which means that $\forall X \subseteq C, \forall C, C' \in C$

$$\hat{M}(X \cup \{C\}) - \hat{M}(X) \geq \hat{M}(X \cup \{C, C'\}) - \hat{M}(X \cup \{C'\}).$$

Proof. We know from Lemma A.1 that each each $\text{MBM}(X, \mathcal{S}|R_i)$ is monotone and submodular. This means that $\hat{M}(X)$ is a sum of monotone submodular function, and it is well known that sum is monotone submodular as well. □

We are now in a position to complete the proof of Theorem 3.1.

Proof of Theorem 3.1. Our goal is to bound with high probability the suboptimality $M(X_k) - M(X_k^*)$ between the candidate set X_k output by the model and the optimal candidate set X_k^*.

First, note that the approximation guarantee of the greedy algorithm for monotone submodular maximization with a cardinality constraint [25] guarantees

$$\hat{M}(X_k) \geq (1 - 1/e)\hat{M}(\hat{X}_k^*) \geq (1 - 1/e)\hat{M}(X_k^*), \tag{12}$$

where $\hat{X}_k^* = \text{argmax}_{|X|=k} \hat{M}(X)$ is the optimal set on the Monte-Carlo samples. The second inequality follows from $\hat{M}(X_k^*) \leq \hat{M}(\hat{X}_k^*)$.

To get a bound in terms of $M(.)$ instead of $\hat{M}(.)$, we need to bound the error due to Monte-Carlo sampling. We start with the following equivalent expansion of Equation (12).

$$M(X_k) \geq \left(1 - \frac{1}{e}\right)\left(M(X_k^*)\left(\hat{M}(X_k^*) - M(X_k^*)\right)\right) + \left(M(X_k) - \hat{M}(X_k)\right)$$

First, we upper bound $\hat{M}(X_k^*) - M(X_k^*)$, for which we can use Hoeffding's inequality since for any X it holds that $E_R[\hat{M}(X)] =$

Table 4: Selected Labels in Real-world Benchmark Experiments.

Dataset names	Selected label indices	Selected label names
Medical	0, 23, 41, 44, 32, 24, 31, 9, 4, 31	Class-0-593_70, Class-23-786_50, Class-41-591, Class-44-786_07, Class-32-486, Class-24-596_54, Class-31-780_6, Class-9-599_0, Class-4-753_0, Class-31-780_6
Bibtex	44, 134, 63, 10, 14, 104, 131, 52, 117, 83	TAG_electrochemistry, TAG_statphys23, TAG_immunoassay, TAG_apob, TAG_bibteximport, TAG_ontology, TAG_software, TAG_evolution, TAG_requirements, TAG_mathematics
Delicious	924, 452, 809, 99, 941, 733, 540, 897, 946, 700	TAG_video, TAG_howto, TAG_software, TAG_blog, TAG_web, TAG_reference, TAG_linux, TAG_tutorial, TAG_webdesign, TAG_programming
TMC2007	13, 7, 21, 5, 18, 4, 1, 12, 11, 17	class14, class08, class22, class06, class19, class05, class02, class13, class12, class18
Mediamill	78, 24, 84, 94, 65, 96, 51, 2, 67, 66	Class79, Class25, Class85, Class95, Class66, Class97, Class52, Class3, Class68, Class67
Bookmarks	20, 163, 151, 144, 145, 109, 57, 89, 92, 87	TAG_books, TAG_shipyard, TAG_rssfeedek, TAG_recept, TAG_recipe, TAG_medical, TAG_firefox, TAG_journal, TAG_kultur, TAG_java

$M(X)$ and the $MBM(X, \mathcal{S}|R_i)$ ($i \in \{1, \ldots, n\}$) are i.i.d. Monte-Carlo samples with $0 \le MBM(X, \mathcal{S}|R_i) \le s$.

$$P(M(X) - \hat{M}(X) > \epsilon) \le \exp(-2n\epsilon^2/s^2).$$

We thus get for our particular X_k^* that with probability $0 \le \delta_1 \le 1/2$

$$\hat{M}(X_k^*) - M(X_k^*) \le s\sqrt{\frac{\ln(1/\delta_1)}{2n}} := \epsilon_1.$$

Second, we need to upper bound $M(X_k) - \hat{M}(X_k)$. Since X_k is selected on the same Monte-Carlo sample we evaluate it on, we need to ensure uniform convergence over all X. We thus take the union bound over the set $\mathcal{H}_k$ of all possible candidate sets of size k:

$$\begin{aligned} P(\max_X(M(X) - \hat{M}(X)) > \epsilon) &\le |\mathcal{H}_k| \exp(-2n\epsilon^2/s^2) \\ &\le \left(\sqrt{2\pi k}\left(\frac{k}{e}\right)^k e^{\frac{1}{12k}}\right) \exp\left(\frac{-2n\epsilon^2}{s^2}\right) \end{aligned}$$

The second step uses Stirling's inequality, since $|\mathcal{H}_k| = k!$. By letting the final expression equal δ_2 and solving for ϵ, we get that with probability $0 \le \delta_2 \le 1/2$ it holds for all X (and therefore also for any X_k our algorithm picks) that

$$M(X_k) - \hat{M}(X_k) \le s\sqrt{\frac{(k\ln k - k + O(\ln k)) + \ln(1/\delta_2)}{2n}} := \epsilon_2.$$

Putting these bounds together, we get that for all $0 < \delta_1, \delta_2 < 1/2$, we have with probability $1 - \delta_1 - \delta_2$,

$$\begin{aligned} M(X_k) &\ge \left(1 - \frac{1}{e}\right)(M(X_k^*) - \epsilon_1) - \epsilon_2 \\ &\ge \left(1 - \frac{1}{e}\right) M(X_k^*) - (\epsilon_1 + \epsilon_2). \end{aligned}$$

Setting $\delta_1 = \delta_2 = \delta/2$ gives the claimed bound.

□

B SELECTED LABELS IN REAL-WORLD BENCHMARK EXPERIMENTS

We report the selected labels for our real-world benchmark experiments in Table 4. We pick these labels with a consideration of both prediction precision (these labels are not hard to predict by logistic regression models) and sufficient competition among candidates (the positive occurrences of these labels should be larger than the number of slots). We do not change the selected labels when we change the number of slots per label in Table 2.