Approximation Algorithms for Diversified Search Ranking

Nikhil Bansal[1], Kamal Jain[2], Anna Kazeykina[3], and Joseph (Seffi) Naor[4]

[1] IBM Research, Yorktown Heights, NY 10598
nikhil@us.ibm.com
[2] Microsoft Research, Redmond, WA 98052
kamalj@microsoft.com
[3] Moscow State University, Moscow, Russian Federation
filledesoleil@gmail.com
[4] Computer Science Dept., Technion, Haifa, Israel
naor@cs.technion.ac.il

Abstract. In many search query applications, users tend to look at only the top part of the ranked result list in order to find relevant documents. Also, a fundamental issue in Web search is ranking search results based on user logs, since different users may have different preferences and intents with regards to a search query. The setting we consider contains various types of users, each of which is interested in a subset of the search results. The goal is to rank the search results of a query providing highly ranked relevant results. Our performance measure is the *discounted cumulative gain* which offers a graded relevance scale of documents in a search engine result set, and measures the usefulness (gain) of a document based on its position in the result list. We suggest a general approach to developing approximation algorithms for ranking search results based on discounted cumulative gain that captures different aspects of users' intents. We also take into account that the relevance of one document cannot be treated independently of the relevance of other documents in a collection returned by a search engine. For a given search query we assume that there is a set of known types of users, where each user type is satisfied by a certain number of search results. We first consider the scenario where users are interested in only a single search result (navigational queries). We then develop a polynomial time approximation scheme for this case. We further consider the general case where each user type is associated with a different requirement on the number of search results and develop efficient approximation algorithms. Finally, we consider the problem of choosing the top k out of n search results and show that for this problem $1 - 1/e$ is indeed the best approximation factor achievable, thus separating the approximability of the two versions of the problem.

1 Introduction

Satisfying users querying a search engine has become immensely complex given the rapid growth of the Internet, its content diversity, and its usage as a primary

source of information. The satisfaction of users with search results has traditionally been characterized by the notion of *relevance* of a document. Since users tend to look only at the top part of a ranked result list in order to find relevant documents, ranking search results is a fundamental problem in Web search. The goal is to rank the search results of a query providing highly ranked relevant results. A common performance metric for relevance is the *discounted cumulative gain* [8, 11] which offers a graded relevance scale of documents in a search engine result set, and measures the usefulness (gain) of a document based on its position in the result list. The gain is accumulated cumulatively, starting from the top of the result list, with the gain of each result discounted at lower ranks.

Recent approaches to ranking search results have placed more emphasis on inferring relevance and intents of queries from user logs and understanding the distribution of users running a query. One major issue that arises here is that most queries have multiple intents[5], since different users may have different preferences, and therefore a ranking system should provide diverse results that cover a wide spectrum of intents.

It is common practice to partition search queries into three broad categories [11]. The first one is an *informational query*, in which users typically try to assimilate information from multiple web pages. The second one is a *navigational query* where a user looks for a particular (single) website among the search results. The third one is a *transactional query* which is typically a prelude to a transaction on the web. Note that the categories differ (among other things) in the number of search results needed to satisfy a user.

We suggest the following approach for quantifying aggregate user satisfaction. Assume that for a particular search query a set of search results is obtained. There are several user types, corresponding to the multiple intents associated with the query, each of which is a-priori known to be interested in a (known) subset of the results. For each user type we assume that we know the number of relevant search results needed to satisfy it. This number depends, among other things, on the category to which the query belongs. The question is which order should the search results be ranked in, so that all user types are satisfied, and the discounted cumulative gain is maximized. As explained later, this problem turns out to be computationally intractable, and thus we develop approximation algorithms for it.

So far we have made the common assumption that the relevance of one document can be treated independently of the relevance of other documents in the collection returned by a search engine. Carbonell and Goldstein [4] were among the first to note that documents retrieved by a search engine are not necessarily informationally independent and are frequently characterized by a high level of redundancy. Thus, the relevance of a document should be replaced

[5] For example, users searching for "cricket" could be interested in different things: the insect, the sport, the wireless company, ..., etc. Similarly, even for unambiguous queries such as for some researcher's name, say "John Smith", different users might have different intents. Some might be interested in his list of publications, some in his Facebook account, some in his personal web page and so on.

by its *marginal relevance* with respect to other documents in a collection [4, 17, 14]. In [4] it is suggested to maximize marginal relevance (MMR), which is a ranking method that maximizes a linear combination of the relevance of a document and its novelty (a measure of diversity) with respect to already retrieved documents. User studies have shown that MMR is generally preferred to a standard ranking algorithm.

Chen and Karger [6] introduced a probabilistic approach to the relevance problem in which, given a probabilistic metric, the expected value of the metric is optimized. For a variety of metrics used in information retrieval (search length, reciprocal rank, %no, and instance recall), they showed that the optimization problem is computationally intractable. They empirically studied the performance of greedy algorithms and suggested the study of more sophisticated approximation techniques as future work on the topic.

The work of Agrawal et al. [1] is the closest to ours. The authors define their objective function as maximizing the probability of satisfying an average user. They observed that the optimized function is submodular and proposed to use a greedy algorithm to obtain a $(1 - 1/e)$-approximate solution. However, [1] implicitly assumed that users study all the documents returned by a search engine with equal attention, while (as we argued) it is more reasonable to assume the attention decreases while going through the list of results from top to bottom.

1.1 The Model

We assume that there exists a taxonomy of information, and that user intents are modeled at the topical level of this taxonomy. Let us focus on a single search query. Denote the search results by $e_1, \ldots, e_n$ and suppose that there are m user types, or intents. For each user type there is a subset of the search results which are relevant to this type. We model our setting as a hypergraph or an instance of the hitting set problem. The search results correspond to elements $e_1, \ldots, e_n$ in a universe U and each user type corresponds to a set, or a hyperedge, containing the elements in which the user type is interested. The collection of sets is denoted by $\mathcal{S} = \{S_1, \ldots, S_m\}$. For each set S there is a coverage requirement, denoted by $k(S)$, corresponding to the number of relevant search results needed to satisfy user type S. Recall that $k(S) = 1$ corresponds to a navigational query, and $k(S)$ corresponds to an informational query. Given an ordering ϕ of the elements $e_1, \ldots, e_n$, set S is covered at the earliest time (position) $t(S)$ in which $k(S)$ elements from S have already appeared. The goal is to find an ordering maximizing the discounted cumulative gain (DCG):

$$\mathrm{DCG} = \sum_S \frac{1}{\log(t(S) + 1)}.$$

The logarithmic factor in the denominator is to the base e and it is the *discount factor* of the gain of each search result. Intuitively, elements that can provide coverage to many sets should be placed in the beginning of the ordering ϕ so as to maximize the objective function. One can consider more general discounting

functions but the logarithmic discounting seems to be most commonly considered in practice [8, Chapter 8.4.3]. We consider this problem in two settings. First, when there is an bound on the number of elements k that must be chosen. This is motivated by the problem of choosing which query results to display on the first page. Second, when there is no bound on the number of pages to display.

In the case of $k(S) = 1$, we can further consider a more general model when one can model correlations between different search results using information theoretical tools. Given a query q and the set of search results $e_1, e_2, \ldots, e_n$, we define the information function $H_q(e_1, e_2, \ldots, e_n)$ that captures the overall knowledge about q that can be obtained from observing all the search results in the set. For any subset S of search results define $H_q(S)$ to be the function capturing the information about query q contained in S. We assume that $H_q(S)$ has the following entropy properties:

1. $H_q(\emptyset) = 0$;
2. $H_q(S) \leq H_q(T), \quad \forall S \subseteq T$ (*monotonicity*);
3. $H_q(S \cup e) - H_q(S) \geq H_q(T \cup e) - H_q(T), \quad \forall S \subseteq T,\ e \notin T$ (*submodularity*).

Given a ranked list of search results, the *marginal relevance* (MR) of result e_i is defined as

$$\mathrm{MR}(e_i) = H_q(\{e_1, \ldots, e_i\}) - H_q(\{e_1, \ldots, e_{i-1}\}).$$

Since most currently used evaluation metrics in information retrieval make use of relevance labels assigned to search results, we get a natural generalization of these metrics by substituting the relevance of e_i by its marginal relevance $\mathrm{MR}(e_i)$. Then, the generalization of DCG is:

$$\mathrm{GDCG} = \sum_{i=1}^{n} \frac{H_q(\{e_1, \ldots, e_i\}) - H_q(\{e_1, \ldots, e_{i-1}\})}{\log(i+1)}. \tag{1}$$

Note that GDCG captures the model of Agrawal et al. [1], as well as probabilistic ranking models.

1.2 Our Results

We first consider the problem of *choosing* and *ordering* the top k out of n search results so as to maximize the DCG, where k is an input to the problem. We show that for the case in which users are only interested in a single search result, a natural greedy algorithm achieves an approximation factor of $(1 - 1/e) \approx 0.632$. Moreover, we show that this is the best approximation factor achievable, assuming P $\neq$ NP. Our $(1 - 1/e)$ approximation also holds for the more general GDCG measure where each user type can have a monotone submodular utility function (satisfying the properties stated in Section 1.1) over the set of search results. Finally, for the case where the users can be interested in an arbitrary number of search results, i.e. $k(S)$ is arbitrary, we show that the problem is hard in the following sense. It is at least as hard as the notorious Densest-k-Subgraph problem for which the best known approximation is only $O(n^{1/3})$ [10].

Next, we consider the above models in the setting where there is no restriction on the number of search results that we wish to display. Most of our paper is devoted to this setting. In particular, the goal is to simply to find an ordering of all n results maximizing GDCG.

For the case of $k(S) = 1$ (i.e., navigational queries), we develop an approximation algorithm that yields a polynomial-time approximation scheme (ptas). That is, the algorithm achieves a $(1 + \varepsilon)$ approximation given any arbitrarily small constant $\epsilon > 0$, and has running time polynomial in n. This approximation scheme also generalizes to the more general GDCG setting. (I.e., with the Informational function $H_q(S)$.)

For the case of arbitrary $k(S)$, we give a linear programming based $O(\log \log n)$-approximation. We also show how to adapt these ideas to obtain a quasi-polynomial time approximation scheme[6]. The natural linear programming relaxations for this problem have a large integrality gap. We go around this by strengthening the natural relaxation by adding exponentially many so-called *knapsack cover inequalities* [5]. Even though the number of constraints is exponential, it can be solved optimally in polynomial time. Our approximation algorithm is based on rounding these fractional solutions and exploits various properties of the problem structure. While our approach is related to the recent work of [3], there are major differences. In particular, while [3] considers a minimization problem, ours is a maximization problem. This does not seem amenable to a local term-by-term analysis as in [3], and we need a much more complex and global analysis.

Our results can be viewed as showing a strong separation between the two versions of the problem, depending on whether the number of search results to be displayed is fixed or not.

1.3 Related Work

Our modeling of user types as hyperedges with a covering demand follows the work of Azar et al. [2]. They defined a model for re-ranking search results where the goal is to minimize the average time (in a certain sense) of satisfying user types. Their model generalizes classic problems like minimum sum set cover and minimum latency. They developed logarithmic approximation factors for their problem using a linear programming formulation. The approximation factor was improved to a constant by Bansal et al. [3] who strengthened the linear program by adding knapsack cover constraints.

Generating a diverse set of search result is a well studied topic and we mention a few relevant papers. Accounting for different meanings of ambiguous queries and document redundancy was considered by Zhai et al. [17], who proposed a generalization of the classic precision and recall metrics. Ziegler et al. [18] considered the diversification of personalized recommendation lists by reducing the intralist similarity (the sum of pairwise similarities between items in the list). Their user study showed that increased diversity of the recommendation

[6] A $(1 + \epsilon)$-approximation algorithm for any constant $\epsilon > 0$, but with a running time of $n^{\text{polylog}(n)}$.

list improved user satisfaction, but also revealed that human perception can capture the level of diversification inherent to a list only to some extent. Beyond that point, increasing diversity remains unnoticed.

Radlinski and Dumais [14] studied the problem in the context of query reformulations. They proposed to build a diversified set of related queries and then provide the top N results retrieved for each query as an input to a standard ranking algorithm. Relevant studies were also carried out in the domain of online shopping, relational databases [16], and online learning algorithms [15].

In the field of question answering, Clarke et al. [7] developed a probabilistic ranking principle by viewing documents as sets of "information nuggets". They measured the relevance of a document as the probability that it contains at least one relevant and new information nugget. Then, they presented a generalization of the DCG metric and proved that the computation of the ideal gain is NP-hard.

2 Unit Covering Requirements

In this section we consider the case when all the covering requirements $k(S)$ are equal to 1. We will give an approximation scheme for maximizing GDCG. However, it will be easier to describe the scheme in the simpler setting of independent search results where we maximize DCG. The generalization to GDCG is quite straightforward.

2.1 Top k out of n Documents

We begin by describing a simple and efficient greedy algorithm for the problem of maximizing the gain (GDCG) by choosing and ordering k out of n documents. The greedy algorithm simply chooses at each step the element that that maximizes the incremental profit. We show that

Statement 1 *The greedy algorithm for GDCG is a* $(1 - 1/e)$*-approximation.*

Perhaps not too surprisingly, our analysis is similar to that of the classical max k-coverage problem defined as follows: We are given a set of elements $U = \{e_1, \ldots, e_n\}$ and a collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ defined over U. The goal is to choose k elements that cover the maximum number of sets. A set S is said to be covered if some element from S is chosen. This problem is NP-hard and it is well known that the greedy algorithm, that at any point of time chooses the element that covers the maximum number of uncovered sets thus far, gives a $(1 - 1/e)$-approximation.

We prove Statement 1 in Appendix A. It is somewhat surprising that the greedy algorithm still achieves a $(1 - 1/e)$ approximation even in the more general setting of using a submodular information function (e.g. $H_q(S)$) together with a logarithmic discount function, instead of maximizing cardinality. This generalization allows for generalizing the approximation to GDCG.

Finally, in Appendix A, we also show that this result is the best possible unless P=NP.

2.2 No Restriction on the Number of Documents

Here we do not have a restriction on the number of documents we wish to display and the goal is simply to order the pages so as to maximize DCG. A $(1-1/e)$-approximation follows by setting $k = n$ in the result of the previous subsection. Here we describe how to adapt it to obtain a polynomial time approximation scheme (ptas).

The Approximation Scheme. Consider the following algorithm.

- Given the parameter $\epsilon > 0$, we consider all possible candidate choices for the ordering of the first $g(\epsilon) = (2/\epsilon^2)^{(1/\epsilon)}$ elements. There are at most $\binom{n}{g(\epsilon)} g(\epsilon)! \leq n^{g(\epsilon)}$ such choices. Such a choice is determined by the set of elements G and permutation π.
- For each choice G of elements above, let $\mathcal{S}_G$ denote the collection of sets covered by G. Apply the following algorithm to the system $U \setminus G, \mathcal{S} \setminus \mathcal{S}_G$ to obtain the sets $A_0, A_1, \ldots$.
  - Initialize: Let $T_0 = \emptyset$. In general, T_i denotes the sets covered before phase i begins.
  - Phase: For $i = 0, 1, \ldots, \lceil \log n \rceil$, apply the following steps.
    1. Consider the max k-coverage instance defined on the universe $U \setminus G$ and the set collection $(\mathcal{S} \setminus \mathcal{S}_G) \setminus T_i$. Let $k = 2^i$.
    2. Apply the greedy algorithm to the instance above and let A_i denote the $k = 2^i$ elements chosen.
    3. Let $\mathcal{S}_i$ denote the sets in $\mathcal{S} \setminus T_i$ that are covered by elements in A_i. Set $T_{i+1} = T_i \cup \mathcal{S}_i$.
  - Output: Construct the final ordering by concatenating the lists $\pi(G), A_0, A_1, \ldots$ in that order. Within the set A_i the elements can be ordered arbitrarily.
- Over all choices of G and π, output the ordering with the maximum gain.

Analysis. Consider the case when G consists of the first $g(\epsilon)$ elements in the optimum ordering Π_{opt}, and the elements in G are also ordered as in Π_{opt}. Let alg_1 and opt_1 denote the gain obtained by the first $g(\epsilon)$ elements in the respective orderings and let $\mathsf{alg}_2 = \mathsf{alg} - \mathsf{alg}_1$ and $\mathsf{opt}_2 = \mathsf{opt} - \mathsf{opt}_1$ denote the respective gain from the remaining elements (that lie after the first $g(\epsilon)$ elements).

Theorem 1. *The above scheme is a polynomial-time approximation scheme.*

Proof: Clearly $\mathsf{alg}_1 = \mathsf{opt}_1$. We will show that $\mathsf{alg}_2 \geq (1-\epsilon)\mathsf{opt}_2$. Let O_i denote the sets covered by elements that appear in Π_{opt} in positions $[2^i, 2^{i+1} - 1]$ after the first $g(\epsilon)$ elements (i.e. in absolute positions $[g(\epsilon) + 2^i, g(\epsilon) + 2^{i+1} - 1]$). Similarly, let T_{i+1} denote the sets covered by elements $A_0 \cup \ldots, A_i$. We claim the following:

Claim. For any $0 \leq i \leq n$, after phase i, the number of sets covered by the algorithm, i.e. $|T_{i+1}|$, satisfies

$$|T_{i+1}| \geq \sum_{j=0}^{i} (1 - e^{j-i}) |O_j|. \tag{2}$$

We move the proof of this claim to Appendix B. Given this claim,

$$\begin{aligned}
\mathsf{alg}_2 &\geq \sum_{i=0}^{\infty} \frac{(|T_{i+1}| - |T_i|)}{\log(g(\epsilon) + 2^{i+1})} \\
&= \sum_{i=0}^{\infty} \left(\frac{1}{\log(g(\epsilon) + 2^{i+1})} - \frac{1}{\log(g(\epsilon) + 2^{i+2})} \right) |T_{i+1}| \\
&\geq \sum_{i=0}^{\infty} \left(\frac{1}{\log(g(\epsilon) + 2^{i+1})} - \frac{1}{\log(g(\epsilon) + 2^{i+2})} \right) \cdot \sum_{j=0}^{i} (1 - e^{j-i}) |O_j| \\
&= \sum_{j=0}^{\infty} \sum_{i=j}^{\infty} \left(1 - e^{j-i}\right) \cdot \left(\frac{1}{\log(g(\epsilon) + 2^{i+1})} - \frac{1}{\log(g(\epsilon) + 2^{i+2})} \right) |O_j| \\
&\geq \sum_{j=0}^{\infty} \sum_{i=j+2\ln(1/\epsilon)}^{\infty} (1 - \epsilon^2) \cdot \left(\frac{1}{\log(g(\epsilon) + 2^{i+1})} - \frac{1}{\log(g(\epsilon) + 2^{i+2})} \right) |O_j| \\
&= \sum_{j=0}^{\infty} (1 - \epsilon^2) \frac{1}{\log(g(\epsilon) + (2^{j+1}/\epsilon^2))} |O_j|. \qquad (3)
\end{aligned}$$

The last step follows since the terms are telescoping.

To complete the proof, we note that the contribution of O_j to the optimum ordering is at most $|O_j|/(\log(g(\epsilon) + 2^i + 1))$. On the other hand, by (3) the contribution of O_j to alg_2 is at least:

$$\begin{aligned}
&(1 - \epsilon^2)(1/\log(g(\epsilon) + 2^{j+1}/\epsilon^2))|O_j| \;\geq\; (1 - \epsilon^2)(1/\log((g(\epsilon) + 2^j)(2/\epsilon^2))|O_j| \\
&= (1 - \epsilon^2) \frac{1}{\log(g(\epsilon) + 2^j) + \log(2/\epsilon^2)} |O_j| \;\geq\; (1 - \epsilon) \frac{1}{\log(g(\epsilon) + 2^j)} |O_j|.
\end{aligned}$$

The last step follows as $\log(2/\epsilon^2) \leq \epsilon \log(g(\epsilon)) \leq \epsilon \log(g(\epsilon) + 2^j)$ by our choice of $g(\epsilon)$. ■

The algorithm has a huge running time of $n^{2^{O(1/\epsilon)}}$, but this is still a polynomial-time approximation scheme. It would be an interesting open problem to see if the dependence of the running time on ϵ can be made to be singly exponential.

3 General Requirements

In this section we consider the scenario when $k(S)$ can be arbitrary. We first consider the setting where only k documents can be displayed, and show that the problem is extremely hard even for very special cases of the problem. We then consider the case when there is no bound on the number of documents that can be displayed. We describe an $O(\log \log n)$-approximation for maximizing DCG in this setting. Our algorithm is based on an LP formulation using knapsack cover inequalities. Finally, we will show how to adapt the algorithm to obtain a quasi-polynomial time approximation scheme.

3.1 Top k out of n Documents

We show that in this setting, the DCG problem already captures the notoriously hard densest k-subgraph problem. In the densest k-subgraph problem, we are given an undirected graph $G = (V, E)$ and the goal is to find a subset of k vertices V' that maximizes the number of induced edges (i.e. edges that have both end points in V'). The best known algorithm for the densest k-subgraph problem essentially achieves a guarantee of only $n^{1/3}$ [10]. It is an outstanding open question whether this bound can be improved. To see the connection to Densest-k-Subgraph, consider the following for a given graph $G = (V, E)$. Define the elements as the vertices V and sets S as the edges E, where each S has a covering requirement of $k(S) = 2$. Clearly, finding k elements that cover the maximum number of sets for this instance is equivalent to solving the densest k-subgraph problem. Since the $1/\log(t+1)$ term in the objective function of the DCG is bounded between 1 and $\log(k+1)$ this implies that the problem is as hard to approximate as the Densest-k-Subgraph problem within an $O(\log(k))$ factor. Moreover, note that this is a very special case of our problem, since $k(S) = 2$ for all sets, and each set has size 2.

3.2 No Restriction on the Number of Documents

Given the discussion above, we note that the approach taken in the previous section of solving the max k-coverage problem for geometrically increasing values of k, and then concatenating the resulting sets, will yield a very poor approximation. Given this limitation, our approximation algorithm will use quite different ideas which are based on a linear programming (LP) formulation of the problem. In particular, we will strongly use the fact that we do not have a bound on the number of elements we need to choose (i.e., we can avoid the max k-coverage approach), and also exploit the specific nature of our objective function.

Our approach is based on the recent work of [3], however, there are crucial differences. Most notably, [3] considers a minimization problem, while here we are interested in a maximization problem. Our LP formulation, described below, is based on *knapsack cover inequalities* first introduced by [5]. It is known that a naive LP relaxation for our problem that does not use these inequalities can have a very large integrality gap (we defer this discussion here, and refer the interested reader to [3]).

An LP Relaxation. Let $[n] = \{1, 2, \ldots, n\}$, where $n = |U|$, the number of elements in the universe. In the following, x_{et} is the indicator variable for whether element $e \in U$ is selected at time $t \in [n]$, and y_{St} is the indicator variable for whether set S has been covered by time $t \in [n]$.

$$\begin{aligned} \text{Maximize} \quad & \sum_{1 \le t \le |U|} \sum_{S \in \mathcal{S}} (y_{S,t} - y_{S,t-1}) / \log(t+1) \\ \text{subject to} \quad & \sum_{e \in U} x_{et} = 1 \qquad \forall\, t \in [n] \end{aligned} \tag{4}$$

$$\sum_{t\in[n]} x_{et} = 1 \qquad \forall\, e \in U \tag{5}$$

$$\sum_{e\in S\setminus A} \sum_{t'<t} x_{et'} \geq (k(S) - |A|) \cdot y_{St} \tag{6}$$

$$\forall\, S \in \mathcal{S}, \forall\, A \subseteq S, \forall t \in [n]$$

$$x_{et}, y_{St} \in [0,1] \qquad \forall\, e \in U,\, S \in \mathcal{S},\, t \in [n]$$

If x_{et} and y_{St} are restricted to take $\{0,1\}$ values, then this is easily seen to be a valid formulation for the problem. Constraints (4) require that only one element can be assigned to a time slot and constraints (5) require that each element must be assigned to some time slot. Constraints (6) correspond to the knapsack cover inequalities and require that if $y_{St} = 1$, then for every subset of elements A, at least $k(S) - |A|$ elements must be chosen from $S \setminus A$ before time t.

The Algorithm. Let (x^*, y^*) denote an optimal (fractional) solution to the above linear programming formulation and let opt denote its value. Clearly, opt is an upper bound on an integral optimal solution. Our rounding algorithm proceeds in $O(\log n)$ stages, with the i^{th} stage operating in the time interval $[1, 2^{i+1} - 1]$. In stage i, for $i = 0, 1, \ldots$, we perform one round of randomized rounding (as described below) on the fractional solution restricted to the interval $[1, 2^{i+1} - 1]$ and obtain a set A_i of elements. At the conclusion of these stages, we output the elements of A_0, followed by the elements of A_1, and then $A_2, \ldots,$, with the elements of any set A_j being output in an arbitrary order.

The rounding process for stage i that generates set A_i is the following:

- Let $t_i = 2^i$.
- Define the fractional extent to which e is selected before time t_i, for each $e \in U$, to be: $z_{e,i} \leftarrow \sum_{t' \leq t_i} x^*_{et'}$.
- Define for all $e \in U$, $p_{e,i} \leftarrow \min(1, 8\log^2(n+1) z_{e,i})$.
- Pick each element $e \in U$ independently with probability $p_{e,i}$. Let A_i be the set of these elements. If $A_i > 16\log^2(n+1) \cdot 2^i$, then set $A_i = \emptyset$.

Analysis. In the solution (x^*, y^*), for each set S, let $t^*(S)$ denote the earliest time that S has been allocated to an extent of $1/\log(n+1)$, i.e., $y_{t^*(S),S} \geq 1/\log(n+1)$. The next lemma shows that $t^*(S)$ is a good estimate for the time when a set is covered.

Lemma 1. $\mathsf{opt} \leq \sum_S 2/\log(t^*(S)+1)$.

Proof: Since any set S is covered to an extent of at most $1/\log(n+1)$ by time $t^*(S)$, this fraction can yield a gain of at most $(1/\log(n+1)) \cdot (1/\log 2) \leq 1/\log(n+1)$. This is at most $(1/\log(t^*(S)+1))$, since $t^*(S) \leq n$ (as each set is covered by time n). The remaining $1 - 1/\log(n+1)$ portion can yield a total gain of at most $(1-1/\log(n+1)) \cdot (1/\log(t^*(S)+1)) \leq (1/\log(t^*(S)+1))$. Thus, the total gain is at most $2/\log(t^*(S)+1)$. ■

Lemma 2. *For any set S and any stage i such that $t^i \geq t^*_S$, the probability that $k(S)$ elements from S are not picked in stage i is at most $1 - 2/n^2$.*

Due to space constraints, we move the proof to Appendix C.

Theorem 2. *The algorithm above is an $O(\log \log n)$-approximation.*

Proof: Since $|A_i| \leq 16 \log^2(n+1) 2^i$, each element in A_i appears no later than time $16 \log^2(n+1)(2^{i+1} - 1)$ in the ordering produced by the algorithm. By Lemma 2, with probability at least $1 - 2/n^2$, each set S appears in a set A_i where $i \in [t^*(S), 2t^*(S) - 1]$. Thus, with probability $1 - 2/n^2$, set S appears in the ordering of the algorithm by time $64 \log^2(n+1) t^*(S) - 1$.

Thus, the expected contribution of set S to the algorithm is at least

$$\left(1 - \frac{1}{n^2}\right) \cdot \left(\frac{t^*(S)}{\log 64 \log^2(n+1)}\right) = \Omega\left(\frac{1}{\log \log n} \cdot \log(t^* + 1)\right).$$

By Lemma 1, opt can obtain a gain of at most $2/(\log(t^* + 1))$, and hence we get an $O(\log \log n)$-approximation. ■

Next, we show that the algorithm can be modified to obtain a quasi-polynomial time approximation scheme. That is, a $(1+\epsilon)$-approximation for any $\epsilon > 0$, but with running time $O(n^{\text{polylog}(n)})$.

A Quasi-Polynomial Time Approximation Scheme. We modify the algorithm above to first guess (by trying out all possibilities) the first $f(\epsilon) = (\log n)^{4/\epsilon}$ elements in the ordering of opt. Let us assume that $n \geq 1/\epsilon \geq 8$ (otherwise the problem can be solved trivially in $O(1)$ time).

Given the optimum LP solution, let us define $t^*(S)$ to be the earliest time when set S is covered to extent $\epsilon / \log n$. A slight modification of Lemma 1 implies that $\mathsf{opt} \leq (1+\epsilon)/\log(t^* + 1)$.

The analysis of our rounding algorithm guarantees that the expected contribution of set S to the algorithm is at least

$$\left(1 - \frac{1}{n^2}\right) \cdot \left(\frac{t^*(S)}{\log 64 \log^2(n+1)}\right) \geq \frac{1 - \epsilon^2}{\log t^*(S) + 4(\log \log(n+1)},$$

since $n \geq 1/\epsilon \geq 8$. If $\log t^*(S) \leq (4/\epsilon) \log \log n)$, or equivalently $t^*(S) \leq \log n^{4/\epsilon}$, the contribution of S to the algorithm is at least $(1 - \epsilon)$ times its contribution to opt. Thus, we can apply the same idea as the one used for obtaining a ptas for the case of $k(S) = 1$, and guess the first $f(\epsilon) = (\log n)^{4/\epsilon}$ positions in the ordering of opt.

4 Conclusions

A natural question is to consider more general forms of discounting in the definition of DCG, beyond just $\gamma(t) = 1/\log(t+1)$. In [8, Chapter 8.4.3] it is

mentioned that the choice of the logarithmic discount function is somewhat arbitrary and has no theoretical justification, although it does provide a relatively smooth (gradual) reduction. It is easily seen that our $(1 - 1/e)$-approximation for unit covering requirements works for any monotonically decreasing $\gamma(t)$. In the case when there is no limit on the number of documents returned, our ptas can also be extended to the case where $\gamma(t)$ satisfies the following property. For any ϵ, the inequality $\gamma(2t) \geq \gamma(t)/(1+\epsilon)$ holds for all but $\ell = O_\epsilon(1)$ integers t. In this case our algorithm has running time $n^{O(\ell)}$. Note that for the logarithmic function, $\ell = 2^{O(1/\epsilon)}$. The above condition is satisfied for any $\gamma(t)$ of the type $1/\text{polylog}(t)$, but it is not satisfied for a polynomial such as $f(t) = 1/t^\delta$.

References

1. R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM'09*, pages 5–14, 2009.
2. Y. Azar, I. Gamzu, and X. Yin. Multiple intents re-ranking. In *STOC '09: Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 669–678, New York, NY, USA, 2009. ACM.
3. N. Bansal, A. Gupta, and R. Krishnaswamy. A constant factor approximation algorithm for generalized min-sum set cover. In *Symposium on Discrete Algorithms (SODA)*, 2010.
4. J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR'98*, pages 335–336, 1998.
5. R. Carr, L. Fleischer, V. Leung, and C. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Symposium on Discrete Algorithms (SODA)*, pages 106–115, 2000.
6. H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *SIGIR'06*, pages 429–436, 2006.
7. C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Bttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR'08*, pages 659–666, 2008.
8. B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesley, 2009.
9. U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.
10. U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.
11. C. D. Manning, P. Raghavan, and H. Schuetze. *Introduction to Information Retrieval*. Cambridge University Press, New York, 2008.
12. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
13. G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.
14. F. Radlinski and S. T. Dumais. Improving personalized web search using result diversification. In *SIGIR'06*, pages 691–692, 2006.
15. F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *ICML'08*, pages 784–791, 2008.

16. E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. A. Yahia. Efficient computation of diverse query results. In *ICDE'08*, pages 228–236, 2008.
17. C. Zhai, W. W. Cohen, and J. D. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR'03*, pages 10–17, 2003.
18. C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW'05*, pages 22–32, 2005.

A Proofs in Section 2.1

We prove statement 1.

Statement 1. The greedy algorithm for GDCG is a $(1 - 1/e)$-approximation.

Proof: We define $(d_1^*, \ldots, d_k^*)$ to be the optimal solution of the GDCG problem, $(d_1^G, \ldots, d_k^G)$ to be the solution returned by the greedy algorithm and $(\bar{d}_1^i, \ldots, \bar{d}_i^i)$ to be the set of elements which maximizes $H_q(S)$ over all sets S such that $|S| = i$.

Note that we can rewrite (1) in the form $\sum_{i=1}^{\infty} \alpha_i H_q(\{d_1 \cup d_i\})$, where the α_i-s are certain coefficients. In [13] it is shown that for all i

$$H_q(\{d_1^G, \ldots, d_i^G\}) \geq (1 - 1/e) H_q(\bar{d}_1^i, \ldots, \bar{d}_i^i).$$

It follows then that

$$\begin{aligned}\sum_{i=1}^{k} \alpha_i H_q(\{d_1^G \cup \ldots \cup d_i^G\}) &\geq (1 - 1/e) \sum_{i=1}^{k} \alpha_i H_q(\{\bar{d}_1^i \cup \ldots \cup \bar{d}_i^i\}) \\ &\geq (1 - 1/e) \sum_{i=1}^{k} \alpha_i H_q(\{d_1^* \cup \ldots \cup d_i^*\}).\end{aligned}$$

■

A.1 Hardness of DCG

Theorem 3. *We show that assuming $P \neq NP$, the DCG problem is NP-hard hard to approximate within a factor of $1 - 1/e - \epsilon$ for any $\epsilon > 0$. In fact, the hard instance is that of the single query case (in the version where we can only choose k out of n documents).*

Proof: Let $S_1, \ldots, S_m$ be a collection of sets defined over the universe $U = \{e_1, \ldots, e_n\}$. The goal is to choose k elements in U that maximize the DCG function $\sum_{j=1}^{m} 1/\log(t(S_j) + 1)$, where $t(S_j)$ is the time when S_j is covered ($S_j = \infty$, if it is not covered by the chosen k elements).

We use the following class of hard max k-coverage instances from Feige [9]. Given any $\epsilon > 0$, there exists m, k, and a hitting set instance on m sets, where each element hits exactly m/k sets, and it is NP-hard to distinguish between the following two cases:

– *Good case:* There exist k elements such that each element covers m/k disjoint sets; or
– *Bad case:* No k elements can cover even $m(1 - /e + \epsilon)$ sets.

Note that in the good case, the optimum solution has DCG at least $m/\log(k+1)$ since each set can be covered by time k. On the other hand, suppose that there is a solution to DCG with value

$$m(1 + (10/\log(1/\epsilon)))(1 - 1/e)/\log(k+1).$$

We claim that in this case the max k-coverage instance has a solution of value $(1-1/e+5\epsilon)m$, implying that we are not in the bad case. As $(1+(10/\log(1/\epsilon)))$ approaches 1 as ϵ becomes arbitrarily small, this would imply the claimed $(1 - 1/e)$ hardness for DCG.

Since each element can cover at most m/k sets, the DCG contribution until time ϵk can be at most

$$\sum_{j=1}^{\epsilon k} \frac{m}{k} \frac{1}{\log(j+1)} \leq \frac{\epsilon}{m} \left(\frac{1}{\log(\epsilon k)} + \frac{1}{2\log^2(\epsilon k)} \right) \leq 2\epsilon m \frac{1}{\log(k+1)}$$

The last step follows as $k \gg 1/\epsilon$. Thus, at least $m(1+10/\log(1/\epsilon))(1-1/e)/\log(k+1) - 2\epsilon m/\log(k+1) \geq m(1 - 1/e + 5/\log(1/\epsilon))/\log(k+1)$ contribution to DCG must come from sets covered during $[\epsilon k, k]$. However, each such set can contribute at most $1/(\log(\epsilon k)) \leq (1 + 2/\log(1/\epsilon))/\log(k+1)$. Thus, at least $m(1 - 1/e + 2/\log(1/\epsilon) \geq m(1 - 1/e + 2\epsilon)$ sets must be covered during $[\epsilon k, k]$, implying the claim. ■

B Proof of Claim 2.2

Claim 2.2. For any $0 \leq i \leq n$, after phase i, the number of sets covered by the algorithm, i.e. $|T_{i+1}|$, satisfies

$$|T_{i+1}| \geq \sum_{j=0}^{i} (1 - e^{j-i})|O_j|. \tag{7}$$

Proof: We prove the lemma by induction. For $i = 0$, the inequality is trivially true as the right hand side is 0. For $i = 1$, the inequality requires that $|T_2| \geq (1 - 1/e)|O_0|$. By definition, $|T_2| = |\mathcal{S}_0| + |\mathcal{S}_1|$. We will in fact show that $|\mathcal{S}_0| \geq |O_0|$. This happens since $\mathcal{S}_0$ is exactly the number of sets satisfied at time 1 by the greedy algorithm, and hence $|S_0|$ is the maximum number of sets any element can satisfy. On the other hand, $|O_0|$ is also the number of sets that are satisfied by the first element in Π_{opt}. Let us assume that

$$|T_{i+1}| \geq \sum_{j=0}^{i} (1 - e^{j-i})|O_j|$$

holds for $i \geq 1$. We show that it also holds for $i+1$, which will imply the result by induction. Let $Q_i = O_0 \cup \ldots \cup O_i$ denote the union of all the sets covered by the first $2^{i+1}-1$ elements in the ordering Π_{opt}. Note that $|Q_i| = \sum_{j=0}^{i} |O_j|$.

Now, A_{i+1} is obtained by applying the greedy algorithm for the max k-coverage problem with $k = 2^{i+1}$ on the collection of sets $S \setminus T_{i+1}$. The optimum solution to this particular max k-coverage problem has value at least $|Q_i \setminus T_{i+1}| \geq |Q_i| - |T_{i+1}|$, since one particular solution is the first $2^{i+1}-1$ elements of Π_{opt}. By the guarantee for the greedy algorithm for max k-coverage, it follows that

$$|S_{i+1}| \geq \left(1 - \frac{1}{e}\right) \cdot \max(0, |Q_i| - |T_{i+1}|). \tag{8}$$

Thus, we have,

$$\begin{aligned}
|T_{i+2}| = |S_{i+1}| + |T_{i+1}| \;&\geq\; \left(1 - \frac{1}{e}\right)|Q_i| + \frac{1}{e}|T_{i+1}| \\
&= \left(1 - \frac{1}{e}\right)\left(\sum_{j=0}^{i} |O_j|\right) + \left(\frac{1}{e}\right)|T_{i+1}| \\
&\geq \left(1 - \frac{1}{e}\right)\left(\sum_{j=0}^{i} |O_j|\right) + \sum_{j=0}^{i}\left(\frac{1}{e}\right)\left(1 - e^{j-i}\right)|O_j| \\
&= \sum_{j=0}^{i+1}\left(1 - e^{j-(i+1)}\right)|O_j|.
\end{aligned}$$

Here, the second step follows from (8), third step follows from the definition of Q_i, and the fourth step follows from the inductive assumption for i, implying the claimed result. ■

C Proof of Lemma 2

Lemma 2. For any set S and any stage i such that $t^i \geq t^*_S$, the probability that $k(S)$ elements from S are not picked in stage i is at most $1 - 2/n^2$.

Proof: Consider any set S, and let $S_g = \{e \in S \mid z_{e,i} \geq 1/8\log^2(n+1)\}$. By the choice of $p_{e,i}$ in the rounding procedure, all elements in S_g are definitely marked in stage i, and any element $e \in S \setminus S_g$ is independently marked with probability $8\log^2(n+1)z_{e,i}$. Thus, if $|S_g| \geq k(S)$, then clearly the lemma holds.

Hence, we consider the case when $|S_g| < k(S)$. Recall that we are considering a set S and stage i such that $t^*_S \in [1, t_i]$; since t^*_S was the last time t at which $y^*_{S,t} \leq \frac{1}{\log(n+1)}$, it follows that $y^*_{S,t_i} > \frac{1}{\log(n+1)}$. Hence, setting $A = S_g$, constraint (6) implies that

$$\sum_{e \in S \setminus S_g} z_{e,i} = \sum_{e \in S \setminus S_g} \sum_{t' \leq t_i} x^*_{et'} \geq (k(S) - |S_g|) \cdot y^*_{St_i} \geq \frac{1}{\log(n+1)}(k(S) - |S_g|).$$

Therefore, the expected number of elements from $S \setminus S_g$ marked in stage i is

$$\sum_{e \in S \setminus S_g} 8 \log^2(n+1) z_{e,i} \geq 8 \log(n+1)(k(S) - |S_g|).$$

Since these elements are marked independently of each other, we can use the following Chernoff bound [12] (Theorem 4.2): if $X_1, X_2, \ldots, X_n$ are independent $\{0,1\}$-valued random variables with $X = \sum_i X_i$, such that $E[X] = \mu$, then

$$\Pr[X < \mu(1-\beta)] \leq \exp\left(-\frac{\beta^2}{2}\mu\right).$$

For our application, since we have $\mu \geq 8\log(n+1)(k(S)-|S_g|) \geq 8\log(n+1)$, we can substitute $\beta = \frac{3}{4}$ and bound the tail probability that fewer than $(k(S)-|S_g|)$ elements are marked from $S \setminus S_g$ by

$$\exp\left(-\frac{(3/4)^2}{2} \cdot 8\ln(n+1)\right) \leq 1/n^2.$$

As the elements in S_g are always picked (unless $A_i = \emptyset$), it follows that the probability that fewer than $k(S)$ elements are marked from S is also at most $1/n^2$. Finally, by standard Chernoff bounds, the probability that A_i is set to $\emptyset$ in the last step of the algorithm is at most $1/n^2$. ■