Predictive Inference for Learning to Rank

Chirag Gupta

October 7, 2020

Abstract

Ranking is a fundamental problem in Information Retrieval. Given a query and a set of potential documents from a larger corpus, the goal is to rank these documents in decreasing order of relevance. A variety of techniques have been studied in classical machine learning settings where a score function is learnt that produces a relevance value for every query-document pair. The documents are then returned in decreasing score as per this function. However, what has been missing in this literature is an attached notion of confidence in the predicted ranking. If such a confidence score were available, it can be used to make high-impact decisions such as if the number of documents shown on the first page of results should be changed. We consider this problem of providing such an uncertainty estimate. Our technique would provide a set of rankings such that with a fixed predefined probability (for example 0.9), at least one of them will be a 'good' ranking as per some standard metrics. Our goal will be to have sets that have interpretability so that they can be acted upon in meaningful ways.

1 Introduction to ranking

Ranking data refers to a set of samples from a distribution Q over $\mathbb{R}^{d,k} \times \mathbb{R}^k$ where d is a fixed natural number and k is a random natural number. A sample from Q is of the form $((f_1, r_1), (f_2, r_2), \ldots (f_k, r_k))$ where each $f_i \in \mathbb{R}^d$ refers to some featurization of a query-document pair (q, d_i) and $r_i \in \mathbb{R}$ refers to a relevance score. Notice that for every sample, the queries are fixed and we have multiple potential documents. We don't assume that the (f_i, r_i)'s are independent of each other, even conditioned on k. The potential set of documents is formed using some underlying algorithm that will be treated as a blackbox. The goal is to use $(f_1, \ldots, f_k)$ to make inference on $(r_1, \ldots, r_k)$. Often, we are interested only in inferring a ranking of the documents $\pi \in S_k$ such that $r_{\pi(1)} \geq r_{\pi(2)} \geq \ldots r_{\pi(k)}$.

Since it is hard to identify the true ranking completely, we consider measures of how good a learnt ranking is. Suppose $r_1 > r_2 > r_3$. An algorithm that returns $\pi(1, 2, 3) = (3, 1, 2)$ is clearly better than one that returns $\pi(1, 2, 3) = (3, 2, 1)$. However incorrect rankings may often not be comparable in this sense. For example if $r_1 > r_2 > r_3 > r_4$, which one of $\pi(1, 2, 3, 4) = (2, 1, 4, 3)$ and $\pi(1, 2, 3, 4) = (1, 3, 2, 4)$ is better? While the latter ranking has only one flipped set of documents, the former gets the top two right (we often value the rank of the top results more than the rest). A number of standard domain dependent ranking measures have been developed to measure how well a ranking does with respect a given a set of relevance scores.

1.1 Ranking metrics

We introduce metrics that measure how good a given ranking does with respect to the ground truth relevance scores. These have been used for benchmarking in the development of many ranking algorithms [Tax et al., 2015, Cao et al., 2007, Burges et al., 2005, Freund et al., 2003, Järvelin and Kekäläinen, 2017, Xu and Li, 2007, Burges et al., 2007, Kuo et al., 2009]:

- CG@k (Cumulative Gain at level k) refers to the total sum of relevances of the top k documents in the

returned ranking:

$$\mathrm{CG}(k) = \sum_{i=1}^{k} r_{\pi(i)}.$$

- DCG@k (Discounted Cumulative Gain at level k) discounts the CG so as to value higher ranked documents more:

$$\mathrm{DCG}(k) = \sum_{i=1}^{k} \frac{2^{r_{\pi(i)}} - 1}{\log_2(i+1)}.$$

- NDCG@k (Normalized Discounted Cumulative Gain at level k) normalized the DCG with the Ideal DCG (IDCG):

$$\mathrm{NDCG}(k) = \frac{\mathrm{DCG}(k)}{\mathrm{IDCG}(k)},$$

  where $\mathrm{IDCG}(k)$ is the $\mathrm{DCG}(k)$ of an optimal ranking.

- P@k (Precision at level k) measures the number of relevant documents in the top k returned documents (here $r_i \in \{0, 1\}$ so that documents are either relevant or irrelavant). It is equal to CG@k but is used only in the context of binary relevance scores.

- MAP (Mean Average Precision) is a way of aggregating Prec@k at different levels:

$$\mathrm{AP} = \frac{\sum_{k=1}^{\#\mathrm{docs}} (P(k)/k) \cdot r_{\pi(k)}}{\sum_{k=1}^{\#\mathrm{docs}} r_{\pi(k)}}.$$

  MAP is the mean value of AP across all queries.

Next, we introduce some machine learning techniques that have been used to learn models that optimize these ranking measures. All of these techniques have been used to predict a single ranking that does well according to one of the metrics above. Later we discuss why these point-predictors are unsatisfactory if the goal is to perform predictive inference.

1.2 Learning point-predictors that optimize ranking metrics

Several empirical predictors have been proposed over the years. Some of the methods that have been discussed in contemporary studies and perform well are ListNet [Cao et al., 2007], RankNet [Burges et al., 2005], LambdaRank [Burges et al., 2007, Burges], AdaRank [Xu and Li, 2007], RankSVM [Joachims, 2002] and BayesRank [Kuo et al., 2009], FRank [Tsai et al., 2007].

We will discuss three techniques that seem to be performing well across datasets [Tax et al., 2015]: LambdaRank, ListNet and BayesRank. Each of these depend on a model class such as neural networks or SVMs whose parameters $\mathbf{w}$ we learn on the training data. A model represents a class of functions $h_{\mathbf{w}} : \mathbb{R}^d \to \mathbb{R}$ that produces a relevance value given a query-document feature. To predict a rank using these models, we then rank all documents for a query in decreasing order of score. The techniques differ in how to train this model class.

- LambdaRank: We will learn $\mathbf{w}$ by a gradient descent like update. Denote an element of $\mathbf{w}$ by w and consider a randomly picked query q along with its documents $\{f_i\}_{i=1}^{k}$ (in practice we average over a mini-batch of the queries). Then we update

$$w \to w - \eta \sum_{i=1}^{k} \lambda_i \frac{\partial h_{\mathbf{w}}(f_i)}{\partial w}$$

  where for every document λ_i is defined as follows. We first identify a set $I \subset [k] \times [k]$ such that for every $(i, j) \in I$ we have that $h_{\mathbf{w}}(f_i) > h_{\mathbf{w}}(f_j)$ but $r_i < r_j$. We then compute for every such pair

of documents the positive change in some score M (from Section 1.1) if they are swapped: $\Delta_M(i,j)$. Then

$$\lambda_i = -\sum_{(i,j)\in I} \frac{\sigma\Delta_M(i,j)}{1+\exp\left(\sigma(h_{\mathbf{w}}(f_i)-h_{\mathbf{w}}(f_j))\right)} + \sum_{(j,i)\in I} \frac{\sigma\Delta_M(j,i)}{1+\exp\left(\sigma(h_{\mathbf{w}}(f_j)-h_{\mathbf{w}}(f_i))\right)}.$$

- ListNet: This algorithm seems to work quite well generally across datasets. ListNet first models the probability of a document being ranked as the top document for a query using a model $h_{\mathbf{w}}$. For some document feature f_i,

$$p(f_i|h_{\mathbf{w}}) = \frac{\phi(h_{\mathbf{w}}(f_i))}{\sum_{j=i}^{n}\phi(h_{\mathbf{w}}(f_j))},$$

  where $\phi(\cdot) = \exp(\cdot)$. They then optimize the empirical cross-entropy loss with respect to the true relevance function r:

$$\underset{\mathbf{w}}{\text{minimize}} \sum_{\text{queries}} \left(-\sum_{i=1}^{k} p(f|r)\log p(f|h_{\mathbf{w}})\right).$$

- BayesNet: BayesNet uses a more refined probability model for the probability of the entire ranking:

$$p_f(\pi|f,h_{\mathbf{w}}) = \prod_{i=1}^{k} \frac{\phi(h_{\mathbf{w}}(f_{\pi(i)}))}{\sum_{j=i}^{n}\phi(h_{\mathbf{w}}(f_{\pi(j)}))}.$$

  We then minimize a decision theoretic notion of risk with respect to one of the metrics M (introduced in Section 1.1) of predicting a ranking σ given the true relevance scores r:

$$\underset{\mathbf{w}}{\text{minimize}} \sum_{\text{queries}} \left(-\sum_{\pi\in S_k} p_f(\pi|f,h_{\mathbf{w}})M(r,\pi)\right).$$

  Note that the gradient here only needs a derivative of the probability model and not the metric M which is often non-differentiable.

While the methods above lead to probability values for rankings as per a probabilistic model, these probability values do not reflect true probabilities without the strong distributional assumptions that the modeling makes. Thus they cannot be used for valid finite-sample predictive inference on the ranking of documents. After we introduce the dataset of interest in Section 2 we will then discuss the method of conformal inference that would enable us to perform predictive inference on rankings.

2 LETOR 4.0 (the dataset)

LETOR 4.0 [Qin and Liu, 2013] is the ranking dataset we will work with. The documents come from the gov2 collection[1] which is a crawl of all documents with a .gov address in 2004. The queries are from the the Million Queries track of the Text Retrieval Conference (TREC) 2007 and 2008 referred to as MQ2007 (about 1700 queries) and MQ2008 (800 queries). This split also enables us to explore interesting covariate shift style domain adaptation aspects of the problem. Allan et al. [2007] (MQ2007) and Allan et al. [2008] (MQ2008) describe the finer details on how the Million Queries track identified queries, documents, and how the relevance score assignment was performed.

LETOR 4.0 is basically a featurization of the existing query-document pairs along with relevance scores from MQ2007 and MQ2008. These are standard NLP features that we describe now. The number of features is $d = 46$. The first 40 features are created using a field element {body, anchor, title, url, whole document} and a featurizer {TF, TF-IDF, BM25, LMIR.ABS, LMIR.DIR, LMIR.JM}. The first set refers to various fields of a particular document:

[1] http://ir.dcs.gla.ac.uk/test$_c$*ollections*/*gov*2 − *summary.htm*

1. Body refers to the text contained in the entire body of the document.
2. Anchor refers to a description of the document from the source which linked to it.
3. Title refers to the title of the document.
4. URL is the URL of the document.
5. Whole document refers to everything including all the 4 above.

The second set refers to a particular kind of featurization between the terms in the query and the particular field of the document we are considering. Consider a function $f : (\text{word}, \text{field}) \to \mathbb{N}_0$. Then using one of the functions described below, the features constructed below corresponds to:

$$F(\text{query } q, \text{field } f) = \sum_{\text{word } w \text{ in } q} f(w, f).$$

Below, we use 'corpus' to refer to the entire set of documents from which a subset is deemed relevant for each query.

1. TF (term frequency): Total number of occurrences of w in f.
2. IDF (inverse document frequency): This feature depends only on the query and the category of the field:
$$\log\left(\frac{0.5 + \#\text{docs in corpus} - \#\text{docs in corpus such that } f \text{ contains } w}{0.5 + \#\text{docs in corpus such that } f \text{ contains } w}\right).$$
3. DL (document length): Length of the field (number of words).
4. TF-IDF (term frequency-inverse document frequency): TF multiplied by IDF.
5. BM25:
$$\frac{3.5 \cdot \text{IDF}(w) \cdot \text{TF}(w, f)}{\text{TF}(w, f) + 2.5 \cdot (0.8 + 0.2 \cdot \text{DL}(f)/\text{avglf})},$$
where avglf = average $\text{DL}(f)$ across the corpus.
6. LMIR: LMIR Zhai and Lafferty [2001] refers to smoothing methods for probabilistic Language Modeling for Informational Retrieval. These are different ways of modeling the probability of a word given the field $p(w|f)$ using priors that depend on the corpus level probability of the word $p(w|C(f))$ for a specific field:
$$p(w|C(f)) = \frac{\#\text{occurences of } w \text{ in corpus for field } f}{\text{total words in corpus for field } f}.$$
Different priors lead to different smoothing techniques (the constants below are specific to the LETOR 4.0 dataset):
   (a) LMIR.JM (Jelinek Mercer method):
   $$p(w|f) = 0.9 \cdot \frac{\text{TF}(w, f)}{\text{DL}(f)} + 0.1 \cdot p(w|C(f)).$$
   (b) LMIR.DIR (Bayesian smoothing using Dirichlet priors):
   $$p(w|f) = \frac{\text{TF}(w, f) + 2000 \cdot p(w|C(f))}{\text{DL}(f) + 2000}.$$
   (c) LMIR.ABS (Absolute discounting):
   $$p(w|f) = \frac{\max(\text{TF}(w, f) - 0.1, 0)}{\text{DL}(f)} + 0.1 \cdot \frac{\#\text{unique terms in } f}{\text{DL}(f)} \cdot p(w|C).$$

The last 6 features are:

41. Pagerank score of the document (independent of the query).
42. Inlink number of the document (number of documents in corpus that link to this document).
43. Outlink number, that is the number of documents linked from this document.
44. Number of slashes in the URL.
45. Length of URL.
46. Number of child pages (identified using the URL). For exmaple https://www.ml.cmu.edu/people/ is a child of https://www.ml.cmu.edu/.

The relevance scores given are either 0:irrelevant, 1:relevant, or 2:very relevant. This leads to a set of optimal rankings where every document with relevance score 2 is above every document with relevance score 1, which in turn is above every document with relevance score 0.

3 Predictive inference for ranking (the experiment)

Previous ranking algorithms have only discussed how to obtain a point-predictions for ranking, while saying nothing about the uncertainty in the estimate. One notion of uncertainty could be a probability value of how confident we are of the particular ranking predicted. However given that the space of rankings is exponentially large, this probability will be very small. We will instead consider if it is possible to return a prediction set of rankings that have some desirable probabilistic properties that can be formally argued. The goal will be to make this prediction set interpretable and meaningful so that it can be actionable. There does not exist any literature in the domain of uncertainty for learning to rank, since it is a statistical perspective not available to most people in the information retrieval community.

Suppose we are interested in maximizing one of the ranking metrics M introduced in Section 1.1 such as NDCG@k or MAP. Given training data $\{Z_i\}_{i=1}^n \overset{iid}{\sim} Q$, we will consider if we can learn a function $C_\alpha : \mathbb{R}^{d,k} \to 2^{S_k}$ such that it has the following predictive property which we term as *validity*:

$$\mathbb{P}_{Z_1,Z_2,\ldots,Z_n,Z\sim Q}\left(\exists \pi \in C_\alpha(f_1,\ldots f_k) : M(\pi, r) \geq s\right) \geq 1-\alpha. \tag{1}$$

Above the randomness of the training data affects the learnt function C_α and Z refers to a test-query $((f_1, r_1), \ldots (f_k, r_k))$. Of course, C_α only has access to f, not r. Validity is a property that says that in the set of rankings C_α at least one of them performs well on the metric (indicated by saying $M(r, \pi) \geq s$ for some $s \in \mathbb{R}$). While the main goal here is to ensure validity, we want to do so while minimizing the average size of the prediction set, which we term as *efficiency*:

$$\mathbb{E}_{Z_1,Z_2,\ldots,Z_n,Z\sim Q}\left|C_\alpha(f_1,\ldots f_k)\right|. \tag{2}$$

The remarkable framework of Conformal inference, introduced next, allows us to provide such guarantees without making any distributional assumptions. It has been shown to perform well in regression and classification settings. The goal of the project will be to extend it to the ranking setting.

3.1 Conformal inference framework

We will now describe an assumption-free method for performing predictive inference: conformal inference [Vovk et al., 2005]. Suppose the training data is $\{Z_i = (X_i, Y_i)\}_{i=1}^n \overset{iid}{\sim} \mathcal{D}$ where $X_i \in \mathcal{X}$ are the features/covariates and $Y_i \in \mathcal{Y}$ is the output that we want to predict. Note that the only non-observed random variable is Y. Consider any training algorithm that can use data to learn a function $g : \mathcal{X} \to \mathcal{P}$ where $\mathcal{P}$ is a set of predictive elements from which inference on $\mathcal{Y}$ can be made in some way (in particular it could be that $\mathcal{P} = \mathcal{Y}$ or $\mathcal{P} = 2^{\mathcal{Y}}$). Conformal prediction relies on a residual score $d : \mathcal{P} \times \mathcal{Y} \to \mathbb{R}$ that

measures some notion of distance between the first argument (the prediction) and the second argument (the output variable). For example in the case of regression, we could have $\mathcal{P} = \mathcal{Y}$ and $d(\widehat{y}, y) = |\widehat{y} - y|$. Given a non-conformity score, we can perform a variety of aggregation schemes that lead to a prediction interval for Y.

For now we describe the scheme of inductive conformal prediction or split conformal [Papadopoulos et al., 2002, Lei et al., 2018]. The first step is to split the training data into a proper training set of size m and a calibration set of size $c = n - m$ such that each point is equally likely to fall within the proper training set. Without loss of generality, let the calibration data be $\{Z_i\}_{i=1}^{c}$. The predictor g is learnt on the proper training data using any algorithm and its residual scores are computed $d_i = d(g(x_i), y_i)$. Let $t = \lceil (1-\alpha)(c+1) \rceil$. Let $d_{(t)}$ denote the t'th residual value in the sorted ordering of the residuals. For a given test point x, we return the prediction interval $C_\alpha(x) = \{y \in \mathcal{Y} : d(g(x), y) \leq d_{(t)}\}$. Then it can be shown that

$$\mathbb{P}_{\{Z_i\}_{i=1}^{c}, Z} \left(Y \in C_\alpha(X) \right) \geq 1 - \alpha.$$

The argument hinges on the fact that the set $\{d_i\}_{i=1}^{c}$ and $d(g(X), Y)$ are exchangeable and hence the rank of $d(g(X), Y)$ among all these is uniformly distributed on $[c+1] := \{1, \ldots c+1\}$ (if ties are broken arbitrarily). If $Y|X$ follows a continuous probability distribution, it can in fact be shown that

$$\mathbb{P}_{\{Z_i\}_{i=1}^{c}, Z} \left(Y \in C_\alpha(X) \right) \leq 1 - \alpha + \frac{1}{n+1},$$

which suggest that $C_\alpha(x)$ is optimal. However, there are two clear drawbacks. First, it is only optimal given a fixed function g that is learnt on only a part of the training data. It may be possible to utilize more (or all) of the training data to obtain functions g that are better predictors, hence reducing the residual value for all points, leading to efficient intervals. Second, the probability guarantee provided is not conditional on the X. Ideally we want guarantees of the form

$$\mathbb{P}_{\{Z_i\}_{i=1}^{c}, Z} \left(Y \in C_\alpha(X) | X = x \right) \geq 1 - \alpha.$$

It was shown by Lei and Wasserman [2014] that finite sample conditional validity is impossible without additional assumptions. There exist practically useful solutions to both these problems with some theoretical guarantees, but we shall delve into them in a future version of the report, depending on what seems to be useful on the LETOR 4.0 dataset.

3.2 Conformal inference for ranking

We will be exploring conformal inference to quantify uncertainty for ranking problems. We list some preliminary ideas to illustrate how this would work in this section. Let us use F_{ij} to denote the featurization corresponding to document j of query i, and R_{ij} to refer to its relevance score. Then $Z_i = (F_{i*}, R_{i*})$. As before, we will continue using $Z = (f, r)$ to refer to the features and unknown relevance scores of a test query. In order to perform predictive inference, we first introduce the notion of a set with metric guarantee. Consider one of the metrics $M : S_k \times \mathbb{R}^k \to \mathbb{R}$ introduced in Section 1.1, and define a 'ball'

$$B_s(M, r) := \{\pi : M(\pi, r) \geq s\}.$$

This ball captures a set of rankings such that they have score at least s with respect to some M such as NDCG@k, Prec@k, etc. The goal is to use conformal inference to learn a function C_α that has the property of validity:

$$\mathbb{P}_{Z_1, Z_2, \ldots, Z_n, Z \sim Q} \left(\exists \pi \in C_\alpha(f_1, \ldots f_k) : \pi \in B_s(M, r) \right) \geq 1 - \alpha.$$

To perform conformal inference, we will identify a prediction set $\mathcal{P}$ and a residual function $d : \mathcal{P} \times S_k \to \mathbb{R}$ as per the framework described in Section 3.1. All the learning algorithms described in Section 1.2 learn a function $h_{\mathbf{w}}$ on the proper training data which thus leads to learnt relevance scores $\mathcal{P} = \mathbb{R}^k$. Alternatively these relevance scores can be directly converted to a ranking to give $\mathcal{P} = S_k$. In each case, we would need to define the function d. This function d would need to be empirically useful. We introduce a couple of candidate residual functions that we could consider and we will be running experiments to identify which of these work for us.

1. **Kernels between rankings** (for the case $\mathcal{P} = S_k$).

   For some space $\mathcal{X}$, a kernel is a positive semidefinite function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that represents an inner product among an implicit feature space representation of elements of $\mathcal{X}$. Although they are usually used to learn functions over $\mathcal{X}$, in our case they will serve as a useful notion of similarity between rankings, when $\mathcal{X} = S_k$. The Kendall and Mallows kernels are standard kernels here (Jiao and Vert [2017] offer a contemporary ML centric discussion), defined as a function of the number of transpositions $T(\pi_1, \pi_2)$ between two rankings $\pi_1, \pi_2 \in S_k$:

   $$T(\pi_1, \pi_2) = \sum_{i \neq j} \mathbb{1}\left\{\pi_1(i) < \pi_1(j)\right\} \mathbb{1}\left\{\pi_2(i) > \pi_2(j)\right\}.$$

   The Kendall kernel is defined as

   $$K(\pi_1, \pi_2) = \frac{1 - T(\pi_1, \pi_2)}{\binom{k}{2}}$$

   and the Mallows kernel is defined as

   $$K(\pi_1, \pi_2) = \exp\left(-\lambda T(\pi_1, \pi_2)\right)$$

   for some λ. Jiao and Vert [2017] showed that these are positive definite and also introduced some versions of these with partial rankings that might be useful for us since the ternary relevance scores only induce a partial ranking. Further, we will be interested in a weighted version of the transposition function that would lead to weighted Kendall and Mallows kernels. This was introduced by Jiao and Vert [2018] and is defined as

   $$T_w(\pi_1, \pi_2) = \sum_{i \neq j} w(\pi_1(i), \pi_1(j)) \mathbb{1}\left\{\pi_1(i) < \pi_1(j)\right\} \mathbb{1}\left\{\pi_2(i) > \pi_2(j)\right\}$$

   for some weighting function w. Notice that a kernel is a similarity function and hence to create a residual function out of it, we would define $d(\pi_1, \pi_2) = -K(\pi_1, \pi_2)$. If we use kernels to measure distance between rankings, the final set that we predict will be of the following form given some point-prediction ranking $\widehat{\pi}$ and a threshold t:

   $$C_\alpha = \{\pi : K(\widehat{\pi}, \pi) \geq t\}$$

2. **Probability balls** (for the case $\mathcal{P} = \mathbb{R}^k$). As discussed in Section 1.2 a function $h_{\mathbf{w}}$ that gives a relevance score to each document can be used to model a probability for every possible ranking

   $$p_f(\pi | h_{\mathbf{w}}(f)) = \prod_{i=1}^{k} \frac{\phi(h_{\mathbf{w}}(f_{\pi(i)}))}{\sum_{j=i}^{n} \phi(h_{\mathbf{w}}(f_{\pi(j)}))}.$$

   This is also known as the Plackett-Luce model. The learning to rank algorithms developed using this model perform well in practice and hence there is some reason to believe its utility. Given a Plackett-Luce model, we can now describe a residual function as follows:

   $$d(h_{\mathbf{w}}(f), \pi) = -p_f(\pi | h_{\mathbf{w}}(f)).$$

   Performing split conformal using would lead to learning a calibrated value t and defining the set of predicted rankings as all rankings π that have $p_f(\pi | h_{\mathbf{w}}(f)) \geq t$. Alternately, we could have an aggregation technique that takes a value t and $h_{\mathbf{w}}(f)$ to give a set of rankings C_α such that

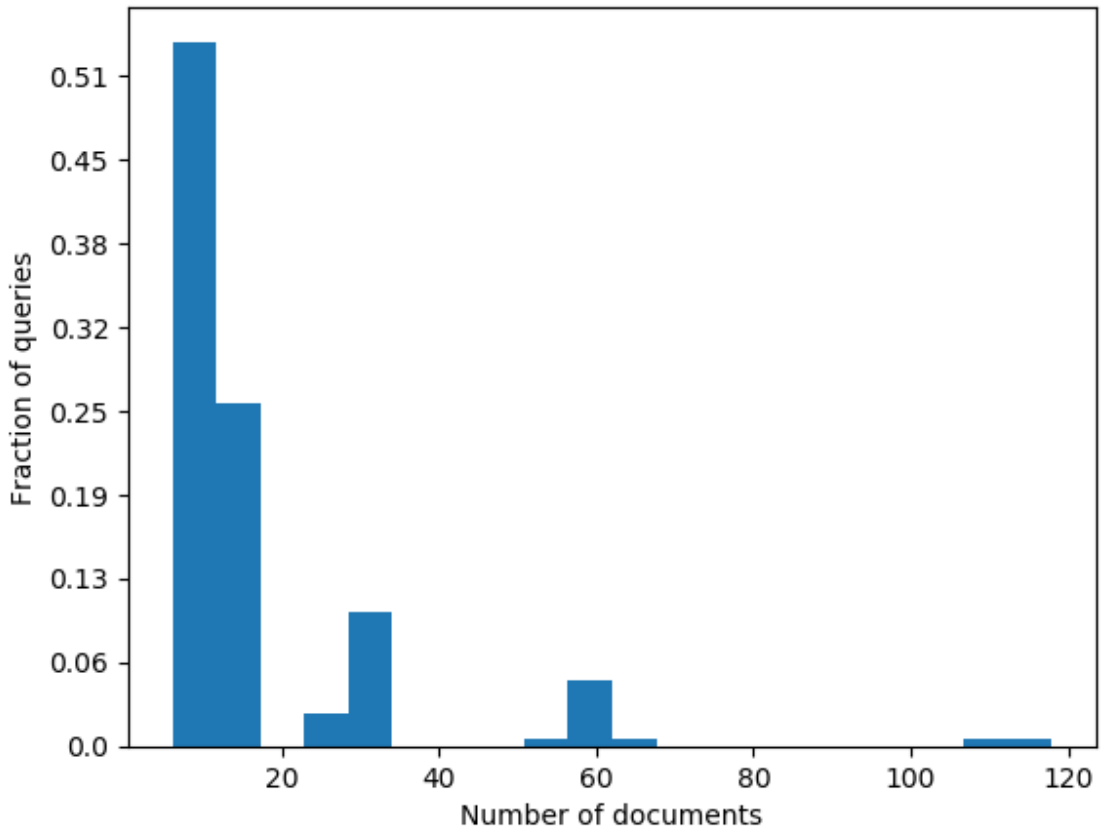
   $$\sum_{\pi \in C_\alpha} p_f(\pi | h_{\mathbf{w}}(f)) \geq t.$$

4 More thoughts

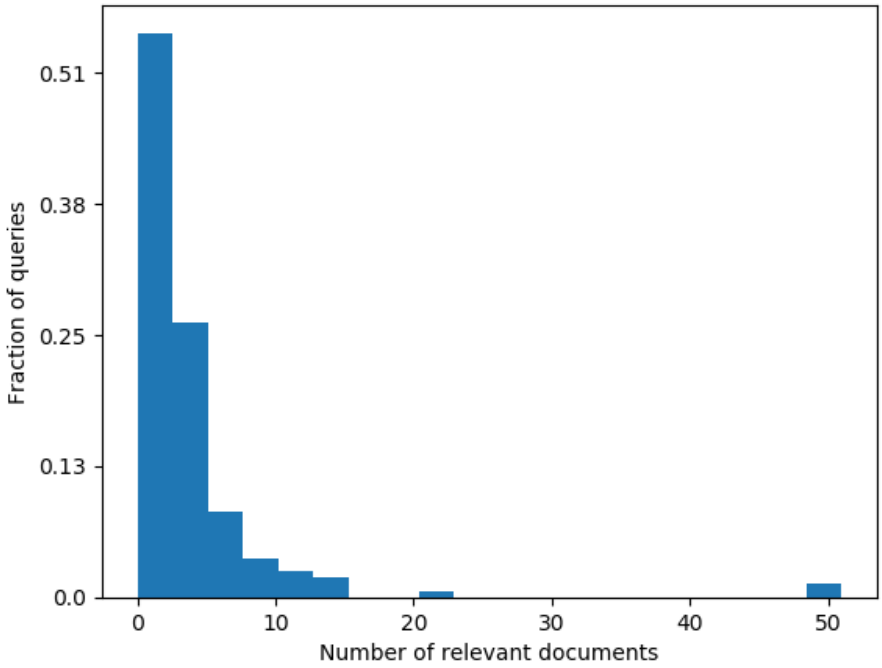
Practically, we want to show a particular number of documents, say r. Thus the set C_α should be a set of the top r documents for some r.

5 Experiments

5.1 About one in five documents are relevant and the number of relevant documents is roughly linear with respect to the total number of documents



Total number of validation documents in Fold 1 = 2096. Total number of relevant validation documents in Fold 1 = 566.



5.2 Parametric estimation

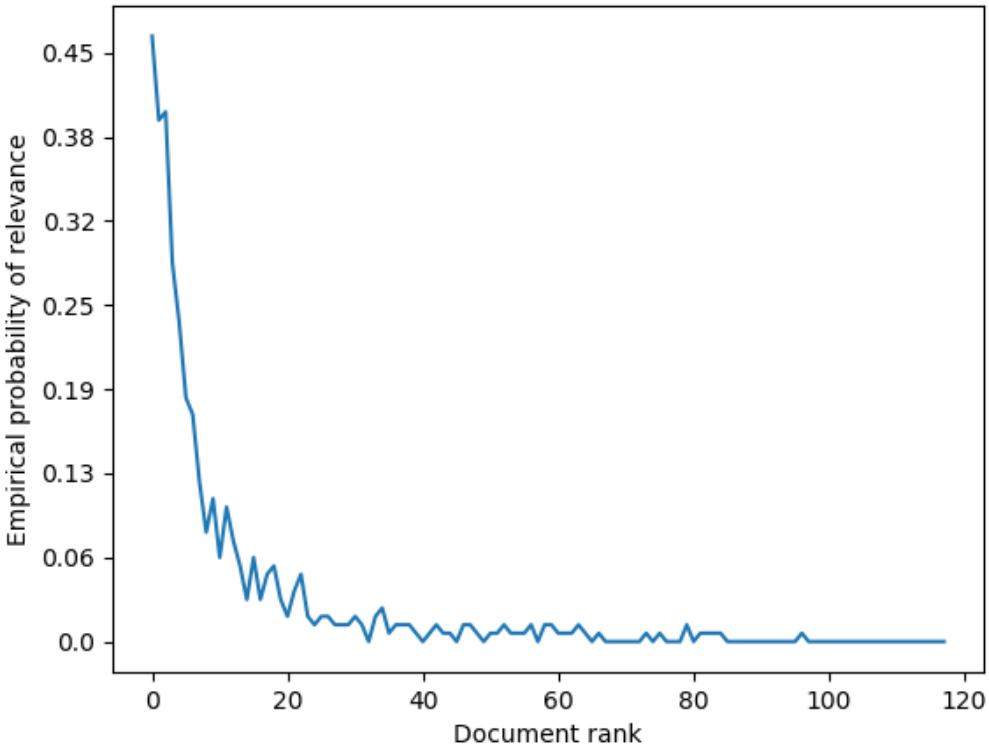
Hoeffding bound for parameter estimation. Say $\delta = 0.1$ With probability $1 - \delta$,

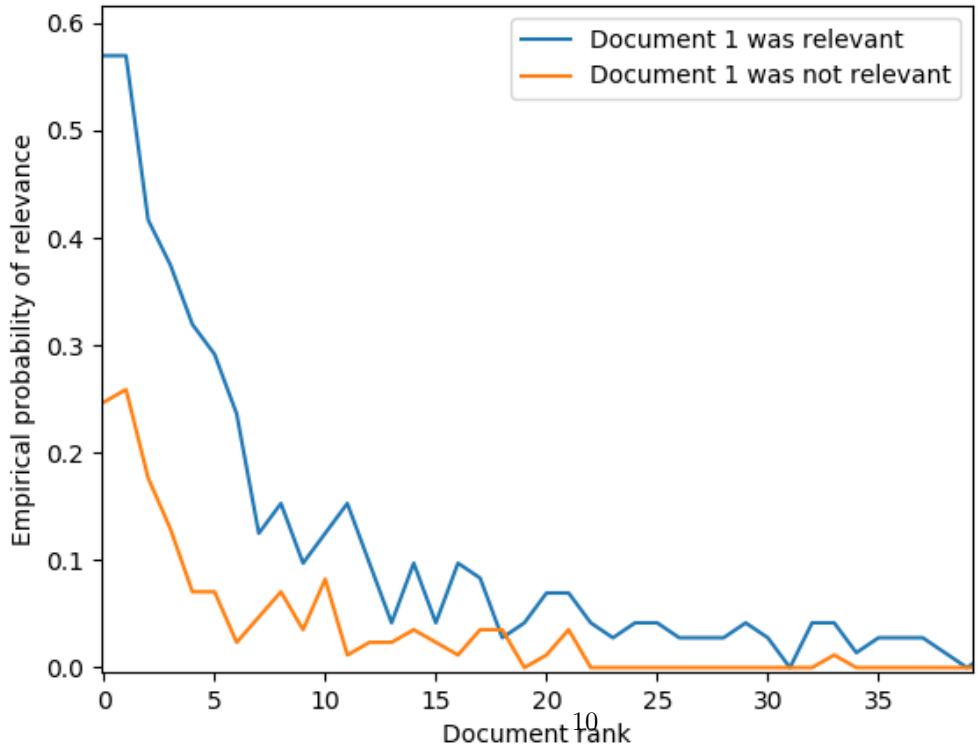
$$P(p > \bar{p} - \epsilon) \leq \exp\left(-2N\epsilon^2\right)$$

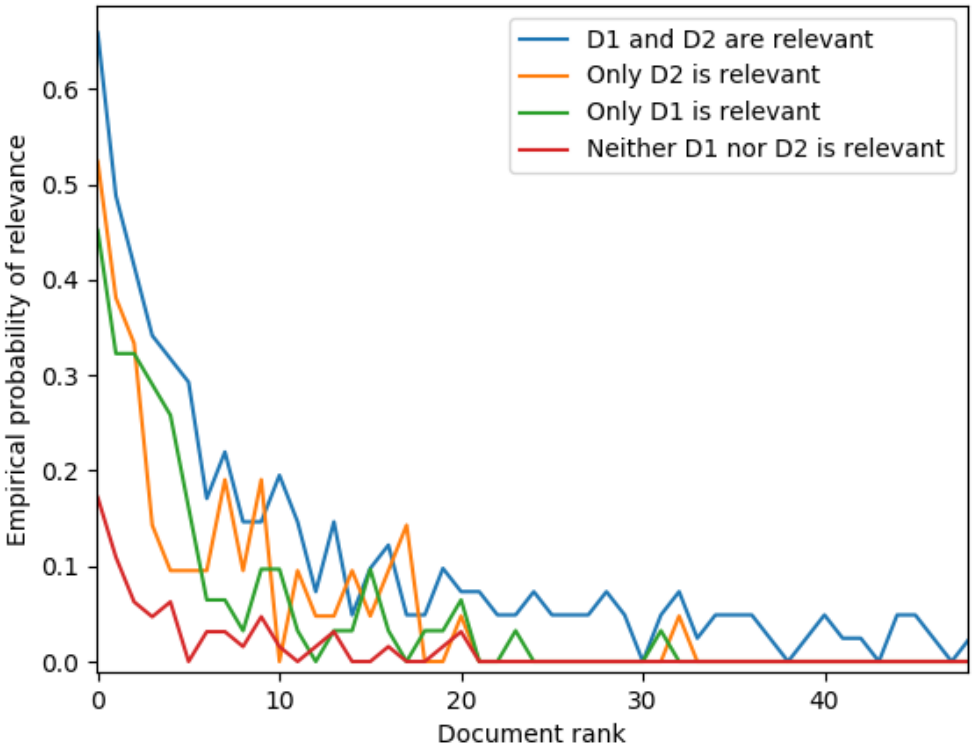
$$P(p > \bar{p} - 0.085) \leq 0.1$$

Event	Number of samples	Empirical value
R_1	157	0.46
R_2	157	0.39
R_3	157	0.40
$R_2 \mid R_1$	72	0.57
$R_2 \mid \neg R_1$	85	0.25
R_3	157	0.40

5.3 Empirical probability of relevance wrt rank







6 References

James Allan, Ben Carterette, Javed A Aslam, Virgil Pavlu, Blagovest Dachev, and Evangelos Kanoulas. Million query track 2007 overview. Technical report, University of Massachusetts Amherst, 2007.

James Allan, Javed A Aslam, Ben Carterette, Virgil Pavlu, and Evangelos Kanoulas. Million query track 2008 overview. Technical report, University of Massachusetts Amherst, 2008.

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

Christopher J Burges, Robert Ragno, and Quoc V Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.

Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136, 2007.

Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, volume 51, pages 243–250. ACM New York, NY, USA, 2017.

Yunlong Jiao and Jean-Philippe Vert. The kendall and mallows kernels for permutations. *IEEE transactions on pattern analysis and machine intelligence*, 40(7):1755–1769, 2017.

Yunlong Jiao and Jean-Philippe Vert. The weighted kendall and high-order kernels for permutations. *arXiv preprint arXiv:1802.08526*, 2018.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

Jen-Wei Kuo, Pu-Jen Cheng, and Hsin-Min Wang. Learning to rank from bayesian decision inference. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 827–836, 2009.

Jing Lei and Larry Wasserman. Distribution-free prediction bands for non-parametric regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1):71–96, 2014.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *European Conference on Machine Learning*, pages 345–356. Springer, 2002.

Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.

Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374, 2010.

Niek Tax, Sander Bockting, and Djoerd Hiemstra. A cross-benchmark comparison of 87 learning to rank methods. *Information processing & management*, 51(6):757–772, 2015.

Ming-Feng Tsai, Tie-Yan Liu, Tao Qin, Hsin-Hsi Chen, and Wei-Ying Ma. Frank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390, 2007.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

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

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, 2001.