Learning to select for a predefined ranking

Aleksei Ustimenko

Alexander Vorobev

Gleb Gusev

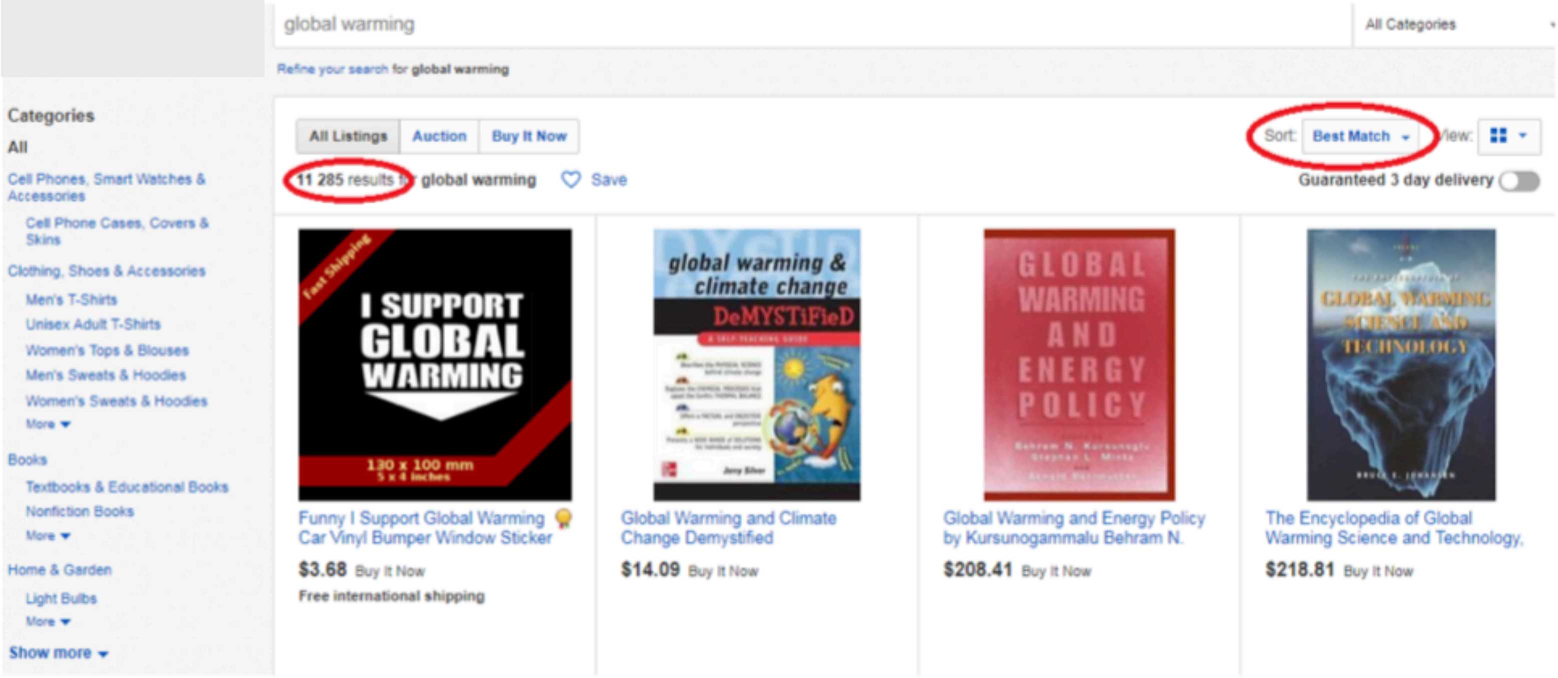
Pavel Serdyukov

From ranking to sorting

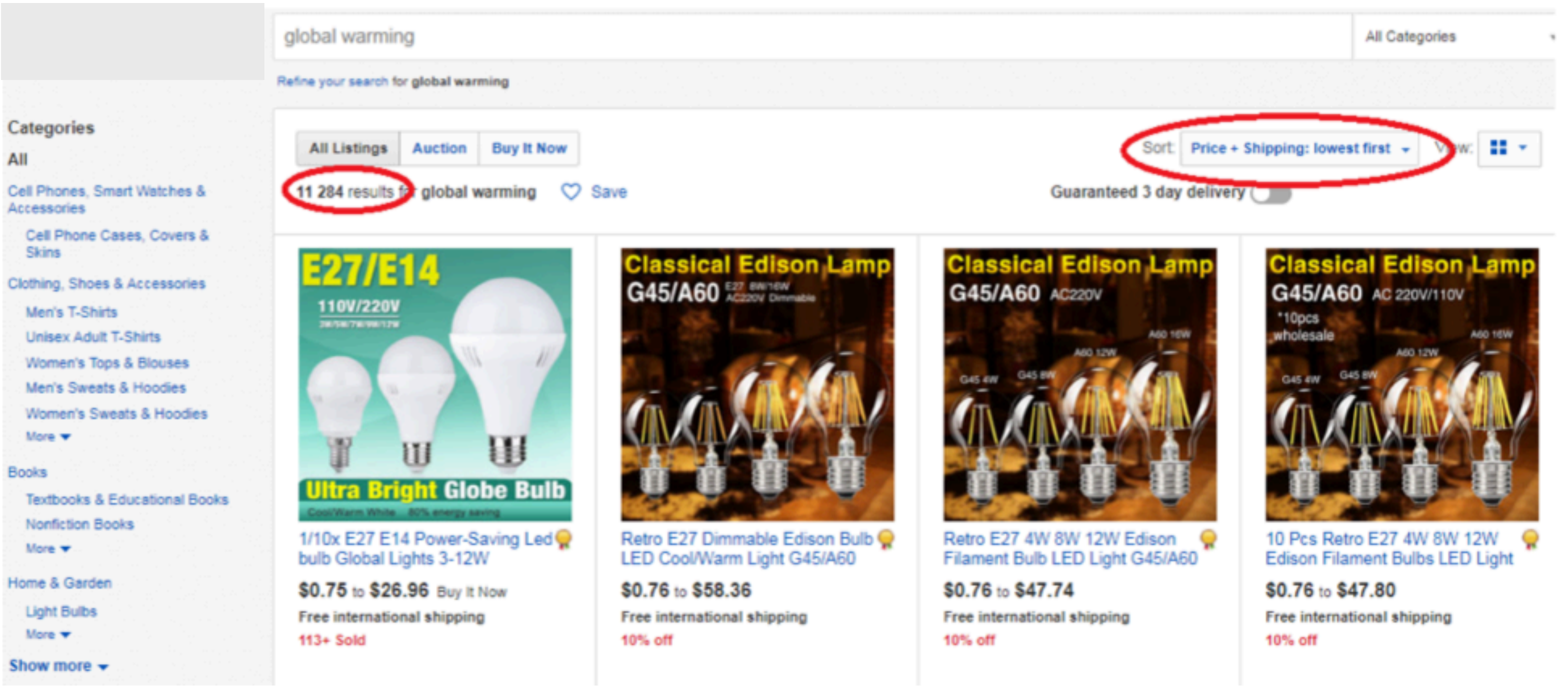
- Search engines typically order the items by some relevance score obtained from a ranker before presenting the items to the user

- Yet, online shops and social networks allow the user to rearrange the items using some dedicated attribute (e.g. price or time)

From ranking



To sorting



Threshold relevance?

- It was proven that filtering with a constant threshold for relevance is **suboptimal** (in terms of ranking quality metrics like DCG)

- The optimal algorithm was suggested by (Spirin et. at at SIGIR 2015), but it has quadratic complexity **O (n^2)**, where **n** – is the list size

- Such algorithms are infeasible for search engines, we need to predict if to filter an item by **just using item features (locally), not the entire list (globally)**

LSO Problem Formulation

- We define a *selection algorithm* as F and the result of its application to a list L to be the *selected* L^F
- L^F - the same ordered list as L, but with some items filtered
- We formulate the problem of **LSO** as learning from D a selection algorithm F that maximizes the expected ranking quality **Q** of L^F, where L is sampled from some P:

$$\mathrm{F}^* = \arg\max \mathbb{E}_{L \sim P} Q(L^F)$$

Optimal Selection Predictor

- First, we suggest to build a model M that predicts the binary decision of the infeasible optimal algorithm

- Then we train a binary classifier M on the training examples obtained from that algorithm $\left\{\left(x_{ij}, Opt_{ij}\right)\right\}_{i:\, L_i \in D,\, j=1..n_i}$ by minimizing logistic loss

- However, the logistic loss of **such a classifier** is still not directly related to ranking quality Q, i.e. it **is not a listwise learning-to-rank algorithm**

Direct Optimization of the Objective

- For a document d with features vector $x_d \in \mathbb{R}^l$ we define probabilistic filtering rule by:

$$P(F(d) = 1) = \sigma(f(x_d)) = \frac{1}{1 + \exp(-f(x_d))}$$

- Assume that decisions $F(d)$ for different d are independent. Denote the space of all so-defined stochastic selection algorithms by $\mathcal{F}$.
- We transform Q to the $Q_{smooth}\,(F, L) = \mathbb{E}_{Z\sim P_F}\, Q(L_Z)$
- And the problem to:

$$F^* = \arg\max_{F\in\mathcal{F}} \mathbb{E}_{L\sim D} Q_{smooth}(F, L)$$

Policy Gradient Approach

- For i.i.d. samples of binary decisions $Z_1, \dots, Z_s \sim P_F$ define the estimate (after applying the log derivative trick):

$$\frac{\partial Q_{smooth}(F, L)}{\partial f(x_j)} \approx \frac{1}{s} \sum_{i=1,s} \left(Q\left(L_{Z_i}\right) - b\right)\left(-p_j\right)^{Z_{ij}} \left(1 - p_j\right)^{1-Z_{ij}}$$

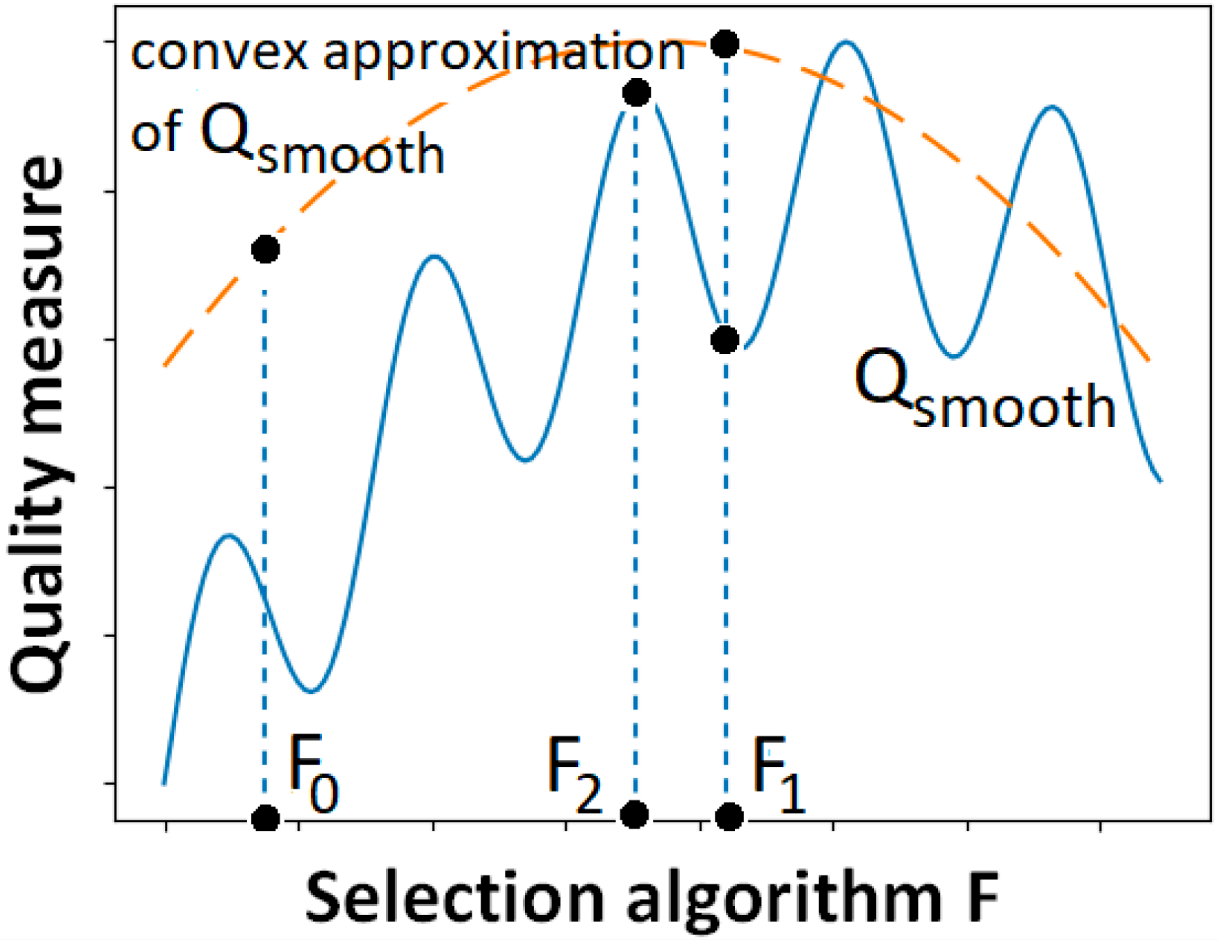
where baseline $b \coloneqq Q(L_{z_F^{0.5}})$ with $z_{F,k}^{0.5} = 1\{p_k > 0.5\}$

- And we use this functional gradient directly in the Gradient Boosted Decision Trees learning algorithm (with CatBoost implementation)

Pre-training

After training OSP model, we use it as a starting point for our approach

Thus, we avoid getting stuck in local maxima



Step by our poster **#228**

Learning to select for a predefined ranking

Aleksei Ustimenko (austimenko@yandex-team.ru)
Alexander Vorobev (alvor88@yandex-team.ru)
Gleb Gusev (gleb57@yandex-team.ru)
Pavel Serdyukov (pavser@yandex-team.ru)
Yandex

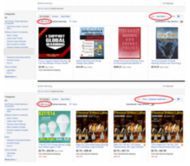
From Ranking to Sorting

Search engines typically order the items by some relevance score obtained from a ranker before presenting the items to user.

Yet, online shops and social networks allow the user to rearrange the items using some dedicated attribute (e.g. price or time).

Ranking

Sorting



In this example, the user sees relevant results after ranking, but they become completely irrelevant after sorting by price.

Thresholding by relevance score is sub-optimal

Table 1. **Example of suboptimality of the cutoff approach**

	Original list	Best cutoff	Best selection
List	d_1, d_2, d_3	d_2	d_2, d_3
Attribute values	1, 2, 3	2	2, 3
Relevance values	2, 7, 1	7	7, 1
DCG@3 of list	6.92	7	7.63

[Spirin et al, SIGIR 2015] proposed an optimal algorithm, but it has quadratic time complexity O(n^2): infeasible for modern search engines.

LSO Problem Formulation

Consider a i.i.d. sample of lists $D = \{L_i\}_{i=1}^m$ from P of ordered sets $L_i = (d_{i1}, \dots, d_{in_i})$ with n_i items $d_{ij} = (x_{ij}, r_{ij}) \in \mathbb{R}^l \times \mathbb{R}$

An item $d = (x, r)$ corresponds to a context-item pair, represented by $x = (x^0, \dots, x^l)$ of its l features and assigned a relevance r (unknown to the system). Assume that the items in each list L_i are ordered by one of the features.

We define a *selection algorithm* as $F : \mathbb{R}^l \to \{0, 1\}$ and the result of its application to a list L_i to be the *selected* list $L_i^F := (d_{i1}, \dots, d_{ik})$, where $i_1 < \dots < i_k$ and $\{i_1, \dots, i_k\} = \{i \in [1, n] : F(x_i) = 1\}$.

We formulate the problem of learning to select with order (**LSO**) as learning from D a selection algorithm F that maximizes the expected ranking quality **Q** of L^F, where L is sampled from P:

$$F^* = \arg\max \mathbb{E}_{L \sim P} Q(L^F)$$

Optimal Selection Prediction (OSP)

First, we suggest to build a model M that for each item predicts the binary decision of the optimal algorithm by [Spirin et al., 2015], that is 1 iff the algorithm decided not to filter out an item.

Then we train a binary classifier M on the training examples $\{(x_{ij}, Opt_{ij})\}_{i:\, L_i \in D, j=1..n_i}$ by minimizing logistic loss. After it, we define F_M on basis of M as $F(x) = 1\{M(x) > t\}$ where t is a constant hyperparameter of F_M

However, the logistic loss is still not directly related to ranking quality Q, i.e. it is not a listwise learning-to-rank algorithm.

Direct Optimization of the Objective

For a document d with features vector $x_d \in \mathbb{R}^l$ we define probabilistic filtering rule by:

$$P(F(d) = 1) = \sigma(f(x_d)) = \frac{1}{1 + \exp(-f(x_d))}$$

Assume that decisions $F(d)$ for different d are independent. Denote the space of all so-defined stochastic selection algorithms by $\mathcal{F}$. We transform Q to the $Q_{smooth}(F, L) = \mathbb{E}_{Z \sim P_F} Q(L_Z)$ (P_F is the distribution of selection decisions). And the problem to:

$$F^* = \arg\max_{F \in \mathcal{F}} \mathbb{E}_{L \sim D} Q_{smooth}(F, L)$$

Policy Gradient Approach (PG)

For i.i.d. samples of binary decisions $Z_1, \dots, Z_s \sim P_F$, we define the gradient estimate (after applying the log derivative trick):

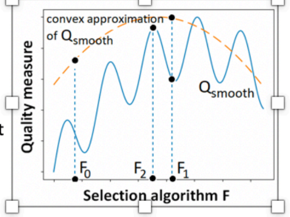
$$\frac{\partial Q_{smooth}(F, L)}{\partial f(x_j)} \approx \frac{1}{s} \sum_{i=1,s} (Q(L_{Z_i}) - b)(-p_j)^{Z_{ij}} (1 - p_j)^{1 - Z_{ij}}$$

where $b := Q(L_{z_F^{0.5}})$ with $z_{F,k}^{0.5} = 1\{p_k > 0.5\}$ is the mode of the distribution $Q(L_Z)$.

Two steps of learning

After training OSP model $F_1(x)$ that started from $F_0(x) \equiv 0$ we use it as a starting point for PG and LBO approaches.

Thus, we avoid getting stuck in local maxima attained if we start optimization from $F_0(x) \equiv 0$ instead.



Learning Algorithm

For an ML algorithm for all the approaches, we chose GBDT as the state-of-the art method for many practical tasks including the learning- to-rank problem in web search and click prediction.

We use GBDT implementation in the open-sourced CatBoost Python package

CatBoost

Experimental Results

We pick DCG–RR$(r_1, \dots, r_k) = \sum_i r_i / i$ as lists quality measure. For the major independent evaluation of the result page relevance, we collected human relevance judgements of 5 grades (from 0 to 4) for top-10 results of each selected list produced by the algorithms trained on train and evaluated $DCG@10, p@10, stup@12$.

Finally, most representative algorithms were compared in online experiments. For evaluation we used Abandoment, MRR and $CTR@12$

Table 2. Performance, absolute for $WeakCutoff$ and relative Δ to $WeakCutoff$, % for others

Approach	DCG-RR	DCG@10	p@10	stup@12
$WeakCutoff$	0.52	1.07	0.73	0.06
$ConstCutoff$	0.05%	2.9%	1.6%	-11.7%
$QueryCutoff$	0.56%	6.3%	4.3%	-17.5%
OSP	3.86%	20.3%	10.7%	-33.2%
$OSP + LBO$	4.17%	**22.4%**	12.0%	**-37.6%**
$OSP + PG$	**4.33%**	**22.4%**	**12.2%**	-36.8%
$Oracle$	14.44%			

Table 3. Online performance, relative Δ to $WeakCutoff$, %

Approach	Abandonment	CTR@12	MRR
$QueryCutoff$	-1.4%	8.7%	5.7%
OSP	-4.5%	19.7%	24.6%
$OSP + PG$	**-5.1%**	**24.8%**	**36.3%**

WeakCutoff – “take all documents”,

ConstCutoff – filtering by global thresholding of relevance prediction,

QueryCutoff – query-wise threshold prediction + filtering by thresholding of relevance

LBO – another proposed algorithm optimizing the lower bound of Q (see its description in the paper)