



Online Learning to Rank for Information Retrieval

Artem Grotov, Maarten de Rijke

Sources

- ▶ Slides based on input and material from Richard Combes, Katja Hofmann, Branislav Kveton, Terry Lam, Anne Schuth, Shimon Whiteson, Masrour Zoghi
- ▶ Slides benefitted from feedback by Alexey Borisov, Damien Lefortier, Ilya Markov, Manos Tsagkias, Wouter Weerkamp

① Introduction

Who we are

Artem Grotov

- ▶ PhD student, University of Amsterdam
- ▶ Research interests: online learning to rank
- ▶ a.grotov@uva.nl

Maarten de Rijke

- ▶ Professor of Computer Science, University of Amsterdam
- ▶ Research interests: semantic search, learning to rank
- ▶ derijke@uva.nl

Materials

These slides

References per section, bibtex file

Draft survey

- ▶ Available in 4–6 weeks

Pointers to experimental resources for evaluating online learning to rank methods

Web site at <https://staff.fnwi.uva.nl/m.derijke/talks-etc/online-learning-to-rank-tutorial/>

Aim

Offline learning to rank had big impact on information retrieval (IR)

Limitations of offline learning to rank for IR have become apparent

Increased attention for **online** learning to rank methods for IR:
learn from user interactions rather than from labeled data that is fully available for training up front

Online learning to rank methods have emerged in machine learning and IR communities

Aim of tutorial: bring these together and offer unified perspective

Objectives

To describe existing online LTR algorithms in **unified** way

Explain importance of **balancing exploration and exploitation** in online LTR setting

Explain how to analyse **performance** of online LTR algorithms

Present **experimental and evaluation methodologies** for online LTR

Present **online learning algorithms** not used yet for LTR

Discuss **future directions of research** in online LTR

Structure of the tutorial

Part 1

- ▶ Introduction
- ▶ Learning to rank for information retrieval
- ▶ Online learning to rank: First encounter
- ▶ Reinforcement learning and bandits
- ▶ Online signals to learn from
- ▶ Online learning to rank: Second encounter

Part 2

- ▶ Experiments and applications
- ▶ Online learning to rank: Third encounter
- ▶ Problems and opportunities
- ▶ Conclusion

② Learning to rank for information retrieval

Learning to rank for information retrieval

Ranking documents: its importance, relationship to other machine learning tasks, and unique challenges of LTR

Current approaches to offline LTR: they all have issues

- ▶ cost of producing labelled data
- ▶ mismatch between manually curated labels and user intent
- ▶ ...

Yandex

Web

Pisa - Отели / booking.com
 Ad booking.com
 Выгодные цены без комиссий! Бронируйте отели в Pisa
 Наиболее популярные С лучшим рейтингом Роскошные отели

W Pisa - Wikipedia, the free encyclopedia
 en.wikipedia.org > **Pisa** +
 Pisa (/ˈpiːzə/; Italian pronunciation: [ˈpiːsa]; ˈpiːza]) is a city in Tuscany, Central Italy, straddling the River Arno just before it empties into the Tyrrhenian Sea. It is the capital city of the Province of Pisa.

Pisa
 en.academic.ru > dic.nsf/enwiki/15093 +
 Pisa's origins are unknown. The city lies at the junction of two rivers, Arno and Serchio in the Ligurian Sea forming a laguna area.

Pisa travel guide - Wikitravel
 wikitravel.org > en/Pisa +
 Pisa is a city in Tuscany, Italy with a population of some 90,000 people. Pisa is best known for the world famous Leaning Tower, but those who come here with their mind already made up that the Tower is the only thing to see may miss the rest of the ...

W Programme for International Student Assessment...
 en.m.wikipedia.org > Programme for International Student Assessment +
 Programme for International Student Assessment. The Programme for International Student Assessment (PISA) is a worldwide study by the Organisation for Economic Co-operation and Development (OECD)...

About PISA - OECD
 oecd.org > pisa/aboutpisa/ +
 The PISA-based Test for Schools is a student assessment tool, used by schools to support research, benchmarking and school improvement.

Program for International Student Assessment (PISA)...
 nces.ed.gov > surveys/pisa/ +
 More information about PISA and resources, including the OECD's PISA reports, PISA assessment frameworks, and international data files, are available at the OECD's website.






Pisa | Article about Pisa by The Free Dictionary
 encyclopedia2.thefreedictionary.com > **Pisa** +
 98,928), capital of Pisa prov., Tuscany, N central Italy, on the Arno River. It is now c.6 mi (9.7 km) from the Tyrrhenian Sea, which once reached the city.

CATHOLIC ENCYCLOPEDIA: Pisa
 newadvent.org > cathen/12110a.htm +

Pisa
 City in Tuscany, Central Italy, straddling the River Arno just before it empties into the Tyrrhenian Sea. It is the capital city of the Province of Pisa. Although Pisa is known worldwide for its leaning tower, the city of over 89,940 residents... [Read more](#)

Population: 89 940
Phone code: 050
Area: 185 km²

Sightseeings

				
Leaning Tower of Pisa	Piazza dei Miracoli	Pisa Baptistry	San Pietro a Grado	Cathedral of Notre-Dame de...

[Wikipedia](#) [Wikidata](#) [Сообщить об ошибке](#)

8 million results found

117		anchor	approach for IR with
118		title	Bayesian smoothing
119		url	using Dirichlet priors
120		whole document	
121	LMIRJM	body	Language model
122		anchor	approach for IR with
123		title	Jelinek-Mercer
124		url	smoothing
125		whole document	
126	Number of slash in URL		
127	Length of URL		
128	Inlink number		
129	Outlink number		
130	PageRank		
131	SiteRank		Site level PageRank
132	QualityScore		The quality score of a web page. The score is outputted by a web page quality classifier.
133	QualityScore2		The quality score of a web page. The score is outputted by a web page quality classifier, which measures the badness of a web page.
134	Query-url click count		The click count of a query-url pair at a search engine in a period
135	url click count		The click count of a url aggregated from user browsing data in a period
136	url dwell time		The average dwell time of a url aggregated from user browsing data in a period

Screendump from <https://www.microsoft.com/en-us/research/project/mslr/>

Ranking at the heart of information retrieval

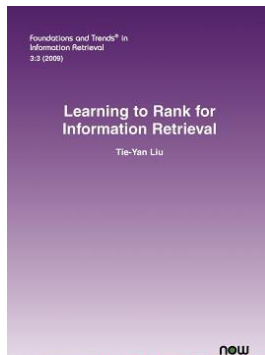
- ▶ Identifying relevant/useful information

Many criteria

- ▶ Text match, semantic match, freshness, popularity, readability, quality score, source score, . . .

Different ways of learning a ranking

- ▶ Point-wise, pair-wise, list-wise



Machine-learned ranking function

- ▶ Assemble a training set of query-document-judgment triples
- ▶ Train classification or regression model on training set
- ▶ For a new query, apply model to all documents (actually: a subset)
- ▶ Rank documents according to model's decisions
- ▶ Return the top K (e.g., $K = 10$) to the user

In principle, any classification/regression method can be used

Big advantage: avoid hand-tuning scoring functions and simply learn them from training data

Bottleneck: need to maintain a representative set of training examples whose relevance assessments must be made by humans



<http://trec.nist.gov>



<http://jugglemum.com/wp-content/uploads/2013/07/011.jpg>

The general picture

- ▶ Documents represented by feature vectors: even if a feature is output of existing retrieval model, one assumes that the parameter in the model is fixed, and only learns the optimal way of combining these features
- ▶ Possibility of combining large number of features very promising: can easily incorporate progress on retrieval model, by including the output of the model as a feature

Core ingredients (for supervised learning to rank)

- ▶ **Input space:** documents represented as feature vectors
- ▶ **Output space:** to facilitate the learning process (e.g., reals even for a binary classification task)
- ▶ **Hypothesis space:** the class of function mapping the input space to the output space
- ▶ **Loss function:** measures to which degree prediction generated by hypothesis is in accordance with ground truth label

Setup

- ▶ n training queries q_i ($i = 1, \dots, n$)
- ▶ associated documents represented by feature vectors $\vec{x}^{(i)} = \{x_j^{(i)}\}_{j=1}^{m^{(i)}}$

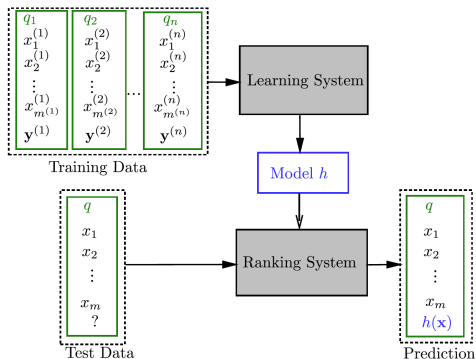


Image taken from Liu [33]

Pointwise approach

- ▶ Input space: feature vector of each single document
- ▶ Output space: relevance degree of each single document
- ▶ Hypothesis space: scoring functions
- ▶ Loss function: regression loss, classification loss, ordinal regression loss

Pros/cons

- Position of documents in the ranked list invisible to loss functions
- Pointwise LTR formulations do not correspond well to IR ranking problem
- Heavily skewed distribution
- + Low complexity
- + Existing classification or regression approaches can be applied directly

Pairwise approach

- ▶ Input space: document pairs (x_u, x_v)
- ▶ Output space: preference $y_{u,v} \in \{-1, +1\}$
- ▶ Hypothesis space: preference function
- ▶ Loss function: pairwise classification loss

Examples

- ▶ Ranking SVM, Rankboost, RankSVM, RankNet

Pros/cons

- Not all mismatches are equally important (top of the ranking vs bottom of the ranking)
- High complexity: quadratic in the number of documents if all documents are considered
- + Problem reduced to binary classification problem

Listwise approach

- ▶ Input space: document set $\vec{x} = \{x_j\}_{j=1}^m$
- ▶ Output space: either scores for all documents or complete permutations of documents
- ▶ Hypothesis space: permutation or scoring function
- ▶ Loss function: IR evaluation measure or smooth approximations of evaluation measures

Examples

- ▶ LambdaRank, LambdaMART, GBDT

Pros/cons

- High complexity
- + Directly optimize for high ranking performance
- + Since we are with lists, listwise properties (document dependencies, diversity, ...) can be learned too

Downsides of offline learning to rank

- ▶ Creating datasets is expensive and therefore infeasible for smaller search engines, such as small web-store search engines
- ▶ May be impossible for experts to annotate documents – personalized search
- ▶ May be substantial mismatch between manually curated labels and user intent
- ▶ Relevance of documents to queries can change over time – news search engine

References for Section 2

- ▶ Dumais [16] – Document relevance can change over time
- ▶ Kohavi et al. [28] – Situations where may be impossible for experts to annotate documents
- ▶ Sanderson [39] – Creating labeled datasets expensive
- ▶ Yue and Joachims [52] – Substantial mismatch between manually curated labels and user intent

3

**Online learning to
rank for information
retrieval**

Online learning to rank

Algorithms that receive input sequentially operate in **online** modality

- ▶ Typical application areas: tasks that involve sequences of decisions, like when you choose how to serve each incoming query in a stream

Batch or **offline** processing does not need human interaction

- ▶ E.g., batch learning proceeds as follows:
 - Initialize the weights
 - Repeat the following steps:
 - Process all the training data
 - Update the weights

Online learning to rank

Algorithms that receive input sequentially operate in **online** modality

- ▶ Typical application areas: tasks that involve sequences of decisions, like when you choose how to serve each incoming query in a stream

Batch or **offline** processing does not need human interaction

- ▶ E.g., batch learning proceeds as follows:

- Initialize the weights
- Repeat the following steps:
 - Process all the training data
 - Update the weights

Typical **offline** computations in information retrieval

- ▶ Any processing that is not query dependent (crawling, document enriching, aggregation, indexing, ...)

Typical **online** computations in information retrieval

- ▶ Any processing that depends on users and their input (query improvement, ranking, presentation, online evaluation, ...)

Ways of learning from interactive feedback

Active learning

- ▶ Start with pool of unlabeled data; pick points at random and get labels
- ▶ Repeat
 - Fit classifier to labels seen so far
 - Query unlabeled point that is closest to boundary (or most uncertain, or most likely to decrease overall uncertainty, ...)

Learning from logged data

- ▶ Challenge is that exploration policy, in which offline data is logged, is not explicitly known

Reinforcement learning

- ▶ Policy and policy evaluation; **much more later**

Contextual bandits

- ▶ Aka associative reinforcement learning
- ▶ Environment chooses context, learner chooses action a , environment responds with reward $r(a)$ – rewards for actions not taken not seen; **much more later**

Back to online learning to rank for information retrieval

Context has a significant influence on peoples' search behavior and goals

- ▶ Addressing all possible settings individually, through supervised learning, is not feasible
- ▶ Need to look for scalable methods that can learn good rankings without expensive tuning

Learn online, i.e., directly from natural interactions with users

- ▶ Continuously adapt and improve rankings to specific deployment setting
- ▶ Continue to learn for as long as there are users

Learning from interactions

Clicks etc. are natural by-product of normal search engine use, but consciously made to reflect users' preferences

Using interaction data to gain insights in user preferences poses several challenges:

- ▶ At most noisy indicators that may correlate with preference.
- ▶ Strongly affected by how results are presented (e.g., position on the result page)
- ▶ Effects may systematically influence (i.e., bias) which results are clicked
- ▶ Distinguishing effects from those of true ranking quality may be hard

Learning directly from user interactions fundamentally different from supervised LTR approaches

- ▶ There, training data assumed sampled i.i.d. from underlying distribution
- ▶ Absolute and reliable labels provided by professional annotators

Learning directly from user interactions fundamentally different from supervised LTR approaches

- ▶ There, training data assumed sampled i.i.d. from underlying distribution
- ▶ Absolute and reliable labels provided by professional annotators

When learning from user interactions, search engine has no control over queries it receives, it only receives feedback on result lists it presents to users, and it has to present high quality result lists while learning, to satisfy user expectations

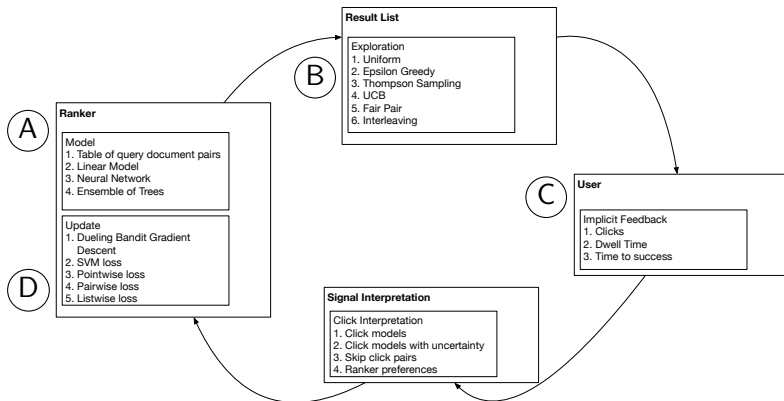
Learning directly from user interactions fundamentally different from supervised LTR approaches (cont'd)

OLTR algorithms must obtain feedback for effective learning while simultaneously utilizing what has already been learned to produce high quality results

- ▶ **exploration-exploitation dilemma**
- ▶ If system presents only document lists that it expects will satisfy user, it cannot obtain feedback on other, potentially better, solutions.
- ▶ However, if it presents document lists from which it can gain lot of new information, it risks presenting bad results to user during learning

Designing an OLTR algorithm

- ranker (A) + exploration strategy (B) + observed signal (C) + way to use observations to update ranker (D)



Choice 1: ranker

- ▶ Typically maps documents to relevance scores used to produce result list
- ▶ Can operate at the level of document id's or at the level of feature representations
- ▶ Working at the level of feature representations allows the ranker to work with previously unseen query-document pairs
- ▶ Well-known examples: linear rankers, neural networks, ensembles of decision trees

Choice 2: exploration strategy

- ▶ In case no exploration is performed, sort documents by relevance score

- ▶ Epsilon Greedy
 - Result list randomly perturbed by injecting random documents in random positions
 - Amount of exploration regulated by parameter ϵ
 - Simple, unbiased data, ignorant of context

- ▶ Thompson Sampling
 - High quality documents should be explored first
 - Docs with uncertain quality estimates should receive higher priority
 - Reasons about doc quality and uncertainty in Bayesian manner

Choice 2: the exploration strategy (cont'd)

- ▶ Upper Confidence Bound
 - Reasons about doc quality and uncertainty in frequentist manner

- ▶ FairPairs
 - Flips adjacent pairs of results in ranking presented to user according to randomized scheme; allows clickthrough data to provide relevance judgments that are unaffected by presentation bias

- ▶ Interleaving
 - Generate candidate ranker, usually from neighborhood of current
 - Create result list that has docs from current and candidate rankers

Choice 3: observed signal

- ▶ User interacts with SERP and OLTR algorithm learns by observing interactions with SERP
- ▶ Several signals the algorithm may consider
 - Raw observations – clicks and dwell times
 - More complex metrics – time to success, or even delayed observations such as engagement
- ▶ Document level feedback (clicks, dwell time) can be interpreted to compute absolute relevance of documents or as relative preference between documents
- ▶ In case result list was constructed using interleaving, document level feedback can be interpreted as preference between rankers

Choice 4: using observations to modify ranker

- ▶ OLTR algorithms updating ranker using the user feedback
- ▶ Rankers that work at level of document id's may update the relevance estimates of the documents for the query for which they received feedback.
- ▶ Feature based rankers optimize a loss function
 - Based on the estimates of the relevance of the documents – point wise, mis-ordered pairs, or the entire list

References for Section 3

- ▶ Hofmann et al. [20, 22] – Importance of balancing exploration and exploitation
- ▶ Hofmann et al. [20], Yue and Joachims [52] – How online LTR addresses the shortcomings of offline LTR
- ▶ Radlinski and Joachims [36] – FairPairs algorithm for randomization
- ▶ Swaminathan and Joachims [46] – How online LTR relates to, and differs from, other tasks such as learning from labelled data, active learning and learning from logged interactions

4 Reinforcement learning and bandits

Reinforcement learning and bandits

What is Reinforcement Learning?

How is OLTR a Reinforcement Learning Problem?

What are Bandits and Contextual Bandits?

OLTR as a Contextual Bandits Problem

What is Reinforcement Learning?

Computational approach to learning from interactions with an environment

- ▶ There is no supervisor, only a reward signal
- ▶ Feedback may be delayed, not instantaneous
- ▶ Time really matters (sequential, non i.i.d data)
- ▶ Agent's actions affect the subsequent data it receives

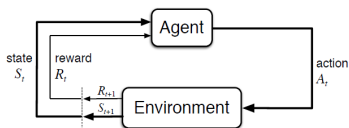


Figure 3.1: The agent–environment interaction in reinforcement learning.

Image taken from Sutton and Barto, 2012.

Key components

- ▶ Agent - our algorithm!
- ▶ Actions: $a \in 1, 2, \dots, k$, we get to choose them
- ▶ Environment – what our algorithm interacts with
- ▶ Rewards $r_a \in [0, 1]$ for action a – aim to get as much rewards as possible
- ▶ States – the world changes as we interact with it
- ▶ Time – always moves forward

Successful applications of Reinforcement Learning:

- ▶ Drive a car
- ▶ Play Atari better than humans
- ▶ Control a power station
- ▶ Fly stunt manoeuvres in a helicopter

See <http://www.youtube.com/playlist?list=PL5nBAYUyJTrM48dViibyi68urttMlUv7e> for examples

How is OLTR a Reinforcement Learning Problem?

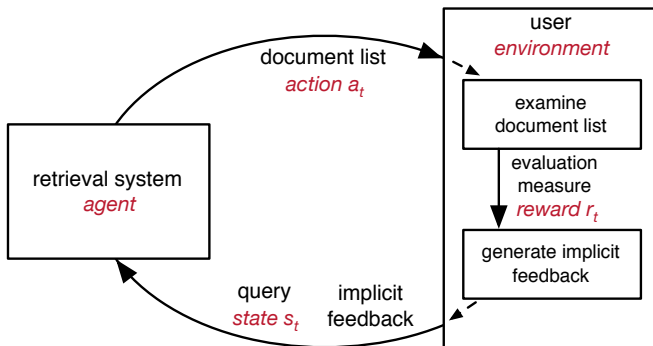
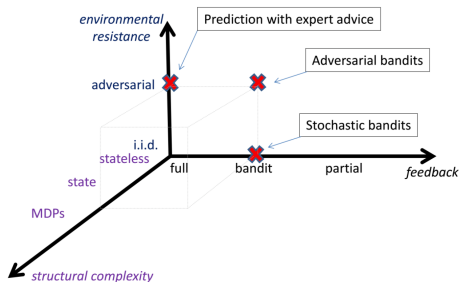


Image taken from K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in learning to rank online. ECIR 2011, April 2011.



- ▶ Contextual bandits (discussed next)
 - Related IR problem: learning to retrieval
- ▶ K -armed bandits
 - Related IR problem: optimizing web advertising revenue
- ▶ Dueling bandits (discussed later)
 - Related IR problem: ranker comparisons ("duels")
- ▶ ...

Image taken from
<https://sites.google.com/site/spaceofonlinelearningproblems/>

Bandit problems

- ▶ RL problem where actions do not affect future states
- ▶ Addresses key challenge: how to balance exploration and exploitation

Given 3-armed bandit and payoff in previous rounds, which arm should we pull?

Minimize **regret**: expected difference between the reward associated with an optimal strategy and the actually collected rewards



Image taken from
<http://www.searchenginepeople.com/wp-content/uploads/2016/06/one-armed-bandits.jpg>

Bandit approaches

► ϵ -greedy

- Explore with probability $\epsilon \in [0, 1]$
- Exploit (best option known so far) with probability $1 - \epsilon$
- Simple, often works well in practice (with parameter tuning).

► Bandit approaches: UCB

- UCB = upper confidence bound
- Assumes stochastic setting (arm payoff is drawn from a probability distribution)

Deterministic policy: UCB1.

Initialization: Play each machine once.

Loop:

- Play machine j that maximizes $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$, where \bar{x}_j is the average reward obtained from machine j , n_j is the number of times machine j has been played so far, and n is the overall number of plays done so far.

Screendump taken from Auer et al. [5]

Bandit approaches (cont'd)

► EXP3

- EXP – exponential weighting
- Regret guarantees against arbitrary (adversarial) opponent strategies

Algorithm Exp3

Parameters: Real $\gamma \in (0, 1]$

Initialization: $w_i(1) = 1$ for $i = 1, \dots, K$.

For each $t = 1, 2, \dots$

1. Set

$$p_i(t) = (1 - \gamma) \frac{w_i(t)}{\sum_{j=1}^K w_j(t)} + \frac{\gamma}{K} \quad i = 1, \dots, K.$$

2. Draw i_t randomly according to the probabilities $p_1(t), \dots, p_K(t)$.

3. Receive reward $x_{i_t}(t) \in [0, 1]$.

4. For $j = 1, \dots, K$ set

$$\begin{aligned} \hat{x}_j(t) &= \begin{cases} x_j(t)/p_j(t) & \text{if } j = i_t \\ 0 & \text{otherwise,} \end{cases} \\ w_j(t+1) &= w_j(t) \exp(\gamma \hat{x}_j(t)/K). \end{aligned}$$

Screendump taken from Auer et al. [6]

Bandit approaches (cont'd)

► Thompson sampling

- Addresses exploration-exploitation challenge using a simple principle
 - maintain a distribution over the reward for actions
 - when taking an action, sample from that distribution and act optimally according to the sample
 - use the observed reward to update belief on reward distribution

What are Contextual Bandits?

State transitions are independent – our actions only determine reward but don't change the world

Much simpler than general RL problem!

- ▶ Immediate reward
- ▶ No planning
- ▶ Observe reward for a single action at a time

Contextual bandit problem

At each timestep, observe side information or context x (e.g., query, user history) before taking an action

- ▶ Distribution $(x, \vec{r}) \sim P$
- ▶ x – context
- ▶ $\vec{r} = (r_1, r_2, \dots, r_k)$
- ▶ $a \in 1, 2, \dots, k$ is the arm to be pulled
- ▶ $r_a \in [0, 1]$ is the reward for arm a

OLTR as a Contextual Bandit Problem

- ▶ Agent – search engine (or: a ranker)
- ▶ Context – user + query
- ▶ Action – SERP
- ▶ Reward – Clicks, dwell time, etc.

References for Section 4

- ▶ Auer [4] – Illustrates the importance of formal analysis and present k-armed bandits
- ▶ Auer [4], Langford and Zhang [31] – Bandit algorithms as an important formalism behind many online LTR methods
- ▶ Kveton et al. [30] – Cascading bandits as a way to formalize the online LTR setting
- ▶ Langford and Zhang [31] – Contextual bandits as a way to formalize the online LTR setting
- ▶ Hofmann et al. [20], Sutton and Barto [45] – Connection to reinforcement learning

5 Online signals to learn from

Online signals to learn from

What do we want to optimize?

What can we observe?

What is the connection between observed feedback and hidden quality?

Types of feedback

Clicks and models

What would we like to be able to do with those signals?

What do we want to optimize?

- ▶ Happiness of the users, revenue of the advertisers, our revenue?
- ▶ Classical offline measures based on assessed relevance – NDCG, ERR, P@10
- ▶ Long term online measures – Engagement, Abandonment
- ▶ Short term online measures – CTR, Time to Success

CTR and user satisfaction

- ▶ Click through rate (CTR) often used as surrogate for metric for user engagement and interest
- ▶ Most content recommendation systems essentially optimize for CTR
- ▶ Dwell time found to be a proxy to user satisfaction for recommended content, and complements or even replaces click-based signals
- ▶ Optimizing dwell time achieves better user engagement metrics and improves CTR as well
- ▶ When optimizing towards dwell-based engaging signals rather than high CTR, users may better like the content recommended, come back to the site and click more

CTR and abandonment

- ▶ Good abandonment: when query was successfully addressed by SERP without requiring clicks or query reformulations
- ▶ Accurately categorizing abandoned queries into good and bad abandonment for learning
- ▶ Studies show 41% of abandonments were bad, 32% abandonments were good, with the remaining 27% associated with alternate reasons (e.g., choosing a better query before considering returned SERP)
- ▶ Successful prediction of abandonment rationale uses topical, linguistic, historic features, session interactions, and in-situ judgments from abandoning searchers (acc. ~ 0.88)
- ▶ Revising CTR definition (omitting predicted good abandonments from total counts) leads to slightly better LTR performance

What can we observe?

- ▶ Clicks
- ▶ Dwell time
- ▶ Skip pairs
- ▶ Mouse movements
- ▶ Eye tracking
- ▶ Page downs
- ▶ ...

Why clicks and skips aren't simple “relevant” or “not relevant” judgments

- ▶ Absence of click may mean good abandonment
- ▶ Click is response (and judgment) to snippet
- ▶ Click may be an expression of trust, absence an expression of distrust of the source
- ▶ ...

Clicks are biased

- ▶ Users won't click on things you didn't show them
- ▶ Users are likely to click on things that appear high
- ▶ Users are influenced by how you present documents: snippets, images, colors, font size, grouped with other documents

Key question: to accurately interpret interaction data

What is the connection between observed feedback and hidden quality?

Pointwise feedback – heavy position bias

- ▶ CTR
- ▶ Dwell time

Pairwise feedback – less position bias, but assumes items are independent

- ▶ Skip pairs

Listwise feedback – more holistic approach

- ▶ Time to success
- ▶ Next page
- ▶ Interleaving

Clicks

- ▶ Noise
- ▶ Presentation bias
- ▶ Position bias

Click models

- ▶ Implicit – CTR, Skip pairs
- ▶ Explicit – Cascade, UBM, DBN

What would we like to be able to do with those signals?

- ▶ Directly optimize what we care about
- ▶ What do we care about?
- ▶ Optimize multiple things at once?
- ▶ Optimize with delayed feedback
- ▶ Move to sessions and long sessions

References for Section 5

- ▶ Diriye et al. [15] – Large-scale user study on abandonment
- ▶ Joachims [25] – Interaction signals can be interpreted in many ways
- ▶ Grotov et al. [18], Hofmann et al. [23] – Connection with online and logged based IR evaluation
- ▶ Kim et al. [27], Lefortier et al. [32] – How to use observed user feedback to train ranking models
- ▶ Song et al. [44] – Abandonment, good and bad prediction
- ▶ Yi et al. [51] – Clicks vs dwell time as an indicator of user satisfaction and engagement

6

**Online learning to rank
for IR: Second
encounter**

OLTR: Second encounter

- ▶ Pairwise LTR
- ▶ Listwise LTR

Pairwise OLTR

- ▶ Ranker
- ▶ Exploration Strategy
- ▶ Observed signal
- ▶ Way to use observations to update ranker

Ranker

- ▶ Linear model parametrized by w .
- ▶ $\text{score}_{q.d} = w_t^T X_{q.d}$, where X are the feature values of a query document pair and w_t are the parameters of the model at time t .
- ▶ Sort documents by score.
- ▶ Could we use a non-linear ranker: neural network, trees?

Exploration Strategy

- ▶ ϵ -greedy
- ▶ With probability ϵ inject a random document at each position
- ▶ Pros
 - Easy to understand
 - Easy to implement
 - Unbiased
 - All items will eventually be explored
- ▶ Cons
 - Naive
 - Takes forever
 - Does not take into account uncertainty of relevance
 - Does explore promising items first

Observed signal – clicks

- ▶ Click-skip pairs - a clicked document is better than skipped documents above it.
- ▶ Cascade assumption - items are examined in the order they are presented.
- ▶ Pros
 - Reduces position bias
 - Easy to understand
 - Easy to implement
- ▶ Cons
 - Only negative feedback - ones you are reach the top you can only go down
 - Naive - no click \neq irrelevant, click \neq relevant, no interaction between documents.

Update rule

x_1 is not clicked, x_2 is clicked and x_2 is below x_1 .

Regularization: If $w_{t-1}^T(x_1 - x_2) < 1.0$

Then $w_t = w_{t-1} + \eta(x_1 - x_2) - \eta\lambda w_{t-1}$, where η is the learning rate and λ is a regularizer.

Pairwise LTR

- ▶ Linear ranker
- ▶ ϵ -greedy exploration
- ▶ SVM-like update rule

Listwise LTR

Dueling Bandit Gradient Descent

- ▶ Introduction
- ▶ What sort of signal does it use/need?
- ▶ Why does it work?
- ▶ Practice
- ▶ Bells and whistles
- ▶ Future

Introduction

- ▶ Linear Model
- ▶ Start with a random ranker
- ▶ Explore by choosing random candidate from neighborhood of current ranker
- ▶ Interleave to see which is better → listwise comparison
- ▶ Update current ranker by taking a step towards a better candidate
- ▶ Regret

$$\Delta_T = \sum_{t=1}^T h(w^*, w_t) + h(w^*, w'_t)$$

(w are rankers (at time t); w^* is best ranker in hindsight; $h(w, w') \in [-0.5, 0.5]$ the distinguishability of w, w')

Algorithm 1 Dueling Bandit Gradient Descent

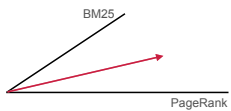
```

1: Input:  $\gamma, \delta, w_1$ 
2: for query  $q_t$  ( $t = 1..T$ ) do
3:   Sample unit vector  $u_t$  uniformly.
4:    $w'_t \leftarrow \mathbf{P}_{\mathcal{W}}(w_t + \delta u_t)$  //projected back into  $\mathcal{W}$ 
5:   Compare  $w_t$  and  $w'_t$ 
6:   if  $w'_t$  wins then
7:      $w_{t+1} \leftarrow \mathbf{P}_{\mathcal{W}}(w_t + \gamma u_t)$  //also projected
8:   else
9:      $w_{t+1} \leftarrow w_t$ 
10:  end if
11: end for

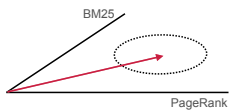
```

Screendump from Yue and Joachims [52]

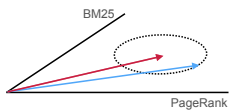
#Clicks



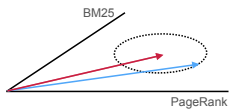
#Clicks



#Clicks

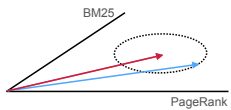


#Clicks



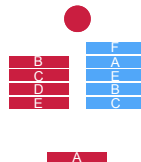
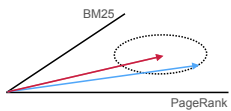
A	F
B	A
C	E
D	B
E	C

#Clicks

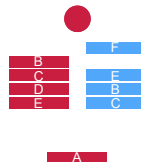
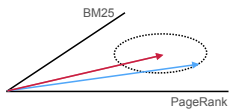


A	F
B	A
C	E
D	B
E	C

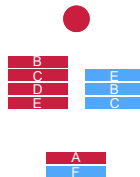
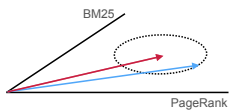
#Clicks



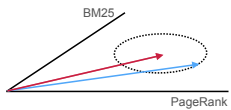
#Clicks



#Clicks



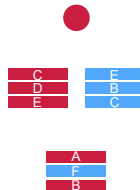
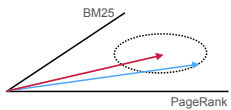
#Clicks



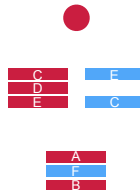
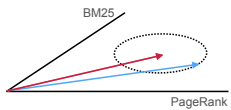
B	E
C	B
D	C
E	

A
F

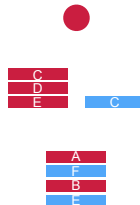
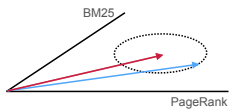
#Clicks



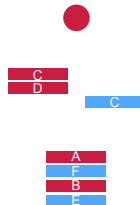
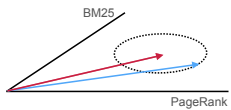
#Clicks



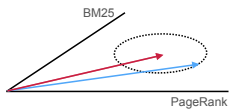
#Clicks



#Clicks



#Clicks



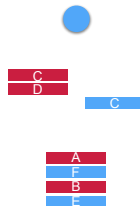
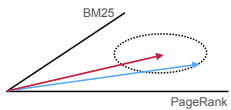
?

C
D

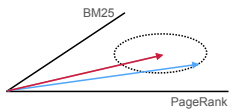
C

A
F
B
E

#Clicks



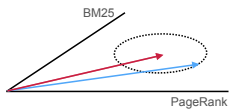
#Clicks



C
D

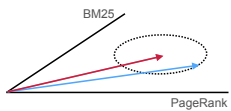
A
F
B
E
C

#Clicks

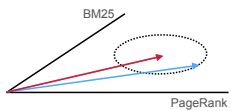


A
F
B
E
C

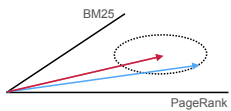
#Clicks



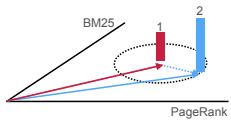
#Clicks



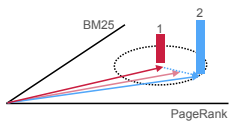
#Clicks



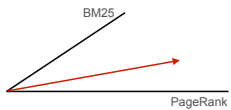
#Clicks



#Clicks



#Clicks



Why does it work?

- ▶ Gradient Descent – compute gradients explicitly
- ▶ Gradient Descent with Bandit Feedback – compute gradients by sampling
- Monte Carlo gradient estimates
- ▶ Gradient Descent with Dueling Bandit Feedback – same as bandit feedback sans the magnitude of the gradients

How well does it work?

- ▶ Regret is sublinear!
- ▶ Proportional to $Time^{\frac{3}{4}}$
- ▶ Proportional to square root of number of dimensions
- ▶ Depends on how erratic our search space is
- ▶ Depends on how noisy the feedback is

Assumptions

- ▶ Differentiable, strictly concave utility function
- ▶ Comparison functions behave like cumulative distribution functions
- ▶ Utility and comparison functions are Lipschitz continuous
- ▶ Comparison function is second order Lipschitz continuous

Parameters

- ▶ Exploration step size
- ▶ Exploitation step size

Probabilistic Interleave

- ▶ Introduce a probabilistic interleave method that constructs result list from distributions over documents
- ▶ Derive sound comparison estimators that exhibit fidelity from the probabilistic interleave model
- ▶ Make estimators more efficient using (1) marginalization and (2) data reuse, and show that fidelity and soundness are maintained

Probabilistic interleave (cont'd)

- Define probability distributions (softmax functions) over docs, e.g.,

$$P_i(d) = \frac{1}{Z} r_i(d)^\tau$$

($r_i(d)$ – rank of d in list l_i ; τ – decay parameter; Z – normalizing constant)

- Interleave per rank:
 - draw softmax (1, 2) to contribute next doc;
 - draw doc from selected distribution
- Any permutation of candidate docs is possible, each ranker can contribute each doc



Images by Katja Hofmann

Candidate Preselection (CPS)

- ▶ Idea 1: Generate several candidate rankers, and select the best one by running a **tournament on historical data**
- ▶ Idea 2: Use probabilistic interleave and importance sampling for ranker comparisons during the tournament
 - Importance sampling needed as candidate ranker may produce different lists from those logged

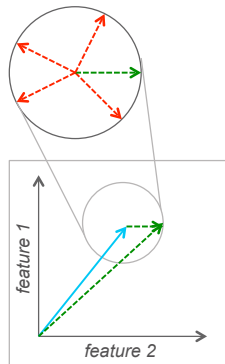


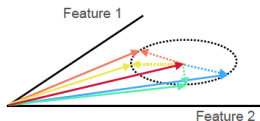
Image by Katja Hofmann

Multileave Gradient Descent

► Why?

- DBGD updates after exploring a single direction – Exploring multiple directions before updating would be beneficial
- Fewer updates would lead to a better ranker – But would be expensive when interleaving was used
- All directions require pairwise comparisons – Multileaved comparisons come to the rescue

- Select a notion of winner: pick a winner and update vs. take mean of all winners and update



A	B	A	D	C
B	D	B	B	A
C	G	H	A	B
D	A	C	I	E
E	C	I	E	G

Image by Anne Schuth

References for Section 6

- ▶ Hofmann et al. [21] – DBGD with Candidate Preselection
- ▶ Oosterhuis et al. [35], Schuth et al. [42] – Probabilistic Multileave Gradient Descent
- ▶ Yue and Joachims [52] – Dueling bandit gradient descent, one of the core methods used in online LTR

Break

Structure of the tutorial

Part 1

- ▶ Introduction
- ▶ Learning to rank for information retrieval
- ▶ Online learning to rank: First encounter
- ▶ Reinforcement learning and bandits
- ▶ Online signals to learn from
- ▶ Online learning to rank: Second encounter

Part 2

- ▶ Experiments and applications
- ▶ Online learning to rank: Third encounter
- ▶ Problems and opportunities
- ▶ Conclusion

7

Experiments and applications

Experiments and applications

Experiments

- ▶ How to run OLTR experiments
- ▶ Sample of outcomes

Applications

- ▶ Some examples
- ▶ Some guidance

How to run OLTR experiments

- ▶ Run a simulation
- ▶ Run someone else's search engine: CLEF LL4IR: Living Labs for IR Evaluation
- ▶ Run someone else's search engine: TREC OpenSearch – Academic Search
- ▶ Run your own search engine

Meet Lerot: <http://bitbucket.org/ilps/lerot>

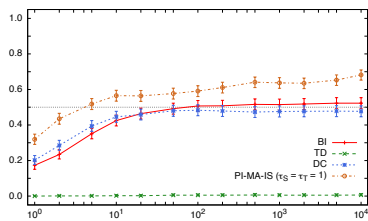


- ▶ Simulated users: produce clicks based on the dependent click model
 - Click probability $P(\text{click} = 1 \mid R)$ given relevance label R
 - Labels R come from human assessors (e.g., LETOR datasets)
 - After click, users stops with $P(\text{stop} = 1 \mid R)$
 - Different choices: perfect, navigational, informational, almost random, random
- ▶ Implements several OLTR and interleaving methods
- ▶ Evaluation: online (discounted cumulative nDCG, per query) and offline (nDCG of current best on held out set)
- ▶ Used in dozens of publications

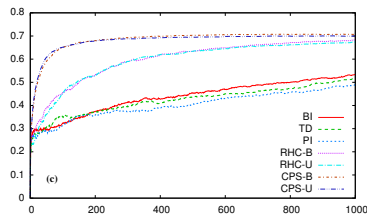


Listing 1: Minimal example of an online learning experiment that uses a list wise learning algorithm and a cascade user model to simulate clicks.

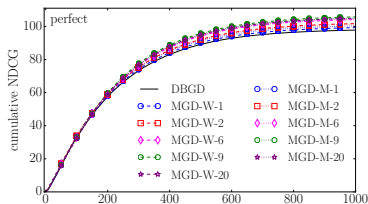
```
import sys, random
import retrieval_system, environment, evaluation,
                                     query
learner = retrieval_system.ListwiseLearningSystem(
                                     [...])
user_model = environment.CascadeUserModel([...])
evaluation = evaluation.NdcgEval([...])
train = query.load_queries(sys.argv[1], [...])
test = query.load_queries(sys.argv[2], [...])
while True:
    q = train[random.choice(train.keys())]
    l = learner.get_ranked_list(q)
    c = user_model.get_clicks(l, q.get_labels())
    s = learner.update_solution(c)
    print evaluation.evaluate_all(s, test)
```



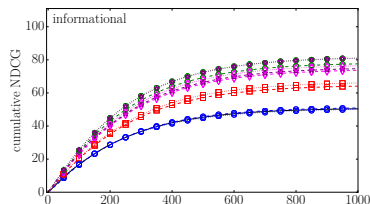
Data reuse



Data reuse (offline performance)



Multileave perfect (online performance)

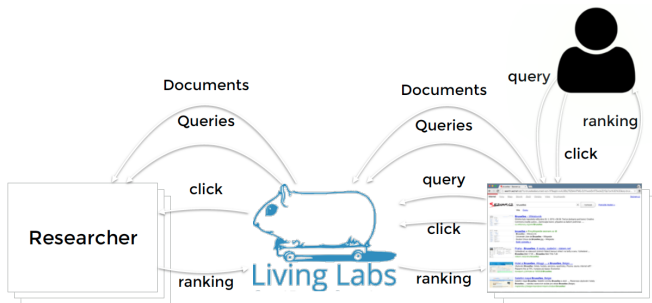


Multileave informational (online performance)



Meet LL4IR: <http://living-labs.net>

- ▶ Overall goal: make information retrieval evaluation more realistic
- ▶ Evaluate retrieval methods in live setting with real users in their natural task environments
- ▶ Focus: medium to large sized organizations with fair amount of search volume
- ▶ Focus on frequent (head) queries
 - Enough traffic on them (both real-time and historical)
 - Ranked result lists can be generated offline



(2015) Product search (Hungarian) and web search (Seznam)



Meet TREC OpenSearch:

<http://trec-open-search.org>

- ▶ Same architecture and setup as LL4IR
- ▶ Different task – Academic Search
 - Easy to comprehend (the setup is already different enough)
 - Connects to current research (e.g., entity linking, normalization)
 - Extendable in future years (related literature, author/venue recommendation)
 - “Subtasks” in the form of several search engines
- ▶ Query and click providers: SSOAR, CiteSeerX, Microsoft Academic Search



Evaluation setup

- ▶ Infer winner: $B > A$ through interleaving
- ▶ Well tested in practice, used at industrial settings
- ▶ Participants are not compared head to head, but transitivity holds in practice: if $A > B$ and $C < B$ then $A > C$
- ▶ Metric: fraction of wins against the search engine

2016 cycle in progress

- ▶ Three evaluation rounds (Jun 1 – July 15, Aug 1 – Sep 15, and Oct 1 – Nov 15)
- ▶ First round a “trial” round, meant to test infrastructure and to allow participants familiarize themselves with the procedure
- ▶ Only CiteSeerX and SOAR are participating in the first round; their test queries are already available

Run your own search engine

- ▶ Meta-search engine – presenting results produced by others, using a public search API
- ▶ Find a niche with a sizable group of users – arXiv
- ▶ If you're at a university, build your own university's (or library's) search engine – Essex, Tilburg, Twente, . . .
- ▶ Web proxy – Intercept, record & modify results before they get to client
- ▶ Browser toolbar – Intercept and modify the page the browser gets

Real world applications of online LTR

Some examples

Some guidance

Example

Recommender system at Netflix

Interactive recommendations

Personalization

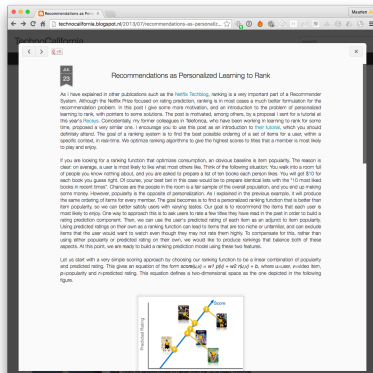


Image from <http://technocalifornia.blogspot.com/2013/07/recommendations-as-personalized.html>

Example

Freshness detection at Yandex

Detect when intent of a query has shifted from non-fresh to fresh

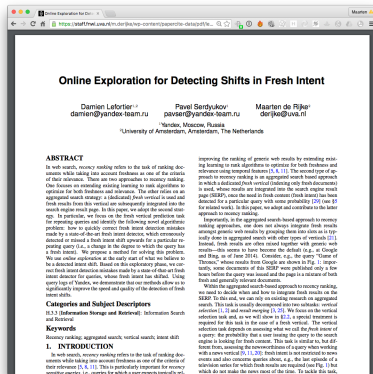


Image from <https://staff.fnwi.uva.nl/m.derijke/wp-content/papercite-data/pdf/lefortier-online-2014.pdf>

Example

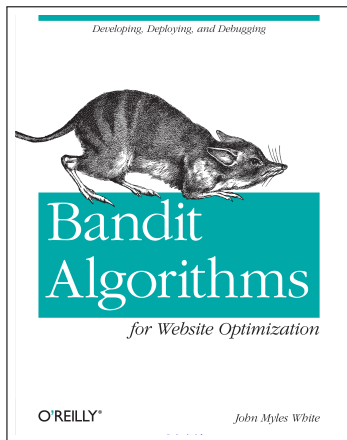


Image from
<http://shop.oreilly.com/product/0636920027393.do>



Startup for self-learning search;
 see SIRIP (Wednesday)

Practicalities

As an aside: great papers by Kohavi et al. on the practicalities of running large-scale A/B tests

Running Controlled Experiments at Scale

Numbers below are approximate to give sense of scale

- In a visit, you're in about 15 experiments

- There is no single Bing.
There are 30B variants (5 per line ^15 lines)
- 90% of users are in experiments.
10% are kept as holdout

- Sensitivity: we need to detect small effects

- 0.1% change in the revenue/user metric > \$1M/year
- Not uncommon to see unintended revenue impact of +/-1% (>\$10M)
- Sessions/UU, a key component of our evaluation criteria ([KDD 2012 paper](#)), is hard to move, so we're looking for small effects
- Important experiments run on 10-20% of users

UI	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5
Ads	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5
Relevance	...				
...					
Feature area					

No such experience reports exist for OLTR (AFAWK)...

Image taken from <http://www.exp-platform.com/Documents/2013-08 ExPScaleKDD.pdf>

Practicalities

- ▶ Performance metrics are not absolute
- ▶ Dependent on volume of user traffic
- ▶ Overhead of experimental framework
- ▶ Model limitations – incrementally updatable
- ▶ Learning limitations – hill climbing

Practicalities

- ▶ **Delayed feedback:** Unlike lab settings, where query A is fired, and clicks are received for A, then query B is fired, and clicks are received, etc., the real-life setting is more like query A is fired, query B is fired, a click for B comes in, query C is fired, a click for A comes in, etc. How to learn from this?
- ▶ **Personalize learning to rank:** How to learn a personalized weight vector? How do you start this, without much evidence for a user? How do you make sure the personalized weights get more important as evidence is collected?
- ▶ **Evaluation:** How feasible is interleaving as evaluation method when your learning method also depends on it?
- ▶ Distributing the learning to handle the scale
- ▶ Handling the risk of the model changing drastically without being able to roll-back to a previous model easily

Practicalities of a different nature

- ▶ Ethics of online experimentation
- ▶ Should you tell your users that they are in one or more experiments?
- ▶ Should you ask them for consent?
- ▶ Societal debate slowly emerging; engage if you are going to run online experiments; involve (educate) your IRB

References for Section 7

Datasets and resources

- ▶ Balog et al. [7] – Running Tilburg University expert search
- ▶ Djoerd Hiemstra – Running University of Twente search engine
- ▶ Joachims [25] – Using a metasearch engine for experimental purposes
- ▶ Radlinski et al. [37] – Running retrieval experiments on arXiv
- ▶ Schuth et al. [40] – How to run online LTR experiments at home
- ▶ Schuth et al. [41] – CLEF LL4IR: Living Labs for IR Evaluation
- ▶ TREC [49] – TREC OpenSearch – Academic Search

Real world applications

- ▶ Kohavi et al. [28, 29] – A/B experiments at scale

8

**Online learning to
rank: third encounter**

Online learning to rank: Third encounter

K-armed bandits theory

- ▶ UCB
- ▶ Thompson Sampling
- ▶ Cascading Bandits

K-armed bandits applications

- ▶ Finite population of candidate rankers
- ▶ Condorcet winner

Upper confidence bound

- ▶ In case of uncertainty behave optimistically
- ▶ A frequentist approach of selecting most promising items
- ▶ Maintain an estimate of quality and a confidence bound around it
- ▶ Confidence bound

$$c_{t,s} = \sqrt{\frac{1.5 \log t}{s}}$$

grows with time t and shrinks each time we explore the item s

Thompson Sampling

- ▶ Maximize the expected reward with respect to a randomly drawn belief
- ▶ A bayesian approach
- ▶ Maintain a probability distribution over the quality of items
- ▶ Sample qualities of items from this distribution
- ▶ Pick item with highest quality
- ▶ Update the probability distribution

Cascading Bandits

Learn a list of K items that maximizes the probability that the user will be satisfied by at least one of them

Similar to pairwise OLTR

- ▶ Cascade assumption

Difference from pairwise OLTR

- ▶ Pointwise feedback
- ▶ No generalization – maintain relevance estimates for query document pairs
- ▶ Smart exploration – rank items by how promising they are

Cascading bandits

Assume there is only one query

- ▶ Ground set $E = 1, \dots, L$ of L items – webpages
- ▶ Distribution of attraction weights $P(w) = \prod_{e \in E} P_e(w(e))$
- ▶ $P_e(w(e))$ Bernoulli with mean $\bar{w}(e)$
- ▶ Number of items in list K
- ▶ E and K are known, P is unknown

Interaction at time t

- ▶ Environment draws attraction weight $w_t \sim T$
- ▶ Agent chooses a set of K items $A_t = a_1^t, \dots, a_k^t$
- ▶ User clicks first attractive item in A_t , $C_t = \min(1 \leq k < K : w_t(a_k^t) = 1)$
- ▶ Agent receives the index of click as feedback
- ▶ Feedback determines the weight of all observed items $f(A_t, w_t)$
- ▶ Reward is $f(A, w) = 1 - \prod_{k=1}^K (1 - w(a_k))$

Learning in Cascading Bandits

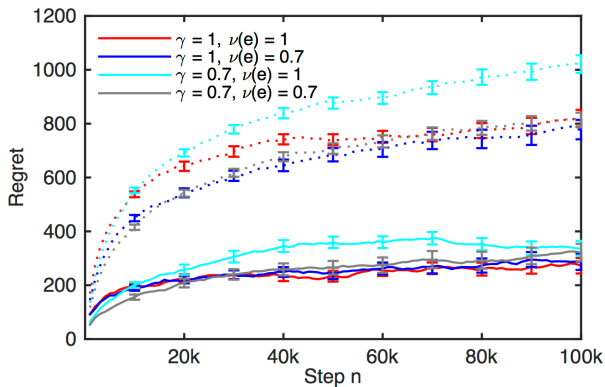
- ▶ Compute UCB on the attraction probabilities of each item
- ▶ Choose the list with the highest UCB
- ▶ Observe the clicks
- ▶ Update the attraction probabilities of each item

Improvements of Cascading Bandits

- ▶ Linear Cascading Bandits
 - UCB
 - Thompson Sampling
- ▶ Learning to rank in the dependent click model, position-based click model
- ▶ Learning to rank with additional structure over users and items

- ▶ UCB $U_t(e) = \hat{w}_{T_{t-1}(e)}(e) + \sqrt{\frac{1.5 \log t}{s}}$
- ▶ Regret of UCB $\mathcal{O}((L - K) \frac{1}{\Delta} \log n)$
- ▶ KL-UCB $U_t(e) = \max(q \in [\hat{w}_{T_{t-1}(e)}(e), 1] : T_{t-1}(e) D_{KL}(\hat{w}_{T_{t-1}(e)}(e) || q) \leq \log t + 3 \log \log t)$
- ▶ Regret of KL-UCB $\mathcal{O}((L - K) \frac{\Delta(1 + \log \frac{1}{\Delta})}{D_{KL}(p - \Delta || p)} \log n)$
- ▶ Regret of any consistent algorithm $\mathcal{O}((L - K) \frac{\Delta}{D_{KL}(p - \Delta || p)} \log n)$

Experiments



Finite population of candidate rankers

- ▶ Why is this relevant?
- ▶ Who is the winner?
- ▶ Relative Upper Confidence Bound
- ▶ mergeRUCB

Finite population of candidate rankers

- ▶ Choose the best ranker from a pool of rankers
- ▶ Speed up AB testing
- ▶ Relative feedback

Best ranker

- ▶ A beats B, B beats C, C beats A
- ▶ Transitivity assumption
- ▶ Condorcet winner

K-armed Dueling Bandits

Preference probabilities $p_{i,j} := \Pr(a_i \text{ beats } a_j)$ for $i, j = 1, \dots, K$

Table: Preference Matrix

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	0.51	0.52	0.53
a_2	0.49	0.5	0.75	0.25
a_3	0.48	0.25	0.5	0.75
a_4	0.47	0.75	0.25	0.5

K-armed Dueling Bandits

Preference probabilities $p_{i,j} := \Pr(a_i \text{ beats } a_j)$ for $i, j = 1, \dots, K$

Goal 1: Find the “best” ranker a_b that beats all other others: i.e. $p_{bj} > 0.5$ for all $j \neq b$

Goal 2: Minimize the number of suboptimal comparisons

Table: Preference Matrix

p_{ij}	a_1	a_2	a_3	a_4
a_1	0.5	0.51	0.52	0.53
a_2	0.49	0.5	0.75	0.25
a_3	0.48	0.25	0.5	0.75
a_4	0.47	0.75	0.25	0.5

Evaluation Measure

Given a comparison between a_i and a_j , define regret as

$$r = \frac{\Delta_i + \Delta_j}{2},$$

where $\Delta_k = p_{bk} - \frac{1}{2}$ and a_b is the best ranker

Cumulative regret = sum of regrets over time

No regret only when a_1 is compared against a_1

Table: Preference Matrix

p_{ij}	a_1	a_2	a_3	a_4
a_1	0.5	0.51	0.52	0.53
a_2	0.49	0.5	0.75	0.25
a_3	0.48	0.25	0.5	0.75
a_4	0.47	0.75	0.25	0.5
Δ_i	0	0.01	0.02	0.03

Relative Upper Confidence Bound

Three phases in each iteration:

1. Run an “optimistic tournament: to pick a **contender**

- Compute $\mu_{ij}(t)$, the frequentist estimate of p_{ij}

Table: Frequentist estimates

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	0.47	0.78	0.67
a_2	0.53	0.5	0.92	0.08
a_3	0.22	0.08	0.5	0.92
a_4	0.33	0.92	0.08	0.5

Relative Upper Confidence Bound

Three phases in each iteration:

1. Run an “optimistic tournament: to pick a **contender**

- ▶ Compute $\mu_{ij}(t)$, the frequentist estimate of p_{ij}
- ▶ Add optimism bonuses to get upper bounds $u_{ij}(t) = \mu_{ij}(t) + \sqrt{\frac{\alpha + \log t}{N_{ij}(t)}}$

Table: Optimistic estimates

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	0.82	1.26	1.15
a_2	0.88	0.5	1.33	0.51
a_3	0.71	0.48	0.5	1.33
a_4	0.82	1.35	0.48	0.5

Relative Upper Confidence Bound

Three phases in each iteration:

1. Run an “optimistic tournament: to pick a **contender**

- ▶ Compute $\mu_{ij}(t)$, the frequentist estimate of p_{ij}
- ▶ Add optimism bonuses to get upper bounds $u_{ij}(t) = \mu_{ij}(t) + \sqrt{\frac{\alpha + \log t}{N_{ij}(t)}}$
- ▶ Choose contender, i.e. an arm that optimistically beats everyone

Table: a_1 and a_2 are potential contenders

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	0.82	1.26	1.15
a_2	0.88	0.5	1.33	0.51
a_3	0.71	0.48	0.5	1.33
a_4	0.82	1.35	0.48	0.5

Relative Upper Confidence Bound

Three phases in each iteration:

1. Run an “optimistic tournament: to pick a **contender**

Table: a_2 chosen as contender

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	0.82	1.26	1.15
a_2	0.88	0.5	1.33	0.51
a_3	0.71	0.48	0.5	1.33
a_4	0.82	1.35	0.48	0.5

Relative Upper Confidence Bound

Three phases in each iteration:

1. Run an “optimistic tournament: to pick a **contender**
2. Pick a **challenger** using UCB relative to the contender
 - ▶ Optimism for challenger = pessimism for contender
 - ▶ Choose challenger most likely to show contender \neq best arm

Table: UCB relative to a_2

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	0.82	1.26	1.15
a_2	0.88	0.5	1.33	0.51
a_3	0.71	0.48	0.5	1.33
a_4	0.82	1.35	0.48	0.5

Relative Upper Confidence Bound

Three phases in each iteration:

1. Run an “optimistic tournament: to pick a **contender**
2. Pick a **challenger** using UCB relative to the contender
3. Compare two arms and update score sheet

Table: UCB relative to a_2

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	0.82	1.26	1.15
a_2	0.88	0.5	1.33	0.51
a_3	0.71	0.48	0.5	1.33
a_4	0.82	1.35	0.48	0.5

Merge Relative Upper Confidence Bound

Main idea: Divide and Conquer

- ▶ Group rankers into small batches

Table: UCB relative to a_2

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	0.51	*	*
a_2	0.49	0.5	*	*
a_3	*	*	0.5	0.75
a_4	*	*	0.25	0.5

Merge Relative Upper Confidence Bound

Main idea: Divide and Conquer

- ▶ Group rankers into small batches
- ▶ Apply RUCB **only inside** each batch and eliminate inferior rankers

Table: UCB relative to a_2

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	0.51	*	*
a_2	0.49	0.5	*	*
a_3	*	*	0.5	0.75
a_4	*	*	0.25	0.5

Merge Relative Upper Confidence Bound

Main idea: Divide and Conquer

- ▶ Group rankers into small batches
- ▶ Apply RUCB **only inside** each batch and eliminate inferior rankers
- ▶ Merge batches whenever the number of rankers is halved and repeat process

Table: UCB relative to a_2

$p_{i,j}$	a_1	a_2	a_3	a_4
a_1	0.5	*	0.52	*
a_2	*	*	*	*
a_3	0.48	*	0.5	*
a_4	*	*	*	*

RUCB Regret bound

Given a dueling bandit problem with a Condorcet winner, the regret accumulated by RUCB takes the form

$$\mathcal{O}(K^2) + \mathcal{O}(K \log T)$$

MergeRUCB Regret bound

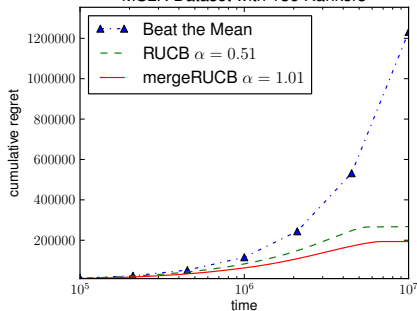
Given a dueling bandit problem with a Condorcet winner and with not “too many” ties, the regret accumulated by mergeRUCB takes the form

$$\mathcal{O}(K \log T)$$

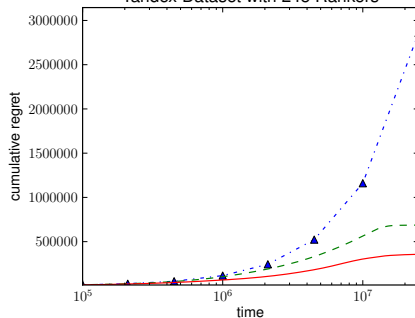
Remark: Main difficulty is dealing with (near) ties

Results: Large-Scale Experiments

MSLR Dataset with 136 Rankers

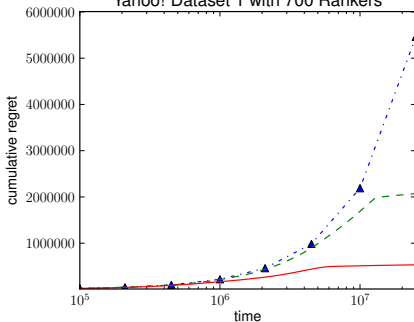


Yandex Dataset with 245 Rankers

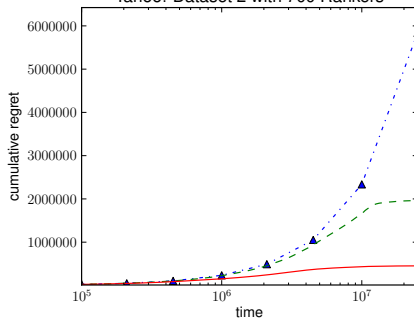


Results: Large-Scale Experiments

Yahoo! Dataset 1 with 700 Rankers



Yahoo! Dataset 2 with 700 Rankers



References for Section 8

- ▶ Busa-Fekete and Hüllermeier [14] – How to perform online LTR with a finite population of candidate rankers, framing it as a K-armed bandits problem
- ▶ Zoghi et al. [56, 57, 58, 59] – Challenges associated with deciding which ranker is the best among a population, the concept of Condorcet winner and Relative Upper Confidence Bound algorithm.

9

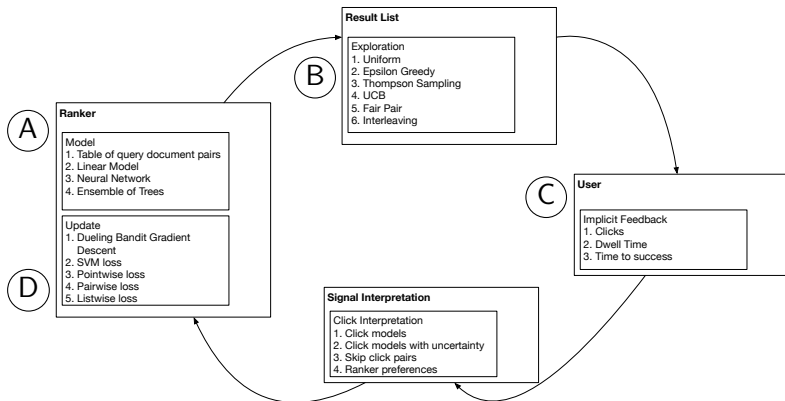
Problems and opportunities

Problems and opportunities

So many things to do!

Designing an OLTR algorithm

- ranker (A) + exploration strategy (B) + observed signal (C) + way to use observations to update ranker (D)



Ranker

Today

- ▶ Mostly focused on linear rankers

We want online versions:

- ▶ Gradient Boosted Regression Trees – current state of art
- ▶ (Deep) (Recurrent) Neural Networks (with Bells and Whistles) – probably next state of art

Gradient Boosted Regression Trees

The main ranker in all search companies, AFAWK

- ▶ Beats the rest on large publicly available datasets
- ▶ A forest of trees, where each tree approximates the error of the trees before
- ▶ Trained on annotated relevances and clicks
- ▶ First trees are much more important than later trees

What do we need to learn?

- ▶ Split values inside each tree
- ▶ Values inside each leave
- ▶ Add new trees

Gradient Boosted Regression Trees Challenges

- ▶ How do we update GBRT incrementally?
 - Add a new tree after collecting a new minibatch?
 - Change the existing trees for each click?
- ▶ How to apply UCB to trees?
- ▶ How to apply Thompson sampling to trees?
- ▶ How to apply DBGD to trees?

Unfolded GBRT

- ▶ GBRT as a non-linear feature transform learned offline
- ▶ Encode the output of GBRT as a binary feature vector – which leaf does the document end up in?
- ▶ Learn a linear model using this representation
- ▶ Challenge – several million features!
- ▶ Bayesian online learning scheme

Neural Networks for IR

- ▶ Still inferior to GBRT offline
- ▶ Can OLTR Neural Networks beat Offline Learning from Relevance Labels GBRT online? Because massive data, right optimisation objective?
- ▶ Drop those features and unleash the real power of Neural Networks! Are you coming to the workshop this Thursday?

OLTR Neural Networks

- ▶ Weight Uncertainty in Neural Networks – Thompson Sampling – works, not yet for ranking but working on it 😊
- ▶ UCB with Neural Networks?
- ▶ DBGD with Neural Networks – scales with $\sqrt{\text{Number of weights}}$ – a lot

Exploration strategy

- ▶ UCB – for GBRT, for Neural Networks
- ▶ Thompson Sampling – for Contextual Bandits – no regret bounds yet
- ▶ DBGD with smart exploration
- ▶ Pairwise OLTR with smart exploration

Exploration strategy + Active Learning

- ▶ Explore items from dense regions
- ▶ Explore items from unexplored regions
- ▶ Explore items which will modify the model the most

Observed signal

- ▶ Today we were focusing on clicks
- ▶ Dwell time, mouse movements, page downs
- ▶ What about mobile?
 - Do the signals mean the same?
 - New signals, zoom-in, swipe, multi-touch?
- ▶ User models
 - Classical – Cascade, DCM, UBM
 - Bayesian – DBN, BBM
 - Rich – intent aware, session aware, non-sequential
- ▶ Sessions
- ▶ Delayed signals
- ▶ Rich calibrated interleavings

What do we want to optimize?

- ▶ Happiness of the users, revenue of the advertisers, our revenue?
- ▶ Long term online measures ? Engagement, Abandonment
- ▶ All of the above?
- ▶ Multi Objective OLTR

We have won! Do we fire the assessors?

- ▶ Combine online and offline
- ▶ Fresh documents

Risk

- ▶ Optimality requires full exploration!
- ▶ Don't show porn to children!
- ▶ How to avoid catastrophic mistakes?

Infrastructure

- ▶ Low lag
- ▶ Distributed OLTR
- ▶ Recent work on parallelizing gradient descent in offline setting – can these methods be used in online setting?

Learning rate

- ▶ Learning rate of a self-learning system is important
- ▶ Prominent in low-traffic settings
 - E.g., site with 500–1,000 queries per month
 - If system takes 1,000 queries and 10,000 clicks to figure out a close-to-optimal weight vector, it means a year of sub-optimal search experience for a low-traffic site
 - Revisit variable-rate gradient descent optimizers for self-learning search
 - As an aside, an experiment that you might want to run: measure user satisfaction at different points while a self-learning system is being trained; ask people what they think of the results they get and whether they would return to the website at 0 queries (when the system starts), when it has been trained on 500 queries, 1,000 queries, etc.
- ▶ Similar challenges in personalization

References for Section 9

- ▶ Burges [11] – Non-linear neural network and tree ensemble-based ranking models
- ▶ Burtini et al. [12], Busa-Fekete and Hüllermeier [13] – Inspiration for online learning algorithms that have not been used in LTR
- ▶ Grotov et al. [19] – Example of state of the art in online LTR and ways to improve it
- ▶ Niu et al. [34], Zinkevich et al. [55] – Influential approaches to paralllizing SGD
- ▶ Tian and Lease [48] – Exploration strategies based on uncertainty estimations [48]

10 **Conclusion**

Let's step back

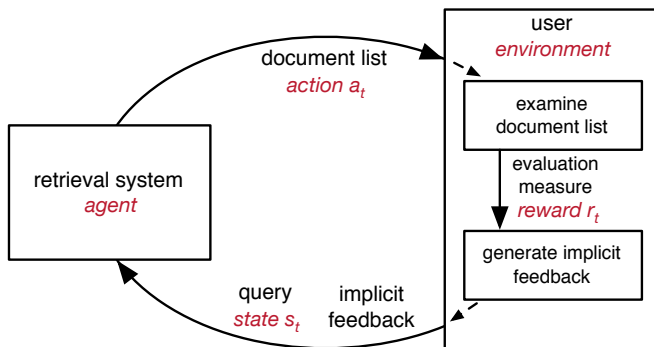
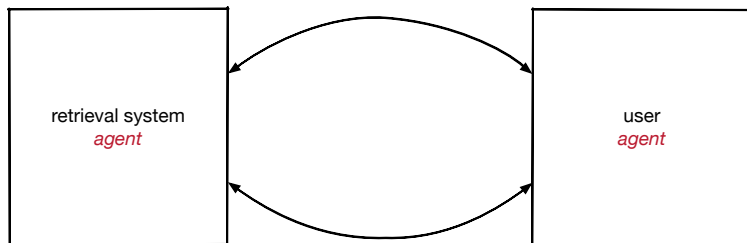
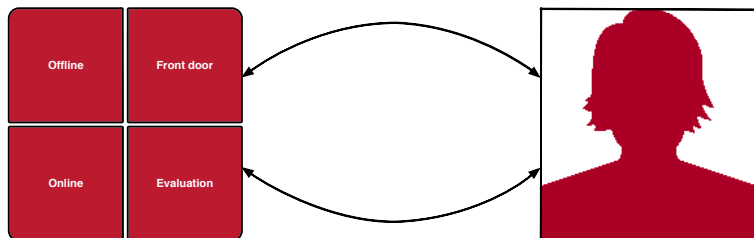
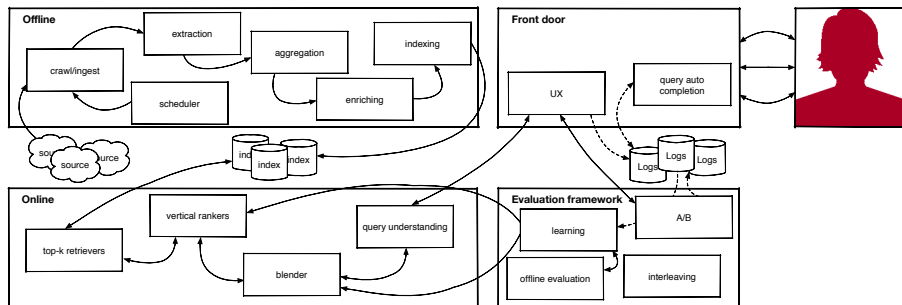


Image taken from K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in learning to rank online. ECIR 2011, April 2011.



Search engine and users are agents that perform actions in response to each other: interactions, result list, interactions, result list, interactions, result list, ...





Thank you!

Acknowledgments



netherlands



All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their employer and/or sponsors.

References I

- [1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*., 2006.
- [2] E. Agichtein, E. Brill, S. Dumais, and R. Rago. Learning user interaction models for prediction web search results preferences. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.
- [3] R. Agrawal, A. Halverson, K. Kenthapadi, N. Mishra, and P. Tsapara. Generating labels from clicks. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 172–181, 2009.
- [4] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003.
- [5] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47, 2002.
- [6] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.
- [7] K. Balog, T. Bogers, L. Azzopardi, M. de Rijke, and A. van den Bosch. Broad expertise retrieval in sparse data environments. In *SIGIR '07*. ACM, July 2007.
- [8] P. Bennett, F. Radlinski, R. White, and E. Yilmaz. Inferring and using location metadata to personalize web search. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, 2011.
- [9] M. Bilenko and R. W. White. Mining the search trails of surfing crowds: Identifying relevant websites from user activity. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, 2008.
- [10] A. Borisov, P. Serdyukov, and M. de Rijke. Using metafeatures to increase the effectiveness of latent semantic models in web search. In *WWW 2016: 25th International World Wide Web Conference*. ACM, April 2016.

References II

- [11] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, June 2010.
- [12] G. Burtini, J. Loeppky, and R. Lawrence. A survey of online experiment design with the stochastic multi-armed bandit. *CoRR*, abs/1510.00757, 2015. URL <http://arxiv.org/abs/1510.00757>.
- [13] R. Busa-Fekete and E. Hüllermeier. A survey of preference-based online learning with bandit algorithms. In *Algorithmic Learning Theory: 25th International Conference, ALT 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, pages 18–39, Cham, 2014. Springer International Publishing.
- [14] R. Busa-Fekete and E. Hüllermeier. A survey of preference-based online learning with bandit algorithms. In *ALT '14*, number 8776 in LNCS, pages 18–39. Springer, 2014.
- [15] A. Diriyee, R. White, G. Buscher, and S. Dumais. Leaving so soon?: Understanding and predicting web search abandonment rationales. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12*, pages 1025–1034, New York, NY, USA, 2012. ACM.
- [16] S. T. Dumais. The web changes everything: Understanding and supporting people in dynamic information environments. In *Research and Advanced Technology for Digital Libraries, 14th European Conference, ECDL 2010*, 2010.
- [17] A. Grotov and M. de Rijke. Online learning to rank for information retrieval: A survey. *Draft*, 2016.
- [18] A. Grotov, S. Whiteson, and M. de Rijke. Bayesian ranker comparison based on historical user interactions. In *SIGIR 2015: 38th international ACM SIGIR conference on Research and development in information retrieval*. ACM, August 2015.
- [19] A. Grotov, M. de Rijke, and S. Whiteson. Online LambdaRank. In *Submitted*, 2016.
- [20] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in learning to rank online. In *ECIR 2011: 33rd European Conference on Information Retrieval*. Springer, April 2011.
- [21] K. Hofmann, A. Schuth, S. Whiteson, and M. de Rijke. Reusing historical interaction data for faster online learning to rank for information retrieval. In *WSDM 2013: International Conference on Web Search and Data Mining*. ACM, February 2013.

References III

- [22] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval Journal*, 16(1):63–90, February 2013.
- [23] K. Hofmann, S. Whiteson, and M. de Rijke. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Transactions on Information Systems*, 31(3):Article 18, October 2013.
- [24] K. Hofmann, S. Whiteson, A. Schuth, and M. de Rijke. Learning to rank for information retrieval from user interactions. *ACM SIGWEB Newsletter*, (Spring):5:1–5:7, April 2014.
- [25] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.
- [26] T. Joachims. Evaluating retrieval performance using clickthrough data. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining*. Physica/Springer Verlag, 2003.
- [27] Y. Kim, A. Hassan, R. W. White, and I. Zitouni. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 193–202, New York, NY, USA, 2014. ACM.
- [28] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: Survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, 2009.
- [29] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann. Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '13, pages 1168–1176, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2174-7. doi: 10.1145/2487575.2488217. URL <http://doi.acm.org/10.1145/2487575.2488217>.
- [30] B. Kveton, C. Szepesvári, Z. Wen, and A. Ashkan. Cascading bandits. *CoRR*, abs/1502.02763, 2015. URL <http://arxiv.org/abs/1502.02763>.
- [31] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 817–824. Curran Associates, Inc., 2008.

References IV

- [32] D. Lefortier, P. Serdyukov, and M. de Rijke. Online exploration for detecting shifts in fresh intent. In *CIKM 2014: 23rd ACM Conference on Information and Knowledge Management*. ACM, November 2014.
- [33] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, Mar. 2009.
- [34] F. Niu, B. Recht, C. Ré, and S. J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011.
- [35] H. Oosterhuis, A. Schuth, and M. de Rijke. Probabilistic multileave gradient descent. In *ECIR 2016: 38th European Conference on Information Retrieval*, LNCS. Springer, March 2016.
- [36] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *AAAI '06*, 2006.
- [37] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*. ACM, 2008.
- [38] F. Radlinski, M. Kurup, and T. Joachims. Evaluating search engine relevance with click-based metrics. In J. Fürnkranz and E. Hüllermeier, editors, *Preference Learning*, pages 337–362. Springer, 2010.
- [39] M. Sanderson. Test collection based evaluation of information retrieval systems. *Found. & Tr. Inform. Retr.*, 4(4):247–375, 2010.
- [40] A. Schuth, K. Hofmann, S. Whiteson, and M. de Rijke. Lerot: an online learning to rank framework. In *Living Labs for Information Retrieval Evaluation workshop at CIKM'13*, 2013.
- [41] A. Schuth, K. Balog, and L. Kelly. Overview of the living labs for information retrieval evaluation (Il4ir) clef lab 2015. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 484–496. Springer, 2015.
- [42] A. Schuth, H. Oosterhuis, S. Whiteson, and M. de Rijke. Multileave gradient descent for fast online learning to rank. In *WSDM 2016: The 9th International Conference on Web Search and Data Mining*. ACM, February 2016.
- [43] A. Slivkins, F. Radlinski, and S. Gollapudi. Ranked bandits in metric spaces: learning optimally diverse rankings over large document collections. Technical report, arXiv preprint arXiv:1005.5197, 2010.

References V

- [44] Y. Song, X. Shi, R. White, and A. H. Awadallah. Context-aware web search abandonment prediction. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 93–102, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2257-7. doi: 10.1145/2600428.2609604. URL <http://doi.acm.org/10.1145/2600428.2609604>.
- [45] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press., 1998.
- [46] A. Swaminathan and T. Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. *CoRR*, abs/1502.02362, 2015. URL <http://arxiv.org/abs/1502.02362>.
- [47] J. Teevan, S. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, 1995.
- [48] A. Tian and M. Lease. Active learning to maximize accuracy vs. effort in interactive information retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 145–154, New York, NY, USA, 2011. ACM.
- [49] TREC. OpenSearch track. <http://trec-open-search.org>, 2016.
- [50] A. Vorobev, D. Lefortier, G. Gusev, and P. Serdyukov. Gathering additional feedback on search results by multi-armed bandits with respect to production ranking. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1177–1187. ACM, 2015.
- [51] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan. Beyond clicks: Dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 113–120. ACM, 2014.
- [52] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML '09*, 2009.
- [53] W. Zhang and R. Jones. Comparing click logs and editorial labels for training query rewriting. In *Proceedings of the International Conference on the World Wide Web (WWW)*, 2007.
- [54] Z. Zhu, W. Chen, T. Minka, C. Zhu, and Z. Chen. A novel click model and its applications to online advertising. In *ACM international conference on Web search and data mining (WSDM '10)*, 2010.

References VI

- [55] M. A. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized stochastic gradient descent. In *NIPS*, 2010.
- [56] M. Zoghi, S. Whiteson, M. de Rijke, and R. Munos. Relative confidence sampling for efficient on-line ranker evaluation. In *7th ACM WSDM Conference (WSDM2014)*. ACM, February 2014.
- [57] M. Zoghi, S. Whiteson, R. Munos, and M. de Rijke. Relative upper confidence bound for the k-armed dueling bandit problem. In *ICML 2014: International Conference on Machine Learning*, June 2014.
- [58] M. Zoghi, S. Whiteson, and M. de Rijke. Mergerucb: A method for large-scale online ranker evaluation. In *WSDM 2015: The Eighth International Conference on Web Search and Data Mining*. ACM, February 2015.
- [59] M. Zoghi, S. Whiteson, Z. Karnin, and M. de Rijke. Copeland dueling bandits. In *NIPS 2015*, December 2015.
- [60] M. Zoghi, T. Tunys, L. Li, D. Jose, J. Chen, C. M. Chin, and M. de Rijke. Click-based hot fixes for underperforming torso queries. In *SIGIR 2016: 39th international ACM SIGIR conference on Research and development in information retrieval*. ACM, July 2016.