Toloka

Toloka

ICWE 2022 hands-on tutorial

Web Engineering with Human-in-the-Loop

Dmitry Ustalov, Nikita Pavlichenko, Boris Tseytlin,
Daria Baidakova and Alexey Drutsa

Toloka

Part II
Ranking
and Quality Metrics

Dr. Dmitry Ustalov,
Head of Research

Tutorial Schedule

Part I Intro: 45 min
The Role of HITL in building Search Engines

Part II: 45 min
Ranking and Quality Metrics

Coffee Break: 30 min

Part III: 45 min
Human-in-the-Loop Essentials

Part IV: 45 min
Practice: Websites relevance

Lunch Break: 90 min

Part V: 45 min
Results aggregation and implementation into ML pipeline

Part VI: 90 min
Metric Computation

Coffee Break: 30 min

Part VII: 90 min
Results discussion and conclusion

Plan

1. Introduction
2. Signals
3. Metrics
4. How to Sample Queries?
5. Real Life Examples
6. What Can Go Wrong?
7. Why Crowdsourcing?
8. Datasets
9. Literature

Why use offline quality evaluation?

Why we use offline

1. Online tests are not enough:

- ► Implicit signal
- ► Delayed response
- ► Slow experimentation

Why we use offline

2. DSAT (Dissatisfaction Analysis):

- Why users are dissatisfied
- What's exactly wrong with our service
- Insights for improvement

Why we use offline

3. Users are prone to manipulation:

- Clickbait
- Fraud
- Other manipulations

Why you might want offline

1

Baseline for product launch

Poor quality = zero retention

2

Detect malfunction before release

Saves money and reputation

3

Draw insights

Where and how to improve your service

Signals

How to measure quality in offline setting?

Model

- Assume we have a user u who interacts with a service by sending some sort of a query q

- Service respondes to query q with array of objects $r_1, \ldots, r_n$ (or a single object r_1)

Model

What we need to do

1. Evaluate every response object r_i with some quality measure s_i (create a signal)

2. Aggregate s_i to overall measure of quality (create a metric)

Signals

Model

Examples

1

Search engine

- Text search
- Image search
- Ecommerce goods search

2

Recommendations

- Music feed
- Content feed
- Social media feed

3

Moderation

- Service quality assurance
- Social media business account behavior

Signals

In order to calculate metric, we need to estimate response objects.

It can be done through multiple approaches

- Pointwise
- Listwise
- Pairwise

Signals are usually obtained through experts or crowdsource platforms, less commonly — from precomputed data

Pointwise

Given a query q and a single response r_i, we can judge how well does this object match to a user query

Pros	Cons
Easy to obtain	Low resolution

Pointwise

Examples

1

Binary relevance

- 1 or 0

2

Multiple grade relevance

- Relevant
- Semi-relevant
- Non-relevant
- Etc.

3

Match score from 0 to 100%

Listwise

Order all objects at once and use ranks as signal

Useful in training ML algorithms

Pros	Cons
► Provides full information ► Judge has all available context	► Expensive ► Inconsistent ► Relative

Pairwise

Pointwise is of low resolution, listwise is inconsistent

Pairwise comparisons tackle both of this problems, they are a perfect example of task decomposition.

Pros	Cons
► Consistent	► Still quite expensive
► Simple	► Relative signal

1. Ranking: Compare, don't score https://ieeexplore.ieee.org/abstract/document/6120246

Which one?

1. In the beginning, use **pointwise** as baseline
2. When you have a working service, use **pairwise** (for incremental improvements)

We will focus on pointwise evaluation in our practice, but later we will show how to address the more advanced pairwise setting.

Metrics

Metrics

From signal to metric — how to aggregate?

Ranking metrics

1. Mean Average Precision (mAP) measures trade-off between precision and recall going down through service response
2. Normalized Discounted Cumulative Gain (nDCG) measures quality of objects with discount factor
3. Expected Reciprocal Rank (ERR) is a cascade model of user interaction with service response

mAP (mean average precision)

Let us recall some definitions from binary classifier ($s_i \in \{0, 1\}$):

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision@k and Recall@k: precision and recall over top-k elements

		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

mAP (mean average precision)

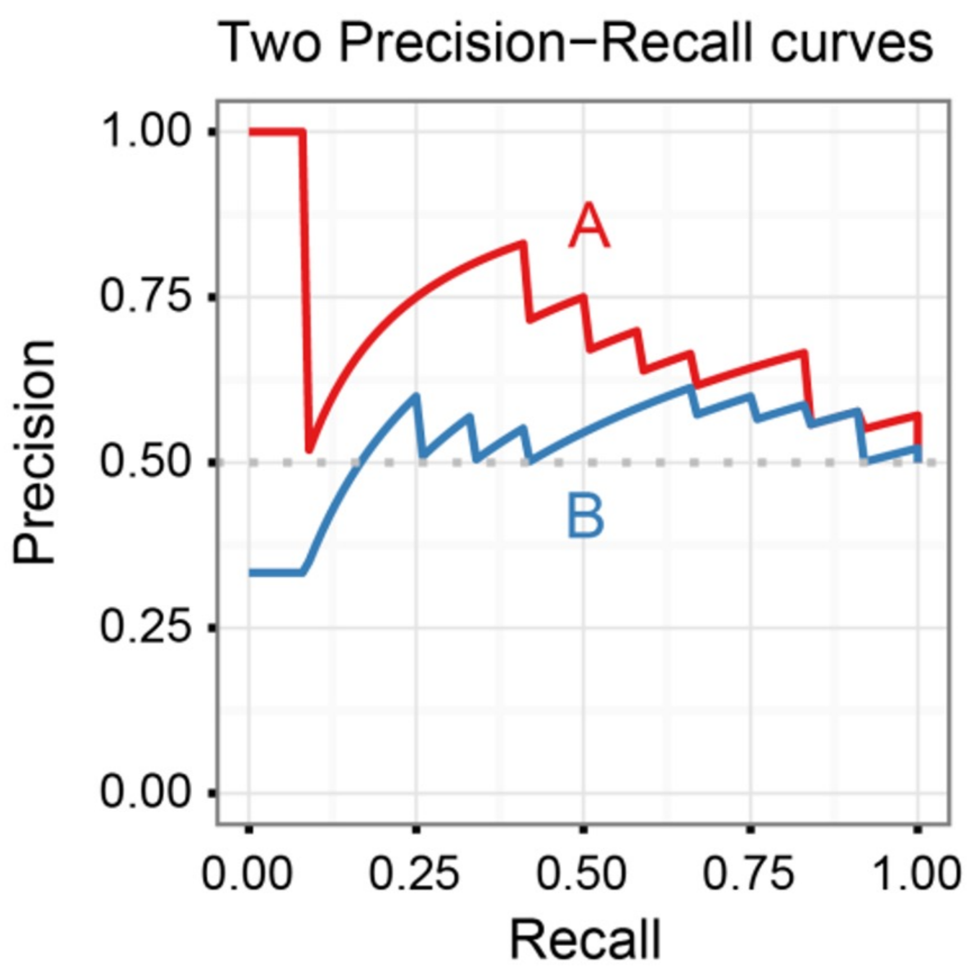
How precision and recall changes going down the list?

1. Recall increases (non-decreasing function)
2. Precision can be arbitrary

Area under precision-recall curve is:

1. Maximum for perfect order
   (positive objects on top, negative on bottom)
2. Minimum for the worst order

We can define precision as function of recall $p(r)$



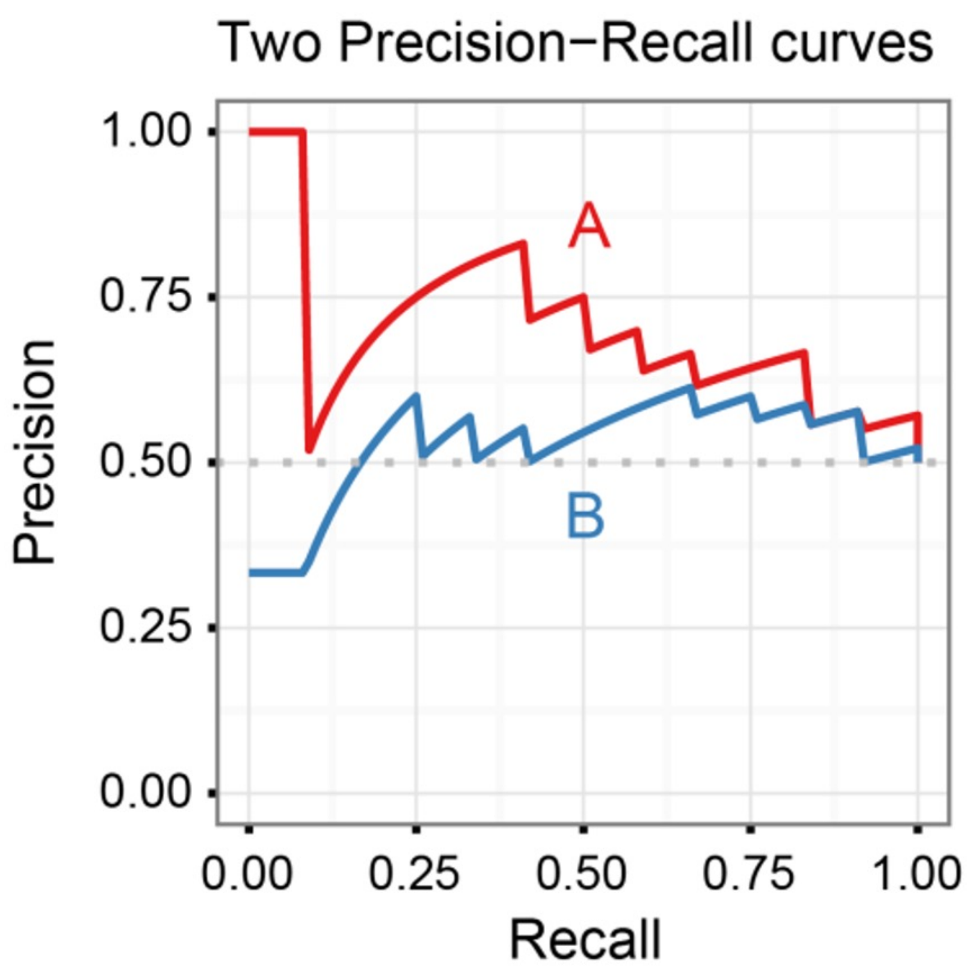
mAP (mean average precision)

We can define Average Precision as the following:

$$AP = \int_0^1 p(r)dr\,.$$

r is recall
$p(r)$ is precision

AP is the area under precision-recall curve (**precision-recall AUC**)



mAP (mean average precision)

In a simple discrete case, previous equation can be transformed into:

$$AP = \sum_{i=1}^{n} Precision@i \cdot \Delta Recall@i \, ,$$

where $\Delta Recall@i$=Recall@i−Recall@(i−1)

mAP (mean average precision)

Since $\Delta Recall@i$ is positive iff included object is true positive, we can simplify AP to

$$AP = \frac{1}{n} \sum_{i=1}^{n} Precision@i[s_i = 1].$$

Mean average precision is defined as mean AP over set of queries:

$$mAP = \frac{1}{Q} \sum_{q} AP(q).$$

nDCG (normalized discounted cumulative gain)

- Good ranking puts the best objects on top
- Idea: sum signal values of ordered response with some discounter
- The lower the object, the lower its impact on metric is

nDCG (normalized discounted cumulative gain)

We can define discounted cumulative gain (DCG[3]) as following:

$$DCG@k = \sum_{i=1}^{k} \frac{s_i}{d(i)},$$

where $d(i)$ is a discounting factor

3. Cumulated gain-based evaluation of IR techniques https://doi.org/10.1145/582415.582418

nDCG (normalized discounted cumulative gain)

Example of discounters:

Linear: i

Logarithmic: $\log_2(i+1)$

Exponential: 2^i

nDCG (normalized discounted cumulative gain)

Raw DCG cannot be compared between queries, normalization is required

To align values of DCG we can normalized it by ideal DCG:

$$IDCG@k = \sum_{i=1}^{k} \frac{s_{(i)}}{d(i)},$$

where $s_{(i)}$ is i-th object with largest signal available

nDCG (normalized discounted cumulative gain)

Thus, nDCG is defined as following:

$$nDCG@k = \frac{DCG@k}{IDCG@k}$$

Now values are between 0 and 1 and thus cross-query comparable

ERR (expected reciprocal rank)

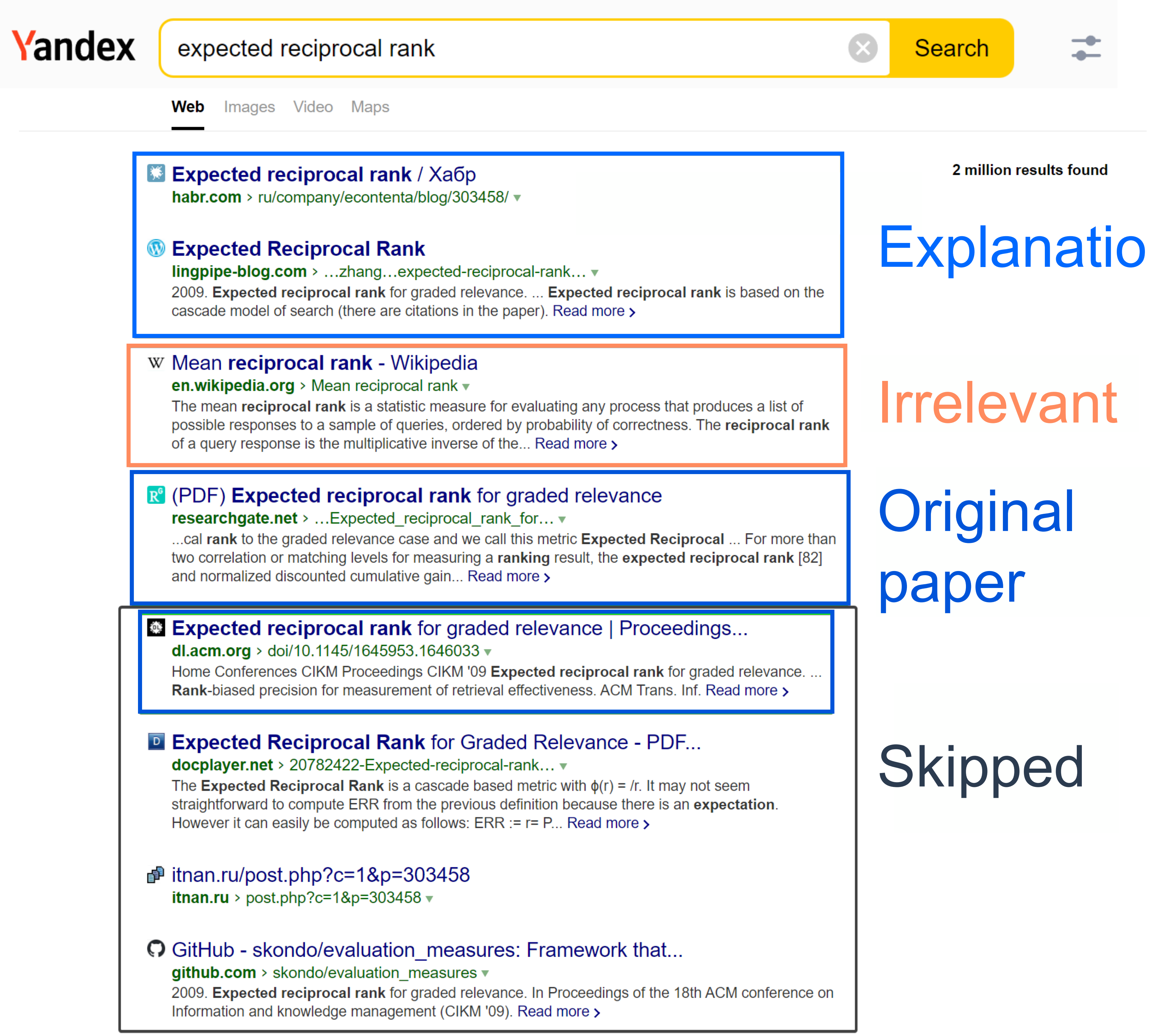
mAP and nDCG metrics

1. Gain profit even on lower positions
2. When user has found answer, everything else doesn’t matter

Improvement: cascade model

1. User go down the ranked list until he finds satisfying result
2. The lower user has to go, the worse performance of a ranker is

ERR (expected reciprocal rank)



Explanations

Irrelevant

Original paper

Skipped

ERR (expected reciprocal rank)

Suppose we have signal values s_i

1. Map s_i to probability of finding answer R_i
2. Use it to model termination rank
   (on which position the user will stop)

ERR (expected reciprocal rank)

Probability of user terminating their session on rank k equals to

$$P(k) = R_k \prod_{i=1}^{k-1} (1 - R_i)\,,$$

where R_i — probability of user to find answer on rank i.

Use 1/s to have a metric with semantic “higher is better”:

$$ERR^4 = \sum_{k=1}^{n} \frac{1}{k} R_k \prod_{i=1}^{k-1} (1 - R_i)\,.$$

4. Expected reciprocal rank for graded relevance https://dl.acm.org/doi/10.1145/1645953.1646033

ERR (expected reciprocal rank)

A few months earlier, another cascade metric was proposed, pFound[5]:

$$pFound = \sum_{i=1}^{n} pLook_i \cdot R_i \,,$$

where:

1. $pLook_i = pLook_{i-1} \cdot (1 - R_{i-1}) \cdot (1 - pBreak)$ — probability that user will interact with object i:
   - ► User looked at object $i-1$
   - ► Did not found answer
   - ► Continued his search
2. $pBreak$ — probability of ending session

5. http://romip.ru/romip2009/15_yandex.pdf

How to Sample Queries?

How to Sample Queries?

What queries to use in offline evaluation?

1

Most popular?

- Beak, simple queries
- Easy to process
- Affect lots of users

2

Unique queries?

- Tail, usually hard or ambiguous
- Huge amount (30%–70% depending on service)

3

Something in the middle?

How to Sample Queries?

Simple idea: take a random sample

1. **Flip a coin with a probability p on every object**

- Heads — use query
- Tails — skip
- On average, $p \cdot N$ queries will be sampled

2. **More sophisticated — reservoir sampling[6]:**

- Every object is considered
- Exactly k objects will be sampled

No guarantee that popular queries will be presented in sample

6. Random Sampling with a Reservoir http://www.cs.umd.edu/~samir/498/vitter.pdf

How to Sample Queries?

Stratified sampling:

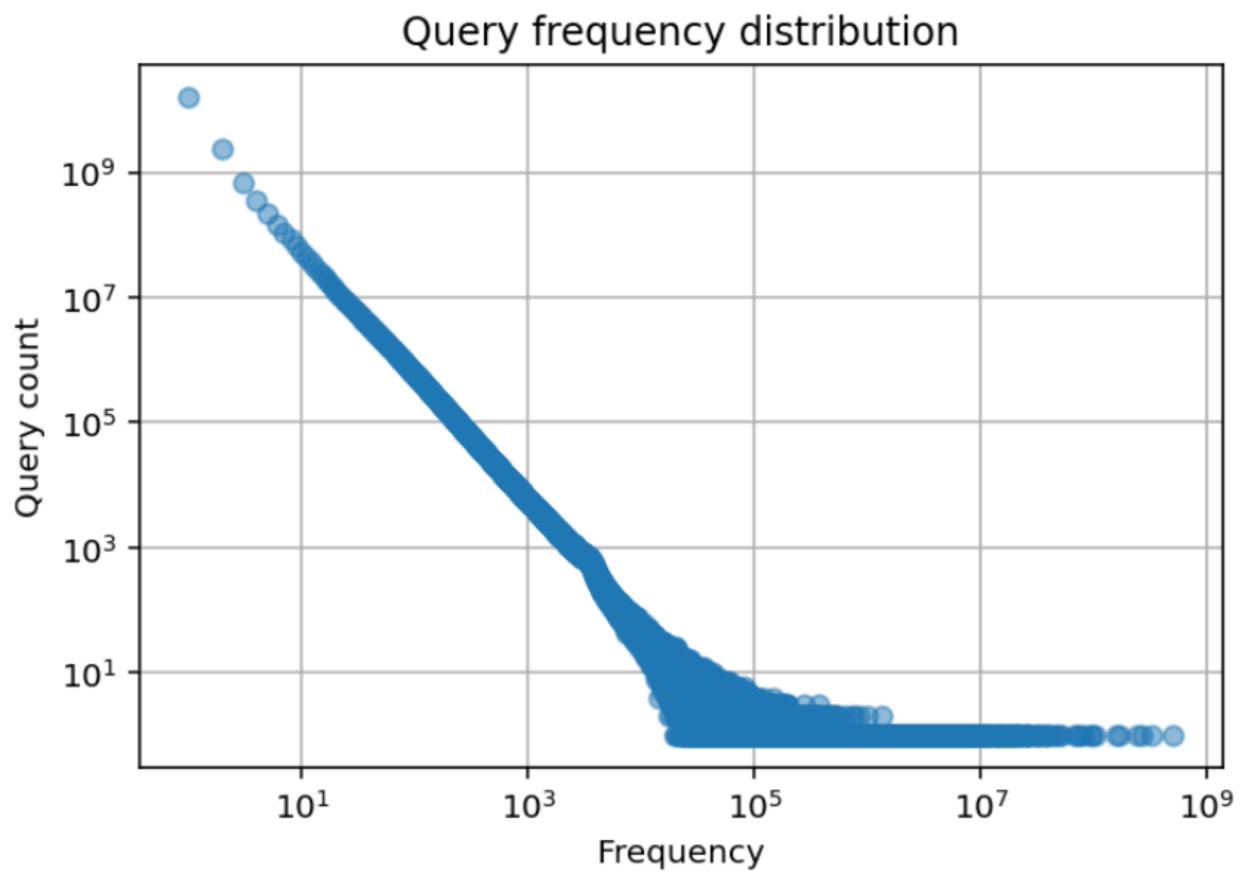
- Each query q_i has frequency f_i
- Order queries by f_i and split them in k buckets Q_k s.t.

$$\sum_{m \in Q_i} f_m \approx \sum_{k \in Q_j} f_k \;\forall i, j,$$

$$\forall i < j \Rightarrow f_m < f_k \;\forall m \in Q_i, k \in Q_j.$$

- After that, sample the necessary amount from every bucket

Guarantees that queries of all frequencies will be presented in a sample



Real Life Examples

Metric purpose

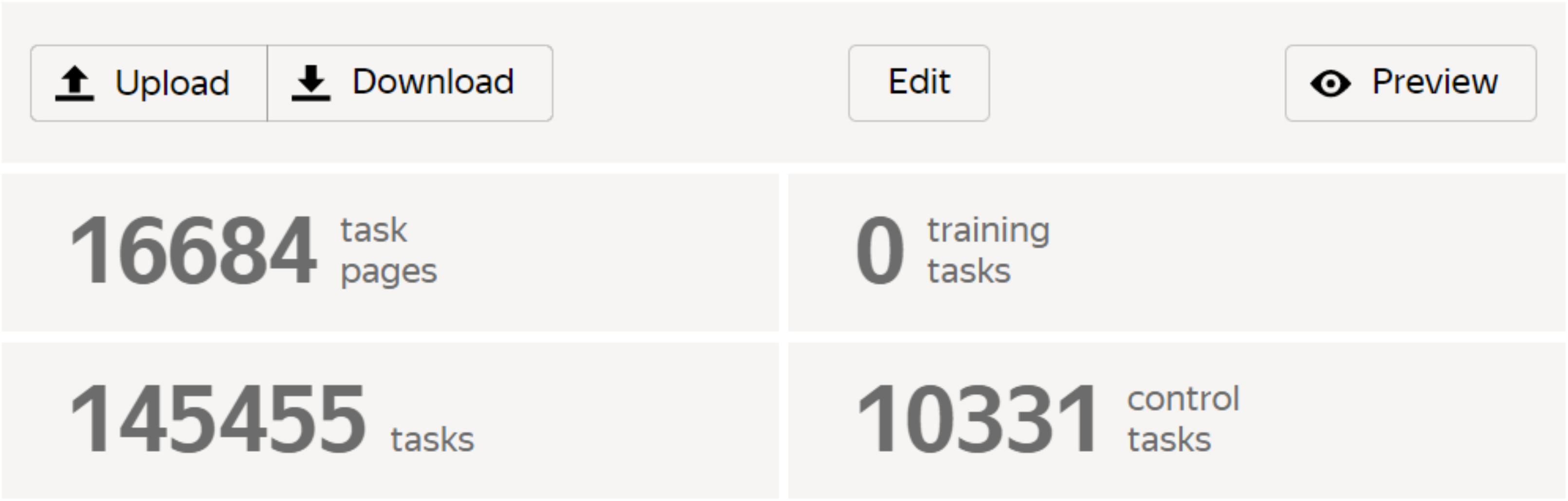
1. Service quality monitoring (KPI metric): when you need to track what is going on with your service
2. Target for supervised learning: for training machine learning algorithms
3. Acceptance metric: final validation before the release of new features

KPI

1. Motivate to increase quality
2. Respectively react to releases
3. Stable
4. Reliable
5. Regular measurements

Target for ML

1. Most informative
2. Huge volume
3. Suitable for trained models



Acceptance

1. Very fast

2. Before release

3. Offline A/B

SYSTEMS ›	754432		769701 pers = 7678...		769706 pers = 7678...		769711 pers = 7678...		769717 pers = 7678...		769723 pers = 7678...		769320 pers = 7678...		769331 pers = 7678...	
	VALUE	% DIFF	VALUE	% DIFF	VALUE	% DIFF	VALUE	% DIFF	VALUE	% DIFF	VALUE	% DIFF	VALUE	% DIFF	VALUE	% DIFF
	1.9491	+0.01%	1.96	+0.57%	1.9467	−0.12%	1.9546	+0.29%	1.9505	+0.08%	1.9639	+0.79%	1.9532	+0.22%	1.9558	+0.35%
	2.2808	+0.01%	2.2969	+0.72%	2.2787	−0.08%	2.2897	+0.40%	2.2838	+0.14%	2.3023	+0.97%	2.288	+0.33%	2.292	+0.50%
	0.8548	−0.01%	0.8535	−0.15%	0.855	+0.03%	0.854	−0.09%	0.8546	−0.02%	0.8531	−0.20%	0.8538	−0.12%	0.8534	−0.16%
	1.3051	+0.01%	1.3101	+0.39%	1.301	−0.31%	1.3066	+0.12%	1.3041	−0.07%	1.3128	+0.62%	1.3051	+0.01%	1.3075	+0.20%
	1.2655	+0.01%	1.2738	+0.67%	1.264	−0.11%	1.27	+0.37%	1.267	+0.13%	1.2766	+0.91%	1.2687	+0.26%	1.2709	+0.43%
	0.917	+0.01%	0.9263	+1.02%	0.9193	+0.26%	0.9234	+0.70%	0.9206	+0.40%	0.9284	+1.26%	0.9236	+0.73%	0.9244	+0.82%
	2.1825	+0.01%	2.2001	+0.82%	2.1834	+0.05%	2.1934	+0.51%	2.1876	+0.24%	2.205	+1.06%	2.1923	+0.46%	2.1953	+0.59%
	0.0983	+0.02%	0.0968	−1.48%	0.0953	−2.98%	0.0963	−2.00%	0.0961	−2.16%	0.0973	−0.98%	0.0957	−2.53%	0.0967	−1.54%
	1.8921	+0.01%	1.9065	+0.77%	1.8901	−0.10%	1.9009	+0.47%	1.8953	+0.17%	1.9112	+1.04%	1.8964	+0.24%	1.9001	+0.43%

What Can Go Wrong?

What can go wrong

Ambiguity and clarity

Example: “local language is more preferable than foreign language”

What went wrong: international porn sites were penalized ☹

Result: service quality decreased

Moral: avoid ambiguity

What can go wrong

Nothing is perfectly reliable!

Basic checks: input and output, presence of judgements, service availability

Advanced checks: A/A testing, comparison with previously known verdicts, re-evaluations, DSAT

Why Crowdsourcing?

Why Crowdsourcing?

Initially: in-house experts (assessors)

Pros	Cons
▶ Trusted	▶ Expensive
▶ Can perform sensitive tasks (signed NDA)	▶ Hard to scale
▶ Easy to train/ control/interact	

Why Crowdsourcing?

What is crowdsource?

1. Lots of annotators
2. Easy to scale
3. Easy to add and remove annotators

Need to control quality
Open market, compete for annotators

Why Crowdsourcing?

It is possible to replicate in-house annotation processes with crowdsourcing!

1. Same quality
2. Cheaper
3. More scalable, higher performance
4. Quality control via in-house pipeline
5. Relevance assessment in pairwise setting

References

Where to read more

1. A Short Survey on Online and Offline Methods for Search Quality Evaluation
2. Pairwise comparisons — https://ieeexplore.ieee.org/abstract/document/6120246
3. Just sort it — https://arxiv.org/abs/1502.05556
4. Cumulated gain-based evaluation of IR techniques — https://doi.org/10.1145/582415.582418
5. ERR — http://dx.doi.org/10.1145/1645953.1646033
6. pFound — http://romip.ru/romip2009/15_yandex.pdf https://catboost.ai/docs/references/pfound.html
7. Reservoir sampling — http://www.cs.umd.edu/~samir/498/vitter.pdf

Datasets

1. Text REtrieval Conference Data —
   https://trec.nist.gov/data.html

2. Toloka Relevance 2 & Relevance 5
   https://toloka.ai/datasets

Join our Slack: icwe_tutorial channel

Dr. Dmitry Ustalov

Head of Research

dustalov@toloka.ai

https://toloka.ai/events/icwe-2022/