



Toloka

Improving Recommender Systems with Human-in-the-Loop

Dmitry Ustalov, Natalia Fedorova, Nikita Pavlichenko,
Maxim Kunakov, Fedor Zhdanov

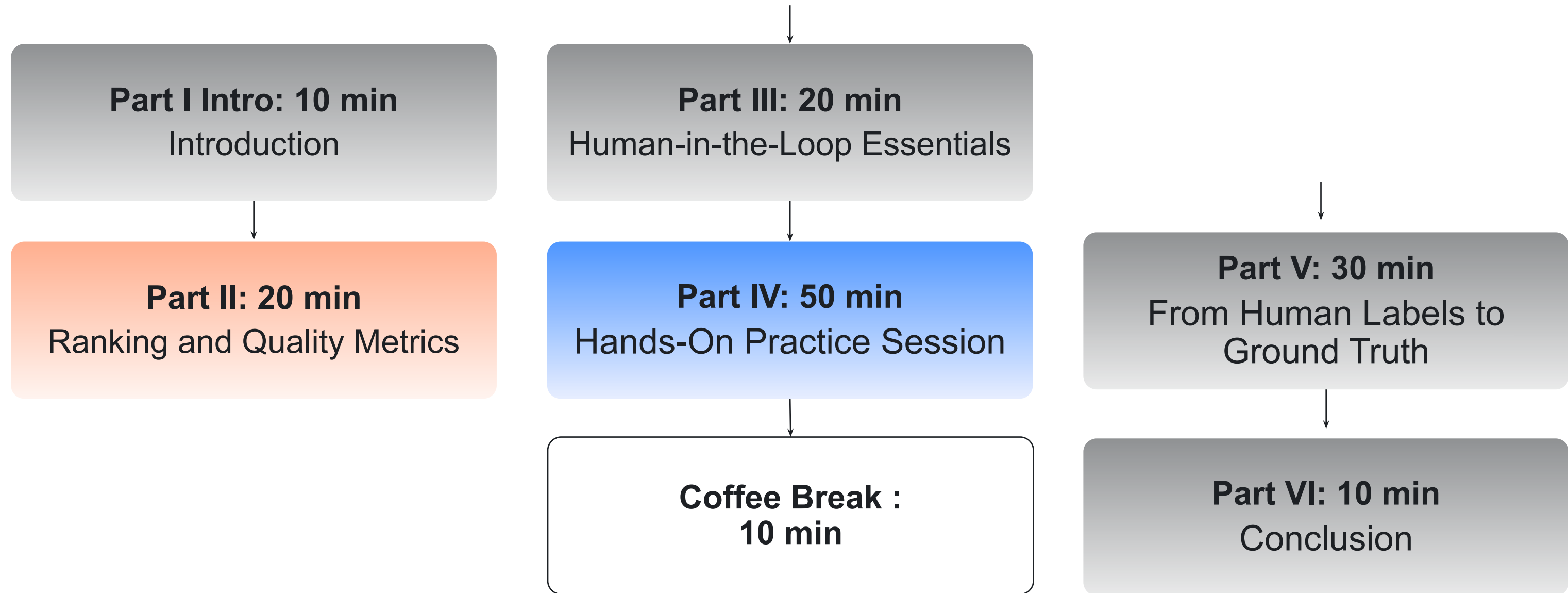


Part II

Ranking and Its Quality

Dr. Dmitry Ustalov,
Head of Research

Tutorial Schedule



Online and Offline Metrics

How labeled data is used

1. Calculating offline metrics to evaluate how a model is performing
2. Training ML models and choosing the best model version

Offline metrics measured with data labeling

Pros

- + Clear signal
- + Measures designated product characteristics
- + Fast results

Cons

- Not actual users (not always a representative sample)
- Can't measure business metrics

Online metrics measured with A/B tests

Pros

- + Results from real users
- + Measures business metrics (clicks, dwell time, leakage)

Cons

- Implicit signal
- Delayed response
- Slow results (long experiments)
- Clickbait
- Fraud

Signals

The background features a series of overlapping, curved, semi-transparent shapes in various shades of blue, ranging from a deep navy to a bright, vibrant blue. These shapes create a sense of depth and movement, resembling stylized waves or a modern architectural design. The overall aesthetic is clean and contemporary.

What We Need to Do

1. Evaluate every response object r with some quality measure s (create a signal)
2. Aggregate s to overall measure of quality (create a metric)

Examples

1

Search engine

- ▶ Text search
- ▶ Image search
- ▶ Ecommerce goods search

2

Recommendations

- ▶ Music feed
- ▶ Content feed
- ▶ Social media feed

3

Content Moderation

- ▶ Service quality assurance
- ▶ Social media business account behavior

Signals

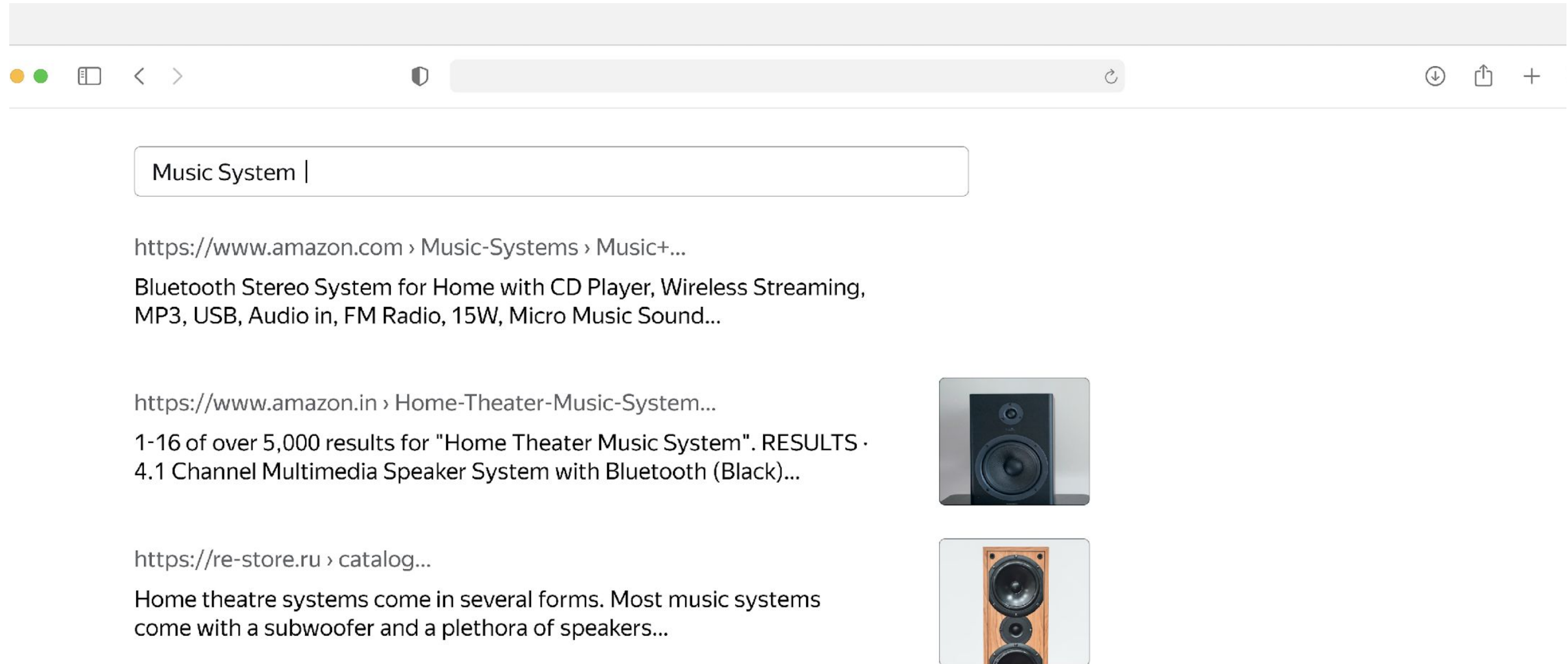
In order to calculate metric,
we need to estimate
response objects.

It can be done through multiple
approaches

- ▶ Pointwise
- ▶ Listwise
- ▶ Pairwise

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

Ranking and Recommender Evaluation




The screenshot shows a web browser window with a search bar containing the text "Music System |". Below the search bar, there are three search results. The first result is from "https://www.amazon.com" and is titled "Bluetooth Stereo System for Home with CD Player, Wireless Streaming, MP3, USB, Audio in, FM Radio, 15W, Micro Music Sound...". The second result is from "https://www.amazon.in" and is titled "1-16 of over 5,000 results for 'Home Theater Music System'. RESULTS · 4.1 Channel Multimedia Speaker System with Bluetooth (Black)...". The third result is from "https://re-store.ru" and is titled "Home theatre systems come in several forms. Most music systems come with a subwoofer and a plethora of speakers...". To the right of the second and third results are small images of speakers. The second image shows a black speaker, and the third image shows a wooden speaker.


Music System |

<https://www.amazon.com> › Music-Systems › Music+...
Bluetooth Stereo System for Home with CD Player, Wireless Streaming, MP3, USB, Audio in, FM Radio, 15W, Micro Music Sound...

<https://www.amazon.in> › Home-Theater-Music-System...
1-16 of over 5,000 results for "Home Theater Music System". RESULTS · 4.1 Channel Multimedia Speaker System with Bluetooth (Black)...



<https://re-store.ru> › catalog...
Home theatre systems come in several forms. Most music systems come with a subwoofer and a plethora of speakers...



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

Easy to obtain

Cons

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

- ▶ Provides full information
- ▶ Judge has all available context

Cons

- ▶ Expensive
- ▶ Inconsistent
- ▶ Relative

Pairwise

Pointwise is of low resolution, listwise is inconsistent
Pairwise comparisons tackle both of these problems,
they are a perfect example of task decomposition.

Pros

- ▶ Consistent
- ▶ Simple

Cons

- ▶ Still quite expensive
- ▶ Relative signal

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 pairwise
evaluation
in our practice.

Metrics

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	
		<i>P</i>	<i>N</i>
Actual Class	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)

mAP (mean average precision)

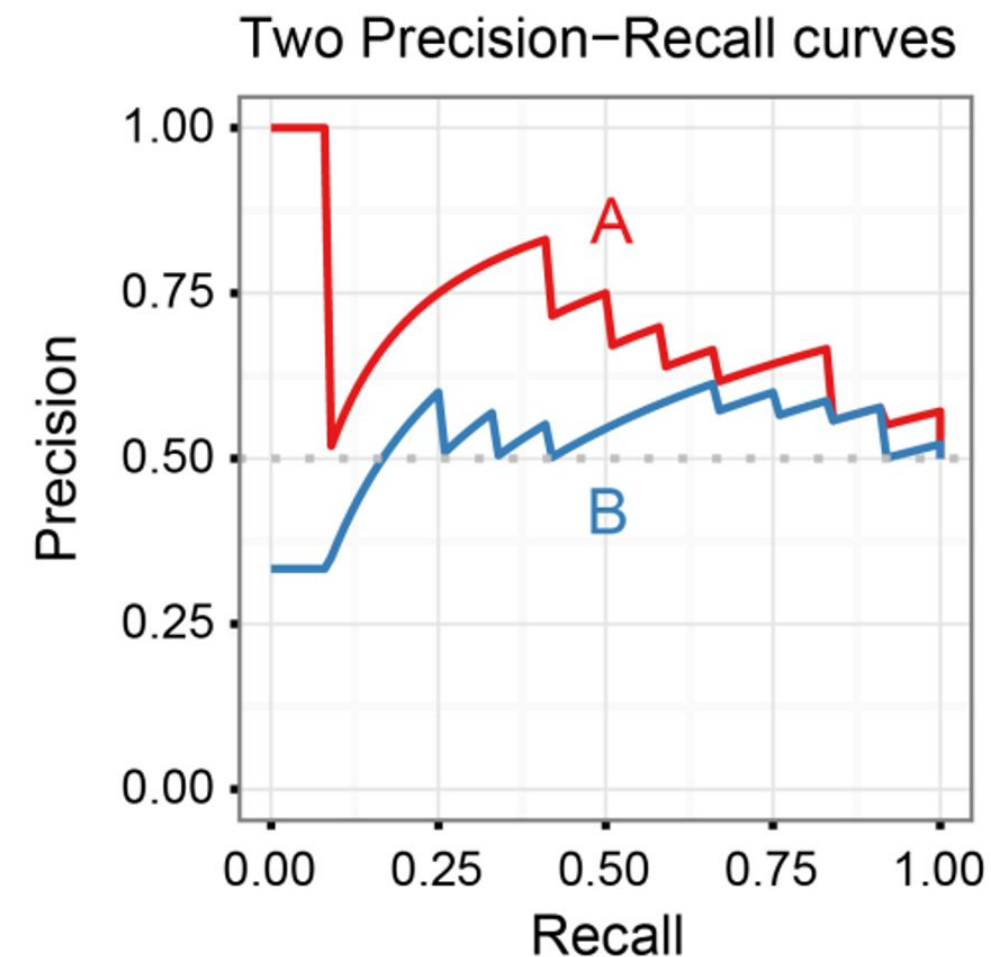
How precision and recall changes going down the list?

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)

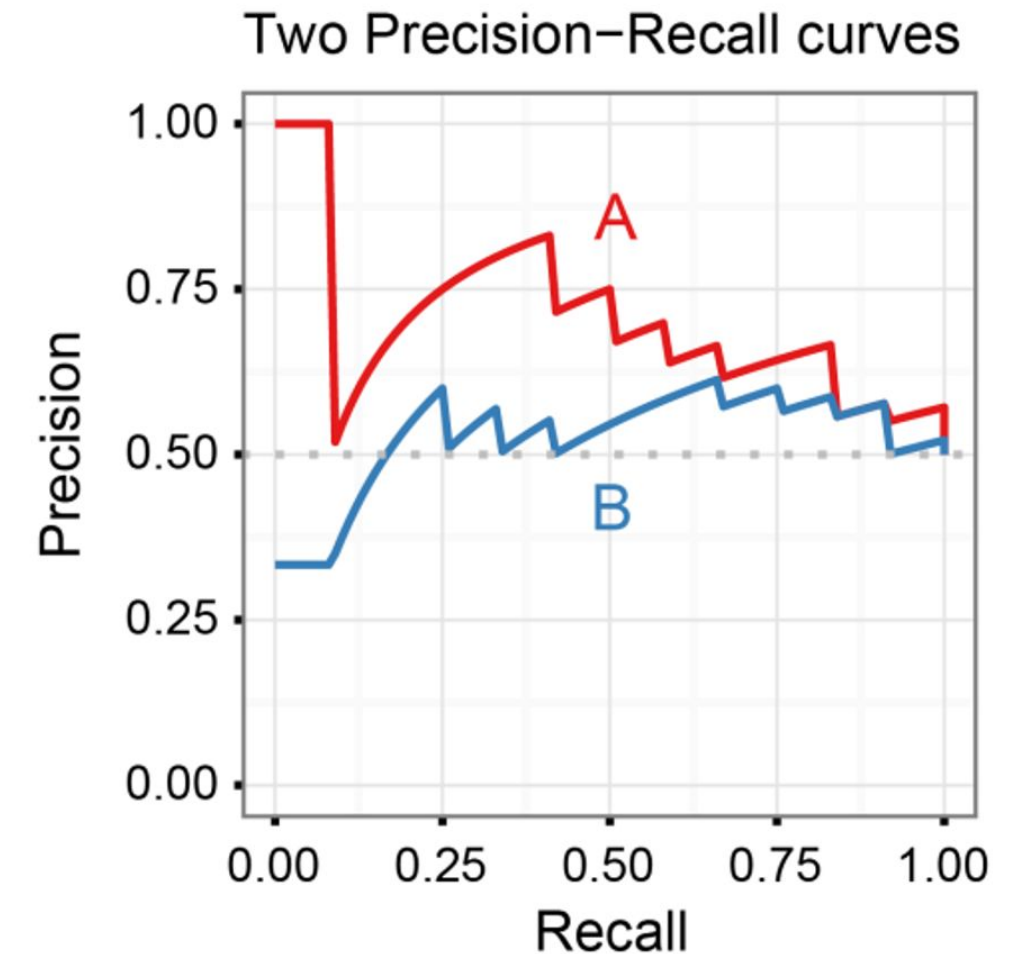
We can define Average Precision as the following:

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

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:

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

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 (DCG³) as following:

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

where $d(i)$ is a discounting factor

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

Expected Reciprocal Rank (ERR)

Web Images Video Maps

2 million results found

- Expected reciprocal rank / Хабр**
habr.com > ru/company/econtenta/blog/303458/
- Expected Reciprocal Rank**
lingpipe-blog.com > ...zhang...expected-reciprocal-rank...
2009. **Expected reciprocal rank** for graded relevance. ... **Expected reciprocal rank** is based on the cascade model of search (there are citations in the paper). Read more >
- Mean reciprocal rank - Wikipedia**
en.wikipedia.org > Mean reciprocal rank
The mean **reciprocal rank** is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness. The **reciprocal rank** of a query response is the multiplicative inverse of the... Read more >
- (PDF) Expected reciprocal rank for graded relevance**
researchgate.net > ...Expected_reciprocal_rank_for...
...cal **rank** to the graded relevance case and we call this metric **Expected Reciprocal** ... For more than two correlation or matching levels for measuring a **ranking** result, the **expected reciprocal rank** [82] and normalized discounted cumulative gain... Read more >
- Expected reciprocal rank for graded relevance | Proceedings...**
dl.acm.org > doi/10.1145/1645953.1646033
Home Conferences CIKM Proceedings CIKM '09 **Expected reciprocal rank** for graded relevance. ... **Rank**-biased precision for measurement of retrieval effectiveness. ACM Trans. Inf. Read more >
- Expected Reciprocal Rank for Graded Relevance - PDF...**
docplayer.net > 20782422-Expected-reciprocal-rank...
The **Expected Reciprocal Rank** is a cascade based metric with $\phi(r) = /r$. It may not seem straightforward to compute ERR from the previous definition because there is an **expectation**. However it can easily be computed as follows: $ERR := r = P...$ Read more >
- itnan.ru/post.php?c=1&p=303458
itnan.ru > post.php?c=1&p=303458
- GitHub - skondo/evaluation_measures: Framework that...**
github.com > skondo/evaluation_measures
2009. **Expected reciprocal rank** for graded relevance. In Proceedings of the 18th ACM conference on Information and knowledge management (CIKM '09). Read more >

Explanations

Irrelevant

Original paper

Skipped

Expected Reciprocal Rank (ERR)

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)

Expected Reciprocal Rank (ERR)

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”:

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

How to Sample Queries?

How to Sample Queries?

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 the 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 the query
 - ▶ Tails: skip
 - ▶ On average, $p \cdot N$ queries will be sampled
- 2. More sophisticated — reservoir sampling⁶:**
 - ▶ Every object is considered
 - ▶ Exactly k objects will be sampled

No guarantee that popular queries will be presented in sample!

How to Sample Queries?

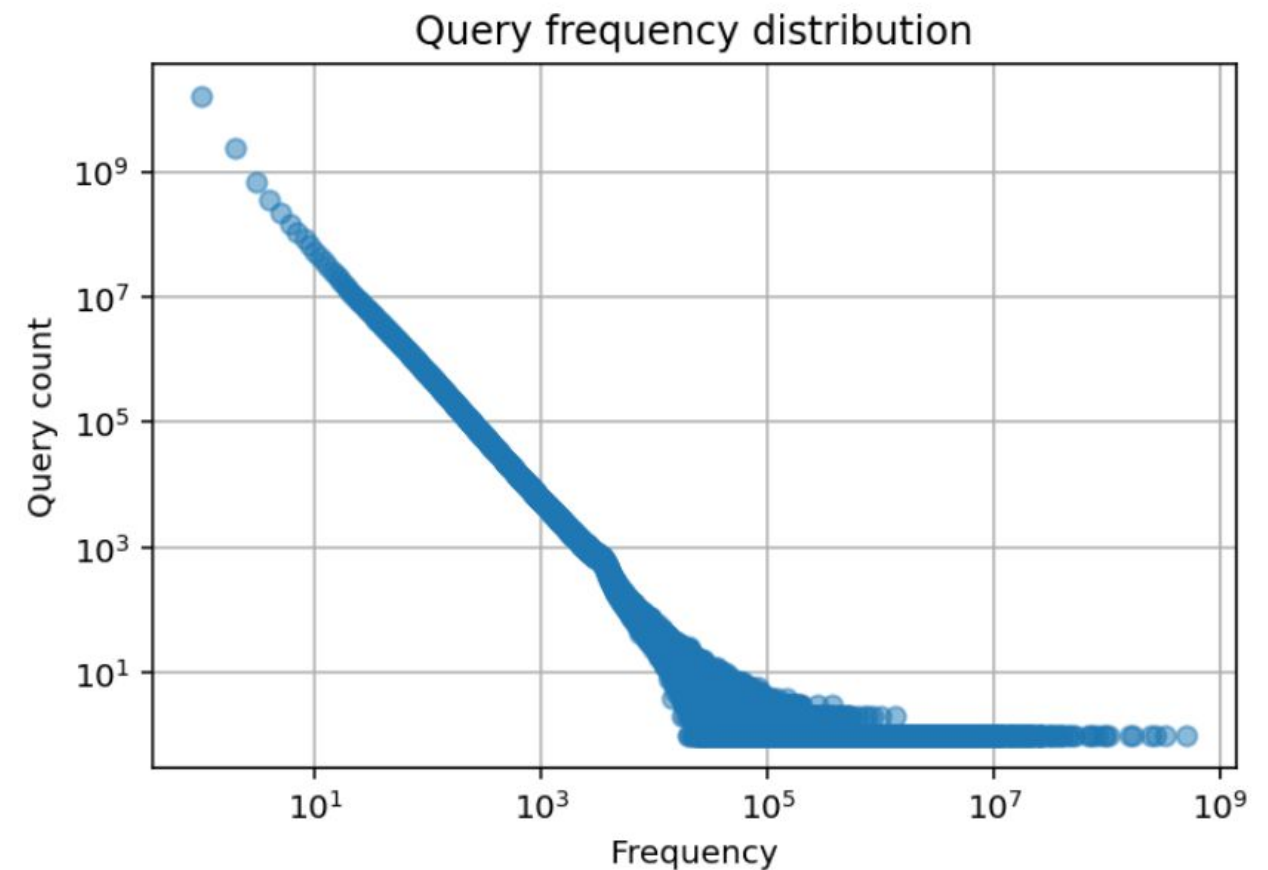
Stratified sampling:

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

$$\sum_{m \in Q_i} f_m \approx \sum_{k \in Q_j} f_k \quad \forall i, j,$$
$$\forall i < j \Rightarrow f_m < f_k \quad \forall m \in Q_i, k \in Q_j.$$

- ▶ After that, sample the necessary amount from every bucket

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



How to Sample Pairs?

How to Sample Pairs?

What is the right number of pairs for evaluation?

1

Lower bound

- ▶ Traditional sorting algorithms like MergeSort run $n \log n$ comparisons

2

Upper bound

- ▶ The number of all possible pairs is bound by n^2

3

Reasonable bound

- ▶ Select $k n \log n$ pairs (sort objects multiple times)
- ▶ k is a hyper-

What to Compare?

- › $n \log n$ is enough when compares are transitive:

$$i > j, j > k \Rightarrow i > k$$

- › Human judgements do not always provide transitivity
- › But we can rely on Bradley-Terry's *linear order*

$$\frac{P(i > k)}{P(i < k)} = \frac{P(i > j) P(j > k)}{P(i < j) P(j < k)}$$

Using latent scores s_i, s_j, s_k from Bradley-Terry model, we have

$$s_i - s_k = s_i - s_j + s_j - s_k$$

Applications and Problems

The background features a series of overlapping, curved, blue shapes that create a sense of depth and movement, resembling a stylized fan or a series of concentric arcs. The colors range from a deep navy blue to a bright, vibrant blue.

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

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

- ▶ Trusted
- ▶ Can perform sensitive tasks (signed NDA)
- ▶ Easy to train/control/interact

Cons

- ▶ Expensive
- ▶ Hard to scale

Why Crowdsourcing?

What is crowdsource?

- Lots of annotators
- Easy to scale
- Easy to add and remove annotators

Cons:

- 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: <https://doi.org/10.1145/1645953.1646033>
6. Reservoir sampling: <http://www.cs.umd.edu/~samir/498/vitter.pdf>
7. RankEval: <https://github.com/hpclab/rankeval>

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: **recsys_2022**

Dr. Dmitry Ustalov

Head of Research



dustalov@toloka.ai



<https://toloka.ai/events/recsys-2022/>



<https://bit.ly/3eYIX2P>