Yandex
Practice of Efficient Data Collection via Crowdsourcing: Aggregation, Incremental Relabelling, and Pricing

Alexey Drutsa, Valentina Fedorova, Dmitry Ustalov, Olga Megorskaya, Evfrosiniya Zerminova, Daria Baidakova
Part V:

Effective quality control and task interface: details

Alexey Drutsa,
Head of Efficiency and Growth Division

Yandex.Toloka is a service of Swiss company Yandex Services AG
Quality control: 
the rate of correct answers
Task sequence

Tasks executed by a performer

Signals of answer correctness
For instance, binary, $y_i \in \{0,1\}$

$n$, window size
Estimation of correctness rate

To estimate the probability of a correct answer use

\[ P(\text{correct}) \approx \frac{1}{n} \sum_{i=1}^{n} y_i \pm \frac{1}{2\sqrt{n}} \]

Window size \((n)\) is a balance between

\> accuracy of the estimate

and

\> fast reaction to changes in performer quality
Sources for correct answer signal

How can we get $y_i$?

- Control tasks
- Agreement with aggregated answer (e.g., Majority Vote)
- Post-verification
Control tasks

Pros:
› Signal is obtained instantly
› Signal has high confidence on tasks where obtained

Cons:
› Tasks for labelling do not provide this signal (=> signal for a fraction of tasks)
› Creation and maintenance of a set of control tasks

Costs (extra charge for quality control)
› Control task creation
› Depends on the frequency of control tasks occurred in the task sequence

You can apply adaptive frequency to optimize costs
Agreement with aggregated answer

Pros:
› Easy to implement

Cons:
› Signal is obtained with latency
› Works well only if most workers have good quality
› Works well for tasks with small # of answer variants (e.g., classification)

Costs (extra charge for quality control)
› Multiplied by the overlap used

You can apply incremental relabelling to optimize costs
Agreement may fail against coordinated attacks

\[ \mathbb{P}(\#m_{\text{bad}} > \frac{n}{2}) = \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n} C_n^k p^k (1 - p)^{n-k} \]

\( p \) is the fraction of coordinated spammers among performers
\( n \) is the overlap for Majority Vote model

For instance:
If \( n = 3 \) and \( p = 0.1 \)

\[ \text{The probability of majority with an incorrect answer is 2.8\%} \]

in fact, is larger since other performers may accidentally agree with spammers
Post-verification

**Pros:**
- Can be applied to any task type (even with a sophisticated answer)

**Cons:**
- Signal is obtained with latency
- Requires efforts to construct a pipeline

**Costs (extra charge for quality control):**
- Cost of verification tasks

You can apply selective verification to optimize costs
Non-binary penalty

You can set different penalty $y_i \in [0,1]$ for different signals

For instance:
› task consists of several answers of different importance
› level of confidence of the aggregated answer
› level of expertise of the performer who post-verifies
Quality control:
undesired behavior
Performer behavior

Correct answers to your tasks are not the sole signal of performer quality

For instance, take care of such characteristics:
› Time of task execution
› Usage of UI control elements within task execution
› CAPTCHA

Use them to filter out (ban) performers with low quality of high confidence
Fast responses

There is a lower bound on time required to execute your task with good quality

› Estimate this time based on behavior of a set of performers
› Calculate the number or the rate of tasks executed too fast
Verification of action execution

Some tasks require usage of certain UI control elements

For instance:
› check whether a link has been visited
› check whether a video has been played
Instead of revoking access to your tasks, you can ask crowdsourcing platform to show CAPTCHA to a performer.

You get an additional signal to decide whether you face a robot or not.
Quality control: skills
Skill is a variable assigned to a performer

Can be used to automatically calculate
› answer correctness rates (via control tasks, agreement, post-verification)
› behavioral features (e.g., fast response rate)
› binary information on execution of particular projects
› any their combinations and other features

Can be used for automatic decision making:
› access control to certain projects and tasks
› e.g., revoke access to your tasks if a skill becomes too low
Thinking (cogitation) vs reflexes

Skills based on a single signal are easy to game

It is difficult to force a performer to think (cogitate) instead of to use/train reflexes

A representative crowd project

Average time within performers

# tasks made by a performer
Best practice for a good skill

Combine different signals to get a skill robust to gaming

› Combine agreement signal with control tasks or post-verification
› Add behavioral information: execution time, CAPTCHA, etc.

Use this skill in quality-based pricing
Quality control: performer life cycle
Training task

Train performers to execute your tasks

› All tasks are control ones
› There are hints that explain incorrect answers
Exam task

Control the results of training

› All tasks are control ones
› No hints and explanations

› A good exam should be:
1. passable
2. regularly updated
3. small
Recommended life cycle of performers

All performers → Training → Exam → Real tasks

Rehabilitation

Access denied
Recommended life cycle of performers

Let quality be controlled by means of a skill $S$

All performers

Set skill $S$

Training

$S > X$

Exam

$S > Y$

Real tasks

$S > Z$

Rehabilitation

$S < Z$

$S < T$

Access denied

Set skill $S$

$S < X$

$S < Y$

$S < Z$

$S < T$
Rehabilitation task

Give a change to those who failed the skill threshold accidentally

- Rehabilitation is similar to an exam task, but with another access criterion
- Remind that there is a chance to observe low quality of a good performer

\[
P(\text{correct}) \approx \frac{1}{n} \sum_{i=1}^{n} y_i \pm \frac{1}{2\sqrt{n}}
\]
Grant initial access to top performers

Access for performers having platform rating > threshold

Access denied

Training → Exam → Real tasks

Rehabilitation
Platform rating *

is calculated based on performer behavior on all existed tasks within the platform

* is available on Yandex.Toloka
Interface. Introduction
Task in the eyes of the performers

Web-page with specific features

› Long run time
› Repetitive actions
› Concentration
› Speed
Structure of a task interface

A joke

When in doubt, mumble.

- Very funny
- Somewhat funny
- Not funny
- Bad joke
Structure of a task interface

- Task block
- Subject of evaluation: When in doubt, mumble.
- Evaluation block:
  - Verdict:
    - Somewhat funny
    - Not funny
    - Bad joke

When in doubt, mumble.
9 golden rules of interface structure
Why is it important?

› Performer’s time
› Speed and data labelling volumes
› Manager’s time
› Quality of the results
› Project’s rating
› Task simplification thanks to the interface
Rule #1. Cross-platform compatibility

Possible limitations for mobile services:
› Task difficulty
› Media Content, Devices, and Browsers
Rule #1. Cross-platform compatibility

Task: evaluate sound quality in wav audio files
Rule #1. Cross-platform compatibility

**Task:** draw a polygon around every road sign
Rule #1. Cross-platform compatibility

**Task:** draw a polygon around every road sign

**Challenge:** to outline every single road sign
Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match

<table>
<thead>
<tr>
<th>Phrase</th>
<th>Ad headline</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>job occupation in New York</td>
<td>New York employment center</td>
<td>Find a stable job on nycjobs.com</td>
</tr>
</tbody>
</table>

Does the phrase match the query?  
- Yes 1  
- No 2
## Rule #1. Cross-platform compatibility

**Task:** evaluate the phrase and search query match

<table>
<thead>
<tr>
<th>Phrase</th>
<th>job occupation in New York</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>New York employment</td>
</tr>
<tr>
<td>Additionally</td>
<td></td>
</tr>
<tr>
<td>Ad headline</td>
<td>New York employment</td>
</tr>
<tr>
<td>Text</td>
<td>Find a stable job on nyc</td>
</tr>
</tbody>
</table>

Does the phrase match the query?

- Yes \(^1\)
- No \(^2\)
Rule #1. Cross-platform compatibility

Task: evaluate the phrase and search query match
Rule #1. Cross-platform compatibility

**Task:** evaluate the phrase and search query match

Phrase
- job occupation in New York

Query
- New York employment center

Additionally
- Ad headline
  - New York employment center
- Text
  - Find a stable job on nycjobs.com

Does the phrase match the query?
- Yes
- No
Rule #2. Hotkeys

› Used by about 28% of performers
› Affect task completion speed
› You can assign hotkeys to any action
› Hidden hotkeys should be documented

Ideal scenario: the task can be completed without using a mouse
Rule #2. Hotkeys

**Task:** evaluate functionality of a game in a browser (works with a keyboard)
Rule #2. Hotkeys

Task: tell whether the game works in a web browser (works with a keyboard)
Rule #2. Hotkeys

Task: tell whether the game works in a web browser (works with a keyboard)
Rule #3. Action and data check

We can check if the performer:

› Watched the video or listened to the audio
› Went to external resources
› Provided correct input data
› Spent enough time on each task

Finish the task as fast as possible!
Rule #3. Action and data check

Game Lets Play!

- Go to game
- Please, go to the game page

- Works ok
- Problems

Keys do not work
- Space
- Enter
- Shift
Rule #4. Test the task

Always test the task before publishing it

› Preview option
› Test task pool in Toloka sandbox
Rule #5. Minimize external resources usage

Spoiler: not always applicable

› Impossible to control performer’s actions outside of the task interface
› External resources might not always work properly
Rule #5. Minimize external resources usage

› Show all information inside the task
› Copy data to your own storage
› Check performers’ actions and their input data

*Idea: show screenshots instead of the links*
Rule #6. Avoid experimental design

Signs:

★ Odd layout of typical interface elements

★ Variety of bright and different colors

★ The presence of conspicuous elements with an exclusively artistic function
Rule #6. Avoid experimental design

<table>
<thead>
<tr>
<th>Phrase</th>
<th>job occupation in New York</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>New York employment center</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additionally</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad headline</td>
<td>Jobs in New York</td>
</tr>
<tr>
<td>Text</td>
<td>Find a stable job on nycjobs.com</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Does the phrase match the query?</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
</table>
Rule #6. Avoid experimental design

- Extra nesting of the blocks
- Unnecessary bright color
- All text is in one font
- A lot of empty space on the right side of the block
- Odd display of verdicts
- 2 types of patterns
Rule #6. Avoid experimental design

<table>
<thead>
<tr>
<th>Phrase</th>
<th>job occupation in New York</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>New York employment center</td>
</tr>
</tbody>
</table>

Additionally

| Ad headline               | Jobs in New York            |
| Text                      | Find a stable job on nycjobs.com |

- The phrase match the query
- The phrase doesn’t match the query
Rule #7. Efficient space usage

› Group the elements within your task block
› Absence of empty spaces
› Highlight most important information

Ideal scenario: one task perfectly fits the size of a monitor
Rule #7. Efficient space usage
Rule #7. Efficient space usage
Rule #7. Efficient space usage
Rule #8. Constructing task suit

Page with many tasks

Check list:
› Absence of empty spaces
› Equal width of the task blocks
› No more than 2 (3) tasks in a row
Rule #8. Constructing task suit

<table>
<thead>
<tr>
<th>Query</th>
<th>borrow a Yota router for a week</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase</td>
<td>Yota router</td>
</tr>
<tr>
<td>Additionally</td>
<td></td>
</tr>
<tr>
<td>Ad headline</td>
<td>Buy Yota router at a super price!</td>
</tr>
<tr>
<td>Text</td>
<td>High-quality wi-fi routers! Installation and configuration. Call us!</td>
</tr>
</tbody>
</table>

Does the meaning of the phrase match the query?  
- Yes 1  
- No 2

<table>
<thead>
<tr>
<th>Query</th>
<th>should I buy an apartment now</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase</td>
<td>buying an apartment</td>
</tr>
<tr>
<td>Additionally</td>
<td></td>
</tr>
<tr>
<td>Ad headline</td>
<td>Buying an apartment on Move.ru</td>
</tr>
<tr>
<td>Text</td>
<td>Selling apartments in your city. Prices straight from the owners</td>
</tr>
</tbody>
</table>

Does the meaning of the phrase match the query?  
- Yes 1  
- No 2
Rule #9. Limit the number of elements in your interface

- Buttons
- Links
- Images
- Other elements, that with a particular function

The presence of any interface element must be justified

Every element of the interface should be useful for the performer
Rule #9. Limit the number of elements in your interface

Task: evaluate which translation from Russian to English is better

Phrase: где правильно переходить улицу
Translation 1: where can I cross the street correctly
Translation 2: where can I cross the street

Check in online translators:
- Yandex
- Google
- Bing
- Lingvo
- PROMT

First translation is better
Second translation is better
Rule #9. Limit the number of elements in your interface

Task: evaluate which translation from Russian to English is better

Phrase: где правильно переходить улицу
Translation 1: where can I cross the street correctly
Translation 2: where can I cross the street

Check in online translators
- Yandex
- Google
- Bing
- Lingvo
- PROMT

First translation is better
Second translation is better
Bonus! Check list

1. Check the adaptability of the task template
2. Test task submission in the preview mode
3. Check the availability and functionality of hotkeys
4. Make sure that the required actions are checked
5. Check for the "not opening" option in tasks with external resources
6. Make sure that there are no experimental design solutions
7. Avoid page interface with a large number of tasks and different sizes of information in it
8. Make sure that there are no unnecessary interface elements in the task
Thank you!
Questions?

Alexey Drutsa
Head of Head of Efficiency and Growth Division

adrutsa@yandex-team.ru
https://research.yandex.com/tutorials/crowd/wsdm-2020