UNIVERSITY OF AMSTERDAM

TREC OpenSearch

Planning Session

Anne Schuth, Krisztian Balog

TREC OpenSearch

> OpenSearch is a **new evaluation paradigm** for IR.

TREC OpenSearch

> OpenSearch is a **new evaluation paradigm** for IR. The experimentation platform *is* an existing search engine.

TREC OpenSearch

> OpenSearch is a **new evaluation paradigm** for IR. The experimentation platform *is* an existing search engine. Researchers have the opportunity to **replace components** of this search engine

TREC OpenSearch

> “OpenSearch is a **new evaluation paradigm** for IR. The experimentation platform *is* an existing search engine. Researchers have the opportunity to **replace components** of this search engine and evaluate these components using interactions with **real, unsuspecting users** of this search engine.”

TREC OpenSearch

"OpenSearch is a **new evaluation paradigm** for IR. The experime[illegible]m *is* an existing search engi[illegible]rs have the opportunity to **replace components** of this search engine and evaluate these components using interactions with **real, unsuspecting users** of this search engine."

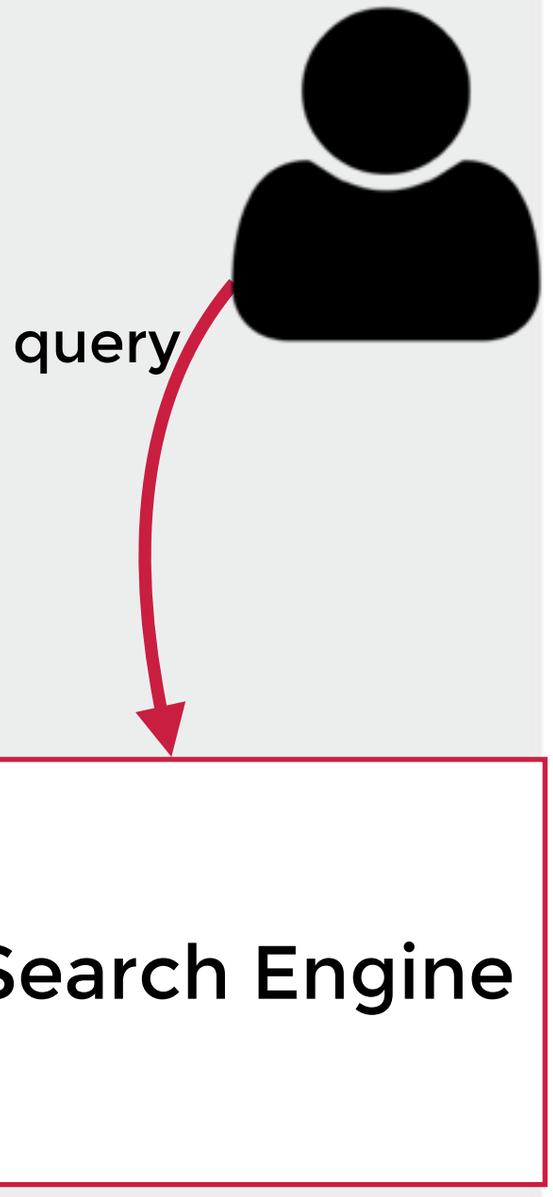
For now, only re-ranking component

Evaluation Setup

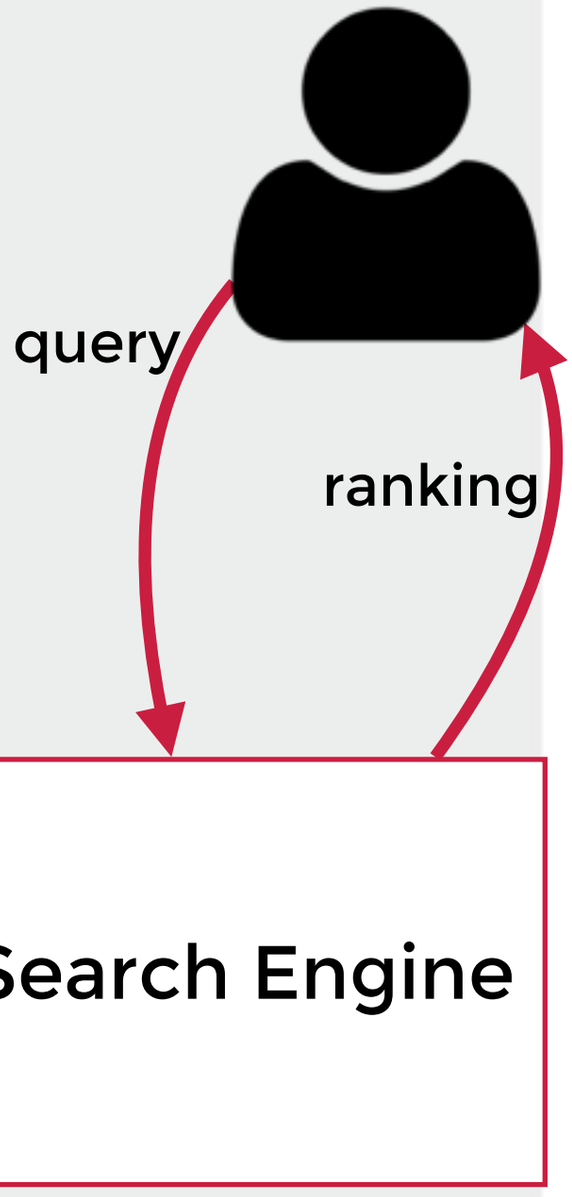
Evaluation Setup

Search Engine

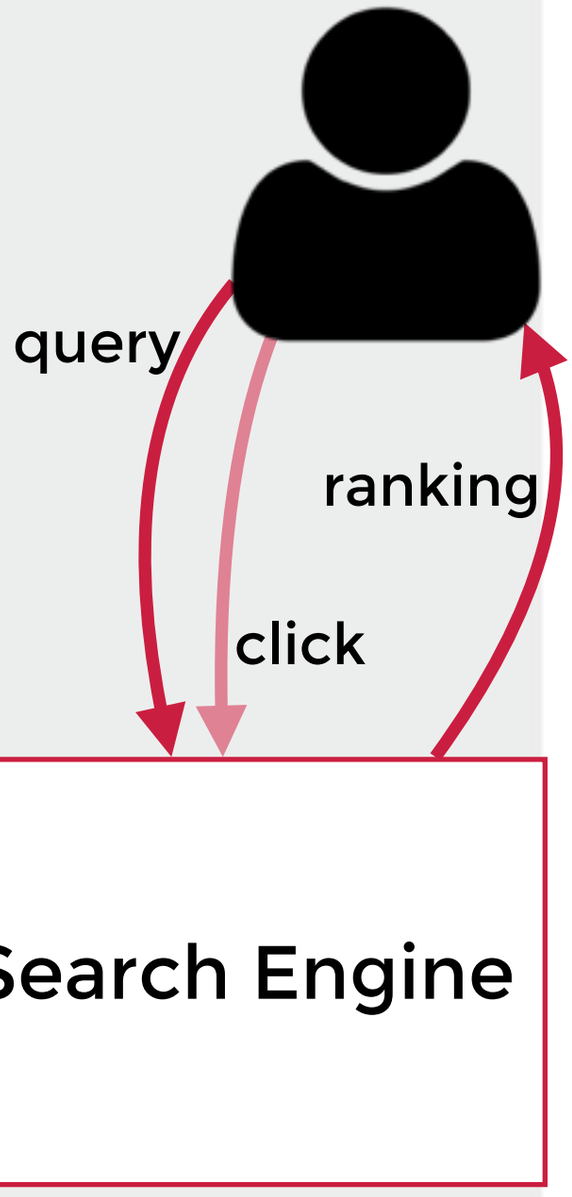
Evaluation Setup



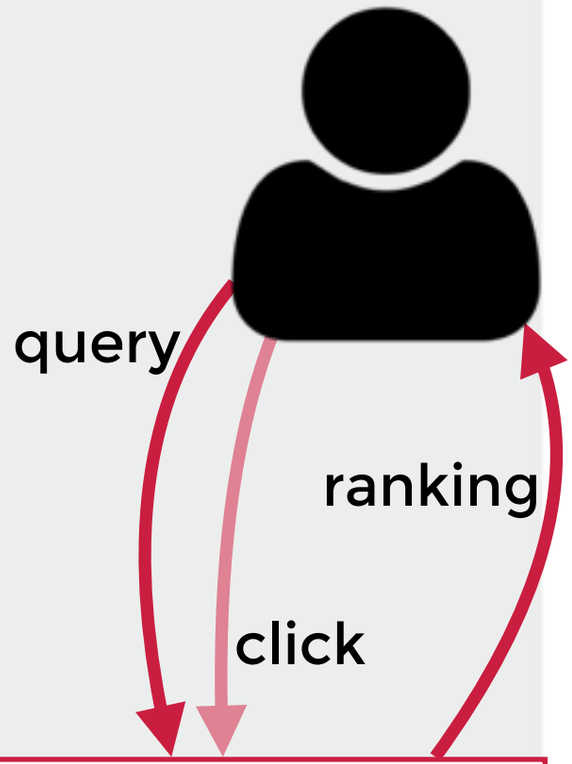
Evaluation Setup



Evaluation Setup

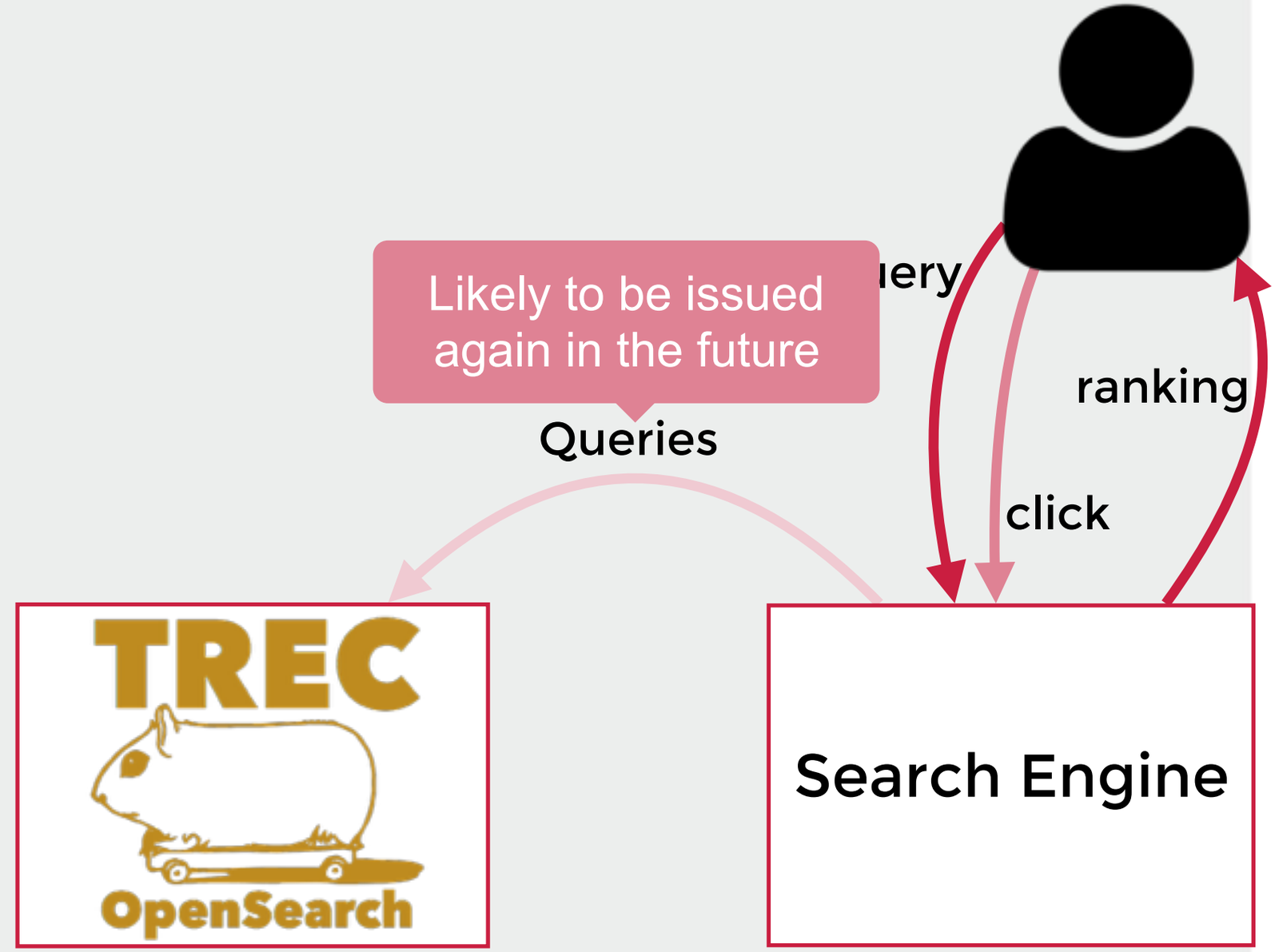


Evaluation Setup

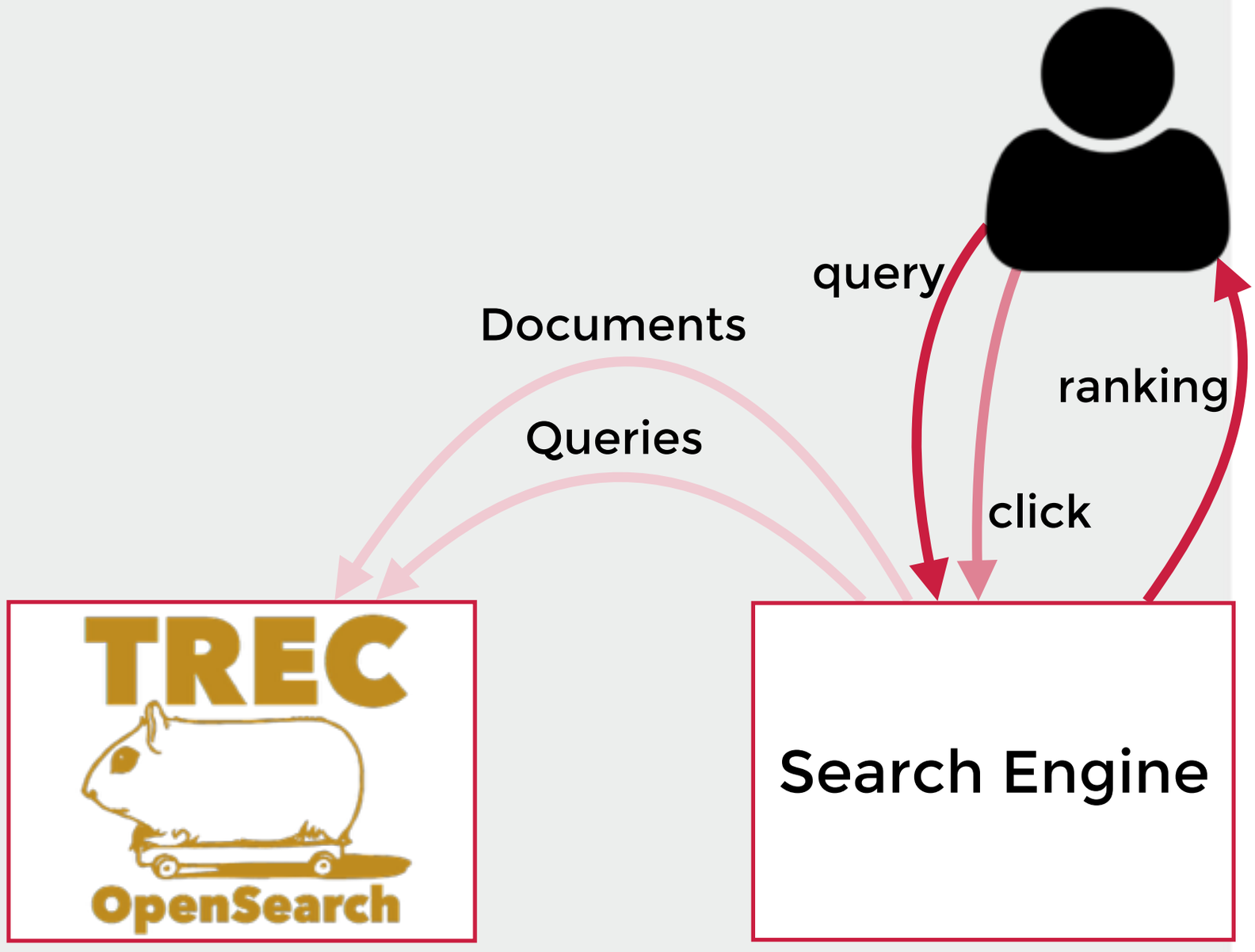


TREC OpenSearch

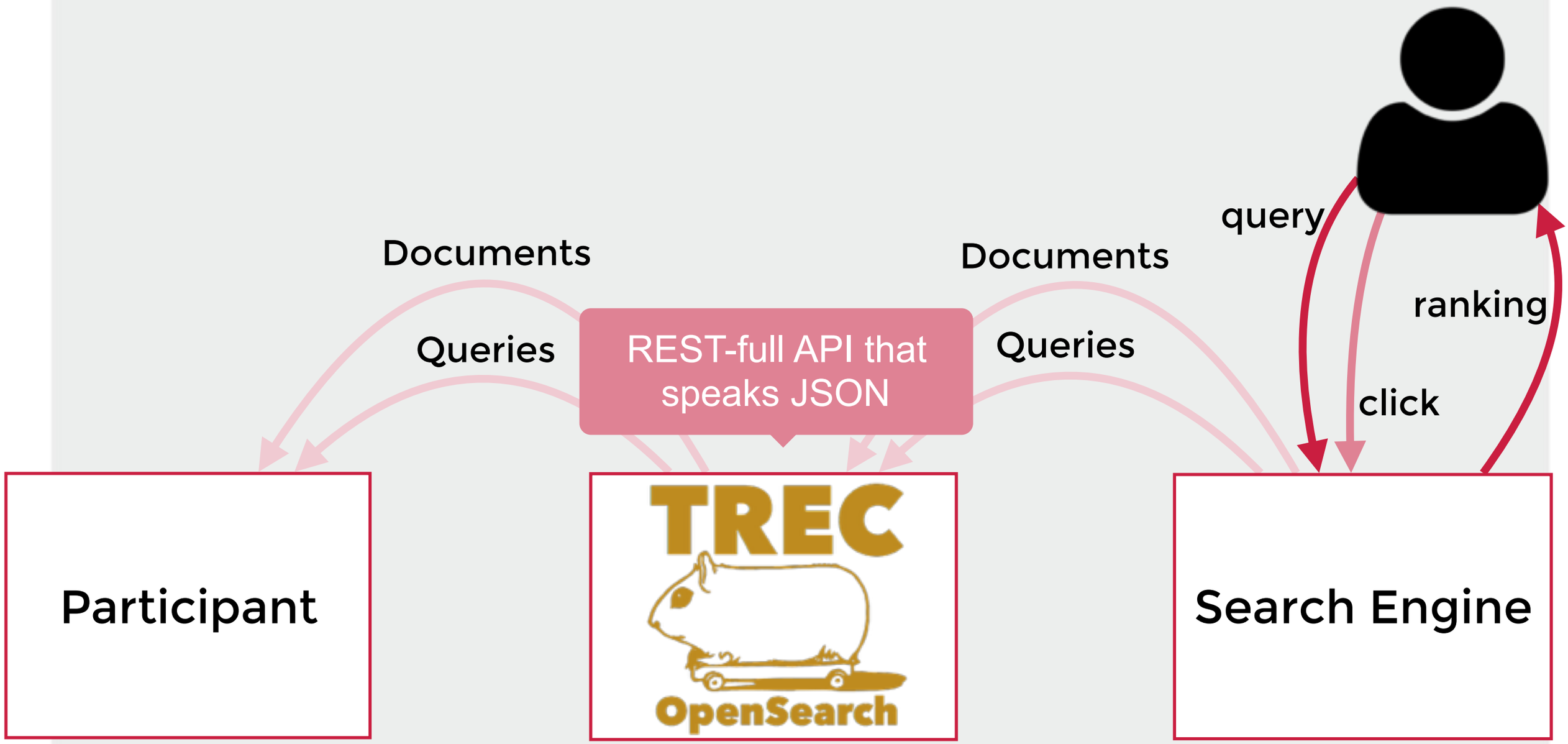
Evaluation Setup



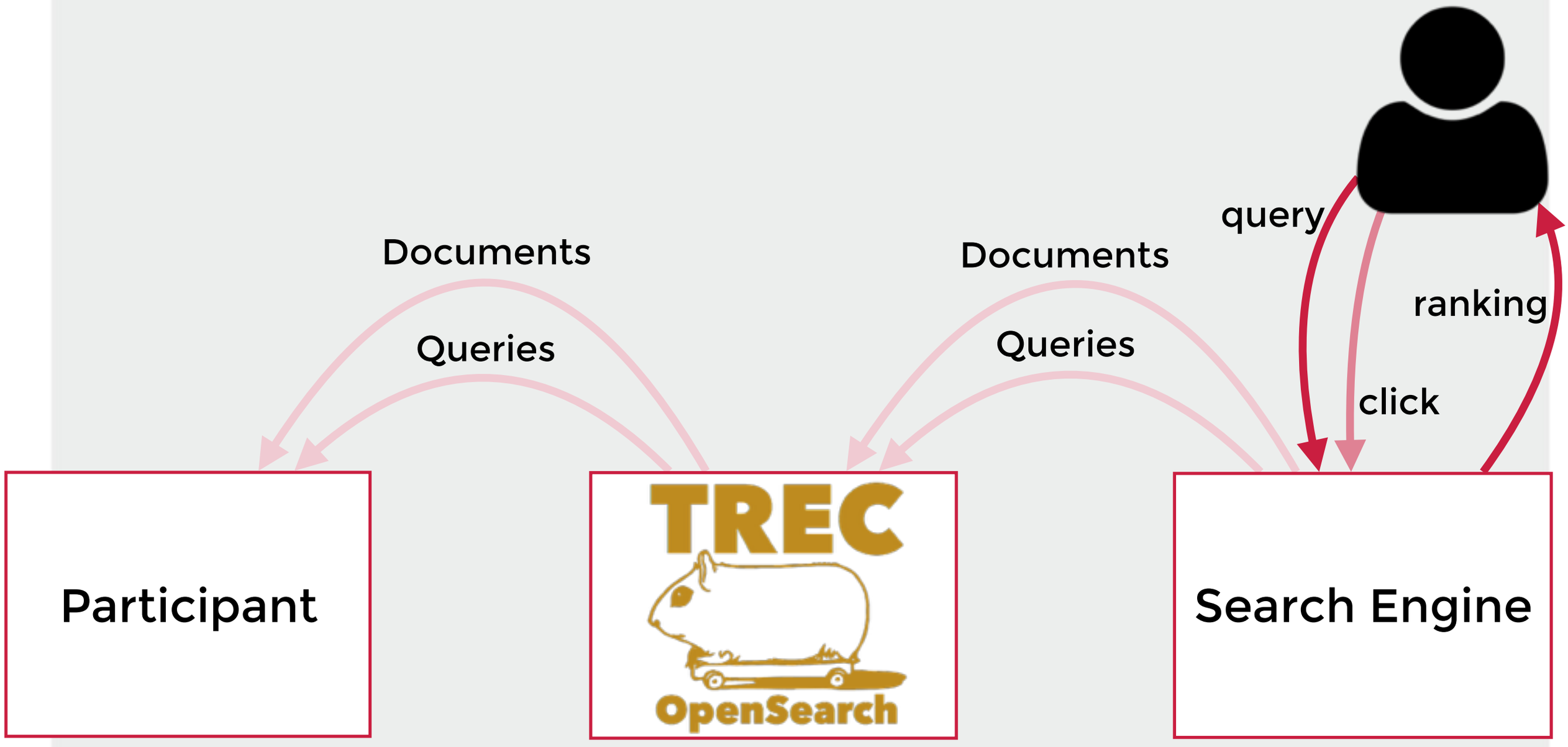
Evaluation Setup



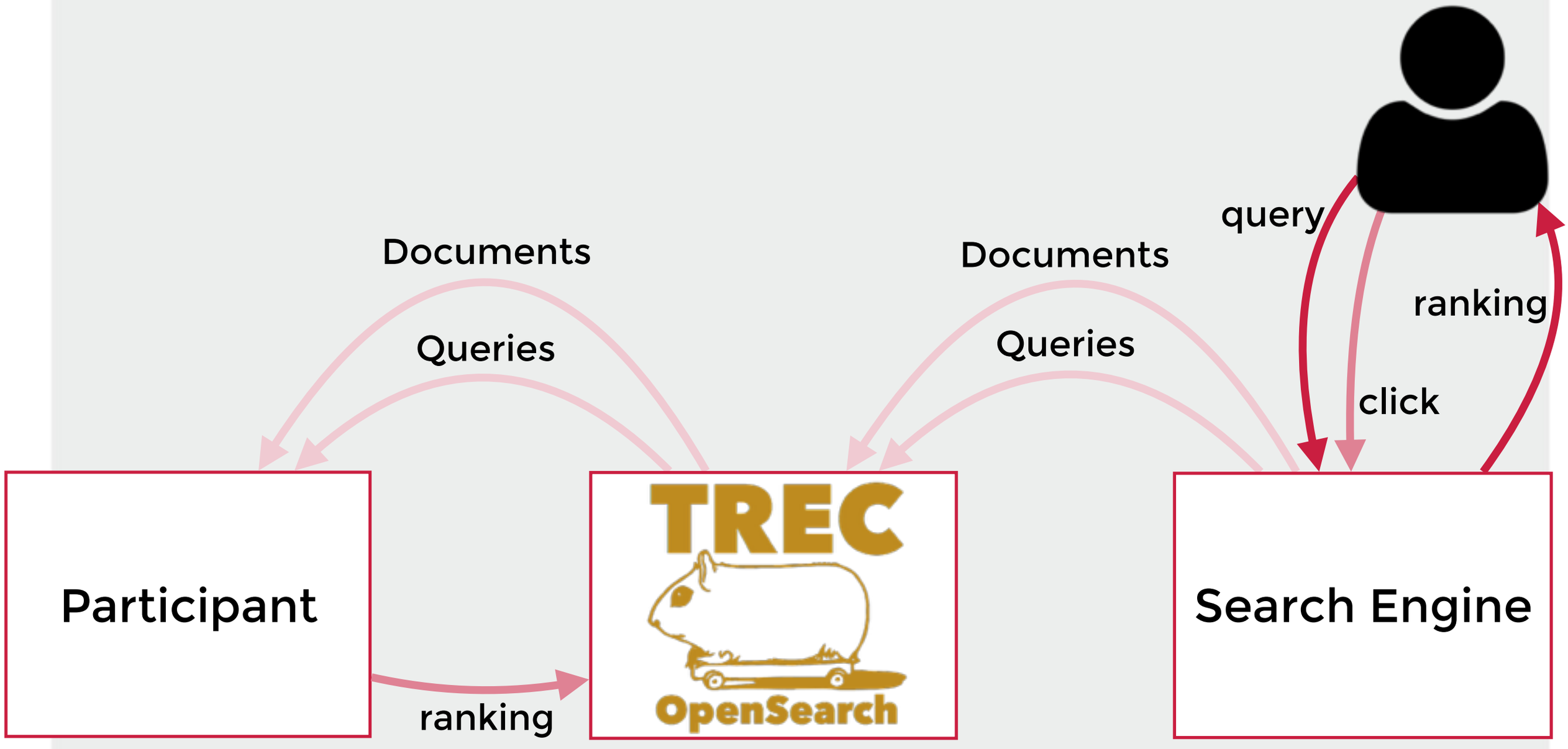
Evaluation Setup



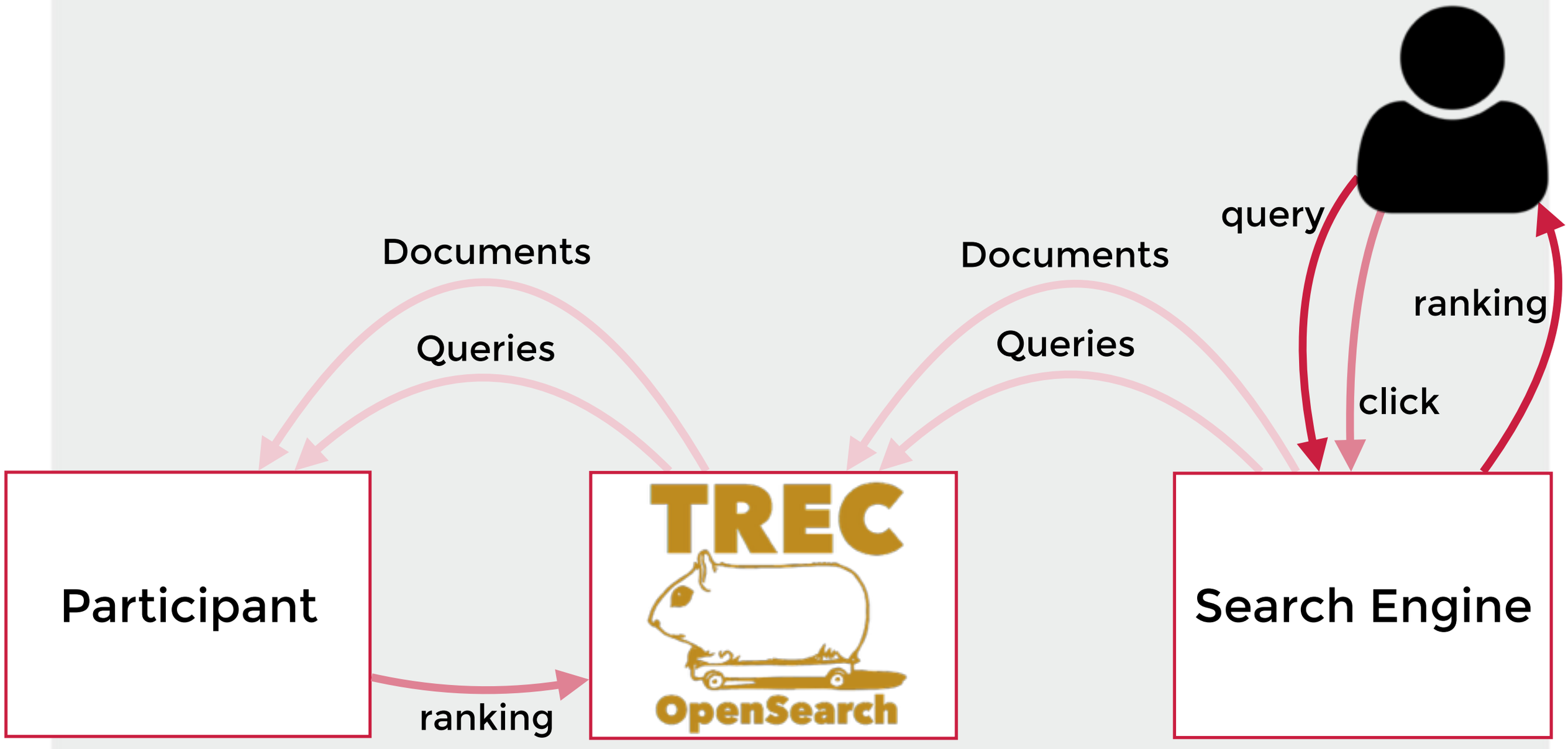
Evaluation Setup



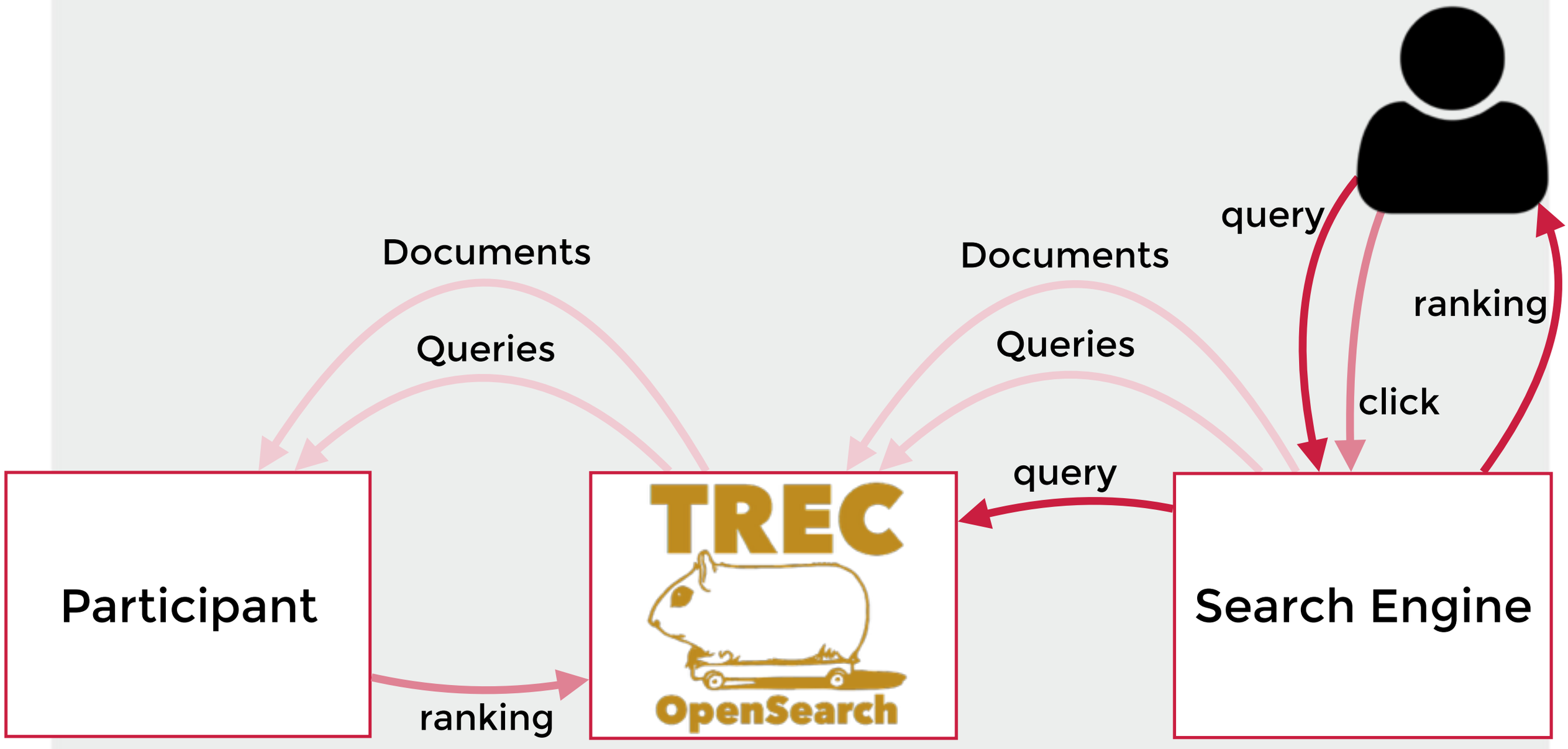
Evaluation Setup



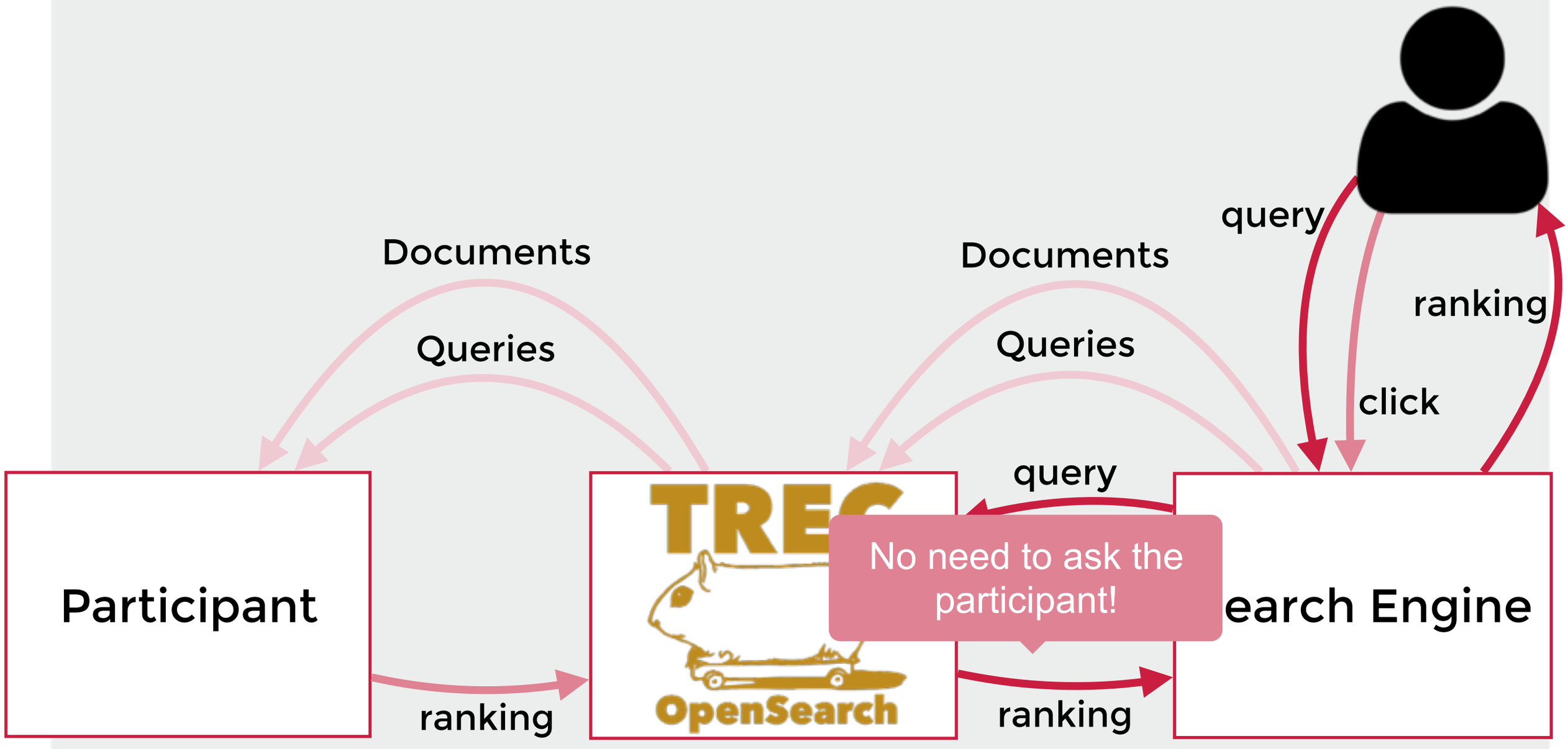
Evaluation Setup



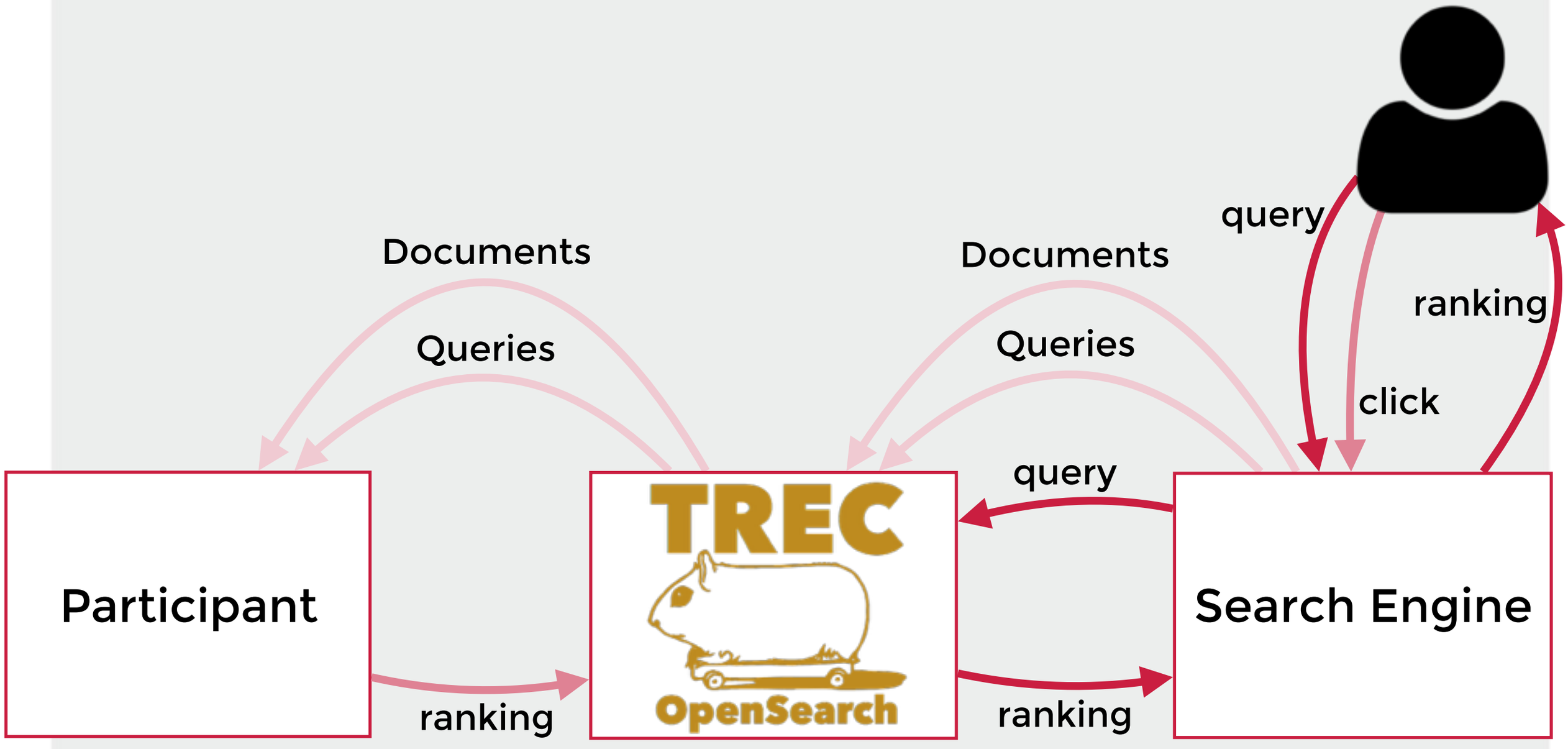
Evaluation Setup



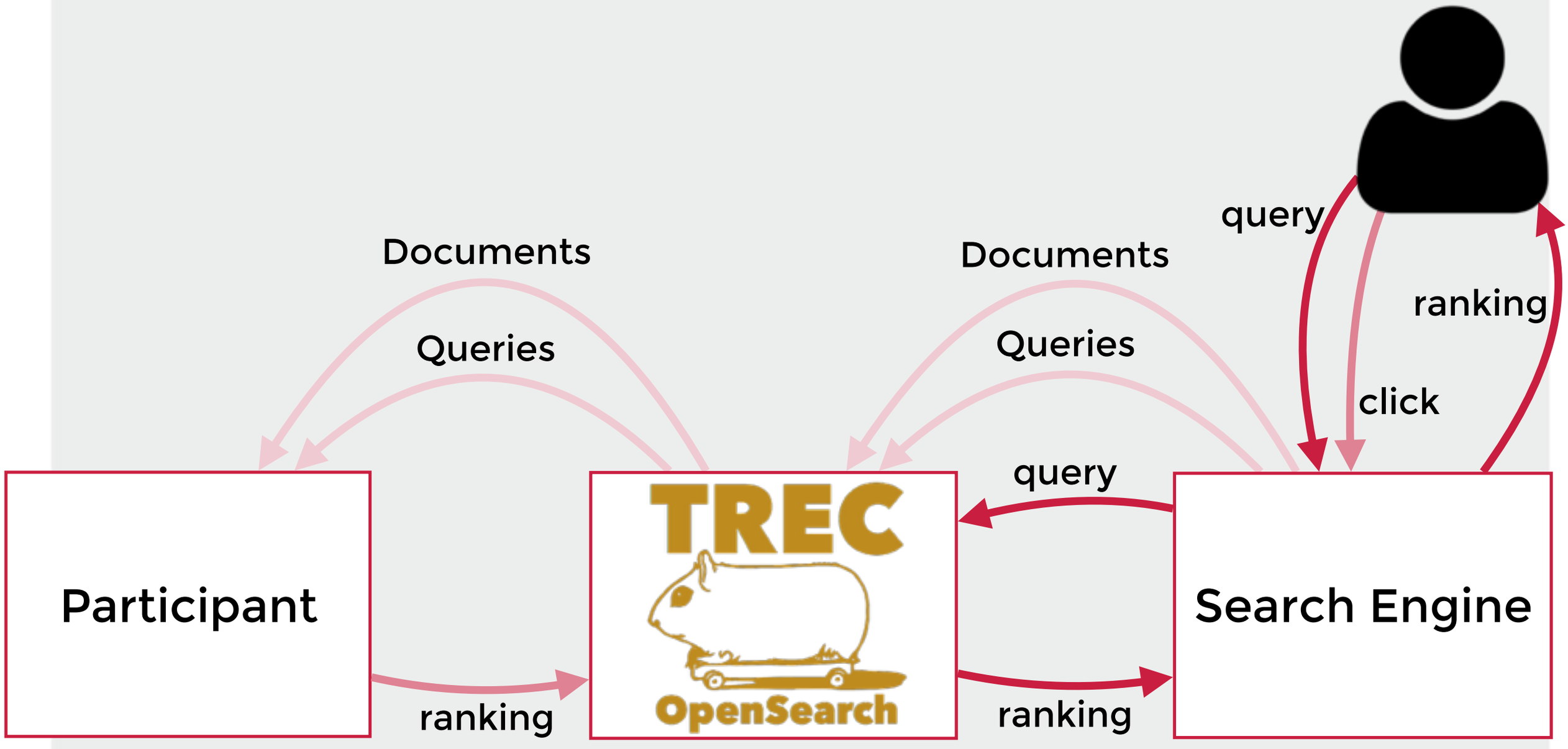
Evaluation Setup



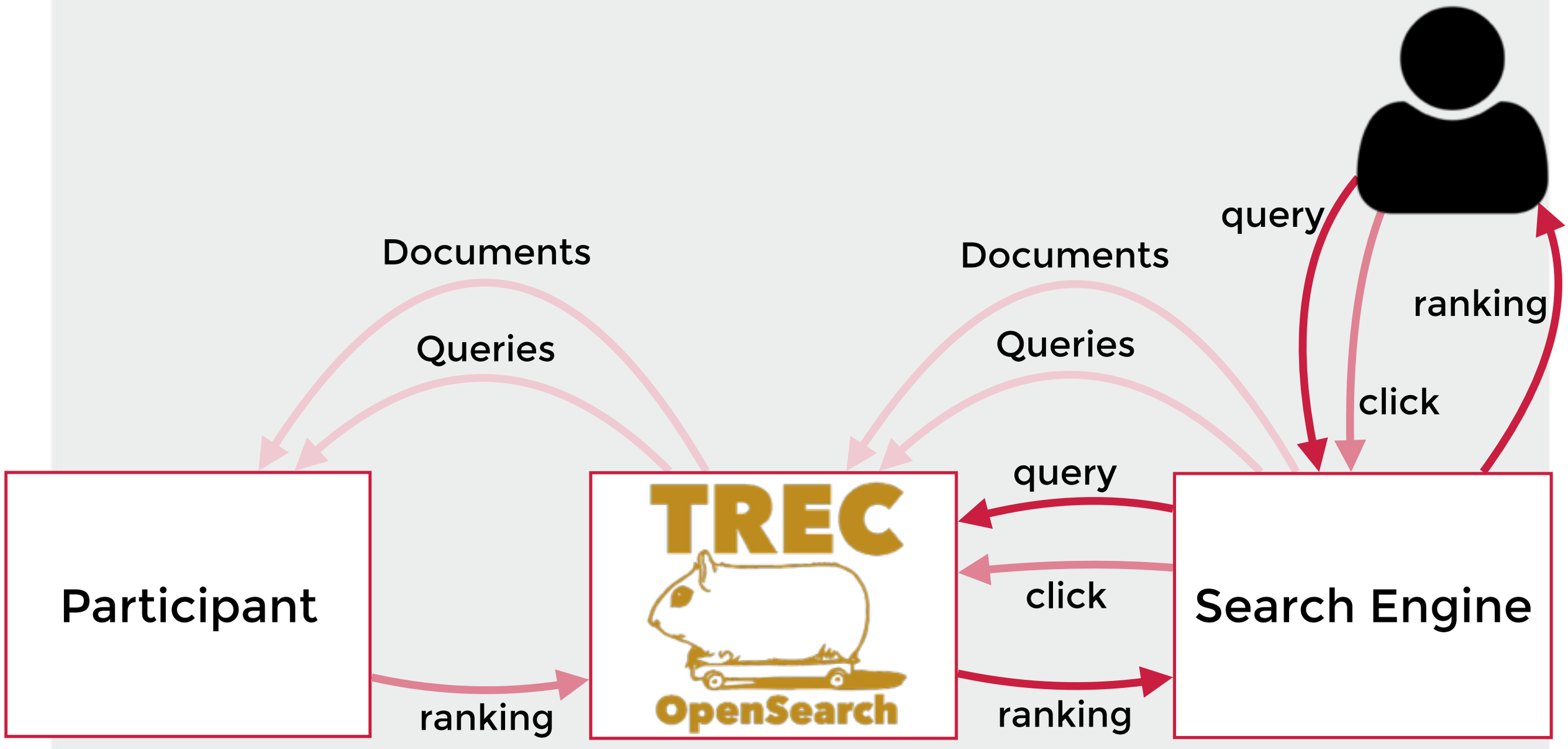
Evaluation Setup



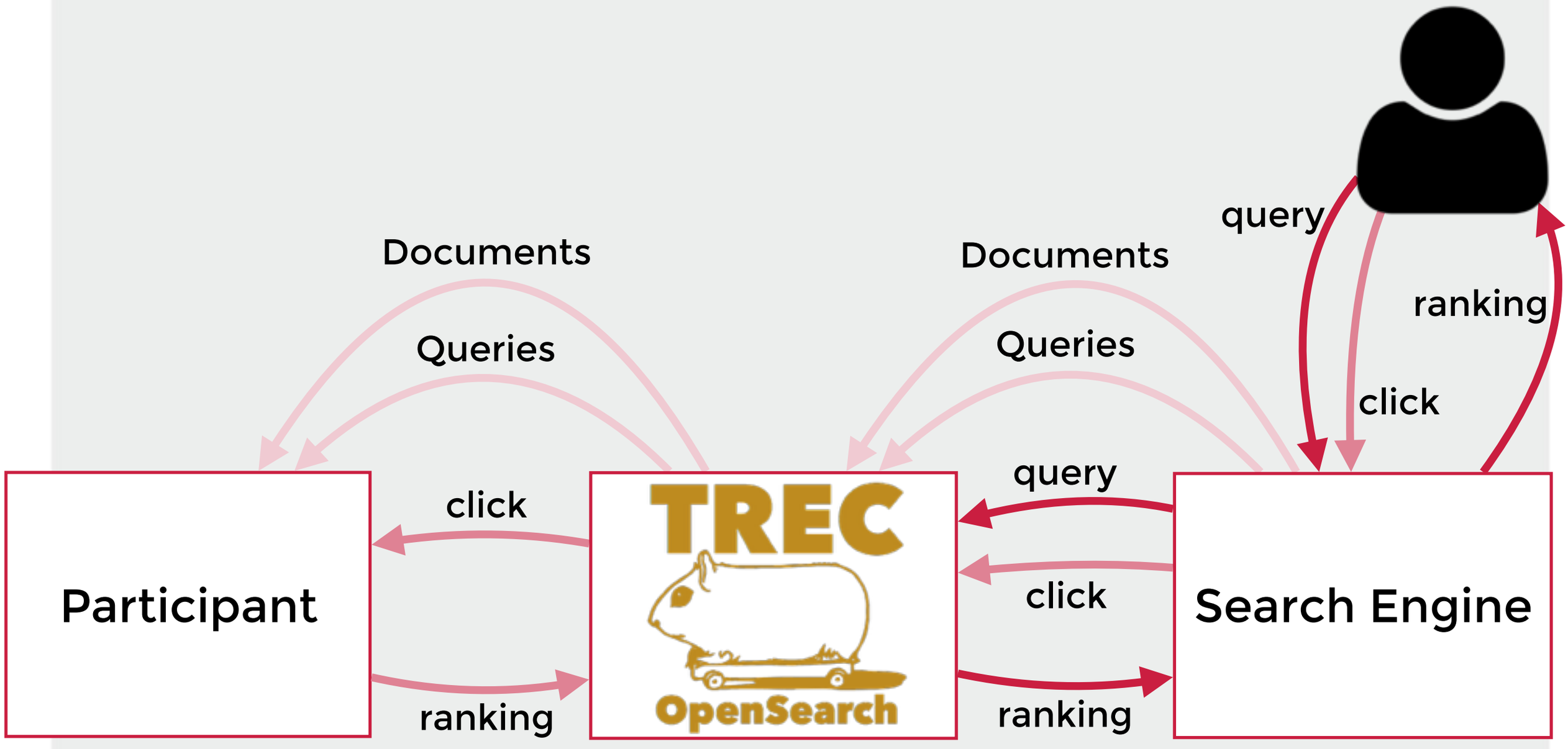
Evaluation Setup



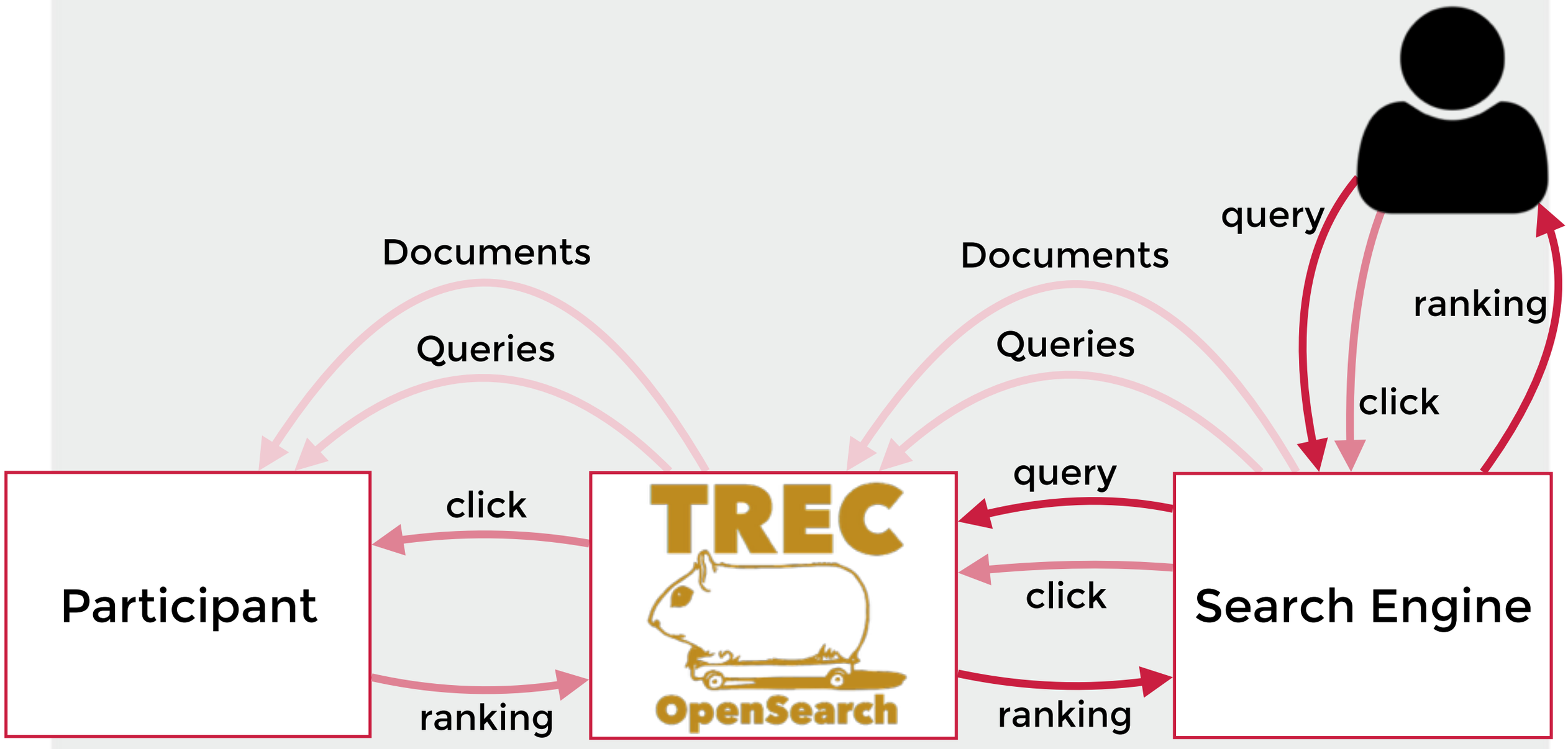
Evaluation Setup



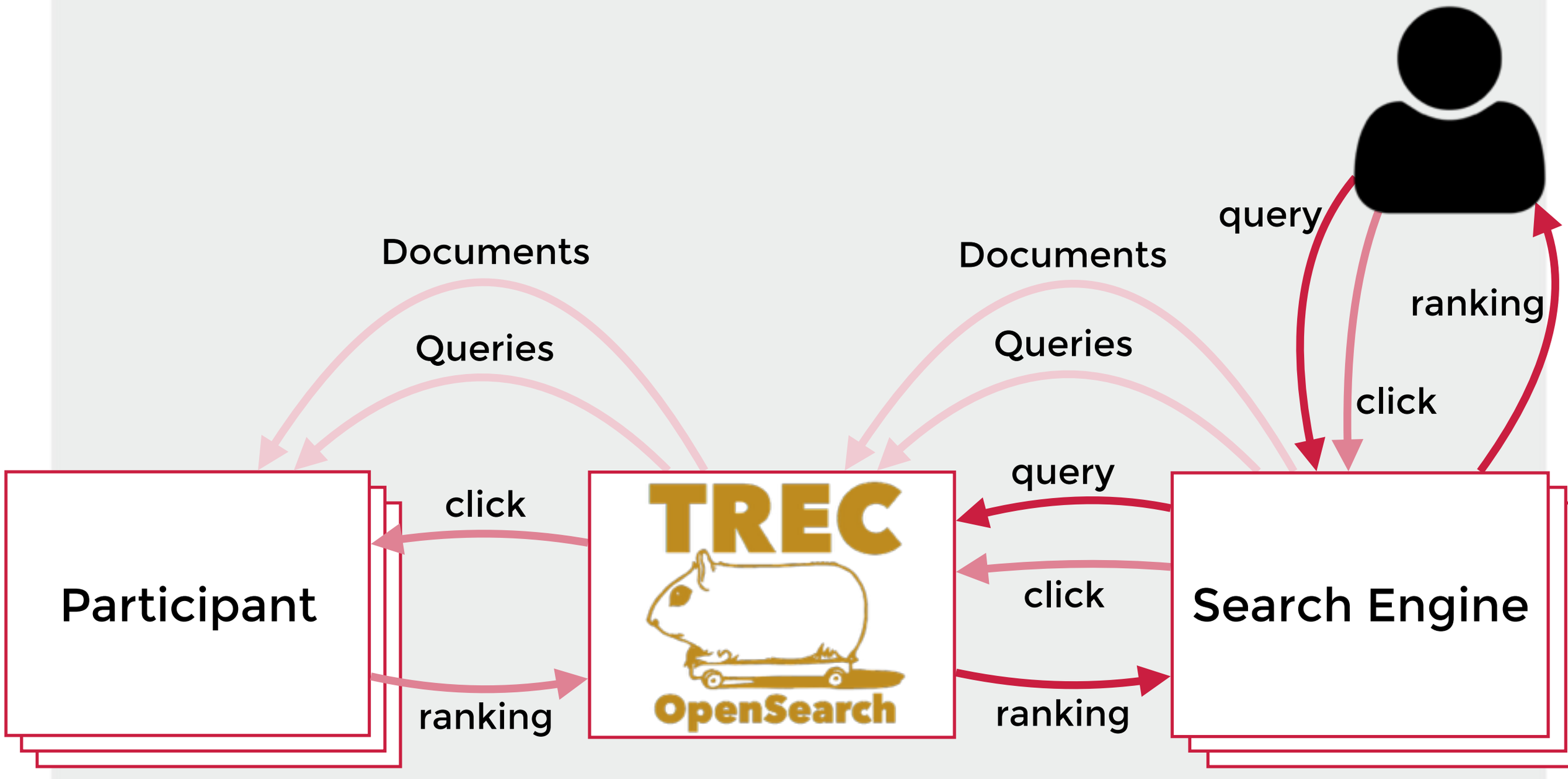
Evaluation Setup



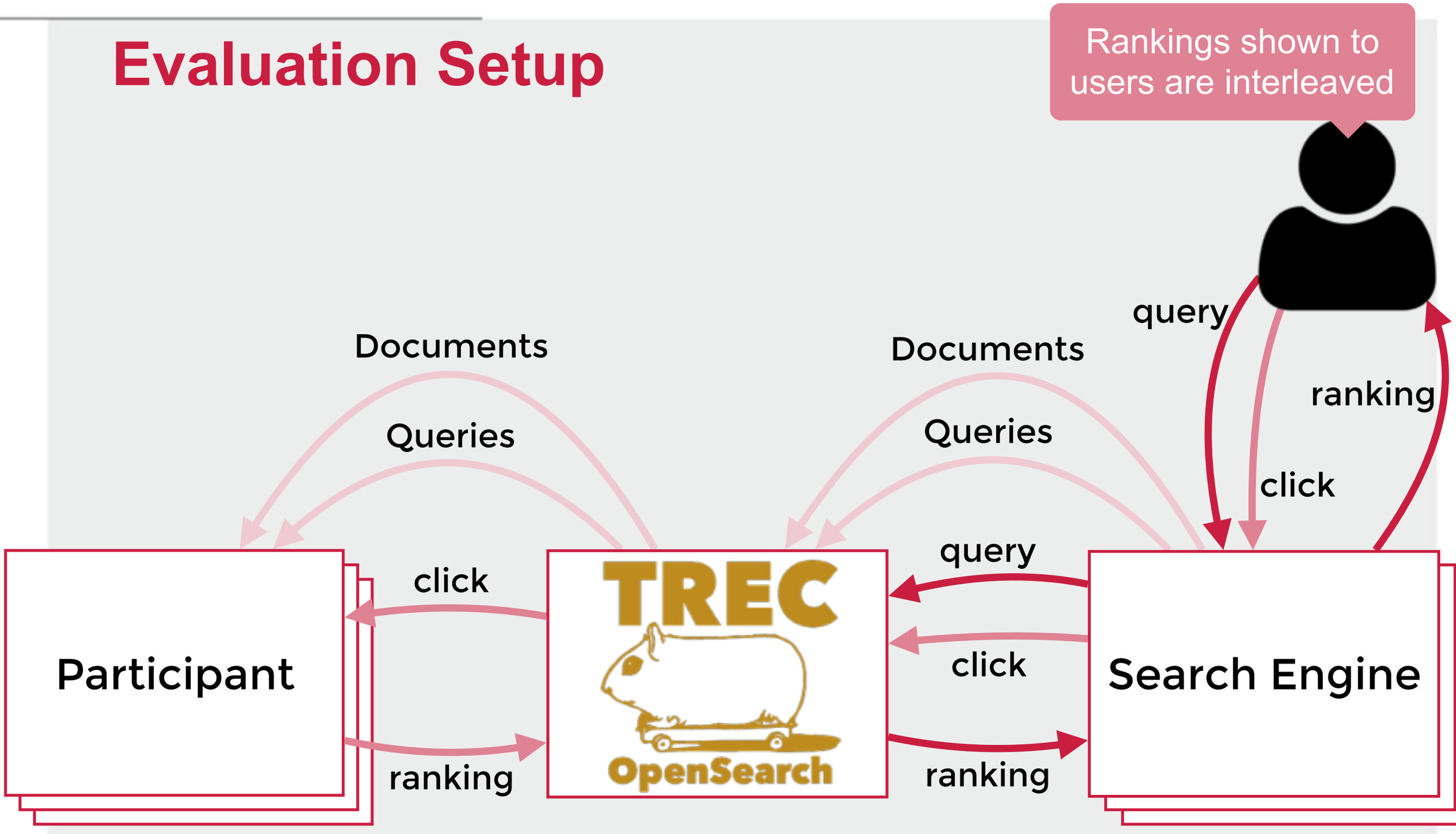
Evaluation Setup



Evaluation Setup



Evaluation Setup



Evaluation Setup - **Interleaving**

A	B
doc 1	doc 2
doc 2	doc 4
doc 3	doc 7
doc 4	doc 1
doc 5	doc 3

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In CIKM'08. 2008

Evaluation Setup - **Interleaving**

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In CIKM'08. 2008

Evaluation Setup - **Interleaving**

A B

doc 1

doc 2

doc 4

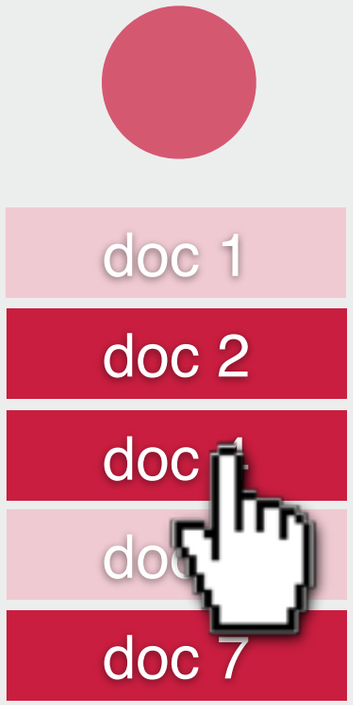
doc 3

doc 7

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In CIKM'08. 2008

Evaluation Setup - **Interleaving**

A B



F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In CIKM'08. 2008

Evaluation Setup - **Interleaving**

- Infer winner: **B** > **A**

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In CIKM'08. 2008

Evaluation Setup - **Interleaving**

- Infer winner: **B** > **A**
- Well tested in practice
  - Used at Bing, Yandex, Seznam

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In CIKM'08. 2008

Evaluation Setup - **Interleaving**

- Infer winner: **B** > **A**
- Well tested in practice
  - Used at Bing, Yandex, Seznam
- Participants are not compared head to head
  - but transitivity holds in practice:
    if A > B and C < B then A > C

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In CIKM'08. 2008

Evaluation Setup - **Interleaving**

- Infer winner: **B** > **A**
- Well tested in practice
  - Used at Bing, Yandex, Seznam
- 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

F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In CIKM'08. 2008

Test/Train Queries/Periods

Query Type
Train

- Feedback **Available**
- **Individual** Feedback
- Update **possible**

Test/Train Queries/Periods

	Query Type	
	Train	Test
Period: Train	- Feedback **Available** - **Individual** Feedback - Update **possible**	- Feedback **Available** - **No** Individual Feedback - Update **possible**

Test/Train Queries/Periods

	Query Type	
	Train	Test
Period: Train	- Feedback **Available** - **Individual** Feedback - Update **possible**	- Feedback **Available** - **No** Individual Feedback - Update **possible**
Period: Test		- **No** Feedback Available - **No** Individual Feedback - Update **not** possible

Documentation

2.1. API Reference for Participants

Note
Please refer the Guide for Participants before reading this.

We provide a basic API for participants of the CLEF Living Labs to perform several actions such as obtaining a key, queries, documents and feedback. The API can also be used to update runs. Everything is implemented as HTTP request, and we use the request types GET, HEAD and PUT. We try to throw appropriate 4XX errors where possible. Furthermore, the content the API returns when a error is thrown should help locate the issue. Please let us know when error messages are not helpful and need clarification.

Note that participants are free to implement their own client to communicate with this API. However, example clients are provided by the organization.

For all operations, an API key is required. Also, we require you to sign an agreement. Details on that process will be shared once you sign up. The dashboard that you can use to obtain an API key is here: http://living-labs.net:5001/

Our API is located at this location: http://living-labs.net:5000/api/.

Note
We have rate limited the API to 300 calls per minute or 10 calls per second, whichever hits first. Please do let us know if this is causing you any problems.

Note
We may sometimes restart our API. You may notice this because the API is down for a few seconds (up to a few minutes). Please implement your client in such a way that this will not cause problems (i.e., add a retry loop with a small sleep to all the API calls).

Documentation

GET /api/participant/query/ (*key*)¶

Obtain the query set for all sites that you have agreed too. If you update the sites you agree too through the dashboard, then the query set will reflect this.

Each query is marked with its type. A query can be a train, test or eval query. Eval queries are supposed to *not* be evaluated online. So, participants will (should) not expect any feedback for them. The default query type is "train".

Parameters:
- **key** – your API key

Status Codes:
- 200 OK – valid key
- 403 Forbidden – invalid key

Return:

```
{
    "queries": [
        {
            "creation_time": "Mon, 10 Nov 2014 17:42:24 -0000",
            "qid": "S-q1",
            "qstr": "jaguar",
            "type": "train"
        },
        {
            "creation_time": "Mon, 10 Nov 2014 17:42:24 -0000",
            "qid": "S-q2",
            "qstr": "apple",
            "type": "test"
        }
    ]
}
```

Documentation

1.4. Implement a Client

We advise you to first familiarize yourself with the API Reference for Participants.

Code that implements a client that talks to this API should approximately take the following logical steps:

1. Obtain queries
2. For each query, obtain a doclist, a list of candidate documents
3. For each document in these doclists, obtain the content of the documents (if any, some uses cases such as Seznam only provides feature vectors as part of the doclist).
4. Create runs, using your ranking algorithm.
5. Upload runs
6. Wait a while to give users a change to interact with your run
7. Download feedback
8. Potentially update your run and repeat from 5.

Examples that implement the above steps are included in the code repository which can be found here: https://bitbucket.org/living-labs/ll-api/

What follows is a *very minimal* example of the above steps. But it should get you up and running. While we used Python, there is no such requirement for you. You are free to use any client that communicate with our API.

Note that this really is a very basic example that is purely exploitative. It sorts documents only by their click counts. While this may be a reasonable baseline, it has a huge risk of getting stuck in local optima (unseen documents never have a change to be clicked). Plus, this approach does not look at the content of document nor at relevance signals (features). Therefore, it will not generalize to unseen queries. Nevertheless, it illustrates how to communicate with the Living Labs API.

1.4.1. Initialize

We start of with some imports and definitions. Replace `KEY` with your own participant key.

```
import requests
import json
import time
import random
```

Advantages of this setup

- Realistic evaluation with real users
- No real-time requirement on participants
- A single implementation to experiment on many search engines
- TREC ad-hoc style queries + documents

Limitations of this setup

- No tail queries
- No sessions
- No context

Task

- Ad-hoc Academic Literature Search

Task

- Ad-hoc Academic Literature Search
- Easy to comprehend

Task

- Ad-hoc Academic Literature Search
- Easy to comprehend
  - The setup is already different enough

Task

- Ad-hoc Academic Literature Search
- Easy to comprehend
  - The setup is already different enough
- Connects to current research

Task

- Ad-hoc Academic Literature Search
- Easy to comprehend
  - The setup is already different enough
- Connects to current research
  - E.g. entity linking, normalization

Task

- Ad-hoc Academic Literature Search
- Easy to comprehend
  - The setup is already different enough
- Connects to current research
  - E.g. entity linking, normalization
- Extendable in future years

Task

- Ad-hoc Academic Literature 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

Task

- Ad-hoc Academic Literature 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

Task

- Ad-hoc Academic Literature 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

Task - **Academic Search Engines**

Task - **Academic Search Engines**

	contact	*enthusiasm*	*commit*	*match timeline*	*implement*	*clicks flowing*

Task - Academic Search Engines

	contact	*enthusiasm*	*commit*	*match timeline*	*implement*	*clicks flowing*
SSOAR						

Task - Academic Search Engines

	contact	*enthusiasm*	*commit*	*match timeline*	*implement*	*clicks flowing*
SSOAR						
OpenEdition						

Task - Academic Search Engines

	contact	*enthusiasm*	*commit*	*match timeline*	*implement*	*clicks flowing*
SSOAR						
OpenEdition						
arXiv						

Task - Academic Search Engines

	contact	*enthusiasm*	*commit*	*match timeline*	*implement*	*clicks flowing*
SSOAR						
OpenEdition						
arXiv						
CiteSeerX						

Task - Academic Search Engines

	contact	*enthusiasm*	*commit*	*match timeline*	*implement*	*clicks flowing*
SSOAR						
OpenEdition						
arXiv						
CiteSeerX						
Google Scholar						

Task - Academic Search Engines

	contact	*enthusiasm*	*commit*	*match timeline*	*implement*	*clicks flowing*
SSOAR						
OpenEdition						
arXiv						
CiteSeerX						
Google Scholar						
MS Academic Search						

Timeline

Timeline

- December: finalize search engine agreements

Timeline

- December: finalize search engine agreements
- January: query + document selection

Timeline

- December: finalize search engine agreements
- January: query + document selection
- March 1: finalize guidelines

Timeline

- December: finalize search engine agreements
- January: query + document selection
- March 1: finalize guidelines
- March 1: release train queries

Timeline

- December: finalize search engine agreements
- January: query + document selection
- March 1: finalize guidelines
- March 1: release train queries
- March 15: clicks start flowing

Timeline

- December: finalize search engine agreements
- January: query + document selection
- March 1: finalize guidelines
- March 1: release train queries
- March 15: clicks start flowing
- May 15: release test queries

Timeline

- December: finalize search engine agreements
- January: query + document selection
- March 1: finalize guidelines
- March 1: release train queries
- March 15: clicks start flowing
- May 15: release test queries
- Jun 1: test period begins

Timeline

- December: finalize search engine agreements
- January: query + document selection
- March 1: finalize guidelines
- March 1: release train queries
- March 15: clicks start flowing
- May 15: release test queries
- Jun 1: test period begins
- July 15: test period ends

Discussion

- Task details
- Query selection
- Relevance assessments (!)
- Simulations
- Reproducibility

Discussion - **Task Details**

- Document format
  - to be discussed
  - full articles may not (always) be possible
  - metadata
- Collection
  - Statistics?
  - Author graph?
- Historical interactions?
- …?

Discussion - **Query Selection**

- Volume
  - Head but few
    - \+ higher sensitivity
    - \- not so interesting?
  - Torso but many
    - \+ more interesting?
    - \- lower sensitivity
  - Mix of both?
- Language
- Type

Discussion - **Relevance Assessments**

- Relevance assessments (for some queries)
  - Compare Cranfield-style evaluation to OpenSearch-style evaluation
- Participants “bidding” on queries?
- No assessments (for the first year)?

Discussion - **Simulations**

- Simulate a search engine
  - Queries sampled according to frequency from log
  - Noisy click model to produce interactions
- So that participants can verify their systems

Discussion - **Repeatability/Reproducibility**

- Share all interactions *after* the test period?
- Have multiple test periods?
- …?