

---

# PapersWithCode Client

*Release 0.3.1*

Viktor Kerkez <alefnula@gmail.com>

Dec 02, 2021



# CONTENTS

<b>1</b>	<b>Contents</b>	<b>3</b>
1.1	Installation . . . . .	3
1.2	Quickstart . . . . .	3
1.3	Mirroring your competition on Papers with Code . . . . .	4
<b>2</b>	<b>API Documentation</b>	<b>5</b>
2.1	PapersWithCode Client api documentation . . . . .	5
<b>3</b>	<b>Indices and tables</b>	<b>39</b>
	<b>Python Module Index</b>	<b>41</b>
	<b>Index</b>	<b>43</b>



This documentation details how to use the `paperswithcode` library to connect with [PapersWithCode](#).



## CONTENTS

## 1.1 Installation

The library requires Python 3.6+ and can be installed via pip:

```
$ pip install paperswithcode-client
```

Usually you would want to create a virtual environment before installing the library:

```
$ python3 -m venv pwc
$ source pwc/bin/activate
$ pip install paperswithcode-client
```

## 1.2 Quickstart

Library is designed to work with python objects. All of the models used as return values in the library are described in the API part of the documentation.

To use the library you only need to import and instantiate the client and start calling methods on it:

```
>>> from paperswithcode import PapersWithCodeClient
>>> client = PapersWithCodeClient()
>>> papers_page = client.paper_list()
>>> papers_page.count
175834
>>> papers_page.next_page
2
>>> paper = papers_page.results[0]
>>> paper.id
'efficient-methods-for-incorporating-knowledge'
>>> paper.title
'Efficient Methods for Incorporating Knowledge into Topic Models'
```

Same principle is used for all models.

```
>>> dataset_page = client.dataset_list()
>>> dataset_page.count
2782
>>> dataset_page.results[0].id
```

(continues on next page)

(continued from previous page)

```
'hci'
>>> dataset_page.results[0].name
'HCI'
```

For nested queries you will need to provide the required id's:

```
>>> conference_page = client.conference_list()
>>> conference = conference_page.results[0]
>>> conference.id
'eccv'
>>> proceedings_page = client.proceeding_list(conference_id=conference.id)
>>> proceeding = proceedings_page.results[0]
>>> proceeding.id
'eccv-2018'
>>> papers = client.proceeding_paper_list(
    conference_id=conference.id, proceeding_id=proceeding.id
)
>>> papers[0].title
'Person Search by Multi-Scale Matching'
```

## 1.3 Mirroring your competition on Papers with Code

For more information on how to mirror your competition, please refer to the [README file in the paperswithcode-client repository](#).



## API DOCUMENTATION

### 2.1 PapersWithCode Client api documentation

#### 2.1.1 PapersWithCode Client Models

##### Paper Models

```
class paperswithcode.models.paper.Paper(*, id: str, arxiv_id: str = None, nips_id: str = None, url_abs: str, url_pdf: str, title: str, abstract: str, authors: List[str], published: datetime.date = None, conference: str = None, conference_url_abs: str = None, conference_url_pdf: str = None, proceeding: str = None)
```

Paper object.

**id**

Paper ID.

**Type** str

**arxiv\_id**

ArXiv ID.

**Type** str, optional

**nips\_id**

NIPS Conference ID.

**Type** str, optional

**url\_abs**

URL to the paper abstract.

**Type** str

**url\_pdf**

URL to the paper PDF.

**Type** str

**title**

Paper title.

**Type** str

**abstract**

Paper abstract.

**Type** str

**authors**

List of paper authors.

**Type** List[str]

**published**

Paper publication date.

**Type** date, optional

**conference**

ID of the conference in which the paper was published.

**Type** str, optional

**conference\_url\_abs**

URL to the conference paper page.

**Type** str, optional

**conference\_url\_pdf**

URL to the conference paper PDF.

**Type** str, optional

**proceeding**

ID of the conference proceeding in which the paper was published.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.paper.Papers(*, count: int, next_page: int = None, previous_page: int = None, results: List[paperswithcode.models.paper.Paper])
```

Object representing a paginated page of papers.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of papers on this page.

**Type** List[Paper]

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

## Repository Models

**class** `paperswithcode.models.repository.Repository`(\*, *url: str, owner: str, name: str, description: str, stars: int, framework: str, is\_official: bool = None*)

Repository object.

**url**

URL of the repository.

**Type** `str`

**owner**

Repository owner.

**Type** `str`

**name**

Repository name.

**Type** `str`

**description**

Repository description.

**Type** `str`

**stars**

Number of repository stars.

**Type** `int`

**framework**

Implementation framework (TensorFlow, PyTorch, MXNet, Torch, Jax, Caffee2...).

**Type** `str`

**is\_official**

Is this an official implementation of the paper. Available only when listing repositories for a specific paper.

**Type** `bool`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

**class** `paperswithcode.models.repository.Repositories`(\*, *count: int, next\_page: int = None, previous\_page: int = None, results: List[paperswithcode.models.repository.Repository]*)

Object representing a paginated page of repositories.

**count**

Number of elements matching the query.

**Type** `int`

**next\_page**

Number of the next page.

**Type** `int, optional`

**previous\_page**

Number of the previous page.

**Type** `int, optional`

**results**

List of repositories on this page.

**Type** List[[Repository](#)]

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

**Paper Repository Models**

```
class paperswithcode.models.paper_repo.PaperRepo(*, paper: paperswithcode.models.paper.Paper,  
                                                    repository:  
paperswithcode.models.repository.Repository =  
None, is_official: bool)
```

Paper <-> Repository object.

**paper**

Paper objects.

**Type** [Paper](#)

**repository**

Repository object.

**Type** [Repository](#), optional

**is\_official**

Is this the official implementation.

**Type** bool

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.paper_repo.PaperRepos(*, count: int, next_page: int = None,  
                                                    previous_page: int = None, results:  
List[paperswithcode.models.paper_repo.PaperRepo])
```

Object representing a paginated page of paper<->repos.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of paper<->repos on this page.

**Type** List[[PaperRepo](#)]

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

## Author Models

**class** paperswithcode.models.author.**Author**(\**id: str, full\_name: str*)

Author object.

**id**

Author ID.

**Type** str

**full\_name**

Author full name.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

**class** paperswithcode.models.author.**Authors**(\**count: int, next\_page: int = None, previous\_page: int = None, results: List[paperswithcode.models.author.Author]*)

Object representing a paginated page of authors.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of authors on this page.

**Type** List[[Author](#)]

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

## Conference Models

**class** paperswithcode.models.conference.**Conference**(\**id: str, name: str*)

Conference object.

**id**

Conference ID.

**Type** str

**name**

Conference name.

**Type** str

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.conference.Conferences(*, count: int, next_page: int = None,  
                                                    previous_page: int = None, results:  
                                                    List[paperswithcode.models.conference.Conference])
```

Object representing a paginated page of conferences.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of conferences on this page.

**Type** List[[Conference](#)]

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.conference.Proceeding(*, id: str, year: int = None, month: int = None)  
Conference proceeding object.
```

**id**

Proceeding ID.

**Type** str

**year**

Year in which the proceeding was held.

**Type** int, optional

**month**

Month in which the proceedingt was held.

**Type** int, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.conference.Proceedings(*, count: int, next_page: int = None,  
                                                    previous_page: int = None, results:  
                                                    List[paperswithcode.models.conference.Proceeding])
```

Object representing a paginated page of proceedings.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of proceedings on this page.

**Type** List[*Proceeding*]

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

**Task Models**

**class** paperswithcode.models.task.**Area**(\*, id: str, name: str)

Area object.

Representing an area of research.

**id**

Area ID.

**Type** str

**name**

Area name.

**Type** str

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

**class** paperswithcode.models.task.**Areas**(\*, count: int, next\_page: int = None, previous\_page: int = None, results: List[paperswithcode.models.task.Area])

Object representing a paginated page of areas.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of areas on this page.

**Type** List[*Area*]

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

**class** paperswithcode.models.task.**Task**(\*, id: str, name: str, description: str)

Task object.

**id**

Task ID.

**Type** str**name**

Task name.

**Type** str**description**

Task description.

**Type** str

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.task.TaskCreateRequest(*, name: str, description: str = "", area: str =  
None, parent_task: str = None)
```

Task object.

**name**

Task name.

**Type** str**description**

Task description.

**Type** str**area**

Task area ID or area name.

**Type** str, optional**parent\_task**

ID of the parent task.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.task.TaskUpdateRequest(*, name: str = None, description: str = None,  
area: str = None, parent_task: str = None)
```

Evaluation table row object.

**name**

Task name.

**Type** str, optional**description**

Task description.

**Type** str, optional**area**

Task area ID.

**Type** str, optional



**parent\_task**

ID of the parent task.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

**class** `paperswithcode.models.task.Tasks`(\*, *count*: int, *next\_page*: int = None, *previous\_page*: int = None, *results*: List[`paperswithcode.models.task.Task`])

Object representing a paginated page of tasks.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of tasks on this page.

**Type** List[`Task`]

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

**Dataset Models**

**class** `paperswithcode.models.dataset.Dataset`(\*, *id*: str, *name*: str, *full\_name*: str = None, *url*: str = None)

Dataset object.

**id**

Dataset ID.

**Type** str

**name**

Dataset name.

**Type** str

**full\_name**

Dataset full name.

**Type** str, optional

**url**

URL for dataset download.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.dataset.DatasetCreateRequest(*, name: str, full_name: str = None, url: str = None)
```

Task object.

**name**

Dataset name.

**Type** str

**full\_name**

Dataset full name.

**Type** str, optional

**url**

Dataset url.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.dataset.DatasetUpdateRequest(*, name: str = None, url: str = None)
```

Evaluation table row object.

**name**

Dataset name.

**Type** str, optional

**url**

Dataset url.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.dataset.Datasets(*, count: int, next_page: int = None, previous_page: int = None, results: List[paperswithcode.models.dataset.Dataset])
```

Object representing a paginated page of datasets.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of datasets on this page.

**Type** List[[Dataset](#)]

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

## Method Models

**class** paperswithcode.models.method.**Method**(\**id: str, name: str, full\_name: str, description: str, paper: str = None*)

Method object.

**id**

Method ID.

**Type** str

**name**

Method short name.

**Type** str

**full\_name**

Method full name.

**Type** str

**description**

Method description.

**Type** str

**paper**

ID of the paper that describes the method.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

**class** paperswithcode.models.method.**Methods**(\**count: int, next\_page: int = None, previous\_page: int = None, results: List[paperswithcode.models.method.Method]*)

Object representing a paginated page of methods.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of methods on this page.

**Type** List[[Method](#)]

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

## Evaluation Table Models

**class** `paperswithcode.models.evaluation.Metric`(\**id: str, name: str, description: str, is\_loss: bool*)  
Metric object.

Metric used for evaluation.

**id**

Metric id.

**Type** `str`

**name**

Metric name.

**Type** `str`

**description**

Metric description.

**Type** `str`

**is\_loss**

Is this a loss metric.

**Type** `bool`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

**class** `paperswithcode.models.evaluation.Metrics`(\**count: int, next\_page: int = None, previous\_page: int = None, results: List[paperswithcode.models.evaluation.metric.Metric]*)

Object representing a paginated page of metrics.

**count**

Number of elements matching the query.

**Type** `int`

**next\_page**

Number of the next page.

**Type** `int`, optional

**previous\_page**

Number of the previous page.

**Type** `int`, optional

**results**

List of metrics on this page.

**Type** `List[Metric]`

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.MetricCreateRequest(*, name: str, description: str, is_loss:
                                                         bool)
```

Metric object.

Metric used for evaluation.

**name**

Metric name.

**Type** str

**description**

Metric description.

**Type** str

**is\_loss**

Is this a loss metric.

**Type** bool

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.MetricUpdateRequest(*, name: str = None, description: str =
                                                         None, is_loss: bool = None)
```

Metric object.

Metric used for evaluation.

**name**

Metric name.

**Type** str, optional

**description**

Metric description.

**Type** str, optional

**is\_loss**

Is this a loss metric.

**Type** bool, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.Result(*, id: str, best_rank: int = None, metrics: dict,
                                                methodology: str, uses_additional_data: bool, paper:
                                                str = None, best_metric: str = None, evaluated_on: str =
                                                None, external_source_url: str = None)
```

Evaluation table row object.

**id**

Result id.

**Type** str

**best\_rank**

Best rank of the row.

**Type** int, optional

**metrics**

Dictionary of metrics and metric values.

**Type** dict

**methodology**

Methodology used for this implementation.

**Type** str

**uses\_additional\_data**

Does this evaluation uses additional data not provided in the dataset used for other evaluations.

**Type** bool

**paper**

Paper describing the evaluation.

**Type** str, optional

**best\_metric**

Name of the best metric.

**Type** str, optional

**evaluated\_on**

Date of the result evaluation in YYYY-MM-DD format.

**Type** str, optional

**external\_source\_url**

The URL to the external source (eg competition).

**Type** str, option

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.Results(*, count: int, next_page: int = None, previous_page:
                                             int = None, results:
                                             List[paperswithcode.models.evaluation.result.Result])
```

Object representing a paginated page of results.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.

**Type** int, optional

**results**

List of results on this page.

**Type** List[[Result](#)]

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.ResultCreateRequest(*, metrics: dict, methodology: str,
                                                         uses_additional_data: bool = False,
                                                         paper: str = None, evaluated_on: str =
                                                         None, external_source_url: str =
                                                         None)
```

Evaluation table row object.

**metrics**

Dictionary of metrics and metric values.

**Type** dict

**methodology**

Methodology used for this implementation.

**Type** str

**uses\_additional\_data**

Does this evaluation uses additional data not provided in the dataset used for other valuations.

**Type** bool, optional

**paper**

Paper describing the evaluation.

**Type** str, optional

**evaluated\_on**

Date of the result evaluation: YYYY-MM-DD format.

**Type** str, optional

**external\_source\_url**

The URL to the external source (eg competition).

**Type** str, option

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.ResultUpdateRequest(*, metrics: dict = None, methodology:
                                                         str = None, uses_additional_data: bool
                                                         = None, paper: str = None,
                                                         evaluated_on: str = None,
                                                         external_source_url: str = None)
```

Evaluation table row object.

**metrics**

Dictionary of metrics and metric values.

**Type** dict, optional

**methodology**

Methodology used for this implementation.

**Type** str, optional

**uses\_additional\_data**

Does this evaluation uses additional data not provided in the dataset used for other evaluations.

**Type** bool, optional

**paper**

Paper describing the evaluation.

**Type** str, optional

**evaluated\_on**

Date of the result evaluation: YYYY-MM-DD format.

**Type** datetime, optional

**external\_source\_url**

The URL to the external source (eg competition).

**Type** str, option

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.EvaluationTable(*, id: str, task: str, dataset: str, description:  
                                                    str = "", mirror_url: str = None)
```

Evaluation table object.

**id**

Evaluation table ID.

**Type** str

**task**

ID of the task used in evaluation.

**Type** str

**dataset**

ID of the dataset used in evaluation.

**Type** str

**description**

Evaluation table description.

**Type** str

**mirror\_url**

URL to the evaluation table that this table is a mirror of.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.EvaluationTables(*, count: int, next_page: int = None,  
                                                         previous_page: int = None, results:  
                                                         List[paperswithcode.models.evaluation.table.EvaluationTable])
```

Object representing a paginated page of evaluation tables.

**count**

Number of elements matching the query.

**Type** int

**next\_page**

Number of the next page.

**Type** int, optional

**previous\_page**

Number of the previous page.



**Type** int, optional

### results

List of evaluation tables on this page.

**Type** List[SotaPartial]

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.EvaluationTableCreateRequest(*, task: str, dataset: str,
                                                                    description: str = "",
                                                                    mirror_url: str = None)
```

Evaluation table create request object.

### task

ID of the task used in evaluation.

**Type** str

### dataset

ID of the dataset used in evaluation.

**Type** str

### description

Evaluation table description.

**Type** str

### mirror\_url

URL to the evaluation table that this table is a mirror of.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.EvaluationTableUpdateRequest(*, task: str = None, dataset:
                                                                    str = None, description: str
                                                                    = None, mirror_url: str =
                                                                    None)
```

Evaluation table update request object.

### task

ID of the task used in evaluation.

**Type** str, optional

### dataset

ID of the dataset used in evaluation.

**Type** str, optional

### description

Evaluation table description.

**Type** str, optional

### mirror\_url

URL to the evaluation table that this table is a mirror of.

**Type** str, optional

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.ResultSyncRequest(*, metrics: dict, methodology: str, paper: str = None, uses_additional_data: bool = False, external_id: str = "", evaluated_on: str, external_source_url: str = None)
```

Evaluation table row object.

**metrics**

Dictionary of metrics and metric values.

**Type** dict

**methodology**

Methodology used for this implementation.

**Type** str

**uses\_additional\_data**

Does this evaluation uses additional data not provided in the dataset used for other evaluations.

**Type** bool

**paper**

Paper describing the evaluation.

**Type** str, optional

**external\_id**

Optional external ID used to identify rows when doing sync.

**Type** str, optional

**evaluated\_on**

Evaluation date in YYYY-MM-DD format

**Type** str

**external\_source\_url**

The URL to the external source (eg competition).

**Type** str, option

Create a new model by parsing and validating input data from keyword arguments.

Raises `ValidationError` if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.MetricSyncRequest(*, name: str, description: str = "", is_loss: bool = True)
```

Metric object.

Metric used for evaluation.

**name**

Metric name.

**Type** str

**description**

Metric description.

**Type** str

**is\_loss**

Is this a loss metric.

**Type** bool

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.EvaluationTableSyncRequest(*, task: str, dataset: str,
    description: str = "",
    mirror_url: str = None,
    external_id: str = None,
    metrics:
        List[paperswithcode.models.evaluation.synchron
    = None, results:
        List[paperswithcode.models.evaluation.synchron
    = None)
```

Evaluation table object.

**task**

ID of the task used in evaluation.

**Type** str

**dataset**

ID of the dataset used in evaluation.

**Type** str

**description**

Evaluation table description.

**Type** str

**mirror\_url**

URL to the evaluation table that this table is a mirror of.

**Type** str, optional

**external\_id**

Optional external ID used to identify rows when doing sync.

**Type** str, optional

**metric**

List of MetricSyncRequest objects used in the evaluation.

**Type** list

**results**

List of ResultSyncRequest objects - results of the evaluation.

**Type** list

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.ResultSyncResponse(*id: str, metrics: dict, methodology:  
str, paper: str = None,  
uses_additional_data: bool = False,  
external_id: str = "", evaluated_on: str =  
None, external_source_url: str = None)
```

Evaluation table row object.

**id**

Result id.

**Type** str

**metrics**

Dictionary of metrics and metric values.

**Type** dict

**methodology**

Methodology used for this implementation.

**Type** str

**uses\_additional\_data**

Does this evaluation uses additional data not provided in the dataset used for other evaluations.

**Type** bool

**paper**

Paper describing the evaluation.

**Type** str, optional

**external\_id**

Optional external ID used to identify rows when doing sync.

**Type** str, optional

**evaluated\_on**

Evaluation date in YYYY-MM-DD format

**Type** str, optional

**external\_source\_url**

The URL to the external source (eg competition)

**Type** str, option

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.MetricSyncResponse(*name: str, description: str = "",  
is_loss: bool = True)
```

Metric object.

Metric used for evaluation.

**name**

Metric name.

**Type** str

**description**

Metric description.

**Type** str

**is\_loss**

Is this a loss metric.

**Type** bool

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

```
class paperswithcode.models.evaluation.EvaluationTableSyncResponse(*, id: str, task: str, dataset:
                                                                    str, description: str = "",
                                                                    mirror_url: str = None,
                                                                    external_id: str = "", metrics:
                                                                    List[paperswithcode.models.evaluation.syncro
                                                                    = None, results:
                                                                    List[paperswithcode.models.evaluation.syncro
                                                                    = None)
```

Evaluation table object.

**id**

Evaluation table ID.

**Type** str

**task**

ID of the task used in evaluation.

**Type** str

**dataset**

ID of the dataset used in evaluation.

**Type** str

**description**

Evaluation table description.

**Type** str

**mirror\_url**

URL to the evaluation table that this table is a mirror of.

**Type** str, optional

**external\_id**

Optional external ID used to identify rows when doing sync.

**Type** str, optional

**metric**

List of metrics sync objects used in the evaluation.

**Type** list

**results**

List of result sync objects - results of the evaluation.

**Type** list

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

## 2.1.2 PapersWithCode Client Class

**class** `paperswithcode.client.PapersWithCodeClient`(*token=None, url=None*)  
PapersWithCode client.

**\_\_init\_\_**(*token=None, url=None*)  
Initialize self. See `help(type(self))` for accurate signature.

**search**(*q: Optional[str] = None, page: int = 1, items\_per\_page: int = 50*) →  
[\*paperswithcode.models.paper\\_repo.PaperRepos\*](#)  
Search in a similar fashion to the frontpage search.

### Parameters

- **q** (*str, optional*) – Filter papers by querying the paper title and abstract.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** `PaperRepos` object.

**Return type** [\*PaperRepos\*](#)

**paper\_list**(*q: Optional[str] = None, arxiv\_id: Optional[str] = None, title: Optional[str] = None, abstract: Optional[str] = None, page: int = 1, items\_per\_page: int = 50*) →  
[\*paperswithcode.models.paper.Papers\*](#)

Return a paginated list of papers.

### Parameters

- **q** (*str, optional*) – Filter papers by querying the paper title and abstract.
- **arxiv\_id** (*str, optional*) – Filter papers by arxiv id.
- **title** (*str, optional*) – Filter papers by part of the title.
- **abstract** (*str, optional*) – Filter papers by part of the abstract.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** `Papers` object.

**Return type** [\*Papers\*](#)

**paper\_get**(*paper\_id: str*) → [\*paperswithcode.models.paper.Paper\*](#)  
Return a paper by it's ID.

**Parameters** **paper\_id** (*str*) – ID of the paper.

**Returns** `Paper` object.

**Return type** [\*Paper\*](#)

**paper\_dataset\_list**(*paper\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
[\*paperswithcode.models.repository.Repositories\*](#)

Return a list of datasets mentioned in the paper..

### Parameters

- **paper\_id** (*str*) – ID of the paper.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Datasets object.

**Return type** *Datasets*

**paper\_repository\_list**(*paper\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
*paperswithcode.models.repository.Repositories*

Return a list of paper implementations.

**Parameters**

- **paper\_id** (*str*) – ID of the paper.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Repositories object.

**Return type** *Repositories*

**paper\_task\_list**(*paper\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
*paperswithcode.models.task.Tasks*

Return a list of tasks mentioned in the paper.

**Parameters**

- **paper\_id** (*str*) – ID of the paper.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Tasks object.

**Return type** *Tasks*

**paper\_method\_list**(*paper\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
*paperswithcode.models.method.Methods*

Return a list of methods mentioned in the paper.

**Parameters**

- **paper\_id** (*str*) – ID of the paper.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Methods object.

**Return type** *Methods*

**paper\_result\_list**(*paper\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
*paperswithcode.models.evaluation.result.Results*

Return a list of evaluation results for the paper.

**Parameters**

- **paper\_id** (*str*) – ID of the paper.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Results object.

**Return type** *Results*

**repository\_list**(*q: Optional[str] = None, owner: Optional[str] = None, name: Optional[str] = None, stars: Optional[int] = None, framework: Optional[str] = None, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.paper.Papers*

Return a paginated list of repositories.

**Parameters**

- **q** (*str, optional*) – Search all searchable fields.
- **owner** (*str, optional*) – Filter repositories by owner.
- **name** (*str, optional*) – Filter repositories by name.
- **stars** (*int, optional*) – Filter repositories by minimum number of stars.
- **framework** (*str, optional*) – Filter repositories by framework. Available values: tf, pytorch, mxnet, torch, caffe2, jax, paddle, mindspore.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Repositories object.

**Return type** *Repositories*

**repository\_owner\_list**(*owner: str*) → *paperswithcode.models.repository.Repositories*

List all repositories for a specific repo owner.

**Parameters** **owner** (*str*) – Repository owner.

**Returns** Repositories object.

**Return type** *Repositories*

**repository\_get**(*owner: str, name: str*) → *paperswithcode.models.repository.Repository*

Return a repository by it's owner/name pair.

**Parameters**

- **owner** (*str*) – Owner name.
- **name** (*str*) – Repository name.

**Returns** Repository object.

**Return type** *Repository*

**repository\_paper\_list**(*owner: str, name: str, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.paper.Papers*

List all papers connected to the repository.

**Parameters**

- **owner** (*str*) – Owner name.
- **name** (*str*) – Repository name.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Papers object.

**Return type** *Papers*



**author\_list**(*q: Optional[str] = None, full\_name: Optional[str] = None, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.author.Authors*

Return a paginated list of paper authors.

**Parameters**

- **q** (*str, optional*) – Search all searchable fields.
- **full\_name** (*str, optional*) – Filter authors by part of their full name.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Repositories object.

**Return type** *Repositories*

**author\_get**(*author\_id: str*) → *paperswithcode.models.author.Author*

Return a specific author selected by its id.

**Parameters** **author\_id** (*str*) – Author id.

**Returns** Author object.

**Return type** *Author*

**author\_paper\_list**(*author\_id: str, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.paper.Papers*

List all papers connected to the author.

**Parameters**

- **author\_id** (*str*) – Author id.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Papers object.

**Return type** *Papers*

**conference\_list**(*q: Optional[str] = None, name: Optional[str] = None, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.conference.Conferences*

Return a paginated list of conferences.

**Parameters**

- **q** (*str, optional*) – Search all searchable fields.
- **name** (*str, optional*) – Filter conferences by part of the name.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Conferences object.

**Return type** *Conferences*

**conference\_get**(*conference\_id: str*) → *paperswithcode.models.conference.Conference*

Return a conference by it's ID.

**Parameters** **conference\_id** (*str*) – ID of the conference.

**Returns** Conference object.

**Return type** *Conference*

**proceeding\_list**(*conference\_id: str, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.conference.Proceedings*

Return a paginated list of conference proceedings.

**Parameters**

- **conference\_id** (*str*) – ID of the conference.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Proceedings object.

**Return type** *Proceedings*

**proceeding\_get**(*conference\_id: str, proceeding\_id: str*) → *paperswithcode.models.conference.Proceeding*

Return a conference proceeding by it's ID.

**Parameters**

- **conference\_id** (*str*) – ID of the conference.
- **proceeding\_id** (*str*) – ID of the proceeding.

**Returns** Proceeding object.

**Return type** *Proceeding*

**proceeding\_paper\_list**(*conference\_id: str, proceeding\_id: str, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.paper.Papers*

Return a list of papers published in a confrence proceeding.

**Parameters**

- **conference\_id** (*str*) – ID of the conference.
- **proceeding\_id** (*str*) – ID of the proceeding.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Papers object.

**Return type** *Papers*

**area\_list**(*q: Optional[str] = None, name: Optional[str] = None, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.task.Areas*

Return a paginated list of areas.

**Parameters**

- **q** (*str, optional*) – Filter areas by querying the area name.
- **name** (*str, optional*) – Filter areas by part of the name.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Areas object.

**Return type** *Areas*

**area\_get**(*area\_id: str*) → *paperswithcode.models.task.Area*

Return an area by it's ID.

**Parameters** **area\_id** (*str*) – ID of the area.

**Returns** Area object.

**Return type** *Area*

**area\_task\_list**(*area\_id*: str, *page*: int = 1, *items\_per\_page*: int = 50) → *paperswithcode.models.task.Tasks*

Return a paginated list of tasks in an area.

**Parameters**

- **area\_id** (str) – ID of the area.
- **page** (int) – Desired page.
- **items\_per\_page** (int) – Desired number of items per page. Default: 50.

**Returns** Tasks object.

**Return type** *Tasks*

**task\_list**(*q*: Optional[str] = None, *name*: Optional[str] = None, *page*: int = 1, *items\_per\_page*: int = 50) → *paperswithcode.models.task.Tasks*

Return a paginated list of tasks.

**Parameters**

- **q** (str, optional) – Filter tasks by querying the task name.
- **name** (str, optional) – Filter tasks by part of th name.
- **page** (int) – Desired page.
- **items\_per\_page** (int) – Desired number of items per page. Default: 50.

**Returns** Tasks object.

**Return type** *Tasks*

**task\_get**(*task\_id*: str) → *paperswithcode.models.task.Task*

Return a task by it's ID.

**Parameters** **task\_id** (str) – ID of the task.

**Returns** Task object.

**Return type** *Task*

**task\_add**(*task*: *paperswithcode.models.task.TaskCreateRequest*) → *paperswithcode.models.task.Task*

Add a task.

**Parameters** **task** (*TaskCreateRequest*) – Task create request.

**Returns** Created task.

**Return type** *Task*

**task\_update**(*task\_id*: str, *task*: *paperswithcode.models.task.TaskUpdateRequest*) → *paperswithcode.models.task.Task*

Update a task.

**Parameters**

- **task\_id** (str) – ID of the task.
- **task** (*TaskUpdateRequest*) – Task update request.

**Returns** Updated task.

**Return type** *Task*

**task\_delete**(*task\_id: str*)

Delete a task.

**Parameters** **task\_id** (*str*) – ID of the task.

**task\_parent\_list**(*task\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
*paperswithcode.models.task.Tasks*

Return a paginated list of parent tasks for a selected task.

**Parameters**

- **task\_id** (*str*) – ID of the task.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Tasks object.

**Return type** *Tasks*

**task\_child\_list**(*task\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
*paperswithcode.models.task.Tasks*

Return a paginated list of child tasks for a selected task.

**Parameters**

- **task\_id** (*str*) – ID of the task.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Tasks object.

**Return type** *Tasks*

**task\_paper\_list**(*task\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
*paperswithcode.models.paper.Papers*

Return a paginated list of papers for a selected task.

**Parameters**

- **task\_id** (*str*) – ID of the task.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Papers object.

**Return type** *Papers*

**task\_evaluation\_list**(*task\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
*paperswithcode.models.evaluation.table.EvaluationTables*

Return a list of evaluation tables for a selected task.

**Parameters**

- **task\_id** (*str*) – ID of the task.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** EvaluationTables object.

**Return type** *EvaluationTables*

**dataset\_list**(*q: Optional[str] = None, name: Optional[str] = None, full\_name: Optional[str] = None, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.dataset.Datasets*

Return a paginated list of datasets.

**Parameters**

- **q** (*str, optional*) – Filter datasets by querying the dataset name.
- **name** (*str, optional*) – Filter datasets by their name.
- **full\_name** (*str, optional*) – Filter datasets by their full name.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Datasets object.

**Return type** *Datasets*

**dataset\_get**(*dataset\_id: str*) → *paperswithcode.models.dataset.Dataset*

Return a dataset by its ID.

**Parameters** **dataset\_id** (*str*) – ID of the dataset.

**Returns** Dataset object.

**Return type** *Dataset*

**dataset\_add**(*dataset: paperswithcode.models.dataset.DatasetCreateRequest*) → *paperswithcode.models.dataset.Dataset*

Add a dataset.

**Parameters** **dataset** (*DatasetCreateRequest*) – Dataset create request.

**Returns** Created dataset.

**Return type** *Dataset*

**dataset\_update**(*dataset\_id: str, dataset: paperswithcode.models.dataset.DatasetUpdateRequest*) → *paperswithcode.models.dataset.Dataset*

Update a dataset.

**Parameters**

- **dataset\_id** (*str*) – ID of the dataset.
- **dataset** (*DatasetUpdateRequest*) – Dataset update request.

**Returns** Updated dataset.

**Return type** *Dataset*

**dataset\_delete**(*dataset\_id: str*)

Delete a dataset.

**Parameters** **dataset\_id** (*str*) – ID of the dataset.

**dataset\_evaluation\_list**(*dataset\_id: str, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.evaluation.table.EvaluationTables*

Return a list of evaluation tables for a selected dataset.

**Parameters**

- **dataset\_id** (*str*) – ID of the dataset.
- **page** (*int*) – Desired page.

- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** EvaluationTables object.

**Return type** *EvaluationTables*

**method\_list**(*q: Optional[str] = None, name: Optional[str] = None, full\_name: Optional[str] = None, page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.method.Methods*

Return a paginated list of methods.

**Parameters**

- **q** (*str, optional*) – Search all searchable fields.
- **name** (*str, optional*) – Filter methods by part of the name.
- **full\_name** (*str, optional*) – Filter methods by part of the full name.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Methods object.

**Return type** *Methods*

**method\_get**(*method\_id*) → *paperswithcode.models.method.Method*

Return a method by it's ID.

**Parameters** **method\_id** (*str*) – ID of the method.

**Returns** Method object.

**Return type** *Method*

**evaluation\_list**(*page: int = 1, items\_per\_page: int = 50*) → *paperswithcode.models.evaluation.table.EvaluationTables*

Return a paginated list of evaluation tables.

**Parameters**

- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Evaluation table page object.

**Return type** *EvaluationTables*

**evaluation\_get**(*evaluation\_id: str*) → *paperswithcode.models.evaluation.table.EvaluationTable*

Return a evaluation table by it's ID.

**Parameters** **evaluation\_id** (*str*) – ID of the evaluation table.

**Returns** Evaluation table object.

**Return type** *EvaluationTable*

**evaluation\_create**(*evaluation: paperswithcode.models.evaluation.table.EvaluationTableCreateRequest*) → *paperswithcode.models.evaluation.table.EvaluationTable*

Create an evaluation table.

**Parameters** **evaluation** (*EvaluationTableCreateRequest*) – Evaluation table create request object.

**Returns** The new created evaluation table.

**Return type** *EvaluationTable*

**evaluation\_update**(*evaluation\_id: str, evaluation:*  
     *paperswithcode.models.evaluation.table.EvaluationTableUpdateRequest*) →  
     *paperswithcode.models.evaluation.table.EvaluationTable*

Update an evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **evaluation** (*EvaluationTableUpdateRequest*) – Evaluation table update request object.

**Returns** The updated evaluation table.

**Return type** *EvaluationTable*

**evaluation\_delete**(*evaluation\_id: str*)

Delete an evaluation table.

**Parameters** **evaluation\_id** (*str*) – ID of the evaluation table.

**evaluation\_metric\_list**(*evaluation\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
     *paperswithcode.models.evaluation.metric.Metrics*

List all metrics used in the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Metrics object.

**Return type** *Metrics*

**evaluation\_metric\_get**(*evaluation\_id: str, metric\_id: str*) →  
     *paperswithcode.models.evaluation.metric.Metric*

Get a metrics used in the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **metric\_id** (*str*) – ID of the metric.

**Returns** Requested metric.

**Return type** *Metric*

**evaluation\_metric\_add**(*evaluation\_id: str, metric:*  
     *paperswithcode.models.evaluation.metric.MetricCreateRequest*) →  
     *paperswithcode.models.evaluation.metric.Metric*

Add a metrics to the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **metric** (*MetricCreateRequest*) – Metric create request.

**Returns** Created metric.

**Return type** *Metric*

**evaluation\_metric\_update**(*evaluation\_id: str, metric\_id: str, metric:*  
*paperswithcode.models.evaluation.metric.MetricUpdateRequest*) →  
*paperswithcode.models.evaluation.metric.Metric*

Update a metrics in the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **metric\_id** (*str*) – ID of the metric.
- **metric** (*MetricCreateRequest*) – Metric update request.

**Returns** Updated metric.

**Return type** *Metric*

**evaluation\_metric\_delete**(*evaluation\_id: str, metric\_id: str*)

Delete a metrics from the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **metric\_id** (*str*) – ID of the metric.

**evaluation\_result\_list**(*evaluation\_id: str, page: int = 1, items\_per\_page: int = 50*) →  
*paperswithcode.models.evaluation.result.Results*

List all results from the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **page** (*int*) – Desired page.
- **items\_per\_page** (*int*) – Desired number of items per page. Default: 50.

**Returns** Results object.

**Return type** *Results*

**evaluation\_result\_get**(*evaluation\_id: str, result\_id: str*) →  
*paperswithcode.models.evaluation.result.Result*

Get a result from the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **result\_id** (*str*) – ID of the result.

**Returns** Requested result.

**Return type** *Result*

**evaluation\_result\_add**(*evaluation\_id: str, result:*  
*paperswithcode.models.evaluation.result.ResultCreateRequest*) →  
*paperswithcode.models.evaluation.result.Result*

Add a result to the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **result** (*ResultCreateRequest*) – Result create request.

**Returns** Created result.



**Return type** *Result*

**evaluation\_result\_update**(*evaluation\_id: str, result\_id: str, result:*  
*paperswithcode.models.evaluation.result.ResultUpdateRequest*) →  
*paperswithcode.models.evaluation.result.Result*

Update a result in the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **result\_id** (*str*) – ID of the result.
- **result** (*ResultUpdateRequest*) – Result update request.

**Returns** Updated result.

**Return type** *Result*

**evaluation\_result\_delete**(*evaluation\_id: str, result\_id: str*)

Delete a result from the evaluation table.

**Parameters**

- **evaluation\_id** (*str*) – ID of the evaluation table.
- **result\_id** (*str*) – ID of the result.



## INDICES AND TABLES

- `genindex`
- `modindex`



## PYTHON MODULE INDEX

### p

- `paperswithcode.client`, 26
- `paperswithcode.models.author`, 9
- `paperswithcode.models.conference`, 9
- `paperswithcode.models.dataset`, 13
- `paperswithcode.models.evaluation`, 16
- `paperswithcode.models.method`, 15
- `paperswithcode.models.paper`, 5
- `paperswithcode.models.paper_repo`, 8
- `paperswithcode.models.repository`, 7
- `paperswithcode.models.task`, 11



## Symbols

`__init__()` (*paperswithcode.client.PapersWithCodeClient* method), 26

## A

`abstract` (*paperswithcode.models.paper.Paper* attribute), 5

`Area` (class in *paperswithcode.models.task*), 11

`area` (*paperswithcode.models.task.TaskCreateRequest* attribute), 12

`area` (*paperswithcode.models.task.TaskUpdateRequest* attribute), 12

`area_get()` (*paperswithcode.client.PapersWithCodeClient* method), 30

`area_list()` (*paperswithcode.client.PapersWithCodeClient* method), 30

`area_task_list()` (*paperswithcode.client.PapersWithCodeClient* method), 31

`Areas` (class in *paperswithcode.models.task*), 11

`arxiv_id` (*paperswithcode.models.paper.Paper* attribute), 5

`Author` (class in *paperswithcode.models.author*), 9

`author_get()` (*paperswithcode.client.PapersWithCodeClient* method), 29

`author_list()` (*paperswithcode.client.PapersWithCodeClient* method), 28

`author_paper_list()` (*paperswithcode.client.PapersWithCodeClient* method), 29

`Authors` (class in *paperswithcode.models.author*), 9

`authors` (*paperswithcode.models.paper.Paper* attribute), 6

## B

`best_metric` (*paperswithcode.models.evaluation.Result* attribute), 18

`best_rank` (*paperswithcode.models.evaluation.Result* attribute), 17

## C

`Conference` (class in *paperswithcode.models.conference*), 9

`conference` (*paperswithcode.models.paper.Paper* attribute), 6

`conference_get()` (*paperswithcode.client.PapersWithCodeClient* method), 29

`conference_list()` (*paperswithcode.client.PapersWithCodeClient* method), 29

`conference_url_abs` (*paperswithcode.models.paper.Paper* attribute), 6

`conference_url_pdf` (*paperswithcode.models.paper.Paper* attribute), 6

`Conferences` (class in *paperswithcode.models.conference*), 10

`count` (*paperswithcode.models.author.Authors* attribute), 9

`count` (*paperswithcode.models.conference.Conferences* attribute), 10

`count` (*paperswithcode.models.conference.Proceedings* attribute), 10

`count` (*paperswithcode.models.dataset.Datasets* attribute), 14

`count` (*paperswithcode.models.evaluation.EvaluationTables* attribute), 20

`count` (*paperswithcode.models.evaluation.Metrics* attribute), 16

`count` (*paperswithcode.models.evaluation.Results* attribute), 18

`count` (*paperswithcode.models.method.Methods* attribute), 15

`count` (*paperswithcode.models.paper.Papers* attribute), 6

`count` (*paperswithcode.models.paper\_repo.PaperRepos* attribute), 8

`count` (*paperswithcode.models.repository.Repositories* attribute), 7

`count` (*paperswithcode.models.task.Areas* attribute), 11

count	(paperswithcode.models.task.Tasks attribute), 13	16	
<b>D</b>		description	(paperswithcode.models.evaluation.MetricCreateRequest attribute), 17
Dataset	(class in paperswithcode.models.dataset), 13	description	(paperswithcode.models.evaluation.MetricSyncRequest attribute), 22
dataset	(paperswithcode.models.evaluation.EvaluationTable attribute), 20	description	(paperswithcode.models.evaluation.MetricSyncResponse attribute), 24
dataset	(paperswithcode.models.evaluation.EvaluationTableCreateRequest attribute), 21	description	(paperswithcode.models.evaluation.MetricUpdateRequest attribute), 17
dataset	(paperswithcode.models.evaluation.EvaluationTableSyncRequest attribute), 23	description	(paperswithcode.models.method.Method attribute), 15
dataset	(paperswithcode.models.evaluation.EvaluationTableSyncResponse attribute), 25	description	(paperswithcode.models.repository.Repository attribute), 7
dataset	(paperswithcode.models.evaluation.EvaluationTableUpdateRequest attribute), 21	description	(paperswithcode.models.task.Task attribute), 12
dataset_add()	(paperswithcode.client.PapersWithCodeClient method), 33	description	(paperswithcode.models.task.TaskCreateRequest attribute), 12
dataset_delete()	(paperswithcode.client.PapersWithCodeClient method), 33	description	(paperswithcode.models.task.TaskUpdateRequest attribute), 12
dataset_evaluation_list()	(paperswithcode.client.PapersWithCodeClient method), 33	<b>E</b>	
dataset_get()	(paperswithcode.client.PapersWithCodeClient method), 33	evaluated_on	(paperswithcode.models.evaluation.Result attribute), 18
dataset_list()	(paperswithcode.client.PapersWithCodeClient method), 32	evaluated_on	(paperswithcode.models.evaluation.ResultCreateRequest attribute), 19
dataset_update()	(paperswithcode.client.PapersWithCodeClient method), 33	evaluated_on	(paperswithcode.models.evaluation.ResultSyncRequest attribute), 22
DatasetCreateRequest	(class in paperswithcode.models.dataset), 14	evaluated_on	(paperswithcode.models.evaluation.ResultSyncResponse attribute), 24
Datasets	(class in paperswithcode.models.dataset), 14	evaluated_on	(paperswithcode.models.evaluation.ResultUpdateRequest attribute), 20
DatasetUpdateRequest	(class in paperswithcode.models.dataset), 14	evaluation_create()	(paperswithcode.client.PapersWithCodeClient method), 34
description	(paperswithcode.models.evaluation.EvaluationTable attribute), 20	evaluation_delete()	(paperswithcode.client.PapersWithCodeClient method), 35
description	(paperswithcode.models.evaluation.EvaluationTableCreateRequest attribute), 21	evaluation_get()	(paperswithcode.client.PapersWithCodeClient method), 34
description	(paperswithcode.models.evaluation.EvaluationTableSyncRequest attribute), 23	evaluation_list()	(paperswithcode.client.PapersWithCodeClient method), 34
description	(paperswithcode.models.evaluation.EvaluationTableSyncResponse attribute), 25		
description	(paperswithcode.models.evaluation.EvaluationTableUpdateRequest attribute), 21		
description	(paperswithcode.models.evaluation.Metric attribute), 16		



<b>Index</b>	<b>45</b>
--------------	-----------

`id` (`paperswithcode.models.task.Task` attribute), 11  
`is_loss` (`paperswithcode.models.evaluation.Metric` attribute), 16  
`is_loss` (`paperswithcode.models.evaluation.MetricCreateRequest` attribute), 17  
`is_loss` (`paperswithcode.models.evaluation.MetricSyncRequest` attribute), 19  
`is_loss` (`paperswithcode.models.evaluation.MetricSyncRequest` attribute), 22  
`is_loss` (`paperswithcode.models.evaluation.MetricSyncResponse` attribute), 24  
`is_loss` (`paperswithcode.models.evaluation.MetricUpdateRequest` attribute), 17  
`is_official` (`paperswithcode.models.paper_repo.PaperRepo` attribute), 8  
`is_official` (`paperswithcode.models.repository.Repository` attribute), 7

## M

`Method` (class in `paperswithcode.models.method`), 15  
`method_get()` (`paperswithcode.client.PapersWithCodeClient` method), 34  
`method_list()` (`paperswithcode.client.PapersWithCodeClient` method), 34  
`methodology` (`paperswithcode.models.evaluation.Result` attribute), 18  
`methodology` (`paperswithcode.models.evaluation.ResultCreateRequest` attribute), 19  
`methodology` (`paperswithcode.models.evaluation.ResultSyncRequest` attribute), 22  
`methodology` (`paperswithcode.models.evaluation.ResultSyncResponse` attribute), 24  
`methodology` (`paperswithcode.models.evaluation.ResultUpdateRequest` attribute), 19  
`Methods` (class in `paperswithcode.models.method`), 15  
`Metric` (class in `paperswithcode.models.evaluation`), 16  
`metric` (`paperswithcode.models.evaluation.EvaluationTableSyncRequest` attribute), 23  
`metric` (`paperswithcode.models.evaluation.EvaluationTableSyncResponse` attribute), 25  
`MetricCreateRequest` (class in `paperswithcode.models.evaluation`), 16  
`Metrics` (class in `paperswithcode.models.evaluation`), 16  
`metrics` (`paperswithcode.models.evaluation.Result` attribute), 17  
`metrics` (`paperswithcode.models.evaluation.ResultCreateRequest` attribute), 19  
`metrics` (`paperswithcode.models.evaluation.ResultSyncRequest` attribute), 22  
`metrics` (`paperswithcode.models.evaluation.ResultSyncResponse` attribute), 24  
`metrics` (`paperswithcode.models.evaluation.ResultUpdateRequest` attribute), 17  
`MetricSyncRequest` (class in `paperswithcode.models.evaluation`), 22  
`MetricSyncResponse` (class in `paperswithcode.models.evaluation`), 24  
`MetricUpdateRequest` (class in `paperswithcode.models.evaluation`), 17  
`mirror_url` (`paperswithcode.models.evaluation.EvaluationTable` attribute), 20  
`mirror_url` (`paperswithcode.models.evaluation.EvaluationTableCreateRequest` attribute), 21  
`mirror_url` (`paperswithcode.models.evaluation.EvaluationTableSyncRequest` attribute), 23  
`mirror_url` (`paperswithcode.models.evaluation.EvaluationTableSyncResponse` attribute), 25  
`mirror_url` (`paperswithcode.models.evaluation.EvaluationTableUpdateRequest` attribute), 21  
`module`  
`paperswithcode.client`, 26  
`paperswithcode.models.author`, 9  
`paperswithcode.models.conference`, 9  
`paperswithcode.models.dataset`, 13  
`paperswithcode.models.evaluation`, 16  
`paperswithcode.models.method`, 15  
`paperswithcode.models.paper`, 5  
`paperswithcode.models.paper_repo`, 8  
`paperswithcode.models.repository`, 7  
`paperswithcode.models.task`, 11  
`month` (`paperswithcode.models.conference.Proceeding` attribute), 10

## N

`name` (`paperswithcode.models.conference.Conference` attribute), 9  
`name` (`paperswithcode.models.dataset.Dataset` attribute), 13  
`name` (`paperswithcode.models.dataset.DatasetCreateRequest` attribute), 14  
`name` (`paperswithcode.models.dataset.DatasetUpdateRequest` attribute), 14  
`name` (`paperswithcode.models.evaluation.Metric` attribute), 16  
`name` (`paperswithcode.models.evaluation.MetricCreateRequest` attribute), 17

**P**  
 name (paperswithcode.models.evaluation.MetricSyncRequest attribute), 22  
 name (paperswithcode.models.evaluation.MetricSyncResponse attribute), 24  
 name (paperswithcode.models.evaluation.MetricUpdateRequest attribute), 17  
 name (paperswithcode.models.method.Method attribute), 15  
 name (paperswithcode.models.repository.Repository attribute), 7  
 name (paperswithcode.models.task.Area attribute), 11  
 name (paperswithcode.models.task.Task attribute), 12  
 name (paperswithcode.models.task.TaskCreateRequest attribute), 12  
 name (paperswithcode.models.task.TaskUpdateRequest attribute), 12  
 next\_page (paperswithcode.models.author.Authors attribute), 9  
 next\_page (paperswithcode.models.conference.Conferences attribute), 10  
 next\_page (paperswithcode.models.conference.Proceedings attribute), 10  
 next\_page (paperswithcode.models.dataset.Datasets attribute), 14  
 next\_page (paperswithcode.models.evaluation.EvaluationTables attribute), 20  
 next\_page (paperswithcode.models.evaluation.Metrics attribute), 16  
 next\_page (paperswithcode.models.evaluation.Results attribute), 18  
 next\_page (paperswithcode.models.method.Methods attribute), 15  
 next\_page (paperswithcode.models.paper.Papers attribute), 6  
 next\_page (paperswithcode.models.paper\_repo.PaperRepos attribute), 8  
 next\_page (paperswithcode.models.repository.Repositories attribute), 7  
 next\_page (paperswithcode.models.task.Areas attribute), 11  
 next\_page (paperswithcode.models.task.Tasks attribute), 13  
 nips\_id (paperswithcode.models.paper.Paper attribute), 5  
**O**  
 owner (paperswithcode.models.repository.Repository attribute), 7  
 Paper (class in paperswithcode.models.paper), 5  
 paper (paperswithcode.models.evaluation.Result attribute), 18  
 paper (paperswithcode.models.evaluation.ResultCreateRequest attribute), 19  
 paper (paperswithcode.models.evaluation.ResultSyncRequest attribute), 22  
 paper (paperswithcode.models.evaluation.ResultSyncResponse attribute), 24  
 paper (paperswithcode.models.evaluation.ResultUpdateRequest attribute), 19  
 paper (paperswithcode.models.method.Method attribute), 15  
 paper (paperswithcode.models.paper\_repo.PaperRepo attribute), 8  
 paper\_dataset\_list() (paperswithcode.client.PapersWithCodeClient method), 26  
 paper\_get() (paperswithcode.client.PapersWithCodeClient method), 26  
 paper\_list() (paperswithcode.client.PapersWithCodeClient method), 26  
 paper\_method\_list() (paperswithcode.client.PapersWithCodeClient method), 27  
 paper\_repository\_list() (paperswithcode.client.PapersWithCodeClient method), 27  
 paper\_result\_list() (paperswithcode.client.PapersWithCodeClient method), 27  
 paper\_task\_list() (paperswithcode.client.PapersWithCodeClient method), 27  
 PaperRepo (class in paperswithcode.models.paper\_repo), 8  
 PaperRepos (class in paperswithcode.models.paper\_repo), 8  
 Papers (class in paperswithcode.models.paper), 6  
 paperswithcode.client module, 26  
 paperswithcode.models.author module, 9  
 paperswithcode.models.conference module, 9  
 paperswithcode.models.dataset module, 13  
 paperswithcode.models.evaluation module, 16  
 paperswithcode.models.method module, 15

`paperswithcode.models.paper`  
     module, 5  
`paperswithcode.models.paper_repo`  
     module, 8  
`paperswithcode.models.repository`  
     module, 7  
`paperswithcode.models.task`  
     module, 11  
`PapersWithCodeClient` (class in `paperswith-`  
     `code.client`), 26  
`parent_task` (paperswith-  
     `code.models.task.TaskCreateRequest` attribute),  
     12  
`parent_task` (paperswith-  
     `code.models.task.TaskUpdateRequest` at-  
     tribute), 12  
`previous_page` (paperswithcode.models.author.Authors  
     attribute), 9  
`previous_page` (paperswith-  
     `code.models.conference.Conferences` attribute),  
     10  
`previous_page` (paperswith-  
     `code.models.conference.Proceedings` attribute),  
     10  
`previous_page` (paperswith-  
     `code.models.dataset.Datasets` attribute),  
     14  
`previous_page` (paperswith-  
     `code.models.evaluation.EvaluationTables`  
     attribute), 20  
`previous_page` (paperswith-  
     `code.models.evaluation.Metrics` attribute),  
     16  
`previous_page` (paperswith-  
     `code.models.evaluation.Results` attribute),  
     18  
`previous_page` (paperswith-  
     `code.models.method.Methods` attribute),  
     15  
`previous_page` (paperswithcode.models.paper.Papers  
     attribute), 6  
`previous_page` (paperswith-  
     `code.models.paper_repo.PaperRepos` at-  
     tribute), 8  
`previous_page` (paperswith-  
     `code.models.repository.Repositories` attribute),  
     7  
`previous_page` (paperswithcode.models.task.Areas at-  
     tribute), 11  
`previous_page` (paperswithcode.models.task.Tasks at-  
     tribute), 13  
`Proceeding` (class in `paperswith-`  
     `code.models.conference`), 10  
`proceeding` (paperswithcode.models.paper.Paper  
     attribute), 6  
`proceeding_get()` (paperswith-  
     `code.client.PapersWithCodeClient` method),  
     30  
`proceeding_list()` (paperswith-  
     `code.client.PapersWithCodeClient` method),  
     29  
`proceeding_paper_list()` (paperswith-  
     `code.client.PapersWithCodeClient` method),  
     30  
`Proceedings` (class in `paperswith-`  
     `code.models.conference`), 10  
`published` (paperswithcode.models.paper.Paper at-  
     tribute), 6

## R

`Repositories` (class in `paperswith-`  
     `code.models.repository`), 7  
`Repository` (class in `paperswith-`  
     `code.models.repository`), 7  
`repository` (paperswith-  
     `code.models.paper_repo.PaperRepo` attribute),  
     8  
`repository_get()` (paperswith-  
     `code.client.PapersWithCodeClient` method),  
     28  
`repository_list()` (paperswith-  
     `code.client.PapersWithCodeClient` method),  
     27  
`repository_owner_list()` (paperswith-  
     `code.client.PapersWithCodeClient` method),  
     28  
`repository_paper_list()` (paperswith-  
     `code.client.PapersWithCodeClient` method),  
     28  
`Result` (class in `paperswithcode.models.evaluation`), 17  
`ResultCreateRequest` (class in `paperswith-`  
     `code.models.evaluation`), 18  
`Results` (class in `paperswithcode.models.evaluation`), 18  
`results` (paperswithcode.models.author.Authors at-  
     tribute), 9  
`results` (paperswithcode.models.conference.Conferences  
     attribute), 10  
`results` (paperswithcode.models.conference.Proceedings  
     attribute), 11  
`results` (paperswithcode.models.dataset.Datasets  
     attribute), 14  
`results` (paperswithcode.models.evaluation.EvaluationTables  
     attribute), 21  
`results` (paperswithcode.models.evaluation.EvaluationTableSyncRequest  
     attribute), 23  
`results` (paperswithcode.models.evaluation.EvaluationTableSyncResponse  
     attribute), 25

[results](#) ([paperswithcode.models.evaluation.Metrics](#) attribute), 16  
[results](#) ([paperswithcode.models.evaluation.Results](#) attribute), 18  
[results](#) ([paperswithcode.models.method.Methods](#) attribute), 15  
[results](#) ([paperswithcode.models.paper.Papers](#) attribute), 6  
[results](#) ([paperswithcode.models.paper\\_repo.PaperRepos](#) attribute), 8  
[results](#) ([paperswithcode.models.repository.Repositories](#) attribute), 7  
[results](#) ([paperswithcode.models.task.Areas](#) attribute), 11  
[results](#) ([paperswithcode.models.task.Tasks](#) attribute), 13  
[ResultSyncRequest](#) (class in [paperswithcode.models.evaluation](#)), 22  
[ResultSyncResponse](#) (class in [paperswithcode.models.evaluation](#)), 23  
[ResultUpdateRequest](#) (class in [paperswithcode.models.evaluation](#)), 19

## S

[search\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 26  
[stars](#) ([paperswithcode.models.repository.Repository](#) attribute), 7

## T

[Task](#) (class in [paperswithcode.models.task](#)), 11  
[task](#) ([paperswithcode.models.evaluation.EvaluationTable](#) attribute), 20  
[task](#) ([paperswithcode.models.evaluation.EvaluationTableCreateRequest](#) attribute), 21  
[task](#) ([paperswithcode.models.evaluation.EvaluationTableSyncRequest](#) attribute), 23  
[task](#) ([paperswithcode.models.evaluation.EvaluationTableSyncResponse](#) attribute), 25  
[task](#) ([paperswithcode.models.evaluation.EvaluationTableUpdateRequest](#) attribute), 21  
[task\\_add\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 31  
[task\\_child\\_list\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 32  
[task\\_delete\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 31  
[task\\_evaluation\\_list\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 32  
[task\\_get\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 31  
[task\\_list\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 31  
[task\\_paper\\_list\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 32  
[task\\_parent\\_list\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 32  
[task\\_update\(\)](#) ([paperswithcode.client.PapersWithCodeClient](#) method), 31  
[TaskCreateRequest](#) (class in [paperswithcode.models.task](#)), 12  
[Tasks](#) (class in [paperswithcode.models.task](#)), 13  
[TaskUpdateRequest](#) (class in [paperswithcode.models.task](#)), 12  
[title](#) ([paperswithcode.models.paper.Paper](#) attribute), 5

## U

[url](#) ([paperswithcode.models.dataset.Dataset](#) attribute), 13  
[url](#) ([paperswithcode.models.dataset.DatasetCreateRequest](#) attribute), 14  
[url](#) ([paperswithcode.models.dataset.DatasetUpdateRequest](#) attribute), 14  
[url](#) ([paperswithcode.models.repository.Repository](#) attribute), 7  
[url\\_abs](#) ([paperswithcode.models.paper.Paper](#) attribute), 5  
[url\\_pdf](#) ([paperswithcode.models.paper.Paper](#) attribute), 5  
[uses\\_additional\\_data](#) ([paperswithcode.models.evaluation.Result](#) attribute), 18  
[uses\\_additional\\_data](#) ([paperswithcode.models.evaluation.ResultCreateRequest](#) attribute), 19  
[uses\\_additional\\_data](#) ([paperswithcode.models.evaluation.ResultSyncRequest](#) attribute), 22  
[uses\\_additional\\_data](#) ([paperswithcode.models.evaluation.ResultSyncResponse](#) attribute), 24  
[uses\\_additional\\_data](#) ([paperswithcode.models.evaluation.ResultUpdateRequest](#) attribute), 19

## Y

[year](#) ([paperswithcode.models.conference.Proceeding](#) attribute), 10