model-index

Release 0.1.8

Robert Stojnic <hello@paperswithcode.com>

Mar 03, 2021

GETTING STARTED

The model-index library has two goals:

- Make it easy to maintain a source-of-truth index of ML model metadata
- Enable the community to browse this model metadata on Papers with Code

You can use this library locally or choose to upload the metadata to Papers with Code to have your library featured on the website.

CHAPTER

ONE

INSTALLATION

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

```
$ pip install model-index
```

CHAPTER

TWO

HOW IT WORKS

There is a root file for the model index: `model-index.yml` that contains (or links to) all the metadata in a consistent format. An example with a single model:

```
Models:
  - Name: Inception v3
    Metadata:
      FLOPs: 5731284192
      Parameters: 23834568
      Epochs: 90
      Batch Size: 32
      Training Data: ImageNet
      Training Techniques:
        - RMSProp
        - Weight Decay
        - Gradient Clipping
        - Label Smoothing
      Training Resources: 8x V100 GPUs
      Training Time: 24 hours
      Architecture:
        - Auxiliary Classifier
        - Inception-v3 Module
    Results:
      - Task: Image Classification
        Dataset: ImageNet
        Metrics:
          Top 1 Accuracy: 74.67%
          Top 5 Accuracy: 92.1%
    Paper: https://arxiv.org/abs/1512.00567v3
    Code: https://github.com/rwightman/pytorch-image-models/blob/timm/models/
↪inception_v3.py#L442
    Weights: https://download.pytorch.org/models/inception_v3_google-1a9a5a14.pth
    README: docs/inception-v3-readme.md
```

Every field except for `Name` is **optional**.

The fields you can see above are **common fields** that are automatically recognized by Papers with Code and enable comparison across different models. You can also add any number of **custom fields** that are specific to your model or library. For example:

```
Models:
  - Name: My new model
    Metadata:
      Training Time: 24 hours
      my parameter: 120
```

(continues on next page)

(continued from previous page)

```
      my parameter2:
        sub parameter1: value 1
        sub parameter2: value 2
```

So you can mix-and-match from our set of common field and any other field you want to add.

We recommend putting the `model-index.yml` file in the root of your repository (so that relative links such as `docs/inception-v3-readme.md` are easier to write), but you can also put it anywhere else in the repository (e.g. int your `docs/` or `models/` folder).

2.1 Storing metadata in markdown files

Metadata can also be directly stored in model's README file. For example in this `docs/rexnet.md` file:

```
<!--
Type: model-index
Name: RexNet
Metadata:
  Epochs: 400
  Batch Size: 512
Paper: https://arxiv.org/abs/2007.00992v1
-->

# Summary

Rank Expansion Networks (ReXNets) follow a set of new design
principles for designing bottlenecks in image classification models.

## Usage

import timm
m = timm.create_model('rexnet_100', pretrained=True)
m.eval()
```

In this case, you just need to include this markdown file into the global `model-index.yml` file:

```
Models:
  - docs/rexnet.md
```

CHAPTER

THREE

EXAMPLES

You can find examples in our GitHub repository.

CHAPTER

FOUR

IMPORTING METADATA

It might not be convenient to maintain a huge `model-index.yml` file in your repository. So `model-index` make it easy to split data into smaller file.

Data can be imported from other YAML files, JSON files and from Markdown files.

4.1 Importing fields

For Models, Metadata and Results you can provide a filename instead of actual values. For example:

```yaml
Models:
  - Name: Inception v3
    Metadata: models/metadata/inception_v3.yml
    Results: models/metadata/inception_v3_results.json
```

Both YAML and JSON are supported. The content of the file needs to conform to the expected structure specified for Models, Metadata and Results.

So in this case, `inception_v3.yml` could look like this:

```yaml
Batch Size: 120
Epochs: 100
Dropout: 0.1
```

and `inception_v3_results.json` could look like this:

```json
[
  {
    "Task": "Image Classification",
    "Dataset": "ImageNet ReaL",
    "Metrics": {
      "Accuracy": "90.2%"
    }
  }
]
```

4.2 Importing a whole YAML file

You can split the model index into smaller files (e.g. containing a single collection of related models) and then import them into the main `model-index.yml` file:

```
Import:
  - models/metadata/inception_models.yml
```

Here the `inception_models.yml` file has to be in ModelIndex format (i.e. the same as the root `model-index.yml` file). For example:

```
Collection:
  - Name: Inception models
Models:
  - Name: Inception v3 (original)
    In Collection: Inception models
    Metadata:
      Epochs: 90
  - Name: Inception v3 (retrained)
    In Collection: Inception models
    Metadata:
      Epochs: 120
```

4.3 Importing a whole JSON file

The imported file can also be in JSON format, as long as it has the same structure as the YAML:

```
Import:
  - models/metadata/mnasnet_100.json
  - models/metadata/mnasnet_200.json
```

So one of these JSON files could look like:

```
{
  "Epochs": 120,
  "Dropout": 0.2
}
```

4.4 Importing a whole Markdown file

```
Import:
  - docs/rexnet.md
```

Instead of specifying the `README` field of the model, you can add the model metadata directly into the markdown file.

Include the metadata in YAML format in a comment, with a `Type: model-index` key to indicate that it should be parsed by the `model-index` library. So, the `rexnet.md` could look like this:

```
<!--
Type: model-index
Models:
  - Name: RexNet
```

(continues on next page)

(continued from previous page)

```
    Metadata: ../models/metadata/rexnet.json
    Results: ../models/metadata/rexnet_results.json
-->
# Summary

Rank Expansion Networks (ReXNets) follow a set of new design principles for
designing bottlenecks in image classification models.

## How to use this model

....
```

4.5 Using wildcards to import multiple files

Specify wildcards to import multiple files:

```
Import:
  - docs/*.md
  - models/metadata/*.json
```

CHAPTER

FIVE

CHECKING THE INDEX

To ensure the model index doesn't have any errors in its metadata you can run an automated check on the whole index, or an individual file:

via the CLI:

```
$ mi check  # Check the entire index (model-index.yml in current directory)
$ mi check models/metadata/inception_v3.yml    # check one file
```

or programatically:

```
import modelindex

# check entire index
mi = modelindex.load("<path to model-index.yml>")
mi.check()

# check one file
mi = modelindex.load("models/metadata/inception_v3.yml")
mi.check()
```

CHAPTER

SIX

PUBLISHING ON PAPERS WITH CODE

To feature your library on Papers with Code, get in touch with `hello@paperswithcode.com` and the model index of your library will be automatically included into Papers with Code.

CHAPTER

SEVEN

MODELINDEX

The `model-index.yml` is the root file for the model index. It contains or links to all the models available in the index.

ModelIndex is a dict with three possible fields:

- `Models` - a list of Model dicts, or a list of files to import Model dicts from.
- `Collections` - a list of Collections dicts, or a list of files to import Collection dicts from.
- `Import` - a list of files to import additional ModelIndex files from (e.g. if you want to split the `model-index.yml` file into multiple parts)

All field names are case-insensitive.

7.1 A full example

An example of the full ModelIndex dict:

```
Collections:
  - Name: Mask R-CNN
    Metadata:
      Training Data: COCO
      Training Techniques:
        - SGD with Momentum
      Training Resources: 8x V100 GPUs
      Architecture:
        - RoI Align
        - RPN
    Paper: https://arxiv.org/abs/1703.06870v3
    README: docs/maskrcnn.md

Models:
  - Name: Mask R-CNN (R101-C4, 3x)
    In Collection: Mask R-CNN
    Metadata:
      inference time (s/im): 0.145
      train time (s/iter): 0.652
      Training Memory (GB): 6.3
    Results:
      - Task: Object Detection
        Dataset: COCO minival
        Metrics:
          box AP: 42.6
```

(continues on next page)

(continued from previous page)

```
      - Task: Instance Segmentation
        Dataset: COCO minival
        Metrics:
          mask AP: 36.7
    Weights: https://dl.fbaipublicfiles.com/detectron2/COCO-InstanceSegmentation/mask_
↪rcnn_R_101_C4_3x/138363239/model_final_a2914c.pkl

  - Name: Mask R-CNN (R50-C4, 3x)
    In Collection: Mask R-CNN
    Metadata:
      inference time (s/im): 0.111
      train time (s/iter): 0.575
      Training Memory (GB): 5.2
    Results:
      - Task: Object Detection
        Dataset: COCO minival
        Metrics:
          box AP: 39.8
      - Task: Instance Segmentation
        Dataset: COCO minival
        Metrics:
          mask AP: 34.4
    Weights: https://dl.fbaipublicfiles.com/detectron2/COCO-InstanceSegmentation/mask_
↪rcnn_R_50_C4_3x/137849525/model_final_4ce675.pkl

Import:
  - additional_models.yml
```

Where `additional_models.yml` is a file in the same format and that will be merged in.

CHAPTER

EIGHT

MODEL

A model represents all the information about a Machine Learning model.

Model is represented by a dict with these common fields:

- `Name` - the only required field, with the name of the model.
- `Metadata` - a dict of *Metadata*, or a link to a file (yml/json) containing it.
- `Results` - a list of *Result* dicts.
- `Paper` - URL to the paper, or a dict with paper title, url and
- `Code` - link to code snippet snippet
- `Weights` - link to download the pretrained weights
- `Config` - link to the config file used for training
- `README` - the content of, or a link to the README.md file for the model
- `Image` - path or URL to an image for this model
- `In Collection`- name of the Collection to which this model belongs.

The fields above will be automatically recognized by model-index, but you can also add any number of custom fields to it.

All field names are case-insensitive.

8.1 Metadata

Metadata is a dict of common and custom metadata. The common fields are:

- `FLOPs` - The number of FLOPs of the model
- `Parameters` - The total number of parameters of the model
- `Epochs` - Number of training epochs
- `Batch Size` - Input batch size
- `Training Data` - Names of dataset on which the models is trained on
- `Training Techniques` - A list of training techniques (for the full list see Methods on Papers with Code)
- `Training Resources` - The hardware used for training
- `Training Time` - How many hours or days it takes to train.

- `Architecture` - A list of architectural features of the model (for the full list see Methods on Papers with Code)

You can also add any other custom field that is specific to your model.

8.2 Result

Result is a dict capturing the evaluation results of the model. It has these fields:

- `Task` - Name of the task (for full see Benchmarks on Papers with Code)
- `Dataset` - Name of the dataset (for full see Datasets on Papers with Code)
- `Metrics` - a list of dictionaries with metrics. For relevant metrics consult the see Benchmarks on Papers with Code.

8.3 A full example

An example of the full model dict is shown below:

```
Name: Inception v3
Metadata:
  FLOPs: 5731284192
  Parameters: 23834568
  Epochs: 90
  Batch Size: 32
  Training Data: ImageNet
  Training Techniques:
    - RMSProp
    - Weight Decay
    - Gradient Clipping
    - Label Smoothing
  Training Resources: 8x V100 GPUs
  Training Time: 24 hours
  Architecture:
    - Auxiliary Classifier
    - Inception-v3 Module
Results:
  - Task: Image Classification
    Dataset: ImageNet
    Metrics:
      Top 1 Accuracy: 74.67%
      Top 5 Accuracy: 92.1%
Paper: https://arxiv.org/abs/1512.00567v3
Code: https://github.com/rwightman/pytorch-image-models/blob/timm/models/inception_v3.
↪py#L442
Weights: https://download.pytorch.org/models/inception_v3_google-1a9a5a14.pth
Config: configs/inception-v3-config.json
README: docs/inception-v3-readme.md
```

CHAPTER

NINE

COLLECTION

To keep related model together, you can create Collections. The metadata format for collections is the same as for models and all member models **inherit** all the metadata and can override/add to it.

The fields of a collection are exactly the same as fields of a Model - see the description there.

9.1 A full example

```
Collections:
  - Name: Mask R-CNN
    Metadata:
      Training Data: COCO
      Training Techniques:
        - SGD with Momentum
      Training Resources: 8x V100 GPUs
      Architecture:
        - RoI Align
        - RPN
    Paper: https://arxiv.org/abs/1703.06870v3
    README: docs/maskrcnn.md

Models:
  - Name: Mask R-CNN (R101-C4, 3x)
    In Collection: Mask R-CNN
    Metadata:
      inference time (s/im): 0.145
      train time (s/iter): 0.652
      Training Memory (GB): 6.3
    Results:
      - Task: Object Detection
        Dataset: COCO minival
        Metrics:
          box AP: 42.6
      - Task: Instance Segmentation
        Dataset: COCO minival
        Metrics:
          mask AP: 36.7
    Weights: https://dl.fbaipublicfiles.com/detectron2/COCO-InstanceSegmentation/mask_
↪rcnn_R_101_C4_3x/138363239/model_final_a2914c.pkl

  - Name: Mask R-CNN (R50-C4, 3x)
    In Collection: Mask R-CNN
    Metadata:
```

(continues on next page)

(continued from previous page)

```
      inference time (s/im): 0.111
      train time (s/iter): 0.575
      Training Memory (GB): 5.2
    Results:
      - Task: Object Detection
        Dataset: COCO minival
        Metrics:
          box AP: 39.8
      - Task: Instance Segmentation
        Dataset: COCO minival
        Metrics:
          mask AP: 34.4
    Weights: https://dl.fbaipublicfiles.com/detectron2/COCO-InstanceSegmentation/mask_
↪rcnn_R_50_C4_3x/137849525/model_final_4ce675.pkl
```

In this example we have two variants of the `Mask R-CNN` model, one with a ResNet-50 backbone and one with a ResNet-101 backbone. These models belong together as variants of the Mask R-CNN, so we link them via a `Mask R-CNN` model collection.

CHAPTER

TEN

MODELINDEX CLASS REFERENCE

10.1 ModelIndex Class

class modelindex.models.ModelIndex.**ModelIndex**(*data: Optional[dict] = None*, *filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*, *is_root: bool = False*)

ModelIndex is the root object for the whole model index.

Parameters

- **data** (*dict*) – The root model index as a dictionary
- **filepath** (*str*) – The path from which it was loaded
- **_path_to_readme** (*str*) – The path to the readme file (if loaded from there)
- **is_root** (*bool*) – If this is the root ModelIndex instance for the whole project

__init__(*data: Optional[dict] = None*, *filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*, *is_root: bool = False*)

Parameters

- **data** (*dict*) – The root model index as a dictionary
- **filepath** (*str*) – The path from which it was loaded
- **_path_to_readme** (*str*) – The path to the readme file (if loaded from there)
- **is_root** (*bool*) – If this is the root ModelIndex instance for the whole project

static from_dict(*d: dict*, *filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*, *is_root: bool = False*)

Construct a ModelIndex from a dictionary

Parameters

- **data** (*dict*) – The root model index as a dictionary
- **filepath** (*str*) – The path from which it was loaded
- **_path_to_readme** (*str*) – Path to the README.md file if loaded from there
- **is_root** (*str*) – If this is the root ModelIndex for the whole project

property models

Get the list of models in the ModelIndex.

property collections

Get the list of collections in the ModelIndex.

10.2 Model Class

class modelindex.models.Model.**Model**(*name: Optional[str] = None*, *metadata: Optional[Union[Dict, modelindex.models.Metadata.Metadata, str]] = None*, *results: Optional[Union[List, modelindex.models.ResultList.ResultList, modelindex.models.Result.Result, Dict, str]] = None*, *paper: Optional[Union[str, Dict]] = None*, *code: Optional[str] = None*, *weights: Optional[str] = None*, *config: Optional[str] = None*, *readme: Optional[str] = None*, *in_collection: Optional[Union[str, List[str]]] = None*, *image: Optional[str] = None*, *_filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*, ***kwargs*)

Model represents the ML model.

Parameters

- **name** (*str*) – Name of the model
- **metadata** (Metadata, *dict*, *str*) – Metadata object, metadata dict or a filepath to the metadata file
- **results** (ResultList, Result, *list*, *dict*, *str*) – ResultList, a single Results, a list of result dicts, a single result dict, or a filepath to the result file
- **paper** (*str*, *dict*) – URL to the paper, or a structured dict with paper metadata (title, url)
- **code** (*str*) – URL to the code snippet
- **weights** (*str*) – URL to the pretrained weights
- **config** (*str*) – URL to the config file
- **readme** (*str*) – path to the README file for the model
- **in_collection** (*str*, *List*) – name of the collection to which the model belongs to
- **image** (*str*) – path or URL to an image for the model
- **_filepath** – The file path to where the data was loaded from
- **_path_to_readme** – Path to the markdown readme file if data is coming from there
- ****kwargs** – Any other custom fields

__init__(*name: Optional[str] = None*, *metadata: Optional[Union[Dict, modelindex.models.Metadata.Metadata, str]] = None*, *results: Optional[Union[List, modelindex.models.ResultList.ResultList, modelindex.models.Result.Result, Dict, str]] = None*, *paper: Optional[Union[str, Dict]] = None*, *code: Optional[str] = None*, *weights: Optional[str] = None*, *config: Optional[str] = None*, *readme: Optional[str] = None*, *in_collection: Optional[Union[str, List[str]]] = None*, *image: Optional[str] = None*, *_filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*, ***kwargs*)

Parameters

- **name** (*str*) – Name of the model
- **metadata** (Metadata, *dict*, *str*) – Metadata object, metadata dict or a filepath to the metadata file

- **results** (*ResultList, Result, list, dict, str*) – ResultList, a single Results, a list of result dicts, a single result dict, or a filepath to the result file
- **paper** (*str, dict*) – URL to the paper, or a structured dict with paper metadata (title, url)
- **code** (*str*) – URL to the code snippet
- **weights** (*str*) – URL to the pretrained weights
- **config** (*str*) – URL to the config file
- **readme** (*str*) – path to the README file for the model
- **in_collection** (*str, List*) – name of the collection to which the model belongs to
- **image** (*str*) – path or URL to an image for the model
- **_filepath** – The file path to where the data was loaded from
- **_path_to_readme** – Path to the markdown readme file if data is coming from there
- ****kwargs** – Any other custom fields

classmethod from_dict(*d: Dict*, *_filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*)

Create a Model from a dictionary.

Parameters

- **d** (*dict*) – dictionary containing models data
- **_filepath** (*str*) – The file path to where the data was loaded from
- **_path_to_readme** (*str*) – Path to the README file if metadata was extracted from a README

static from_file(*filepath: Optional[str] = None*, *parent_filepath: Optional[str] = None*)

Load a Model from a file.

Parameters

- **filepath** (*str*) – File from which to load the model
- **parent_filepath** (*str*) – Parent filename (if file is imported from another file)

readme_content()

Get the content of the README file (instead of just the path as returned by .readme())

property full_model

Get the model with all the inherited properties from the collection (read-only property)

property name

Get the model name

property paper

Get the model paper

property code

Get the URL to code

property weights

Get the URL to weights

property config

Get the URL to the config file

property readme
Get the path to the model README

property in_collection
Get the name of the collection of which this model is part of.

property image
Get the path or URL to the image for the model

property metadata
Get the metadata as a Metadata object

property results
Get the results as a Result object

10.3 Result Class

class modelindex.models.Result.**Result**(*task: str*, *dataset: str*, *metrics: Dict*, *_filepath: Optional[str] = None*)
Result represents a model result on a particular benchmark.

Parameters

- **task** (*str*) – The name of the ML task
- **dataset** (*str*) – The name of the dataset
- **metrics** (*dict*) – A dictionary of metrics
- **_filepath** (*str*) – Path to the file where the data was loaded from

__init__(*task: str*, *dataset: str*, *metrics: Dict*, *_filepath: Optional[str] = None*)

Parameters

- **task** (*str*) – The name of the ML task
- **dataset** (*str*) – The name of the dataset
- **metrics** (*dict*) – A dictionary of metrics
- **_filepath** (*str*) – Path to the file where the data was loaded from

static from_dict(*d: Dict*, *_filepath: Optional[str] = None*)
Create a Result from a dict.

Parameters

- **d** (*dict*) – dictionary containing result data
- **_filepath** (*str*) – The file path to where the data was loaded from

static from_file(*filepath: Optional[str] = None*, *parent_filepath: Optional[str] = None*)
Load a Result from a file.

Parameters

- **filepath** (*str*) – File from which to load the result
- **parent_filepath** (*str*) – Parent filename (if file is imported from another file)

property dataset
Get the dataset name

property task
Get the ML task name

property metrics
Get the dictionary of metrics

10.4 Metadata Class

class modelindex.models.Metadata.**Metadata**(*flops: Union[str, int] = None, parameters: Union[str, int] = None, epochs: Union[str, int] = None, batch_size: Union[str, int] = None, training_data: [<class 'str'>, typing.List] = None, training_techniques: [<class 'str'>, typing.List] = None, training_resources: str = None, architecture: [<class 'str'>, typing.List] = None, _filepath: str = None, **kwargs*)

Metadata for a model.

Parameters

- **flops** (*str, int*) – number of FLOPs
- **parameters** (*str, int*) – total number of parameters for the model
- **epochs** (*str, int*) – how many epochs the model was trained
- **batch_size** (*str, int*) – batch size for the model
- **training_data** (*str, list*) – one or a list of datasets used in training
- **training_techniques** (*str, list*) – one or a list of training techniques
- **training_resources** (*str*) – hardware used to train
- **architecture** (*str, List*) – one or a list of architectures used in the model
- **_filepath** (*str*) – path to the file where the data is coming from
- ****kwargs** – any other custom metadata

__init__(*flops: Union[str, int] = None, parameters: Union[str, int] = None, epochs: Union[str, int] = None, batch_size: Union[str, int] = None, training_data: [<class 'str'>, typing.List] = None, training_techniques: [<class 'str'>, typing.List] = None, training_resources: str = None, architecture: [<class 'str'>, typing.List] = None, _filepath: str = None, **kwargs*)

Parameters

- **flops** (*str, int*) – number of FLOPs
- **parameters** (*str, int*) – total number of parameters for the model
- **epochs** (*str, int*) – how many epochs the model was trained
- **batch_size** (*str, int*) – batch size for the model
- **training_data** (*str, list*) – one or a list of datasets used in training
- **training_techniques** (*str, list*) – one or a list of training techniques
- **training_resources** (*str*) – hardware used to train
- **architecture** (*str, List*) – one or a list of architectures used in the model
- **_filepath** (*str*) – path to the file where the data is coming from

- ****kwargs** – any other custom metadata

static from_dict(*d: Dict*, *_filepath: Optional[str] = None*)
Construct Metadata from a dict.

Parameters

- **d** (*dict*) – A dictionary of values
- **_filepath** (*str*) – path to the file where the data is coming from

static from_file(*filepath: Optional[str] = None*, *parent_filepath: Optional[str] = None*)
Load a Metadata from a file.

Parameters

- **filepath** (*str*) – File from which to load the metadata
- **parent_filepath** (*str*) – Parent filename (if file is imported from another file)

property flops
Get the FLOPs

property parameters
Get number of parameters

property epochs
Get epochs

property batch_size
Get batch size

property training_data
Get training data used

property training_techniques
Get techniques used

property training_resources
Get training resources used

property architecture
Get the architecture(s) used.

10.5 Collection Class

class modelindex.models.Collection.**Collection**(*name: Optional[str] = None*, *metadata: Optional[Union[Dict, modelindex.models.Metadata.Metadata, str]] = None*, *results: Optional[Union[List, modelindex.models.ResultList.ResultList, modelindex.models.Result.Result, Dict, str]] = None*, *paper: Optional[Union[str, Dict]] = None*, *code: Optional[str] = None*, *weights: Optional[str] = None*, *config: Optional[str] = None*, *readme: Optional[str] = None*, *in_collection: Optional[Union[str, List[str]]] = None*, *image: Optional[str] = None*, *_filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*, ***kwargs*)

A collection of models with common characteristics.

Parameters

- **name** (*str*) – Name of the model
- **metadata** (*Metadata, dict, str*) – Metadata object, metadata dict or a filepath to the metadata file
- **results** (*ResultList, Result, list, dict, str*) – ResultList, a single Results, a list of result dicts, a single result dict, or a filepath to the result file
- **paper** (*str, dict*) – URL to the paper, or a structured dict with paper metadata (title, url)
- **code** (*str*) – URL to the code snippet
- **weights** (*str*) – URL to the pretrained weights
- **config** (*str*) – URL to the config file
- **readme** (*str*) – path to the README file for the model
- **in_collection** (*str, List*) – name of the collection to which the model belongs to
- **image** (*str*) – path or URL to an image for the model
- **_filepath** – The file path to where the data was loaded from
- **_path_to_readme** – Path to the markdown readme file if data is coming from there
- ****kwargs** – Any other custom fields

static from_file(*filepath: Optional[str] = None*, *parent_filepath: Optional[str] = None*)

Load a Collection from a file.

Parameters

- **filepath** (*str*) – File from which to load the collection
- **parent_filepath** (*str*) – Parent filename (if file is imported from another file)

10.6 ResultList Class

class modelindex.models.ResultList.**ResultList**(*results: Optional[Union[List[Union[Dict, modelindex.models.Result.Result, str]], modelindex.models.Result.Result, Dict]] = None*, *_filepath: Optional[str] = None*)

ResultList is a list of Result objects.

Parameters

- **results** (*list, Result, dict*) – Either a list of results, a single Result object or a dict representing a result
- **_filepath** (*str*) – path to the file where the data is coming from

__init__(*results: Optional[Union[List[Union[Dict, modelindex.models.Result.Result, str]], modelindex.models.Result.Result, Dict]] = None*, *_filepath: Optional[str] = None*)

Parameters

- **results** (*list, Result, dict*) – Either a list of results, a single Result object or a dict representing a result
- **_filepath** (*str*) – path to the file where the data is coming from

static from_file(*filepath: Optional[str] = None*, *parent_filepath: Optional[str] = None*)

Load a ResultList from a file.

Parameters

- **filepath** (*str*) – File from which to load the result list
- **parent_filepath** (*str*) – Parent filename (if file is imported from another file)

10.7 ModelList Class

class modelindex.models.ModelList.**ModelList**(*models: Optional[Union[List[Union[modelindex.models.Model.Model, Dict, str]], modelindex.models.Model.Model, Dict]] = None*, *_filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*)

ModeList is a list of Models.

Parameters

- **models** (*list, Model, dict*) – Either a list of models, and individual model or a dict representing a model
- **_filepath** (*str*) – The path of the file from which the list is initialized
- **_path_to_readme** (*str*) – Path to README if loaded from there

__init__(*models: Optional[Union[List[Union[modelindex.models.Model.Model, Dict, str]], modelindex.models.Model.Model, Dict]] = None*, *_filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*)

Parameters

- **models** (*list, Model, dict*) – Either a list of models, and individual model or a dict representing a model

- **_filepath** (*str*) – The path of the file from which the list is initialized
- **_path_to_readme** (*str*) – Path to README if loaded from there

property models
Get the list of models.

add(*model: Union[modelindex.models.Model.Model, Dict]*, *_filepath: Optional[str] = None*)
Add a model to the list.

Parameters

- **model** (*Model, dict*) – Either a Model or a dict representing a model
- **_filepath** (*str*) – The path from which it was loaded

static from_file(*filepath: Optional[str] = None*, *parent_filepath: Optional[str] = None*)
Load a ModelList from a file.

Parameters

- **filepath** (*str*) – File from which to load the list of models.
- **parent_filepath** (*str*) – Parent filename (if file is imported from another file)

10.8 CollectionList Class

class modelindex.models.CollectionList.**CollectionList**(*collections: Optional[Union[List[Union[modelindex.models.Collectio Dict]], modelindex.models.Collection.Collection, Dict]] = None*, *_filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*)
A list of Collection objects.

Parameters

- **collections** (*list, Collection, dict*) – Either a list of Collection objects, a single Collection object or a dict representing a Collection
- **_filepath** (*str*) – The file path to where the data was loaded from
- **_path_to_readme** (*str*) – Path to README if it was loaded from there

__init__(*collections: Optional[Union[List[Union[modelindex.models.Collection.Collection, Dict]], modelindex.models.Collection.Collection, Dict]] = None*, *_filepath: Optional[str] = None*, *_path_to_readme: Optional[str] = None*)

Parameters

- **collections** (*list, Collection, dict*) – Either a list of Collection objects, a single Collection object or a dict representing a Collection
- **_filepath** (*str*) – The file path to where the data was loaded from
- **_path_to_readme** (*str*) – Path to README if it was loaded from there

property collections
Get the list of Collection objects

add(*col: Union[*modelindex.models.Collection.Collection*, Dict]*, *_filepath: Optional[str] = None*)
Add a Collection to the list.

Parameters

- **col** (*Collection, dict*) – Either a Collection or a dict representing a collection
- **_filepath** (*str*) – The path from which it was loaded

static from_file(*filepath: Optional[str] = None*, *parent_filepath: Optional[str] = None*)
Load a Collection from a file.

Parameters

- **filepath** (*str*) – File from which to load the collection
- **parent_filepath** (*str*) – Parent filename (if file is imported from another file)

PYTHON MODULE INDEX

m

INDEX

Symbols

A

B

C

D

E

F