

Elliot

Release Elliot

**Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara,
Daniele Malitesta, Felice Antonio Merra, Claudio Pomo,
Tommaso Di Noia**

Mar 01, 2021

1	elliott package	5
1.1	Subpackages	5
1.2	Submodules	93
1.3	elliott.run module	93
1.4	Module contents	93
	 Python Module Index	 95
	 Index	 101

#ELLIOT

«Every hacker has her fixation. You hack people, I hack time.»

![PyPI - Python Version](https://img.shields.io/pypi/pyversions/scikit-daisy)
![Version](https://img.shields.io/badge/version-v1.1.2-orange)](https://github.com/sisinflab/elliott)
![GitHub repo size](https://img.shields.io/github/repo-size/sisinflab/elliott)
![GitHub](https://img.shields.io/github/license/sisinflab/elliott)

Elliot is a comprehensive recommendation framework that analyzes the recommendation problem from the researcher's perspective. It conducts a whole experiment, from dataset loading to results gathering. The core idea is to feed the system with a simple and straightforward configuration file that drives the framework through the experimental setting choices. Elliot untangles the complexity of combining splitting strategies, hyperparameter model optimization, model training, and the generation of reports of the experimental results.

The framework loads, filters, and splits the data considering a vast set of strategies (splitting methods and filtering approaches, from temporal training-test splitting to nested K-folds Cross-Validation). Elliot optimizes hyperparameters for several recommendation algorithms, selects the best models, compares them with the baselines providing intra-model statistics, computes metrics spanning from accuracy to beyond-accuracy, bias, and fairness, and conducts statistical analysis (Wilcoxon and Paired t-test).

Elliot aims to keep the entire experiment reproducible and put the user in control of the framework.

Installation Elliot works with the following operating systems:

- Linux
- Windows 10
- macOS X

Elliot requires Python version 3.6 or later.

RecBole requires tensorflow version 2.4.1 or later. If you want to use Elliot with GPU, please ensure that CUDA or cudatoolkit version is XXX.XXX or later. This requires NVIDIA driver version >= XXX.XXX (for Linux) or >= XXX.XXX (for Windows10).

Install from source

```
#### CONDA `bash git clone https://github.com/sisinflab/elliott.git && cd  
elliott conda create --name elliott_env python=3.8 conda activate pip install  
-e . --verbose`
```

```
#### VIRTUALENV `bash git clone https://github.com/sisinflab/elliott.git &&  
cd elliott python3 -m venv ./venv source venv/bin/activate pip install -e .  
--verbose`
```

Quick Start

Elliot's entry point is the function *run_experiment*, which accepts a configuration file that drives the

whole experiment. In the following, a sample configuration file is shown to demonstrate how a sample and explicit structure can generate a rigorous experiment.

```
python from elliot.run import run_experiment
run_experiment("configuration/file/path")
```

The following file is a simple configuration for an experimental setup. It contains all the instructions to get the MovieLens-1M catalog from a specific path and perform a train test split in a random sample way with a ratio of 20%.

This experiment provides a hyperparameter optimization with a grid search strategy for an Item-KNN model. Indeed, it is seen that the possible values of neighbors are closed in squared brackets. It indicates that two different models equipped with two different neighbors' values will be trained and compared to select the best configuration. Moreover, this configuration obliges Elliot to save the recommendation lists with at most 10 items per user as suggest by top_k property.

In this basic experiment, only a simple metric is considered in the final evaluation study. The candidate metric is nDCG for a cutoff equal to top_k, unless otherwise noted.

```
yaml experiment:
  dataset: movielens_1m data_config:
    strategy: dataset dataset_path: ../data/movielens_1m/dataset.tsv splitting:
      test_splitting:
        strategy: random_subsampling test_ratio: 0.2
  models:
    ItemKNN:
      meta:
        hyper_opt_alg: grid save_recs: True
        neighbors: [50, 100] similarity: cosine
  evaluation:
    simple_metrics: [nDCG]
    top_k: 10
```

Contributing

There are many ways to contribute to Elliot! You can contribute code, make improvements to the documentation, report or investigate [bugs and issues](<https://github.com/sisinflab/elliot/issues>)

We welcome all contributions from bug fixes to new features and extensions.

Feel free to share with us your custom configuration files. We are creating a vault of reproducible experiments, and we would be glad of mentioning your contribution.

Reference Elliot in your blogs, papers, and articles.

Talk about Elliot on social media with the hashtag **#elliotsr**.

The Team Elliot is developed by * Vito Walter Anelli^{[*](#f1)} (vitowalter.anelli@poliba.it) * Alejandro Bellogín (alejandro.bellogin@uam.es) * Antonio Ferrara (antonio.ferrara@poliba.it) * Daniele Malitesta (daniele.malitesta@poliba.it) * Felice Antonio Merra (felice.merra@poliba.it) * Claudio Pomo^{[*](#f1)} (claudio.pomo@poliba.it) * Tommaso Di Noia (tommaso.dinoia@poliba.it)

It is maintained by [SisInfLab Group](<http://sisinflab.poliba.it/>) and [Information Retrieval Group](<http://ir.ii.uam.es/>).

<b id="f1">^{*} Corresponding authors ## License ELLIOT uses [APACHE2 License](./LICENSE).

Acknowledgements

We refer to the following repositories to improve our code:

- Slim and KNN-CF parts with
[RecSys2019_DeepLearning_Evaluation](https://github.com/MaurizioFD/RecSys2019_DeepLearning_Ev)

elliott package

1.1 Subpackages

1.1.1 `elliott.dataset` package

Subpackages

elliott.dataset.dataloader package

Submodules

elliott.dataset.dataloader.knowledge_aware_chains module

Module description:

class

`elliott.dataset.dataloader.knowledge_aware_chains.KnowledgeChainsDataObject (config, data_tuple, side_information_data, *args, **kwargs)`

Bases: `object`

Load train and test dataset

`build_dict (dataframe, users)`

`build_sparse ()`

`build_sparse_ratings ()`

`dataframe_to_dict (data)`

`get_test ()`

`get_validation ()`

`class elliott.dataset.dataloader.knowledge_aware_chains.KnowledgeChainsLoader (config, *args, **kwargs)`

Bases: `object`

Load train and test dataset

`check_timestamp (d: pandas.core.frame.DataFrame) → pandas.core.frame.DataFrame`

`generate_dataobjects () → List[object]`

`generate_dataobjects_mock () → List[object]`

`load_attribute_file (attribute_file, separator='\t')`

```
load_dataset_dataframe ( file_ratings, separator='\t', attribute_file=None, feature_file=None,
properties_file=None, column_names=['userId', 'itemId', 'rating', 'timestamp'], additive=True,
threshold=10 )
```

```
load_dataset_dict ( file_ratings, separator='\t', attribute_file=None, feature_file=None,
properties_file=None, additive=True, threshold=10 )
```

```
load_feature_names ( infile, separator='\t' )
```

```
load_item_set ( ratings_file, separator='\t', itemPosition=1 )
```

```
load_properties ( properties_file )
```

```
read_splitting ( folder_path )
```

```
reduce_attribute_map_property_selection ( map, items, feature_names, properties,
additive, threshold=10 )
```

```
reduce_dataset_by_item_list ( ratings_file, items, separator='\t' )
```

elliott.dataset.dataloader.visual_dataloader module

Module description:

```
class elliott.dataset.dataloader.visual_dataloader.VisualDataObject ( config,
data_tuple, side_information_data, *args, **kwargs )
```

Bases: object

Load train and test dataset

```
build_dict ( dataframe, users )
```

```
build_sparse ( )
```

```
build_sparse_ratings ( )
```

```
dataframe_to_dict ( data )
```

```
get_test ( )
```

```
get_validation ( )
```

```
read_images ( images_folder, image_set, size_tuple )
```

```
read_images_multiprocessing ( images_folder, image_set, size_tuple )
```

```
static read_single_image ( images_folder, image_set, size_tuple, image_path )
```

```
class elliott.dataset.dataloader.visual_dataloader.VisualLoader ( config, *args,
**kwargs )
```

Bases: object

Load train and test dataset

```
check_timestamp ( d: pandas.core.frame.DataFrame ) → pandas.core.frame.DataFrame
```

```
generate_dataobjects ( ) → List[object]
```

```
generate_dataobjects_mock ( ) → List[object]
```

```
load_dataset_dataframe ( file_ratings, separator='\t', visual_feature_set=None,
column_names=['userId', 'itemId', 'rating', 'timestamp'] )
```

```
read_splitting ( folder_path )
```

```
reduce_dataset_by_item_list ( ratings_file, items, separator='\t' )
```

Module contents

elliott.dataset.samplers package

Submodules

elliott.dataset.samplers.custom_pointwise_sparse_sampler module

Module description:

```
class    elliott.dataset.samplers.custom_pointwise_sparse_sampler.Sampler    (
indexed_ratings, sp_i_train )
    Bases: object

    step ( events: int, batch_size: int )
```

elliott.dataset.samplers.custom_sampler module

Module description:

```
class elliott.dataset.samplers.custom_sampler.Sampler ( indexed_ratings )
    Bases: object

    step ( events: int, batch_size: int )
```

elliott.dataset.samplers.custom_sparse_sampler module

Module description:

```
class elliott.dataset.samplers.custom_sparse_sampler.Sampler ( indexed_ratings, sp_i_train )
    Bases: object

    step ( events: int, batch_size: int )
```

elliott.dataset.samplers.pairwise_sampler module

Module description:

```
class elliott.dataset.samplers.pairwise_sampler.Sampler ( ratings, users, items )
    Bases: object

    step ( events: int )
```

elliott.dataset.samplers.pipeline_sampler module

Module description:

```
class elliott.dataset.samplers.pipeline_sampler.Sampler ( indexed_ratings, item_indices,
images_path, output_image_size, epochs )
    Bases: object

    pipeline ( num_users, batch_size )

    pipeline_eval ( batch_size )

    read_image ( item )

    read_images_triple ( user, pos, neg )

    step ( events: int, batch_size: int )
```

elliott.dataset.samplers.pointwise_cfgan_sampler module

Module description:

```
class elliot.dataset.samplers.pointwise_cfgan_sampler.Sampler ( indexed_ratings,  
sp_i_train, s_zr, s_pm )
```

Bases: object

```
step ( events: int, batch_size: int )
```

elliot.dataset.samplers.pointwise_pos_neg_ratings_sampler module

Module description:

```
class elliot.dataset.samplers.pointwise_pos_neg_ratings_sampler.Sampler (   
indexed_ratings, sparse_i_ratings )
```

Bases: object

```
step ( events: int, batch_size: int )
```

elliot.dataset.samplers.pointwise_pos_neg_ratio_ratings_sampler module

Module description:

```
class elliot.dataset.samplers.pointwise_pos_neg_ratio_ratings_sampler.Sampler   
( indexed_ratings, sparse_i_ratings, neg_ratio )
```

Bases: object

```
step ( events: int, batch_size: int )
```

elliot.dataset.samplers.pointwise_pos_neg_sampler module

Module description:

```
class elliot.dataset.samplers.pointwise_pos_neg_sampler.Sampler ( indexed_ratings )
```

Bases: object

```
step ( events: int, batch_size: int )
```

elliot.dataset.samplers.pointwise_wide_and_deep_sampler module

Module description:

```
class elliot.dataset.samplers.pointwise_wide_and_deep_sampler.Sampler ( data )
```

Bases: object

```
step ( events: int, batch_size: int )
```

elliot.dataset.samplers.sparse_sampler module

Module description:

```
class elliot.dataset.samplers.sparse_sampler.Sampler ( sp_i_train )
```

Bases: object

```
step ( users: int, batch_size: int )
```

Module contents

Module description:

Submodules

elliot.dataset.abstract_dataset module

```
class elliot.dataset.abstract_dataset.AbstractDataset ( *args, **kwargs )
```

Bases: object

```
abstract build_dict ( )
```

```
abstract build_sparse ( *args )
```

```
abstract get_test ( *args )
```

```
required_attributes = ['config', 'args', 'kwargs', 'users', 'items', 'num_users', 'num_items',
'private_users', 'public_users', 'private_items', 'public_items', 'transactions', 'train_dict',
'i_train_dict', 'sp_i_train', 'test_dict']
```

```
class elliot.dataset.abstract_dataset.ForceRequiredAttributeDefinitionMeta
```

```
Bases: type
```

```
check_required_attributes ( class_object )
```

elliot.dataset.dataset module

Module description:

```
class elliot.dataset.dataset.DataSet ( *args, **kwargs )
```

```
Bases: elliot.dataset.abstract_dataset.AbstractDataset
```

```
Load train and test dataset
```

```
build_dict ( dataframe, users )
```

```
build_sparse ( )
```

```
build_sparse_ratings ( )
```

```
dataframe_to_dict ( data )
```

```
get_test ( )
```

```
get_validation ( )
```

```
class elliot.dataset.dataset.DataSetLoader ( config, *args, **kwargs )
```

```
Bases: object
```

```
Load train and test dataset
```

```
check_timestamp ( d: pandas.core.frame.DataFrame ) → pandas.core.frame.DataFrame
```

```
generate_dataobjects ( ) → List[object]
```

```
generate_dataobjects_mock ( ) → List[object]
```

```
read_splitting ( folder_path )
```

Module contents

Module description:

1.1.2 elliot.evaluation package

Subpackages

elliot.evaluation.metrics package

Subpackages

elliot.evaluation.metrics.accuracy package

Subpackages

elliot.evaluation.metrics.accuracy.AUC package

Submodules

elliot.evaluation.metrics.accuracy.AUC.auc module

This is the implementation of the global AUC metric. It proceeds from a system-wise computation.

class `elliott.evaluation.metrics.accuracy.AUC.auc.AUC` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the global AUC recommendation metric. Passing 'AUC' to the metrics list will enable the computation of the metric.

Note:

This metric does not calculate group-based AUC which considers the AUC scores averaged across users. It is also not limited to k. Instead, it calculates the scores on the entire prediction results regardless the users.

$$\mathrm{AUC} = \frac{\sum_{i=1}^M \text{rank}_i - \frac{\{M\} \times \{(M+1)\}^2}{2}}{\{M\} \times \{N\}}$$

M is the number of positive samples. N is the number of negative samples. `rank_i` is the ascending rank of the *i*th positive sample.

eval ()

Evaluation function :return: the overall value of AUC

static name ()

Metric Name Getter :return: returns the public name of the metric

static needs_full_recommendations ()

elliott.evaluation.metrics.accuracy.AUC.gauc module

This is the implementation of the GroupAUC metric. It proceeds from a user-wise computation, and average the AUC values over the users.

class `elliott.evaluation.metrics.accuracy.AUC.gauc.GAUC` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the GroupAUC recommendation metric. Passing 'GAUC' to the metrics list will enable the computation of the metric.

Note:

It calculates the AUC score of each user, and finally obtains GAUC by weighting the user AUC. It is also not limited to k. Due to our padding for `scores_tensor` in `RankEvaluator` with `-np.inf`, the padding value will influence the ranks of origin items. Therefore, we use descending sort here and make an identity transformation to the formula of AUC, which is shown in `auc` function. For readability, we didn't do simplification in the code.

$$\mathrm{GAUC} = \frac{\{M\} \times \{(M+N+1)\} - \frac{M \times (M+1)}{2} - \sum_{i=1}^M \text{rank}_i}{\{M\} \times \{N\}}$$

M is the number of positive samples. N is the number of negative samples. `rank_i` is the descending rank of the *i*th positive sample.

eval ()

Evaluation function :return: the overall averaged value of AUC

eval_user_metric ()

Evaluation function :return: the overall averaged value of AUC per user

static name ()

Metric Name Getter :return: returns the public name of the metric

static needs_full_recommendations ()

elliott.evaluation.metrics.accuracy.AUC.lauc module

This is the implementation of the Limited AUC metric. It proceeds from a user-wise computation, and average the values over the users.

```
class elliot.evaluation.metrics.accuracy.AUC.lauc.LAUC ( recommendations, config,
params, eval_objects )
```

Bases: `elliot.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Limited AUC recommendation metric. Passing 'LAUC' to the metrics list will enable the computation of the metric.

```
eval_user_metric ( )
```

Evaluation function :return: the overall averaged value of LAUC per user

```
static name ( )
```

Metric Name Getter :return: returns the public name of the metric

Module contents

elliot.evaluation.metrics.accuracy.DSC package

Submodules

elliot.evaluation.metrics.accuracy.DSC.dsc module

This is the implementation of the Sørensen–Dice coefficient metric. It proceeds from a user-wise computation, and average the values over the users.

```
class elliot.evaluation.metrics.accuracy.DSC.dsc.DSC ( recommendations, config, params,
eval_objects, additional_data )
```

Bases: `elliot.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Sørensen–Dice coefficient recommendation metric. Passing 'DSC' to the metrics list will enable the computation of the metric.

```
. math::
```

$$\mathrm{F1@K} = \frac{1+\beta^2}{\frac{1}{\text{precision@k}} + \frac{\beta^2}{\text{recall@k}}}$$

```
eval_user_metric ( )
```

Evaluation function :return: the overall averaged value of Sørensen–Dice coefficient per user

```
static name ( )
```

Metric Name Getter :return: returns the public name of the metric

Module contents

elliot.evaluation.metrics.accuracy.f1 package

Submodules

elliot.evaluation.metrics.accuracy.f1.extended_f1 module

This is the implementation of the F-score metric. It proceeds from a user-wise computation, and average the values over the users.

```
class elliot.evaluation.metrics.accuracy.f1.extended_f1.ExtendedF1 (
recommendations, config, params, eval_objects, additional_data )
```

Bases: `elliot.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the F-score recommendation metric. Passing 'ExtendedF1' to the metrics list will enable the computation of the metric.

```
eval_user_metric ( )
```

```
get ( )
```

```
static name ( )
```

Metric Name Getter :return: returns the public name of the metric

```
process ( )
```

Evaluation function :return: the overall value of Bias Disparity

elliott.evaluation.metrics.accuracy.f1.f1 module

This is the implementation of the F-score metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.accuracy.f1.f1.F1` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the F-score recommendation metric. Passing 'F1' to the metrics list will enable the computation of the metric.

$\mathrm{F1@K} = \frac{1+\beta^2}{\frac{1}{\text{precision@k}} + \frac{\beta^2}{\text{recall@k}}}$

eval_user_metric ()

Evaluation function :return: the overall averaged value of F-score

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.accuracy.hit_rate package

Submodules

elliott.evaluation.metrics.accuracy.hit_rate.hit_rate module

This is the implementation of the Hit Rate metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.accuracy.hit_rate.hit_rate.HR` (*recommendations: Dict[int, List[Tuple[int, float]]], config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Hit Rate recommendation metric. Passing 'HR' to the metrics list will enable the computation of the metric.

$\mathrm{HR@K} = \frac{\text{Number of Hits@K}}{|GT|}$

HR is the number of users with a positive sample in the recommendation list. GT is the total number of samples in the test set.

eval_user_metric ()

Evaluation function :return: the overall averaged value of Hit Rate per user

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.accuracy.map package

Submodules

elliott.evaluation.metrics.accuracy.map.map module

This is the implementation of the Mean Average Precision metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.accuracy.map.map.MAP` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Mean Average Precision recommendation metric. Passing 'MAP' to the metrics list will enable the computation of the metric.

Note:

In this case the normalization factor used is $\frac{1}{\min(m, N)}$, which prevents your AP score from being unfairly suppressed when your number of recommendations couldn't possibly capture all the correct ones.

$$\mathrm{AP@N} = \frac{1}{\min(m, N)} \sum_{k=1}^N P(k) \cdot$$

$$\mathrm{rel}(k) \quad \backslash \backslash \quad \mathrm{MAP@N} \& = \frac{1}{|U| \sum_{u=1}^{|U|} (\mathrm{AP@N})_u}$$

eval_user_metric ()

Evaluation function :return: the overall averaged value of Mean Average Precision per user

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.accuracy.mar package

Submodules

elliott.evaluation.metrics.accuracy.mar.mar module

This is the implementation of the Mean Average Recall metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.accuracy.mar.mar.MAR (recommendations, config, params, eval_objects)`

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Mean Average Recall recommendation metric. Passing 'MAR' to the metrics list will enable the computation of the metric.

$$\begin{aligned} \mathrm{Recall@N} \& = \frac{1}{\min(m, |rel(k)|)} \sum_{k=1}^N P(k) \\ \cdot \mathrm{rel}(k) \quad \backslash \backslash \quad \mathrm{MAR@N} \& = \frac{1}{|U| \sum_{u=1}^{|U|} (\mathrm{Recall@N})_u} \end{aligned}$$

eval_user_metric ()

Evaluation function :return: the overall averaged value of Mean Average Recall per user

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.accuracy.mrr package

Submodules

elliott.evaluation.metrics.accuracy.mrr.mrr module

This is the implementation of the Mean Reciprocal Rank metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.accuracy.mrr.mrr.MRR (recommendations, config, params, eval_objects)`

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Mean Reciprocal Rank recommendation metric. Passing 'MRR' to the metrics list will enable the computation of the metric.

$$\mathrm{MRR} = \frac{1}{|U|} \sum_{i=1}^{|U|} \frac{1}{\mathrm{rank}_i}$$

U is the number of users, rank_i is the rank of the first item in the recommendation list in the test set results for user i.

eval_user_metric ()

Evaluation function :return: the overall averaged value of Mean Reciprocal Rank per user

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.accuracy.ndcg package

Submodules

elliott.evaluation.metrics.accuracy.ndcg.ndcg module

This is the implementation of the normalized Discounted Cumulative Gain metric. It proceeds from a

user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.accuracy.ndcg.ndcg.NDCG` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the nDCG recommendation metric. Passing 'nDCG' to the metrics list will enable the computation of the metric.

$$\begin{gathered} \mathrm{DCG@K} = \sum_{i=1}^K \frac{2^{\mathrm{rel}_i} - 1}{\log_2(i+1)} \\ \mathrm{IDCG@K} = \sum_{i=1}^K \frac{1}{\log_2(i+1)} \\ \mathrm{NDCG_u@K} = \frac{\mathrm{DCG_u@K}}{\mathrm{IDCG_u@K}} \end{gathered}$$

K stands for recommending K items. And the rel_i is the relevance of the item in position i in the recommendation list. 2^{rel_i} equals to 1 if the item hits otherwise 0. U^{te} is for all users in the test set.

compute_idcg (*user, cutoff: int*) → float

Method to compute Ideal Discounted Cumulative Gain :param gain_map: :param cutoff: :return:

compute_user_ndcg (*user_recommendations: List, user, cutoff: int*) → float

Method to compute normalized Discounted Cumulative Gain :param sorted_item_predictions: :param gain_map: :param cutoff: :return:

eval_user_metric ()

Evaluation function :return: the overall averaged value of normalized Discounted Cumulative Gain per user

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

This is the nDCG metric module.

This module contains and expose the recommendation metric.

`elliott.evaluation.metrics.accuracy.precision.package`

Submodules

`elliott.evaluation.metrics.accuracy.precision.precision module`

This is the implementation of the Precision metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.accuracy.precision.precision.Precision` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Precision recommendation metric. Passing 'Precision' to the metrics list will enable the computation of the metric.

$$\mathrm{Precision@K} = \frac{|\mathrm{Rel}_u \cap \mathrm{Rec}_u|}{|\mathrm{Rec}_u|}$$

Rel_u is the set of items relevant to user U , Rec_u is the top K items recommended to users. We obtain the result by calculating the average $\mathrm{Precision@K}$ of each user.

eval_user_metric ()

Evaluation function :return: the overall averaged value of Precision

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

This is the Precision metric module.

This module contains and expose the recommendation metric.

elliott.evaluation.metrics.accuracy.recall package

Submodules

elliott.evaluation.metrics.accuracy.recall.recall module

This is the implementation of the Recall metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.accuracy.recall.recall.Recall (recommendations, config, params, eval_objects)`

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Recall recommendation metric. Passing 'Recall' to the metrics list will enable the computation of the metric.

. _Recall: https://en.wikipedia.org/wiki/Precision_and_recall#Recall

$\mathrm{Recall@K} = \frac{|\mathrm{Rel}_u \cap \mathrm{Rec}_u|}{|\mathrm{Rel}_u|}$

Rel_u is the set of items relevant to user U , Rec_u is the top K items recommended to users. We obtain the result by calculating the average $\mathrm{Recall@K}$ of each user.

eval_user_metric ()

Evaluation Function :return: the overall averaged value of Recall per user

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

This is the Recall metric implementation.

This module contains and expose the recommendation metric.

Module contents

elliott.evaluation.metrics.bias package

Subpackages

elliott.evaluation.metrics.bias.aclt package

Submodules

elliott.evaluation.metrics.bias.aclt.aclt module

This is the implementation of the Average coverage of long tail items metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.bias.aclt.aclt.ACLT (recommendations, config, params, eval_objects)`

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Average coverage of long tail items recommendation metric. Passing 'ACLT' to the metrics list will enable the computation of the metric.

Himan Abdollahpouri, Robin Burke, Bamshad Mobasher Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, 2019

$\mathrm{ACLt} = \frac{1}{|\mathrm{U}_{\{t\}}|} \sum_{u \in \mathrm{U}_{\{f\}}} \sum_{i \in \mathrm{L}_{\{u\}}} 1(i \in \Gamma)$

$\mathrm{U}_{\{t\}}$ is the number of users in the test set. $\mathrm{L}_{\{u\}}$ is the recommended list of items for user u . $1(i \in \Gamma)$ is an indicator function and it equals to 1 when i is in Γ .

eval_user_metric ()

Evaluation function :return: the overall averaged value of ACLT

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.bias.aplt package

Submodules

elliott.evaluation.metrics.bias.aplt.aplt module

This is the implementation of the Average percentage of long tail items metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.bias.aplt.aplt.APLT` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Average percentage of long tail items recommendation metric. Passing 'APLT' to the metrics list will enable the computation of the metric.

Abdollahpouri, H.; Burke, R.; and Mobasher Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017

$$\mathrm{ACLTL} = \frac{1}{|U_t|} \sum_{u \in U_t} \frac{|\{i, i \in (L(u) \cap \Phi)\}|}{|L(u)|}$$

U_t is the number of users in the test set. L_u is the recommended list of items for user u . Φ medium-tail items.

eval_user_metric ()

Evaluation function :return: the overall averaged value of APLT

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.bias.arp package

Submodules

elliott.evaluation.metrics.bias.arp.arp module

This is the implementation of the Average Recommendation Popularity metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.bias.arp.arp.ARP` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Average Recommendation Popularity recommendation metric. Passing 'ARP' to the metrics list will enable the computation of the metric.

Yin, H.; Cui, B.; Li, J.; Yao, J.; and Chen, C. 2012. Challenging the long tail recommendation. Proceedings of the VLDB Endowment

eval_user_metric ()

Evaluation function :return: the overall averaged value of ARP

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.bias.pop_reo package

Submodules

elliott.evaluation.metrics.bias.pop_reo.extended_pop_reo module

This is the implementation of the Popularity-based Ranking-based Equal Opportunity (REO) metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.bias.pop_reo.extended_pop_reo.ExtendedPopREO` (*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Popularity-based Ranking-based Equal Opportunity (REO) recommendation metric. Passing 'ExtendedPopREO' to the metrics list will enable the computation of the metric.

Zhu, Ziwei, Jianling Wang, and James Caverlee. "Measuring and Mitigating Item Under-Recom-

mendation Bias in Personalized Ranking Systems.” Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.

eval ()

Evaluation function :return: the overall averaged value of PopREO

static name ()

Metric Name Getter :return: returns the public name of the metric

elliott.evaluation.metrics.bias.pop_reo.pop_reo module

This is the implementation of the Popularity-based Ranking-based Equal Opportunity (REO) metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.bias.pop_reo.pop_reo.PopREO` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Popularity-based Ranking-based Equal Opportunity (REO) recommendation metric. Passing ‘PopREO’ to the metrics list will enable the computation of the metric.

Zhu, Ziwei, Jianling Wang, and James Caverlee. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.

$\mathrm{REO} = \frac{\mathrm{std}(\left(P \left(R @ k \mid g=g_{\{1\}}, y=1\right) \ldots P \left(R @ k \mid g=g_{\{A\}}, y=1\right)\right)}{\mathrm{mean}(\left(P \left(R @ k \mid g=g_{\{1\}}, y=1\right) \ldots P \left(R @ k \mid g=g_{\{A\}}, y=1\right)\right)}$

$P \left(R @ k \mid g=g_{\{a\}}, y=1\right)$ is $\frac{\sum_{u=1}^N \sum_{i=1}^k G_{\{g_{\{a\}}\}} \left(R_{\{u, i\}}\right) Y \left(u, R_{\{u, i\}}\right)}{\sum_{u=1}^N \sum_{i \in I \backslash I_{\{u\}}^{+}} G_{\{g_{\{a\}}\}}(i) Y(u, i)}$

$Y \left(u, R_{\{u, i\}}\right)$ identifies the ground-truth label of a user-item pair $\left(u, R_{\{u, i\}}\right)$, if item $R_{\{u, i\}}$ is liked by user u , returns 1, otherwise 0

$\sum_{i=1}^k G_{\{g_{\{a\}}\}} \left(R_{\{u, i\}}\right) Y \left(u, R_{\{u, i\}}\right)$ counts how many items in test set from group $\{g_a\}$ are ranked in top- k for user u

$\sum_{i \in I \backslash I_{\{u\}}^{+}} G_{\{g_{\{a\}}\}}(i) Y(u, i)$ counts the total number of items from group $\{g_a\}$ in test set for user u

eval ()

Evaluation function :return: the overall averaged value of PopREO

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.bias.pop_rsp package

Submodules

elliott.evaluation.metrics.bias.pop_rsp.extended_pop_rsp module

This is the implementation of the Popularity-based Ranking-based Statistical Parity (RSP) metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.bias.pop_rsp.extended_pop_rsp.ExtendedPopRSP` (*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Popularity-based Ranking-based Statistical Parity (RSP) recommendation metric. Passing ‘ExtendedPopRSP’ to the metrics list will enable the computation of the metric.

Zhu, Ziwei, Jianling Wang, and James Caverlee. “Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems.” Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.

eval ()

Evaluation function :return: the overall averaged value of PopRSP

static name ()

Metric Name Getter :return: returns the public name of the metric

elliott.evaluation.metrics.bias.pop_rsp.pop_rsp module

This is the implementation of the Popularity-based Ranking-based Statistical Parity (RSP) metric. It proceeds from a user-wise computation, and average the values over the users.

class *elliott.evaluation.metrics.bias.pop_rsp.pop_rsp.PopRSP* (*recommendations, config, params, eval_objects*)

Bases: *elliott.evaluation.metrics.base_metric.BaseMetric*

This class represents the implementation of the Popularity-based Ranking-based Statistical Parity (RSP) recommendation metric. Passing 'PopRSP' to the metrics list will enable the computation of the metric.

Zhu, Ziwei, Jianling Wang, and James Caverlee. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.

$$\mathrm{RSP} = \frac{\mathrm{std}(\left(P\left(R @ k \mid g=g_{\{1\}}\right), \ldots, P\left(R @ k \mid g=g_{\{A\}}\right)\right))}{\mathrm{mean}(\left(P\left(R @ k \mid g=g_{\{1\}}\right), \ldots, P\left(R @ k \mid g=g_{\{A\}}\right)\right))}$$

$P\left(R @ k \mid g=g_{\{a\}}\right)$ is $\frac{\sum_{u=1}^N \sum_{i=1}^k G_{\{g_{\{a\}}\}}(R_{\{u, i\}})}{\sum_{i=1}^k G_{\{g_{\{a\}}\}}}$

$\sum_{i=1}^k G_{\{g_{\{a\}}\}}(R_{\{u, i\}})$ calculates how many un-interacted items from group $\{g_{\{a\}}\}$ are ranked in top-? for user u .

$\sum_{i \in I} \backslash I_{\{u\}}^{+} G_{\{g_{\{a\}}\}}(i)$ calculates how many un-interacted items belong to group $\{g_{\{a\}}\}$ for u

eval ()

Evaluation function :return: the overall averaged value of PopRSP

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

Module contents

elliott.evaluation.metrics.coverage package

Subpackages

elliott.evaluation.metrics.coverage.item_coverage package

Submodules

elliott.evaluation.metrics.coverage.item_coverage.item_coverage module

This is the implementation of the Item Coverage metric. It directly proceeds from a system-wise computation, and it considers all the users at the same time.

class

elliott.evaluation.metrics.coverage.item_coverage.item_coverage.ItemCoverage (*recommendations, config, params, eval_objects*)

Bases: *elliott.evaluation.metrics.base_metric.BaseMetric*

This class represents the implementation of the Item Coverage recommendation metric. Passing 'ItemCoverage' to the metrics list will enable the computation of the metric.

Ricci F, Rokach L, Shapira B, Kantor P. 2015

Note:

The simplest measure of catalog coverage is the percentage of all items that can ever be recommended. This measure can be computed in many cases directly given the algorithm and the input data set.

eval ()

Evaluation function :return: the overall averaged value of Item Coverage

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

This is the Item Coverage metric module.

This module contains and expose the recommendation metric.

elliott.evaluation.metrics.coverage.num_retrieved package

Submodules

elliott.evaluation.metrics.coverage.num_retrieved.num_retrieved module

This is the implementation of the NumRetrieved metric. It proceeds from a user-wise computation, and average the values over the users.

class

`elliott.evaluation.metrics.coverage.num_retrieved.num_retrieved.NumRetrieved`
(*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the NumRetrieved recommendation metric. Passing 'NumRetrieved' to the metrics list will enable the computation of the metric.

eval_user_metric ()

Evaluation function :return: the overall averaged value of NumRetrieved

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.coverage.user_coverage package

Submodules

elliott.evaluation.metrics.coverage.user_coverage.user_coverage module

This is the implementation of the User Coverage metric. It directly proceeds from a system-wise computation, and it considers all the users at the same time.

class

`elliott.evaluation.metrics.coverage.user_coverage.user_coverage.UserCoverage`
(*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the User Coverage recommendation metric. Passing 'UserCoverage' to the metrics list will enable the computation of the metric.

Ricci F, Rokach L, Shapira B, Kantor P. 2015

Note:

The proportion of users or user interactions for which the system can recommend items. In many applications the recommender may not provide recommendations for some users due to, e.g. low confidence in the accuracy of predictions for that user.

eval ()

Evaluation function :return: the overall averaged value of User Coverage

static name ()

Metric Name Getter :return: returns the public name of the metric

elliott.evaluation.metrics.coverage.user_coverage.user_coverage_at_n module

This is the implementation of the User Coverage metric. It directly proceeds from a system-wise computation, and it considers all the users at the same time.

class

`elliott.evaluation.metrics.coverage.user_coverage.user_coverage_at_n.UserCoverageAtN`
(*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the User Coverage recommendation metric. Passing 'UserCoverageAtN' to the metrics list will enable the computation of the metric.

eval ()

Evaluation function :return: the overall averaged value of User Coverage

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

Module contents

elliott.evaluation.metrics.diversity package

Subpackages

elliott.evaluation.metrics.diversity.SRecall package

Submodules

elliott.evaluation.metrics.diversity.SRecall.srecall module

This is the implementation of the SRecall metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.diversity.SRecall.srecall.SRecall` (*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the SRecall recommendation metric. Passing 'SRecall' to the metrics list will enable the computation of the metric.

3. 24. Zhai, W. W. Cohen, and J. Lafferty, 2003

$$\mathrm{SRecall} = \frac{|\left| \cup_{i=1}^K \{\text{subtopics} \mid \left| d_i \right| \right| \right|}{n_A}$$

eval_user_metric ()

Evaluation function :return: the overall averaged value of SRecall

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.diversity.gini_index package

Submodules

elliott.evaluation.metrics.diversity.gini_index.gini_index module

This is the implementation of the Gini Index metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.diversity.gini_index.gini_index.GiniIndex` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Gini Index recommendation metric. Passing 'Gini' to the metrics list will enable the computation of the metric.

Ricci F, Rokach L, Shapira B, Kantor P. 2015

$$\mathrm{GiniIndex} = \frac{1}{n-1} \sum_{j=1}^n (2j-n-1) p(i_j)$$

i_j is the list of items ordered according to increasing $p(i)$

eval ()

Evaluation function :return: the overall averaged value of Gini Index

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.diversity.shannon_entropy package

Submodules

elliott.evaluation.metrics.diversity.shannon_entropy.shannon_entropy module

This is the implementation of the Shannon Entropy metric. It proceeds from a user-wise computation, and average the values over the users.

class

`elliott.evaluation.metrics.diversity.shannon_entropy.shannon_entropy.ShannonEntropy`
(*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Shannon Entropy recommendation metric.

Passing 'SEntropy' to the metrics list will enable the computation of the metric.

Ricci F, Rokach L, Shapira B, Kantor P. 2015

$\mathrm{ShannonEntropy} = -\sum_{i=1}^n p(i) \log p(i)$

eval ()

Evaluation function :return: the overall value of Shannon Entropy

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

Module contents

elliott.evaluation.metrics.fairness package

Subpackages

elliott.evaluation.metrics.fairness.BiasDisparity package

Submodules

elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBD module

This is the implementation of the Bias Disparity metric. It proceeds from a user-wise computation, and average the values over the users.

class

`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBD.BiasDisparityBD`
(*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Bias Disparity recommendation metric. Passing 'BiasDisparityBD' to the metrics list will enable the computation of the metric.

Tsintzou, Virginia, Evaggelia Pitoura, and Panayiotis Tsaparas. Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th {ACM} Conference on Recommender Systems (RecSys 2019)

$\mathrm{BD}(G, C) = \frac{B_{\{R\}}(G, C) - B_{\{S\}}(G, C)}{B_{\{S\}}(G, C)}$

eval ()

get ()

name ()

Metric Name Getter :return: returns the public name of the metric

process ()

Evaluation function :return: the overall value of Bias Disparity

elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBR module

This is the implementation of the Bias Disparity - Bias Recommendations metric. It proceeds from a user-wise computation, and average the values over the users.

class

`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBR.BiasDisparityBR`
(*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Bias Disparity - Bias Recommendations recommendation metric. Passing 'BiasDisparityBR' to the metrics list will enable the computation of the metric.

Tsintzou, Virginia, Evaggelia Pitoura, and Panayiotis Tsaparas. Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th {ACM} Conference on Recommender Systems (RecSys 2019)

$$\mathrm{BD}(G, C) =$$

$$\frac{B_{\{R\}}(G, C) - B_{\{S\}}(G, C)}{B_{\{S\}}(G, C)}$$

eval ()

get ()

get_BR ()

name ()

Metric Name Getter :return: returns the public name of the metric

process ()

Evaluation function :return: the overall value of Bias Disparity - Bias Recommendations

elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBS module

This is the implementation of the Bias Disparity - Bias Source metric. It proceeds from a user-wise computation, and average the values over the users.

class

`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBS.BiasDisparityBS`
(*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Bias Disparity - Bias Source recommendation metric. Passing 'BiasDisparityBS' to the metrics list will enable the computation of the metric.

Tsintzou, Virginia, Evaggelia Pitoura, and Panayiotis Tsaparas. Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th {ACM} Conference on Recommender Systems (RecSys 2019)

$$\mathrm{B}_{\{S\}}(G, C) =$$

$$\frac{P_{R_{\{S\}}}(G, C)}{P(C)}$$

eval ()

get ()

get_BS ()

name ()

Metric Name Getter :return: returns the public name of the metric

process ()

Evaluation function :return: the overall value of Bias Disparity - Bias Source

Module contents

elliott.evaluation.metrics.fairness.MAD package

Submodules

elliott.evaluation.metrics.fairness.MAD.ItemMADranking module

This is the implementation of the Item MAD ranking metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.fairness.MAD.ItemMADranking.ItemMADranking` (`recommendations, config, params, eval_objects, additional_data`)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Item MAD ranking recommendation metric. Passing 'ItemMADranking' to the metrics list will enable the computation of the metric.

Deldjoo, Yashar, Vito Walter Anelli, Hamed Zamani, Alejandro Bellogin, and Tommaso Di Noia. User Modeling and User-Adapted Interaction (2020): 1-47.

$$\mathrm{MAD} = \{\mathrm{avg}\}_{i,j}(\{\mathrm{MAD}\}(R^{\{i\}}, R^{\{j\}}))$$

Math $\{\mathrm{MAD}\} = \{\mathrm{avg}\}_{i,j}(\{\mathrm{MAD}\} \left(R^{\{i\}}, R^{\{j\}} \right))$

eval ()

Evaluation function :return: the overall averaged value of Item MAD ranking

get ()

name ()

Metric Name Getter :return: returns the public name of the metric

elliott.evaluation.metrics.fairness.MAD.ItemMADrating module

This is the implementation of the Item MAD rating metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.fairness.MAD.ItemMADrating.ItemMADrating` (`recommendations, config, params, eval_objects, additional_data`)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Item MAD rating recommendation metric. Passing 'ItemMADrating' to the metrics list will enable the computation of the metric.

Zhu, Ziwei, Xia Hu, and James Caverlee. Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 2018.

$$\mathrm{MAD} = \{\mathrm{avg}\}_{i,j}(\{\mathrm{MAD}\}(R^{\{i\}}, R^{\{j\}}))$$

Math $\{\mathrm{MAD}\} = \{\mathrm{avg}\}_{i,j}(\{\mathrm{MAD}\} \left(R^{\{i\}}, R^{\{j\}} \right))$

eval ()

Evaluation function :return: the overall averaged value of Item MAD rating

get ()

name ()

Metric Name Getter :return: returns the public name of the metric

elliott.evaluation.metrics.fairness.MAD.UserMADranking module

This is the implementation of the User MAD ranking metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.fairness.MAD.UserMADranking.UserMADranking` (
recommendations, config, params, eval_objects, additional_data)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the User MAD ranking recommendation metric. Passing 'UserMADranking' to the metrics list will enable the computation of the metric.

$$\mathrm{MAD} = \mathrm{avg}_{i,j} (\mathrm{MAD}(R^{\{i\}}, R^{\{j\}}))$$

Math $\mathrm{MAD} = \mathrm{avg}_{i,j} (\mathrm{MAD}(\mathrm{left}(R^{\{i\}}, R^{\{j\}})))$

compute_idcg (*user: int, cutoff: int*) → float

Method to compute Ideal Discounted Cumulative Gain :param gain_map: :param cutoff:
 :return:

compute_user_ndcg (*user_recommendations: List, user: int, cutoff: int*) → float

Method to compute normalized Discounted Cumulative Gain :param sorted_item_predictions:
 :param gain_map: :param cutoff: :return:

eval ()

Evaluation function :return: the overall averaged value of User MAD ranking

get ()

name ()

Metric Name Getter :return: returns the public name of the metric

elliott.evaluation.metrics.fairness.MAD.UserMADrating module

This is the implementation of the User MAD rating metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.fairness.MAD.UserMADrating.UserMADrating` (
recommendations, config, params, eval_objects, additional_data)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the User MAD rating recommendation metric. Passing 'UserMADrating' to the metrics list will enable the computation of the metric.

Zhu, Ziwei, Xia Hu, and James Caverlee. Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 2018.

$$\mathrm{MAD} = \mathrm{avg}_{i,j} (\mathrm{MAD}(R^{\{i\}}, R^{\{j\}}))$$

Math $\mathrm{MAD} = \mathrm{avg}_{i,j} (\mathrm{MAD}(\mathrm{left}(R^{\{i\}}, R^{\{j\}})))$

eval ()

Evaluation function :return: the overall averaged value of User MAD rating

get ()

name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

This is the Precision metric module.

This module contains and expose the recommendation metric.

elliott.evaluation.metrics.fairness.reo package

Submodules

elliott.evaluation.metrics.fairness.reo.reo module

This is the implementation of the Ranking-based Equal Opportunity (REO) metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.fairness.reo.reo.REO` (*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Ranking-based Equal Opportunity (REO) recommendation metric. Passing 'REO' to the metrics list will enable the computation of the metric.

Zhu, Ziwei, Jianling Wang, and James Caverlee. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.

$$\mathrm{REO} = \frac{\mathrm{std}(\left(P(\left(R @ k \mid g=g_{\{1\}}, y=1\right) \ldots P(\left(R(a) k=g_{\{A\}}, y=1\right)\right) \right) \{\mathrm{mean}(\left(P(\left(R @ k \mid g=g_{\{1\}}, y=1\right) \ldots P(\left(R @ k \mid g=g_{\{A\}}, y=1\right)\right) \right) \}}{\mathrm{std}(\left(P(\left(R @ k \mid g=g_{\{1\}}, y=1\right) \ldots P(\left(R @ k \mid g=g_{\{A\}}, y=1\right)\right) \right) \{\mathrm{mean}(\left(P(\left(R @ k \mid g=g_{\{1\}}, y=1\right) \ldots P(\left(R @ k \mid g=g_{\{A\}}, y=1\right)\right) \right) \}}}$$

$$P(\left(R @ k \mid g=g_{\{a\}}, y=1\right) \text{ is } \frac{\sum_{u=1}^N \sum_{i=1}^k G_{\{g_{\{a\}}\}}(R_{\{u, i\}}) Y(u, R_{\{u, i\}})}{\sum_{u=1}^N \sum_{i \in I \backslash I_{\{u\}}^{+}} G_{\{g_{\{a\}}\}}(i) Y(u, i)}$$

$Y(u, R_{\{u, i\}})$ identifies the ground-truth label of a user-item pair $(u, R_{\{u, i\}})$, if item $R_{\{u, i\}}$ is liked by user u , returns 1, otherwise 0

$\sum_{i=1}^k G_{\{g_{\{a\}}\}}(R_{\{u, i\}}) Y(u, R_{\{u, i\}})$ counts how many items in test set from group $\{g_a\}$ are ranked in top- k for user u

$\sum_{i \in I \backslash I_{\{u\}}^{+}} G_{\{g_{\{a\}}\}}(i) Y(u, i)$ counts the total number of items from group $\{g_a\}$ in test set for user u

eval ()

get ()

name ()

Metric Name Getter :return: returns the public name of the metric

process ()

Evaluation function :return: the overall value of Ranking-based Equal Opportunity (REO)

Module contents

elliott.evaluation.metrics.fairness.rsp package

Submodules

elliott.evaluation.metrics.fairness.rsp.rsp module

This is the implementation of the Ranking-based Statistical Parity (RSP) metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.fairness.rsp.rsp.RSP` (*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Ranking-based Statistical Parity (RSP) recommendation metric. Passing 'RSP' to the metrics list will enable the computation of the metric.

Zhu, Ziwei, Jianling Wang, and James Caverlee. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.

$$\mathrm{RSP} =$$

$$\frac{\mathrm{rac}(\mathrm{std}(P(R @ k \mid g=g_{\{1\}}), \ldots, P(R @ k \mid g=g_{\{A\}}))}{\{\mathrm{mean}(P(R @ k \mid g=g_{\{1\}}), \ldots, P(R @ k \mid g=g_{\{A\}}))\}}$$

$$\{ \mathrm{mean}(P(R @ k \mid g=g_{\{1\}}), \ldots, P(R @ k \mid g=g_{\{A\}})) \}$$

$P(R @ k \mid g=g_{\{a\}})$ is`

$$\mathrm{rac}(\sum_{u=1}^N \sum_{i=1}^k G_{\{g_{\{a\}}\}}(R_{\{u, i\}})$$

$\{\sum_{u=1}^N \sum_{i \in I \backslash I_{\{u\}}^{+}} G_{\{g_{\{a\}}\}}(i)\}$

$\sum_{i=1}^k G_{\{g_{\{a\}}\}}(R_{\{u, i\}})$ calculates how many un-interacted items from group $\{g_a\}$ are ranked in top- k for user u .

$\sum_{i \in I \backslash I_{\{u\}}^{+}} G_{\{g_{\{a\}}\}}(i)$ calculates how many un-in-

teracted items belong to group $\{g_a\}$ for u

eval ()

get ()

name ()

Metric Name Getter :return: returns the public name of the metric

process ()

Evaluation function :return: the overall value of Ranking-based Statistical Parity (RSP)

Module contents

Module contents

elliott.evaluation.metrics.novelty package

Subpackages

elliott.evaluation.metrics.novelty.EFD package

Submodules

elliott.evaluation.metrics.novelty.EFD.efd module

This is the implementation of the Expected Free Discovery metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.novelty.EFD.efd.EFD` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Expected Free Discovery recommendation metric. Passing 'EFD' to the metrics list will enable the computation of the metric.

Note:

EFD can be read as the expected ICF of seen recommended items

$$\mathrm{EFD} = C \sum_{i \in R} \{disc\}(k) p(\{rel\} \mid i_{\{k\}}, u) (-\log_{\{2\}} p(i \mid \{seen\}, \eta))$$

eval_user_metric ()

Evaluation function :return: the overall averaged value of Expected Free Discovery per user

static name ()

Metric Name Getter :return: returns the public name of the metric

elliott.evaluation.metrics.novelty.EFD.extended_efd module

This is the implementation of the Expected Free Discovery metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.novelty.EFD.extended_efd.ExtendedEFD` (*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Expected Free Discovery recommendation metric. Passing 'ExtendedEFD' to the metrics list will enable the computation of the metric.

eval_user_metric ()

Evaluation function :return: the overall averaged value of Expected Free Discovery per user

static name ()

Metric Name Getter :return: returns the public name of the metric

Module contents

elliott.evaluation.metrics.novelty.EPC package

Submodules

elliott.evaluation.metrics.novelty.EPC.epc module

This is the implementation of the Expected Popularity Complement metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.novelty.EPC.epc.EPC` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Expected Popularity Complement recommendation metric. Passing 'EPC' to the metrics list will enable the computation of the metric.

S. Vargas and P. Castells Proceedings of RecSys 2011

Note:

EPC can be read as the expected number of seen relevant recommended items not previously seen

$$\mathrm{EPC} = C \sum_{i_k \in R} \{ \mathrm{disc}(k) \cdot p(\mathrm{rel} \mid i_k), u \} \cdot (1 - p(\mathrm{seen} \mid i_k))$$

eval_user_metric ()

Evaluation function :return: the overall averaged value of Expected Popularity Complement per user

static name ()

Metric Name Getter :return: returns the public name of the metric

elliott.evaluation.metrics.novelty.EPC.extended_epc module

This is the implementation of the Expected Popularity Complement metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.novelty.EPC.extended_epc.ExtendedEPC` (*recommendations, config, params, eval_objects, additional_data*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Expected Popularity Complement recommendation metric. Passing 'ExtendedEPC' to the metrics list will enable the computation of the metric.

eval_user_metric ()

Evaluation function :return: the overall averaged value of Expected Popularity Complement per user

static name ()

Metric Name Getter :return: returns the public name of the metric

*Module contents**Module contents**elliott.evaluation.metrics.rating package**Subpackages**elliott.evaluation.metrics.rating.mae package**Submodules**elliott.evaluation.metrics.rating.mae.mae module*

This is the implementation of the Mean Absolute Error metric. It proceeds from a system-wise computation.

class `elliott.evaluation.metrics.rating.mae.mae.MAE` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Mean Absolute Error recommendation metric. Passing 'MAE' to the metrics list will enable the computation of the metric.

$$\mathrm{MAE} = \frac{1}{|T|} \sum_{(u, i) \in T} | \hat{r}_{ui} - r_{ui} |$$

T is the test set, \hat{r}_{u_i} is the score predicted by the model, and r_{u_i} the actual score of the test set.

eval ()

Evaluation function :return: the overall averaged value of Mean Absolute Error

eval_user_metric ()

Evaluation function :return: the overall averaged value of Mean Absolute Error per user

static name ()

Metric Name Getter :return: returns the public name of the metric

static needs_full_recommendations ()

Module contents

elliott.evaluation.metrics.rating.mse package

Submodules

elliott.evaluation.metrics.rating.mse.mse module

This is the implementation of the Mean Squared Error metric. It proceeds from a system-wise computation.

class `elliott.evaluation.metrics.rating.mse.mse.MSE` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Mean Squared Error recommendation metric. Passing 'MSE' to the metrics list will enable the computation of the metric.

$\mathrm{MSE} =$

$\frac{1}{|T|} \sum_{(u,i) \in T} (\hat{r}_{u,i} - r_{u,i})^2$

T is the test set, \hat{r}_{u_i} is the score predicted by the model, and r_{u_i} the actual score of the test set.

eval ()

Evaluation function :return: the overall averaged value of Mean Squared Error

eval_user_metric ()

Evaluation function :return: the overall averaged value of Mean Squared Error per user

static name ()

Metric Name Getter :return: returns the public name of the metric

static needs_full_recommendations ()

Module contents

elliott.evaluation.metrics.rating.rmse package

Submodules

elliott.evaluation.metrics.rating.rmse.rmse module

This is the implementation of the Root Mean Squared Error metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.rating.rmse.rmse.RMSE` (*recommendations, config, params, eval_objects*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

This class represents the implementation of the Root Mean Squared Error recommendation metric. Passing 'RMSE' to the metrics list will enable the computation of the metric.

$\mathrm{RMSE} = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} (\hat{r}_{u,i} - r_{u,i})^2}$

T is the test set, \hat{r}_{u_i} is the score predicted by the model, and r_{u_i} the actual

score of the test set.

eval ()

Evaluation function :return: the overall averaged value of Root Mean Squared Error

eval_user_metric ()

Evaluation function :return: the overall averaged value of Root Mean Squared Error

static name ()

Metric Name Getter :return: returns the public name of the metric

static needs_full_recommendations ()

Module contents

Module contents

Submodules

elliott.evaluation.metrics.base_metric module

This is the implementation of the Precision metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.base_metric.BaseMetric` (*recommendations, config, params, evaluation_objects, additional_data=None*)

Bases: `abc.ABC`

This class represents the implementation of the Precision recommendation metric. Passing 'Precision' to the metrics list will enable the computation of the metric.

eval ()

get ()

abstract name ()

static needs_full_recommendations ()

elliott.evaluation.metrics.metrics_utils module

class `elliott.evaluation.metrics.metrics_utils.ProxyMetric` (*name='ProxyMetric', val=0, needs_full_recommendations=False*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

eval ()

name ()

needs_full_recommendations ()

class `elliott.evaluation.metrics.metrics_utils.ProxyStatisticalMetric` (*name='ProxyMetric', val=0, user_val=0, needs_full_recommendations=False*)

Bases: `elliott.evaluation.metrics.base_metric.BaseMetric`

eval ()

eval_user_metric ()

name ()

needs_full_recommendations ()

elliott.evaluation.metrics.statistical_array_metric module

This is the implementation of the Precision metric. It proceeds from a user-wise computation, and average the values over the users.

class `elliott.evaluation.metrics.statistical_array_metric.StatisticalMetric`

Bases: `object`

This class represents the implementation of the Precision recommendation metric. Passing 'Precision' to the metrics list will enable the computation of the metric.

abstract `eval_user_metric ()`

Module contents

This is the metrics' module.

This module contains and expose the recommendation metrics. Each metric is encapsulated in a specific package.

See the implementation of Precision metric for creating new per-user metrics. See the implementation of Item Coverage for creating new cross-user metrics.

`elliott.evaluation.metrics.parse_metric (metric)`

`elliott.evaluation.metrics.parse_metrics (metrics)`

elliott.evaluation.popularity_utils package

Submodules

elliott.evaluation.popularity_utils.popularity module

Module description: This module provides a popularity class based on number of users who have experienced an item (user-item repetitions in the dataset are counted once)

class `elliott.evaluation.popularity_utils.popularity.Popularity (data, pop_ratio=0.8)`

Bases: `object`

get_custom_pop_obj (`pop_ratio=0.8`)

get_long_tail ()

get_pop_items ()

get_short_head ()

get_sorted_pop_items ()

Module contents

elliott.evaluation.relevance package

Submodules

elliott.evaluation.relevance.relevance module

Module description:

class `elliott.evaluation.relevance.relevance.AbstractRelevanceSingleton`

Bases: `abc.ABC`

abstract `get_rel (user, item)`

static `logarithmic_ranking_discount (k: int) → float`

Method to compute logarithmic discount :param k: :return:

class `elliott.evaluation.relevance.relevance.BinaryRelevance (test, rel_threshold)`

Bases: `elliott.evaluation.relevance.relevance.AbstractRelevanceSingleton`

get_rel (`user, item`)

get_user_rel (`user`)

```
get_user_rel_gains ( user )
```

```
class elliot.evaluation.relevance.relevance.DiscountedRelevance ( test, rel_threshold )
```

```
Bases: elliot.evaluation.relevance.relevance.AbstractRelevanceSingleton
```

```
get_rel ( user, item )
```

```
get_user_rel ( user )
```

```
get_user_rel_gains ( user )
```

```
class elliot.evaluation.relevance.relevance.Relevance ( test, rel_threshold )
```

```
Bases: object
```

```
property binary_relevance
```

```
property discounted_relevance
```

```
get_test ( )
```

Module contents

Module description:

Submodules

elliot.evaluation.evaluator module

Module description:

```
class elliot.evaluation.evaluator.Evaluator ( data: elliot.dataset.dataset.DataSet, params: types.SimpleNamespace )
```

```
Bases: object
```

```
eval ( recommendations )
```

```
Runtime Evaluation of Accuracy Performance (top-k) :return:
```

```
eval_at_k ( recommendations, k )
```

```
get_needed_recommendations ( )
```

elliot.evaluation.statistical_significance module

Module description:

```
class elliot.evaluation.statistical_significance.PairedTTest
```

```
Bases: object
```

```
static common_users ( arr_0: Dict[int, float], arr_1: Dict[int, float] )
```

```
static compare ( arr_0: Dict[int, float], arr_1: Dict[int, float], users: List[int] )
```

```
class elliot.evaluation.statistical_significance.WilcoxonTest
```

```
Bases: object
```

```
static common_users ( arr_0: Dict[int, float], arr_1: Dict[int, float] )
```

```
static compare ( arr_0: Dict[int, float], arr_1: Dict[int, float], users: List[int] )
```

Module contents

Module description:

1.1.3 elliot.hyperoptimization package

Submodules

elliot.hyperoptimization.model_coordinator module

Module description:

class `elliot.hyperoptimization.model_coordinator.ModelCoordinator` (*data_objs*, *base: types.SimpleNamespace*, *params*, *model_class: ClassVar*)

Bases: `object`

This class handles the selection of hyperparameters for the hyperparameter tuning realized with HyperOpt.

objective (*args*)

This function respect the signature, and the return format required for HyperOpt optimization :param *args*: a Dictionary that contains the new hyper-parameter values that will be used in the current run :return: it returns a Dictionary with loss, and status being required by HyperOpt, and *params*, and results being required by the framework

single ()

This function respect the signature, and the return format required for HyperOpt optimization :param *args*: a Dictionary that contains the new hyper-parameter values that will be used in the current run :return: it returns a Dictionary with loss, and status being required by HyperOpt, and *params*, and results being required by the framework

Module contents

Module description:

`elliot.hyperoptimization.parse_algorithms` (*opt_alg*)

`elliot.hyperoptimization.suggest` (*new_ids*, *domain*, *trials*, *seed*, *nbMaxSucessiveFailures=1000*)

1.1.4 elliot.namespace package

Submodules

elliot.namespace.namespace_model module

Module description:

class `elliot.namespace.namespace_model.NamespaceModel` (*config_path*, *base_folder_path_elliot*, *base_folder_path_config*)

Bases: `object`

fill_base ()

fill_model ()

elliot.namespace.namespace_model_builder module

Module description:

class `elliot.namespace.namespace_model_builder.Builder`

Bases: `abc.ABC`

The Builder interface specifies methods for creating the different parts of the Product objects.

abstract property base

abstract models () → None

class `elliott.namespace.namespace_model_builder.NamespaceBuilder` (*config_path*, *base_folder_path_elliott*, *base_folder_path_config*)

Bases: `elliott.namespace.namespace_model_builder.Builder`

property `base`

models () → tuple

Module contents

Module description:

1.1.5 elliott.prefiltering package

Submodules

elliott.prefiltering.standard_prefilters module

class `elliott.prefiltering.standard_prefilters.PreFilter`

Bases: `object`

static filter (*d: pandas.core.frame.DataFrame*, *ns: types.SimpleNamespace*) → `pandas.core.frame.DataFrame`

static filter_items_by_popularity (*d: pandas.core.frame.DataFrame*, *threshold*) → `pandas.core.frame.DataFrame`

static filter_iterative_k_core (*d: pandas.core.frame.DataFrame*, *threshold*) → `pandas.core.frame.DataFrame`

static filter_ratings_by_global_average (*d: pandas.core.frame.DataFrame*) → `pandas.core.frame.DataFrame`

static filter_ratings_by_threshold (*d: pandas.core.frame.DataFrame*, *threshold*) → `pandas.core.frame.DataFrame`

static filter_ratings_by_user_average (*d: pandas.core.frame.DataFrame*) → `pandas.core.frame.DataFrame`

static filter_retain_cold_users (*d: pandas.core.frame.DataFrame*, *threshold*) → `pandas.core.frame.DataFrame`

static filter_rounds_k_core (*d: pandas.core.frame.DataFrame*, *threshold*, *n_rounds*) → `pandas.core.frame.DataFrame`

static filter_users_by_profile_size (*d: pandas.core.frame.DataFrame*, *threshold*) → `pandas.core.frame.DataFrame`

Module contents

Module description:

1.1.6 elliot.recommender package

Subpackages

elliot.recommender.NN package

Subpackages

elliot.recommender.NN.attribute_item_knn package

Submodules

elliot.recommender.NN.attribute_item_knn.attribute_item_knn module

Module description:

class

`elliot.recommender.NN.attribute_item_knn.attribute_item_knn.AttributeItemKNN`

`(data, config, params, *args, **kwargs)`

Bases: `elliot.recommender.recommender_utils_mixin.RecMixin`,
`elliot.recommender.base_recommender_model.BaseRecommenderModel`

`build_feature_sparse ()`

`get_recommendations (k: int = 100)`

property name

`restore_weights ()`

`train ()`

elliot.recommender.NN.attribute_item_knn.attribute_item_knn_similarity module

class

`elliot.recommender.NN.attribute_item_knn.attribute_item_knn_similarity.Similarity`

`(data, attribute_matrix, num_neighbors, similarity)`

Bases: `object`

Simple kNN class

`compute_cosine (i_index, j_index)`

`compute_neighbors ()`

`get_item_neighbors (item)`

`get_model_state ()`

`get_transactions ()`

`get_user_recs (u, k)`

`initialize ()`

This function initialize the data model

`process_similarity (similarity)`

`static score_item (neighs, user_items)`

`set_model_state (saving_dict)`

Module contents

elliot.recommender.NN.attribute_user_knn package

Submodules

elliott.recommender.NN.attribute_user_knn.attribute_user_knn module

Module description:

class

`elliott.recommender.NN.attribute_user_knn.attribute_user_knn.AttributeUserKNN`
 (*data, config, params, *args, **kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

build_feature_sparse ()

build_feature_sparse_values ()

compute_binary_profile (*user_items_dict: Dict*)

get_recommendations (*k: int = 100*)

property name

restore_weights ()

train ()

elliott.recommender.NN.attribute_user_knn.attribute_user_knn_similarity module

class

`elliott.recommender.NN.attribute_user_knn.attribute_user_knn_similarity.Similarity`
 (*data, attribute_matrix, num_neighbors, similarity*)

Bases: `object`

Simple kNN class

compute_neighbors ()

get_model_state ()

get_transactions ()

get_user_neighbors (*item*)

get_user_recs (*u, k*)

initialize ()

This function initialize the data model

process_similarity (*similarity*)

static score_item (*neighs, user_neighs_items*)

set_model_state (*saving_dict*)

elliott.recommender.NN.attribute_user_knn.tfidf_utils module

class `elliott.recommender.NN.attribute_user_knn.tfidf_utils.TFIDF` (*map: Dict[int, List[int]]*)

Bases: `object`

get_profiles (*ratings: Dict[int, Dict[int, float]]*)

tfidf ()

Module contents

elliot.recommender.NN.item_knn package

Submodules

elliot.recommender.NN.item_knn.aiolli_ferrari module

Created on 23/10/17 @author: Maurizio Ferrari Dacrema

```
class elliot.recommender.NN.item_knn.aiolli_ferrari.AiolliSimilarity ( data,
maxk=40, shrink=100, similarity='cosine', normalize=True )
```

Bases: object

get_user_recs (user, k=100)

initialize ()

predict (u, i)

```
class elliot.recommender.NN.item_knn.aiolli_ferrari.Compute_Similarity (
dataMatrix, topK=100, shrink=0, normalize=True, asymmetric_alpha=0.5, toversky_alpha=1.0,
toversky_beta=1.0, similarity='cosine', row_weights=None )
```

Bases: object

applyAdjustedCosine ()

Remove from every data point the average for the corresponding row :return:

applyPearsonCorrelation ()

Remove from every data point the average for the corresponding column :return:

compute_similarity (start_col=None, end_col=None, block_size=100)

Compute the similarity for the given dataset :param self: :param start_col: column to begin with :param end_col: column to stop before, end_col is excluded :return:

useOnlyBooleanInteractions ()

```
elliot.recommender.NN.item_knn.aiolli_ferrari.check_matrix ( X, format='csc', dtype=
=<class 'numpy.float32'> )
```

This function takes a matrix as input and transforms it into the specified format. The matrix in input can be either sparse or ndarray. If the matrix in input has already the desired format, it is returned as-is the dtype parameter is always applied and the default is np.float32 :param X: :param format: :param dtype: :return:

elliot.recommender.NN.item_knn.item_knn module

Module description:

```
class elliot.recommender.NN.item_knn.item_knn.ItemKNN ( data, config, params, *args,
**kwargs )
```

Bases: elliot.recommender.recommender_utils_mixin.RecMixin, elliot.recommender.base_recommender_model.BaseRecommenderModel

get_recommendations (k: int = 100)

property name

restore_weights ()

train ()

elliot.recommender.NN.item_knn.item_knn_similarity module

```
class elliot.recommender.NN.item_knn.item_knn_similarity.Similarity ( data,
num_neighbors, similarity )
```

Bases: object

Simple kNN class

```

compute_cosine ( i_index, j_index )

compute_neighbors ( )

get_item_neighbors ( item )

get_model_state ( )

get_transactions ( )

get_user_recs ( u, k )

initialize ( )
    This function initialize the data model

process_cosine ( )

process_similarity ( similarity )

static score_item ( neighs, user_items )

set_model_state ( saving_dict )

```

Module contents

elliot.recommender.NN.user_knn package

Submodules

elliot.recommender.NN.user_knn.aiolli_ferrari module

Created on 23/10/17 @author: Maurizio Ferrari Dacrema

```

class elliot.recommender.NN.user_knn.aiolli_ferrari.AiolliSimilarity ( data,
maxk=40, shrink=100, similarity='cosine', normalize=True )

```

Bases: object

```

get_user_recs ( user, k=100 )

```

```

initialize ( )

```

```

predict ( u, i )

```

```

class elliot.recommender.NN.user_knn.aiolli_ferrari.Compute_Similarity (
dataMatrix, topK=100, shrink=0, normalize=True, asymmetric_alpha=0.5, tversky_alpha=1.0,
tversky_beta=1.0, similarity='cosine', row_weights=None )

```

Bases: object

```

applyAdjustedCosine ( )

```

Remove from every data point the average for the corresponding row :return:

```

applyPearsonCorrelation ( )

```

Remove from every data point the average for the corresponding column :return:

```

compute_similarity ( start_col=None, end_col=None, block_size=100 )

```

Compute the similarity for the given dataset :param self: :param start_col: column to begin with :param end_col: column to stop before, end_col is excluded :return:

```

useOnlyBooleanInteractions ( )

```

```

elliot.recommender.NN.user_knn.aiolli_ferrari.check_matrix ( X, format='csc', dtype=<class 'numpy.float32'> )

```

This function takes a matrix as input and transforms it into the specified format. The matrix in

input can be either sparse or ndarray. If the matrix in input has already the desired format, it is returned as-is the dtype parameter is always applied and the default is np.float32 :param X: :param format: :param dtype: :return:

elliot.recommender.NN.user_knn.user_knn module

Module description:

```
class elliot.recommender.NN.user_knn.user_knn.UserKNN ( data, config, params, *args,
**kwargs )
```

```
    Bases: elliot.recommender.recommender_utils_mixin.RecMixin,
           elliot.recommender.base_recommender_model.BaseRecommenderModel
```

```
    get_recommendations ( k: int = 100 )
```

```
    property name
```

```
    restore_weights ( )
```

```
    train ( )
```

elliot.recommender.NN.user_knn.user_knn_similarity module

```
class elliot.recommender.NN.user_knn.user_knn_similarity.Similarity ( data,
num_neighbors, similarity )
```

```
    Bases: object
```

```
    Simple kNN class
```

```
    compute_neighbors ( )
```

```
    get_model_state ( )
```

```
    get_transactions ( )
```

```
    get_user_neighbors ( item )
```

```
    get_user_recs ( u, k )
```

```
    initialize ( )
```

```
        This function initialize the data model
```

```
    process_similarity ( similarity )
```

```
    static score_item ( neighs, user_neighs_items )
```

```
    set_model_state ( saving_dict )
```

Module contents

Module contents

elliot.recommender.adversarial package

Subpackages

elliot.recommender.adversarial.AMF package

Submodules

elliot.recommender.adversarial.AMF.AMF module

Module description:

```
class elliot.recommender.adversarial.AMF.AMF.AMF ( data, config, params, *args, **kwargs )
```

```
    Bases: elliot.recommender.recommender_utils_mixin.RecMixin,
           elliot.recommender.base_recommender_model.BaseRecommenderModel
```

```
    get_recommendations ( k: int = 100 )
```

property name

train ()

elliott.recommender.adversarial.AMF.AMF_model module

Module description:

class `elliott.recommender.adversarial.AMF.AMF_model.AMF_model (*args, **kwargs)`

Bases: `tensorflow.python.keras.engine.training.Model`

build_perturbation (batch)

Evaluate Adversarial Perturbation with FGSM-like Approach

call (inputs, training=None)

Calls the model on new inputs.

In this case *call* just reapplies all ops in the graph to the new inputs (e.g. build a new computational graph from the provided inputs).

Arguments:

inputs: A tensor or list of tensors. **training:** Boolean or boolean scalar tensor, indicating whether to run

the *Network* in training mode or inference mode.

mask: A mask or list of masks. A mask can be

either a tensor or None (no mask).

Returns:

A tensor if there is a single output, or a list of tensors if there are more than one outputs.

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_positions (predictions, train_mask, items, inner_test_user_true_mask)

get_top_k (predictions, train_mask, k=100)

predict (start, stop, **kwargs)

Generates output predictions for the input samples.

Computation is done in batches. This method is designed for performance in large scale inputs. For small amount of inputs that fit in one batch, directly using `__call__` is recommended for faster execution, e.g., `model(x)`, or `model(x, training=False)` if you have layers such as `tf.keras.layers.BatchNormalization` that behaves differently during inference. Also, note the fact that test loss is not affected by regularization layers like noise and dropout.

Arguments:

x: Input samples. It could be:

- A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
- A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
- A *tf.data* dataset.

- A generator or *keras.utils.Sequence* instance.

A more detailed description of unpacking behavior for iterator types (Dataset, generator, Sequence) is given in the *Unpacking behavior for iterator-like inputs* section of *Model.fit*.

batch_size: Integer or *None*.

Number of samples per batch. If unspecified, *batch_size* will default to 32. Do not specify the *batch_size* if your data is in the form of dataset, generators, or *keras.utils.Sequence* instances (since they generate batches).

verbose: Verbosity mode, 0 or 1. steps: Total number of steps (batches of samples)

before declaring the prediction round finished. Ignored with the default value of *None*. If *x* is a *tf.data* dataset and *steps* is *None*, *predict* will run until the input dataset is exhausted.

callbacks: List of *keras.callbacks.Callback* instances.

List of callbacks to apply during prediction. See [callbacks](/api_docs/python/tf/keras/callbacks).

max_queue_size: Integer. Used for generator or *keras.utils.Sequence*

input only. Maximum size for the generator queue. If unspecified, *max_queue_size* will default to 10.

workers: Integer. Used for generator or *keras.utils.Sequence* input

only. Maximum number of processes to spin up when using process-based threading. If unspecified, *workers* will default to 1. If 0, will execute the generator on the main thread.

use_multiprocessing: Boolean. Used for generator or

keras.utils.Sequence input only. If *True*, use process-based threading. If unspecified, *use_multiprocessing* will default to *False*. Note that because this implementation relies on multiprocessing, you should not pass non-picklable arguments to the generator as they can't be passed easily to children processes.

See the discussion of *Unpacking behavior for iterator-like inputs* for *Model.fit*. Note that *Model.predict* uses the same interpretation rules as *Model.fit* and *Model.evaluate*, so inputs must be unambiguous for all three methods.

Returns:

Numpy array(s) of predictions.

Raises:

RuntimeError: If *model.predict* is wrapped in *tf.function*. *ValueError*: In case of mismatch between the provided

input data and the model's expectations, or in case a stateful model receives a number of samples that is not a multiple of the batch size.

train_step (*batch*, *user_adv_train=False*)

The logic for one training step.

This method can be overridden to support custom training logic. This method is called by *Model.make_train_function*.

This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.

Configuration details for *how* this logic is run (e.g. *tf.function* and *tf.distribute.Strategy* settings), should be left to *Model.make_train_function*, which can also be overridden.

Arguments:

data: A nested structure of `Tensor`s.

Returns:

A *dict* containing values that will be passed to *tf.keras.callbacks.CallbackList.on_train_batch_end*. Typically, the values of the *Model*'s

metrics are returned. Example: {'loss': 0.2, 'accuracy': 0.7}.

Module contents

Module description:

elliott.recommender.adversarial.AMR package

Submodules

elliott.recommender.adversarial.AMR.AMR module

Module description:

```
class elliott.recommender.adversarial.AMR.AMR.AMR ( data, config, params, *args, **kwargs )
    Bases: elliott.recommender.recommender_utils_mixin.RecMixin,
           elliott.recommender.base_recommender_model.BaseRecommenderModel
```

```
    get_recommendations ( k: int = 100 )
```

property name

```
    train ( )
```

elliott.recommender.adversarial.AMR.AMR_model module

Module description:

```
class elliott.recommender.adversarial.AMR.AMR_model.AMR_model ( *args, **kwargs )
```

```
    Bases: tensorflow.python.keras.engine.training.Model
```

```
    build_perturbation ( batch )
```

Evaluate Adversarial Perturbation with FGSM-like Approach

```
    call ( inputs, training=None, mask=None )
```

Calls the model on new inputs.

In this case *call* just reapplies all ops in the graph to the new inputs (e.g. build a new computational graph from the provided inputs).

Arguments:

inputs: A tensor or list of tensors. **training:** Boolean or boolean scalar tensor, indicating whether to run

the *Network* in training mode or inference mode.

mask: A mask or list of masks. A mask can be

either a tensor or None (no mask).

Returns:

A tensor if there is a single output, or a list of tensors if there are more than one outputs.

```
    get_config ( )
```

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

```
    get_top_k ( preds, train_mask, k=100 )
```

```
    predict ( start, stop )
```

Generates output predictions for the input samples.

Computation is done in batches. This method is designed for performance in large scale

inputs. For small amount of inputs that fit in one batch, directly using `__call__` is recommended for faster execution, e.g., `model(x)`, or `model(x, training=False)` if you have layers such as `tf.keras.layers.BatchNormalization` that behaves differently during inference. Also, note the fact that test loss is not affected by regularization layers like noise and dropout.

Arguments:

x: Input samples. It could be:

- A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
- A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
- A `tf.data` dataset.
- A generator or `keras.utils.Sequence` instance.

A more detailed description of unpacking behavior for iterator types (Dataset, generator, Sequence) is given in the *Unpacking behavior for iterator-like inputs* section of `Model.fit`.

batch_size: Integer or None.

Number of samples per batch. If unspecified, `batch_size` will default to 32. Do not specify the `batch_size` if your data is in the form of dataset, generators, or `keras.utils.Sequence` instances (since they generate batches).

verbose: Verbosity mode, 0 or 1. **steps:** Total number of steps (batches of samples)

before declaring the prediction round finished. Ignored with the default value of `None`. If `x` is a `tf.data` dataset and `steps` is `None`, `predict` will run until the input dataset is exhausted.

callbacks: List of `keras.callbacks.Callback` instances.

List of callbacks to apply during prediction. See [callbacks](/api_docs/python/tf/keras/callbacks).

max_queue_size: Integer. Used for generator or `keras.utils.Sequence`

input only. Maximum size for the generator queue. If unspecified, `max_queue_size` will default to 10.

workers: Integer. Used for generator or `keras.utils.Sequence` input

only. Maximum number of processes to spin up when using process-based threading. If unspecified, `workers` will default to 1. If 0, will execute the generator on the main thread.

use_multiprocessing: Boolean. Used for generator or

`keras.utils.Sequence` input only. If `True`, use process-based threading. If unspecified, `use_multiprocessing` will default to `False`. Note that because this implementation relies on multiprocessing, you should not pass non-picklable arguments to the generator as they can't be passed easily to children processes.

See the discussion of *Unpacking behavior for iterator-like inputs* for `Model.fit`. Note that `Model.predict` uses the same interpretation rules as `Model.fit` and `Model.evaluate`, so inputs must be unambiguous for all three methods.

Returns:

Numpy array(s) of predictions.

Raises:

`RuntimeError`: If `model.predict` is wrapped in `tf.function`. `ValueError`: In case of mismatch between the provided

input data and the model's expectations, or in case a stateful model receives a number of samples that is not a multiple of the batch size.

train_step (batch, user_adv_train=False)

The logic for one training step.

This method can be overridden to support custom training logic. This method is called by `Model.make_train_function`.

This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.

Configuration details for *how* this logic is run (e.g. `tf.function` and `tf.distribute.Strategy` settings), should be left to `Model.make_train_function`, which can also be overridden.

Arguments:

data: A nested structure of `Tensor`s.

Returns:

A `dict` containing values that will be passed to `tf.keras.callbacks.CallbackList.on_train_batch_end`. Typically, the values of the `Model`'s metrics are returned. Example: `{'loss': 0.2, 'accuracy': 0.7}`.

Module contents

Module description:

[Module contents](#)

[elliot.recommender.algebraic package](#)

[Subpackages](#)

[elliot.recommender.algebraic.slope_one package](#)

[Submodules](#)

[elliot.recommender.algebraic.slope_one.slope_one module](#)

Module description: Lemire, Daniel, and Anna Maclachlan. "Slope one predictors for online rating-based collaborative filtering." Proceedings of the 2005 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics

```
class elliot.recommender.algebraic.slope_one.slope_one.SlopeOne ( data, config,
params, *args, **kwargs )
```

```
    Bases: elliot.recommender.recommender_utils_mixin.RecMixin,
           elliot.recommender.base_recommender_model.BaseRecommenderModel
```

```
    get_recommendations ( k: int = 100 )
```

```
    property name
```

```
    restore_weights ( )
```

```
    train ( )
```

[elliot.recommender.algebraic.slope_one.slope_one_model module](#)

Lemire, Daniel, and Anna Maclachlan. "Slope one predictors for online rating-based collaborative filtering." Proceedings of the 2005 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics

```
class elliot.recommender.algebraic.slope_one.slope_one_model.SlopeOneModel (
data )
```

```
    Bases: object
```

```
    get_model_state ( )
```

```
    get_user_recs ( u, k )
```

```
    initialize ( )
```

```
    predict ( user, item )
```

```
    set_model_state ( saving_dict )
```

[Module contents](#)

Module contents

elliott.recommender.attentive package

Subpackages

elliott.recommender.attentive.afm package

Submodules

elliott.recommender.attentive.afm.afm module

Module description:

```
class elliott.recommender.attentive.afm.afm.AFM ( data, config, params, *args, **kwargs )
    Bases: elliott.recommender.recommender_utils_mixin.RecMixin,
            elliott.recommender.base_recommender_model.BaseRecommenderModel
```

property name

train ()

elliott.recommender.attentive.afm.afm_model module

Module description:

```
class elliott.recommender.attentive.afm.afm_model.AFMModel ( *args, **kwargs )
```

Bases: tensorflow.python.keras.engine.training.Model

call (inputs, training=None, mask=None)

get_recs (inputs, training=False, **kwargs)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (preds, train_mask, k=100)

predict (inputs, training=False, **kwargs)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (batch)

Module contents

Module contents

Module description:

elliott.recommender.autoencoders package

Subpackages

elliott.recommender.autoencoders.dae package

Submodules

elliott.recommender.autoencoders.dae.multi_dae module

Module description:

```
class elliott.recommender.autoencoders.dae.multi_dae.MultiDAE ( data, config, params,
*args, **kwargs )
```

Bases: elliott.recommender.recommender_utils_mixin.RecMixin,
elliott.recommender.base_recommender_model.BaseRecommenderModel

property name

train ()

elliott.recommender.autoencoders.dae.multi_dae_model module

Module description:

```
class elliott.recommender.autoencoders.dae.multi_dae_model.Decoder ( *args, **kwargs )
```

Bases: tensorflow.python.keras.engine.base_layer.Layer

Converts z, the encoded vector, back into a user interaction vector.

```
call ( inputs, **kwargs )
```

class

```
elliott.recommender.autoencoders.dae.multi_dae_model.DenoisingAutoEncoder ( *args, **kwargs )
```

Bases: tensorflow.python.keras.engine.training.Model

Combines the encoder and decoder into an end-to-end model for training.

```
call ( inputs, training=None, **kwargs )
```

```
get_config ( )
```

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

```
get_top_k ( preds, train_mask, k=100 )
```

```
predict ( inputs, training=False, **kwargs )
```

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

```
train_step ( batch )
```

```
class elliott.recommender.autoencoders.dae.multi_dae_model.Encoder ( *args, **kwargs )
```

Bases: tensorflow.python.keras.engine.base_layer.Layer

Maps user-item interactions to a triplet (z_mean, z_log_var, z).

```
call ( inputs, training=None )
```

Module contents

Module description:

elliott.recommender.autoencoders.vae package

Submodules

elliott.recommender.autoencoders.vae.multi_vae module

Module description:

```
class elliott.recommender.autoencoders.vae.multi_vae.MultivAE ( data, config, params, *args, **kwargs )
```

Bases: `elliott.recommender.recommender_utils.mixin.RecMixin`, `elliott.recommender.base_recommender_model.BaseRecommenderModel`

property name

train ()

elliott.recommender.autoencoders.vae.multi_vae_model module

Module description:

class `elliott.recommender.autoencoders.vae.multi_vae_model.Decoder (*args, **kwargs)`

Bases: `tensorflow.python.keras.engine.base_layer.Layer`

Converts z, the encoded digit vector, back into a readable digit.

call (inputs, **kwargs)

class `elliott.recommender.autoencoders.vae.multi_vae_model.Encoder (*args, **kwargs)`

Bases: `tensorflow.python.keras.engine.base_layer.Layer`

Maps MNIST digits to a triplet (z_mean, z_log_var, z).

call (inputs, training=None)

class `elliott.recommender.autoencoders.vae.multi_vae_model.Sampling (*args, **kwargs)`

Bases: `tensorflow.python.keras.engine.base_layer.Layer`

Uses (z_mean, z_log_var) to sample z, the vector encoding a digit.

call (inputs)

class

`elliott.recommender.autoencoders.vae.multi_vae_model.VariationalAutoEncoder (*args, **kwargs)`

Bases: `tensorflow.python.keras.engine.training.Model`

Combines the encoder and decoder into an end-to-end model for training.

call (inputs, training=None, **kwargs)

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_top_k (preds, train_mask, k=100)

predict (inputs, training=False, **kwargs)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (batch, anneal_ph=0.0, **kwargs)

Module contents

Module description:

Module contents

Module description:

elliott recommender.content_based package

Subpackages

elliott recommender.content_based.VSM package

Submodules

elliott recommender.content_based.VSM.tfidf_utils module

```
class elliott.recommender.content_based.VSM.tfidf_utils.TFIDF ( map: Dict[int,
List[int]] )
```

Bases: object

get_profiles (ratings: Dict[int, Dict[int, float]])

tfidf ()

elliott recommender.content_based.VSM.vector_space_model module

Module description:

```
class elliott.recommender.content_based.VSM.vector_space_model.VSM ( data, config,
params, *args, **kwargs )
```

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

build_feature_sparse (feature_dict, num_entities)

build_feature_sparse_values (feature_dict, num_entities)

compute_binary_profile (user_items_dict: Dict)

get_recommendations (k: int = 100)

property name

restore_weights ()

train ()

elliott recommender.content_based.VSM.vector_space_model_similarity module

class

```
elliott.recommender.content_based.VSM.vector_space_model_similarity.Similarity
( data, user_profile_matrix, item_attribute_matrix, similarity )
```

Bases: object

Simple kNN class

get_model_state ()

get_transactions ()

get_user_recs (u, k)

initialize ()

This function initialize the data model

process_similarity (similarity)

set_model_state (saving_dict)

Module contents

Module contents

elliott.recommender.gan package

Subpackages

elliott.recommender.gan.CFGAN package

Submodules

elliott.recommender.gan.CFGAN.cfgan module

Module description:

```
class elliott.recommender.gan.CFGAN.cfgan.CFGAN ( data, config, params, *args, **kwargs )
    Bases: elliott.recommender.recommender_utils_mixin.RecMixin,
            elliott.recommender.base_recommender_model.BaseRecommenderModel

    get_recommendations ( k: int = 100 )

    property name

    train ( )
```

elliott.recommender.gan.CFGAN.cfgan_model module

Module description:

```
class elliott.recommender.gan.CFGAN.cfgan_model.CFGAN_model ( *args, **kwargs )
    Bases: tensorflow.python.keras.engine.training.Model

    get_config ( )
        Returns the config of the layer.
        A layer config is a Python dictionary (serializable) containing the configuration of a layer.
        The same layer can be reinstantiated later (without its trained weights) from this configuration.
        The config of a layer does not include connectivity information, nor the layer class name.
        These are handled by Network (one layer of abstraction above).
        Returns:
            Python dictionary.

    get_top_k ( predictions, train_mask, k=100 )

    predict ( start, stop, **kwargs )
        Generates output predictions for the input samples.
        Computation is done in batches. This method is designed for performance in large scale
        inputs. For small amount of inputs that fit in one batch, directly using __call__ is recommended
        for faster execution, e.g., model(x), or model(x, training=False) if you have layers such
        as tf.keras.layers.BatchNormalization that behaves differently during inference. Also, note the
        fact that test loss is not affected by regularization layers like noise and dropout.

    Arguments:
        x: Input samples. It could be:
            • A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
            • A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
            • A tf.data dataset.
            • A generator or keras.utils.Sequence instance.

        A more detailed description of unpacking behavior for iterator types (Dataset, generator, Sequence)
        is given in the Unpacking behavior for iterator-like inputs section of Model.fit.
```

batch_size: Integer or None.

Number of samples per batch. If unspecified, *batch_size* will default to 32. Do not specify the *batch_size* if your data is in the form of dataset, generators, or *keras.utils.Sequence* instances (since they generate batches).

verbose: Verbosity mode, 0 or 1. steps: Total number of steps (batches of samples)

before declaring the prediction round finished. Ignored with the default value of None. If *x* is a *tf.data* dataset and *steps* is None, *predict* will run until the input dataset is exhausted.

callbacks: List of *keras.callbacks.Callback* instances.

List of callbacks to apply during prediction. See [callbacks](/api_docs/python/tf/keras/callbacks).

max_queue_size: Integer. Used for generator or *keras.utils.Sequence*

input only. Maximum size for the generator queue. If unspecified, *max_queue_size* will default to 10.

workers: Integer. Used for generator or *keras.utils.Sequence* input

only. Maximum number of processes to spin up when using process-based threading. If unspecified, *workers* will default to 1. If 0, will execute the generator on the main thread.

use_multiprocessing: Boolean. Used for generator or

keras.utils.Sequence input only. If *True*, use process-based threading. If unspecified, *use_multiprocessing* will default to *False*. Note that because this implementation relies on multiprocessing, you should not pass non-picklable arguments to the generator as they can't be passed easily to children processes.

See the discussion of *Unpacking behavior for iterator-like inputs* for *Model.fit*. Note that *Model.predict* uses the same interpretation rules as *Model.fit* and *Model.evaluate*, so inputs must be unambiguous for all three methods.

Returns:

Numpy array(s) of predictions.

Raises:

RuntimeError: If *model.predict* is wrapped in *tf.function*. *ValueError*: In case of mismatch between the provided

input data and the model's expectations, or in case a stateful model receives a number of samples that is not a multiple of the batch size.

train_step(*batch*)

The logic for one training step.

This method can be overridden to support custom training logic. This method is called by *Model.make_train_function*.

This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.

Configuration details for *how* this logic is run (e.g. *tf.function* and *tf.distribute.Strategy* settings), should be left to *Model.make_train_function*, which can also be overridden.

Arguments:

data: A nested structure of *Tensor*'s.

Returns:

A *dict* containing values that will be passed to *tf.keras.callbacks.CallbackList.on_train_batch_end*. Typically, the values of the *Model*'s metrics are returned. Example: {'loss': 0.2, 'accuracy': 0.7}.

```
class elliot.recommender.gan.CFGAN.cfgan_model.Discriminator(*args,**kwargs)
```

Bases: tensorflow.python.keras.engine.training.Model

```
discriminate_fake_data(X)
```

train_step (batch)

The logic for one training step.

This method can be overridden to support custom training logic. This method is called by *Model.make_train_function*.

This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.

Configuration details for *how* this logic is run (e.g. *tf.function* and *tf.distribute.Strategy* settings), should be left to *Model.make_train_function*, which can also be overridden.

Arguments:

data: A nested structure of `Tensor`'s.

Returns:

A *dict* containing values that will be passed to *tf.keras.callbacks.CallbackList.on_train_batch_end*. Typically, the values of the *Model*'s metrics are returned. Example: `{'loss': 0.2, 'accuracy': 0.7}`.

```
class elliot.recommender.gan.CFGAN.cfGAN_model.Generator ( *args, **kwargs )
```

Bases: `tensorflow.python.keras.engine.training.Model`

```
generate_fake_data ( mask, C_u )
```

```
infer ( C_u )
```

train_step (batch)

The logic for one training step.

This method can be overridden to support custom training logic. This method is called by *Model.make_train_function*.

This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.

Configuration details for *how* this logic is run (e.g. *tf.function* and *tf.distribute.Strategy* settings), should be left to *Model.make_train_function*, which can also be overridden.

Arguments:

data: A nested structure of `Tensor`'s.

Returns:

A *dict* containing values that will be passed to *tf.keras.callbacks.CallbackList.on_train_batch_end*. Typically, the values of the *Model*'s metrics are returned. Example: `{'loss': 0.2, 'accuracy': 0.7}`.

Module contents

elliot.recommender.gan.IRGAN package

Submodules

elliot.recommender.gan.IRGAN.irgan module

Module description:

```
class elliot.recommender.gan.IRGAN.irgan.IRGAN ( data, config, params, *args, **kwargs )
```

Bases: `elliot.recommender.recommender_utils_mixin.RecMixin`,
`elliot.recommender.base_recommender_model.BaseRecommenderModel`

```
get_recommendations ( k: int = 100 )
```

property name

```
train ( )
```

elliot.recommender.gan.IRGAN.irgan_model module

Module description:

```
class elliot.recommender.gan.IRGAN.irgan_model.Discriminator ( *args, **kwargs )
    Bases: tensorflow.python.keras.engine.training.Model

    call ( inputs, training=None, mask=None )
        Calls the model on new inputs.
        In this case call just reapplies all ops in the graph to the new inputs (e.g. build a new computational graph from the provided inputs).

        Arguments:
            inputs: A tensor or list of tensors. training: Boolean or boolean scalar tensor, indicating whether to run
                    the Network in training mode or inference mode.
            mask: A mask or list of masks. A mask can be
                  either a tensor or None (no mask).

        Returns:
            A tensor if there is a single output, or a list of tensors if there are more than one outputs.

    train_step ( batch )
        The logic for one training step.
        This method can be overridden to support custom training logic. This method is called by Model.make_train_function.
        This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.
        Configuration details for how this logic is run (e.g. tf.function and tf.distribute.Strategy settings), should be left to Model.make_train_function, which can also be overridden.

        Arguments:
            data: A nested structure of Tensors.

        Returns:
            A dict containing values that will be passed to tf.keras.callbacks.CallbackList.on_train_batch_end. Typically, the values of the Model's metrics are returned. Example: {'loss': 0.2, 'accuracy': 0.7}.
```

```
class elliot.recommender.gan.IRGAN.irgan_model.Generator ( *args, **kwargs )
    Bases: tensorflow.python.keras.engine.training.Model

    call ( inputs, training=None, mask=None )
        Calls the model on new inputs.
        In this case call just reapplies all ops in the graph to the new inputs (e.g. build a new computational graph from the provided inputs).

        Arguments:
            inputs: A tensor or list of tensors. training: Boolean or boolean scalar tensor, indicating whether to run
                    the Network in training mode or inference mode.
            mask: A mask or list of masks. A mask can be
                  either a tensor or None (no mask).

        Returns:
            A tensor if there is a single output, or a list of tensors if there are more than one outputs.

    train_step ( batch )
        The logic for one training step.
        This method can be overridden to support custom training logic. This method is called by Model.make_train_function.
        This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.
```

Configuration details for *how* this logic is run (e.g. *tf.function* and *tf.distribute.Strategy* settings), should be left to *Model.make_train_function*, which can also be overridden.

Arguments:

data: A nested structure of `Tensor`'s.

Returns:

A *dict* containing values that will be passed to *tf.keras.callbacks.CallbackList.on_train_batch_end*. Typically, the values of the *Model*'s metrics are returned. Example: `{'loss': 0.2, 'accuracy': 0.7}`.

train_step_with_reward (batch)

class `elliot.recommender.gan.IRGAN.irgan_model.IRGAN_model (*args, **kwargs)`

Bases: `tensorflow.python.keras.engine.training.Model`

call (inputs, training=None)

Calls the model on new inputs.

In this case *call* just reapplies all ops in the graph to the new inputs (e.g. build a new computational graph from the provided inputs).

Arguments:

inputs: A tensor or list of tensors. training: Boolean or boolean scalar tensor, indicating whether to run

the *Network* in training mode or inference mode.

mask: A mask or list of masks. A mask can be

either a tensor or None (no mask).

Returns:

A tensor if there is a single output, or a list of tensors if there are more than one outputs.

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_positions (predictions, train_mask, items, inner_test_user_true_mask)

get_top_k (predictions, train_mask, k=100)

pre_train_discriminator ()

pre_train_generator ()

predict (start, stop, **kwargs)

Generates output predictions for the input samples.

Computation is done in batches. This method is designed for performance in large scale inputs. For small amount of inputs that fit in one batch, directly using `__call__` is recommended for faster execution, e.g., *model(x)*, or *model(x, training=False)* if you have layers such as *tf.keras.layers.BatchNormalization* that behaves differently during inference. Also, note the fact that test loss is not affected by regularization layers like noise and dropout.

Arguments:

x: Input samples. It could be:

- A Numpy array (or array-like), or a list of arrays (in case the model has

multiple inputs).

- A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
- A *tf.data* dataset.
- A generator or *keras.utils.Sequence* instance.

A more detailed description of unpacking behavior for iterator types (Dataset, generator, Sequence) is given in the *Unpacking behavior for iterator-like inputs* section of *Model.fit*.

batch_size: Integer or *None*.

Number of samples per batch. If unspecified, *batch_size* will default to 32. Do not specify the *batch_size* if your data is in the form of dataset, generators, or *keras.utils.Sequence* instances (since they generate batches).

verbose: Verbosity mode, 0 or 1. steps: Total number of steps (batches of samples)

before declaring the prediction round finished. Ignored with the default value of *None*. If *x* is a *tf.data* dataset and *steps* is *None*, *predict* will run until the input dataset is exhausted.

callbacks: List of *keras.callbacks.Callback* instances.

List of callbacks to apply during prediction. See [callbacks](/api_docs/python/tf/keras/callbacks).

max_queue_size: Integer. Used for generator or *keras.utils.Sequence*

input only. Maximum size for the generator queue. If unspecified, *max_queue_size* will default to 10.

workers: Integer. Used for generator or *keras.utils.Sequence* input

only. Maximum number of processes to spin up when using process-based threading. If unspecified, *workers* will default to 1. If 0, will execute the generator on the main thread.

use_multiprocessing: Boolean. Used for generator or

keras.utils.Sequence input only. If *True*, use process-based threading. If unspecified, *use_multiprocessing* will default to *False*. Note that because this implementation relies on multiprocessing, you should not pass non-picklable arguments to the generator as they can't be passed easily to children processes.

See the discussion of *Unpacking behavior for iterator-like inputs* for *Model.fit*. Note that *Model.predict* uses the same interpretation rules as *Model.fit* and *Model.evaluate*, so inputs must be unambiguous for all three methods.

Returns:

Numpy array(s) of predictions.

Raises:

RuntimeError: If *model.predict* is wrapped in *tf.function*. *ValueError*: In case of mismatch between the provided

input data and the model's expectations, or in case a stateful model receives a number of samples that is not a multiple of the batch size.

train_step()

The logic for one training step.

This method can be overridden to support custom training logic. This method is called by *Model.make_train_function*.

This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.

Configuration details for *how* this logic is run (e.g. *tf.function* and *tf.distribute.Strategy* settings), should be left to *Model.make_train_function*, which can also be overridden.

Arguments:

data: A nested structure of `Tensor`'s.

Returns:

A `dict` containing values that will be passed to `tf.keras.callbacks.CallbackList.on_train_batch_end`. Typically, the values of the `Model`'s metrics are returned. Example: `{'loss': 0.2, 'accuracy': 0.7}`.

Module contents

Module contents

elliott.recommender.graph_based package

Subpackages

elliott.recommender.graph_based.lightgcn package

Submodules

elliott.recommender.graph_based.lightgcn.LightGCN module

Module description:

class `elliott.recommender.graph_based.lightgcn.LightGCN.LightGCN` (`data`, `config`, `params`, `*args`, `**kwargs`)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`, `elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (`k`: `int` = 100)

property name

train ()

elliott.recommender.graph_based.lightgcn.LightGCN_model module

Module description:

class `elliott.recommender.graph_based.lightgcn.LightGCN_model.LightGCNModel` (`*args`, `**kwargs`)

Bases: `tensorflow.python.keras.engine.training.Model`

call (`inputs`, `**kwargs`)

Generates prediction for passed users and items indices

Args:

inputs: user, item (batch) the *Network* in training mode or inference mode.

Returns:

prediction and extracted model parameters

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_top_k (`preds`, `train_mask`, `k`=100)

predict (`start`, `stop`, `**kwargs`)

train_step (`batch`)

Apply a single training step on one batch.

Args:

batch: batch used for the current train step

Returns:

loss value at the current batch

Module contents

elliott.recommender.graph_based.ngcf package

Submodules

elliott.recommender.graph_based.ngcf.NGCF module

Module description:

```
class elliott.recommender.graph_based.ngcf.NGCF.NGCF ( data, config, params, *args, **kwargs )
```

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (k: int = 100)

property name

train ()

elliott.recommender.graph_based.ngcf.NGCF_model module

Module description:

```
class elliott.recommender.graph_based.ngcf.NGCF_model.NGCFModel ( *args, **kwargs )
```

Bases: `tensorflow.python.keras.engine.training.Model`

call (inputs, **kwargs)

Generates prediction for passed users and items indices

Args:

inputs: user, item (batch) the *Network* in training mode or inference mode.

Returns:

prediction and extracted model parameters

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_top_k (preds, train_mask, k=100)

predict (start, stop, **kwargs)

train_step (batch)

Apply a single training step on one batch.

Args:

batch: batch used for the current train step

Returns:

loss value at the current batch

Module contents

Module contents

elliott.recommender.knowledge_aware package

Subpackages

elliott.recommender.knowledge_aware.kaHFM package

Submodules

elliott.recommender.knowledge_aware.kaHFM.ka_hfm module

class `elliott.recommender.knowledge_aware.kaHFM.ka_hfm.KaHFM` (*data, config, params, *args, **kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

predict (*u: int, i: int*)

Get prediction on the user item pair.

Returns:

A single float vaue.

restore_weights ()

train ()

train_step ()

update_factors (*u: int, i: int, j: int*)

class `elliott.recommender.knowledge_aware.kaHFM.ka_hfm.MF` (*ratings: Dict, map: Dict, tfidf: Dict, user_profiles: Dict, random: Any, *args*)

Bases: `object`

Simple Matrix Factorization class

get_factors ()

get_item_bias (*item: int*)

get_item_factors (*item: int*)

get_model_state ()

get_transactions ()

get_user_bias (*user: int*)

get_user_factors (*user: int*)

get_user_recs (*user: int, k: int*)

get_user_recs_argpartition (*user: int, k: int*)

initialize (*loc: float = 0, scale: float = 0.1*)

This function initialize the data model :param loc: :param scale: :return:

property name

predict (*user: int, item: int*)

set_item_bias (*item: int, v: float*)

```
set_item_factors ( item: int, v: float )
```

```
set_model_state ( saving_dict )
```

```
set_user_bias ( user: int, v: float )
```

```
set_user_factors ( user: int, v: float )
```

elliott.recommender.knowledge_aware.kaHFM.tfidf_utils module

```
class elliott.recommender.knowledge_aware.kaHFM.tfidf_utils.TFIDF ( map: Dict[int, List[int]] )
```

```
Bases: object
```

```
get_profiles ( ratings: Dict[int, Dict[int, float]] )
```

```
tfidf ( )
```

Module contents

elliott.recommender.knowledge_aware.kaHFM_batch package

Submodules

elliott.recommender.knowledge_aware.kaHFM_batch.kahfm_batch module

Module description:

```
class elliott.recommender.knowledge_aware.kaHFM_batch.kahfm_batch.KaHFMBatch ( data, config, params, *args, **kwargs )
```

```
Bases: elliott.recommender.recommender_utils_mixin.RecMixin, elliott.recommender.base_recommender_model.BaseRecommenderModel
```

```
get_recommendations ( k: int = 100 )
```

property name

```
train ( )
```

elliott.recommender.knowledge_aware.kaHFM_batch.kahfm_batch_model module

Module description:

class

```
elliott.recommender.knowledge_aware.kaHFM_batch.kahfm_batch_model.KaHFM_model ( *args, **kwargs )
```

```
Bases: tensorflow.python.keras.engine.training.Model
```

```
call ( inputs, training=None, **kwargs )
```

```
get_config ( )
```

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

```
get_top_k ( preds, train_mask, k=100 )
```

```
predict ( inputs, training=False, **kwargs )
```

```
predict_all ( )
```

predict_batch (*start*, *stop*)

train_step (*batch*)

elliott.recommender.knowledge_aware.kaHFM_batch.tfidf_utils module

class *elliott.recommender.knowledge_aware.kaHFM_batch.tfidf_utils.TFIDF* (*map*:
Dict[int, List[int]])

Bases: *object*

get_profiles (*ratings*: *Dict[int, Dict[int, float]]*)

tfidf ()

Module contents

elliott.recommender.knowledge_aware.kahfm_embeddings package

Submodules

elliott.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings module

Module description:

class

elliott.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings.KaHFMEEmbeddings
(*data*, *config*, *params*, **args*, ***kwargs*)

Bases: *elliott.recommender.recommender_utils_mixin.RecMixin*,
elliott.recommender.base_recommender_model.BaseRecommenderModel

get_recommendations (*k*: *int* = 100)

property name

train ()

elliott.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings_model module

Module description:

class

elliott.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings_model.KaHFMEEmbeddings
(**args*, ***kwargs*)

Bases: *tensorflow.python.keras.engine.training.Model*

call (*inputs*, *training*=None, ***kwargs*)

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer.
The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name.
These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_top_k (*preds*, *train_mask*, *k*=100)

predict (*inputs*, *training*=False, ***kwargs*)

predict_batch (*start*, *stop*)

train_step (*batch*)

elliott.recommender.knowledge_aware.kahfm_embeddings.tfidf_utils module

```
class elliott.recommender.knowledge_aware.kahfm_embeddings.tfidf_utils.TFIDF (
    map: Dict[int, List[int]] )
```

Bases: object

```
get_profiles ( ratings: Dict[int, Dict[int, float]] )
```

```
tfidf ( )
```

Module contents

Module contents

elliott.recommender.latent_factor_models package

Subpackages

elliott.recommender.latent_factor_models.BPRMF package

Submodules

elliott.recommender.latent_factor_models.BPRMF.BPRMF module

Module description:

```
class elliott.recommender.latent_factor_models.BPRMF.BPRMF.BPRMF ( data, config,
    params, *args, **kwargs )
```

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

```
get_recommendations ( k: int = 100 )
```

property name

```
predict ( u: int, i: int )
```

Get prediction on the user item pair.

Returns:

A single float vaue.

```
restore_weights ( )
```

```
train ( )
```

```
train_step ( )
```

```
update_factors ( u: int, i: int, j: int )
```

```
class elliott.recommender.latent_factor_models.BPRMF.BPRMF.MF ( F, ratings, random,
    *args )
```

Bases: object

Simple Matrix Factorization class

```
get_item_bias ( item: int )
```

```
get_item_factors ( item: int )
```

```
get_model_state ( )
```

```
get_transactions ( )
```

```
get_user_bias ( user: int )
```

```
get_user_factors ( user: int )
```

```
get_user_recs ( user, k )
```

```

get_user_recs_argpartition ( user: int, k: int )

initialize ( loc: float = 0, scale: float = 0.1 )
    This function initialize the data model :param loc: :param scale: :return:

property name

predict ( user, item )

set_item_bias ( item: int, v: float )

set_item_factors ( item: int, v: float )

set_model_state ( saving_dict )

set_user_bias ( user: int, v: float )

set_user_factors ( user: int, v: float )

```

Module contents

Module description:

elliott.recommender.latent_factor_models.BPRMF_batch package

Submodules

elliott.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch module

Module description:

```

class
elliott.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch.BPRMF_batch
( data, config, params, *args, **kwargs )
    Bases: elliott.recommender.recommender_utils_mixin.RecMixin,
elliott.recommender.base_recommender_model.BaseRecommenderModel

    get_recommendations ( k: int = 100 )

    property name

    restore_weights ( )

    train ( )

```

elliott.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch_model module

Module description:

```

class
elliott.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch_model.BPRMF_batch_model
( *args, **kwargs )
    Bases: tensorflow.python.keras.engine.training.Model

    call ( inputs, training=None )

    get_config ( )
        Returns the config of the layer.
        A layer config is a Python dictionary (serializable) containing the configuration of a layer.
        The same layer can be reinstantiated later (without its trained weights) from this configuration.
        The config of a layer does not include connectivity information, nor the layer class name.
        These are handled by Network (one layer of abstraction above).

    Returns:
        Python dictionary.

```



```

get_positions ( predictions, train_mask, items, inner_test_user_true_mask )

get_top_k ( predictions, train_mask, k=100 )

predict ( start, stop, **kwargs )

train_step ( batch )

```

Module contents

Module description:

elliott.recommender.latent_factor_models.BPRSlim package

Submodules

elliott.recommender.latent_factor_models.BPRSlim.bprslim module

Module description:

```

class elliott.recommender.latent_factor_models.BPRSlim.bprslim.BPRSlim ( data,
config, params, *args, **kwargs )

```

```

    Bases:
        elliott.recommender.recommender_utils_mixin.RecMixin,
        elliott.recommender.base_recommender_model.BaseRecommenderModel

```

```

get_recommendations ( k: int = 100 )

```

property name

```

predict ( u: int, i: int )

```

Get prediction on the user item pair.

Returns:

A single float vaue.

```

restore_weights ( )

```

```

train ( )

```

elliott.recommender.latent_factor_models.BPRSlim.bprslim_model module

Module description:

class

```

elliott.recommender.latent_factor_models.BPRSlim.bprslim_model.BPRSlimModel (
data, num_users, num_items, lr, lj_reg, li_reg, sampler, random_seed=42 )

```

```

    Bases: object

```

```

get_model_state ( )

```

```

get_user_recs ( user, k=100 )

```

```

predict ( u, i )

```

```

set_model_state ( saving_dict )

```

```

train_step ( batch )

```

Module contents

Module description:

elliott.recommender.latent_factor_models.CML package

Submodules

elliott.recommender.latent_factor_models.CML.CML module

Module description:

```
class elliot.recommender.latent_factor_models.CML.CML.CML ( data, config, params, *args,
**kwargs )
    Bases: elliot.recommender.recommender_utils_mixin.RecMixin,
            elliot.recommender.base_recommender_model.BaseRecommenderModel

    get_recommendations ( k: int = 100 )

    property name

    restore_weights ( )

    train ( )
```

elliot.recommender.latent_factor_models.CML.CML_model module

Module description:

```
class elliot.recommender.latent_factor_models.CML.CML_model.CML_model ( *args,
**kwargs )
    Bases: tensorflow.python.keras.engine.training.Model

    call ( inputs, training=None )

    get_config ( )
        Returns the config of the layer.
        A layer config is a Python dictionary (serializable) containing the configuration of a layer.
        The same layer can be reinstantiated later (without its trained weights) from this configuration.
        The config of a layer does not include connectivity information, nor the layer class name.
        These are handled by Network (one layer of abstraction above).
        Returns:
            Python dictionary.

    get_positions ( predictions, train_mask, items, inner_test_user_true_mask )

    get_top_k ( predictions, train_mask, k=100 )

    predict ( start, stop, **kwargs )

    train_step ( batch )
```

```
class elliot.recommender.latent_factor_models.CML.CML_model.LatentFactor (
*args, **kwargs )
    Bases: tensorflow.python.keras.layers.embeddings.Embedding

    censor ( censor_id )
```

Module contents

Module description:

elliot.recommender.latent_factor_models.FFM package

Submodules

elliot.recommender.latent_factor_models.FFM.field_aware_factorization_machine module

Module description:

```
class
elliot.recommender.latent_factor_models.FFM.field_aware_factorization_machine.FFM
( data, config, params, *args, **kwargs )
    Bases: elliot.recommender.recommender_utils_mixin.RecMixin,
            elliot.recommender.base_recommender_model.BaseRecommenderModel
```

```
get_recommendations ( k: int = 100 )
```

property name

```
predict ( u: int, i: int )
```

```
restore_weights ( )
```

```
train ( )
```

elliot.recommender.latent_factor_models.FFM.field_aware_factorization_machine_model module

Module description:

class

```
elliot.recommender.latent_factor_models.FFM.field_aware_factorization_machine_model.Fie  
( *args, **kwargs )
```

Bases: tensorflow.python.keras.engine.training.Model

```
call ( inputs, training=None, mask=None )
```

```
get_recs ( inputs, training=False, **kwargs )
```

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

```
get_top_k ( preds, train_mask, k=100 )
```

```
predict ( inputs, training=False, **kwargs )
```

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

```
train_step ( batch )
```

Module contents

elliot.recommender.latent_factor_models.FISM package

Submodules

elliot.recommender.latent_factor_models.FISM.FISM module

Module description:

```
class elliot.recommender.latent_factor_models.FISM.FISM.FISM ( data, config, params,  
*args, **kwargs )
```

Bases: [elliot.recommender.recommender_utils_mixin.RecMixin](#),
[elliot.recommender.base_recommender_model.BaseRecommenderModel](#)

```
get_recommendations ( k: int = 100, auc_compute: bool = False )
```

property name

```
restore_weights ( )
```

```
train ( )
```

elliot.recommender.latent_factor_models.FISM.FISM_model module

Module description:

```
class elliot.recommender.latent_factor_models.FISM.FISM_model.FISM_model (   
*args, **kwargs )
```

Bases: tensorflow.python.keras.engine.training.Model

batch_predict (*user_start*, *user_stop*, ***kwargs*)

call (*inputs*, *training=None*)

create_history_item_matrix ()

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_positions (*predictions*, *train_mask*, *items*, *inner_test_user_true_mask*)

get_top_k (*predictions*, *train_mask*, *k=100*)

predict (*user*, ***kwargs*)

train_step (*batch*)

class `elliott.recommender.latent_factor_models.FISM.FISM_model.LatentFactor` (**args*, ***kwargs*)

Bases: `tensorflow.python.keras.layers.embeddings.Embedding`

__init__ (*tensor_id*)

Module contents

Module description:

`elliott.recommender.latent_factor_models.FM` package

Submodules

`elliott.recommender.latent_factor_models.FM.factorization_machine` module

Module description:

class `elliott.recommender.latent_factor_models.FM.factorization_machine.FM` (*data*, *config*, *params*, **args*, ***kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`, `elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

predict (*u: int*, *i: int*)

restore_weights ()

train ()

`elliott.recommender.latent_factor_models.FM.factorization_machine_model` module

Module description:

class

`elliott.recommender.latent_factor_models.FM.factorization_machine_model.FactorizationMac` (**args*, ***kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs*, *training=None*, *mask=None*)

get_recs (*inputs*, *training=False*, ***kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds*, *train_mask*, *k=100*)

predict (*inputs*, *training=False*, ***kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

Module contents

elliott.recommender.latent_factor_models.FunkSVD package

Submodules

elliott.recommender.latent_factor_models.FunkSVD.funk_svd module

Module description:

class `elliott.recommender.latent_factor_models.FunkSVD.funk_svd.FunkSVD` (*data*, *config*, *params*, **args*, ***kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

predict (*u: int*, *i: int*)

restore_weights ()

train ()

elliott.recommender.latent_factor_models.FunkSVD.funk_svd_model module

Module description:

class

`elliott.recommender.latent_factor_models.FunkSVD.funk_svd_model.FunkSVDModel` (**args*, ***kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs*, *training=None*, *mask=None*)

get_recs (*inputs*, *training=False*, ***kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds*, *train_mask*, *k=100*)

predict (*inputs*, *training=False*, ***kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (batch)

Module contents

elliott.recommender.latent_factor_models.LogisticMF package

Submodules

elliott.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization module

Module description:

class

`elliott.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization.LogisticMF`
 (data, config, params, *args, **kwargs)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (k: int = 100)

property name

predict (u: int, i: int)

restore_weights ()

train ()

elliott.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization_model module

Module description:

class

`elliott.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization_model.LogisticMF`
 (*args, **kwargs)

Bases: `tensorflow.python.keras.engine.training.Model`

call (inputs, training=None, mask=None)

get_top_k (preds, train_mask, k=100)

predict_batch (start, stop, **kwargs)

set_update_user (update_user)

train_step (batch)

Module contents

elliott.recommender.latent_factor_models.MF package

Submodules

elliott.recommender.latent_factor_models.MF.matrix_factorization module

Module description:

class `elliott.recommender.latent_factor_models.MF.matrix_factorization.MF` (data, config, params, *args, **kwargs)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (k: int = 100)

property name

predict (u: int, i: int)

restore_weights ()

train ()

elliott.recommender.latent_factor_models.MF.matrix_factorization_model module

Module description:

class

elliott.recommender.latent_factor_models.MF.matrix_factorization_model.**MatrixFactorizationModel**
(*args, **kwargs)

Bases: *tensorflow.python.keras.engine.training.Model*

call (inputs, training=None, mask=None)

get_recs (inputs, training=False, **kwargs)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (preds, train_mask, k=100)

predict (inputs, training=False, **kwargs)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (batch)

Module contents

elliott.recommender.latent_factor_models.NonNegMF package

Submodules

elliott.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization module

Module description:

class

elliott.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization.**NonNegativeMatrixFactorizationModel**
(data, config, params, *args, **kwargs)

Bases: *elliott.recommender.recommender_utils.mixin.RecMixin*,
elliott.recommender.base_recommender_model.BaseRecommenderModel

get_recommendations (k: int = 100)

property name

predict (u: int, i: int)

Get prediction on the user item pair.

Returns:

A single float value.

restore_weights ()

train ()

elliott.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization_model module

Module description:

class

```
elliot.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization_model
( data, num_users, num_items, global_mean, embed_mf_size, lambda_weights, learning_rate=0.01,
random_seed=42 )
```

Bases: object

get_model_state ()

get_user_recs (user, k=100)

predict (user, item)

set_model_state (saving_dict)

train_step ()

Module contents

Module description:

elliot.recommender.latent_factor_models.PMF package

Submodules

elliot.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization module

Module description:

Mnih, Andriy, and Russ R. Salakhutdinov. "Probabilistic matrix factorization." Advances in neural information processing systems 20 (2007)

class

```
elliot.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization.PMF
( data, config, params, *args, **kwargs )
```

Bases: *elliot.recommender.recommender_utils.mixin.RecMixin*,
elliot.recommender.base_recommender_model.BaseRecommenderModel

get_recommendations (k: int = 100)

property name

predict (u: int, i: int)

restore_weights ()

train ()

elliot.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization_model module

Module description:

Mnih, Andriy, and Russ R. Salakhutdinov. "Probabilistic matrix factorization." Advances in neural information processing systems 20 (2007)

class

```
elliot.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization_model.PMF
( *args, **kwargs )
```

Bases: tensorflow.python.keras.engine.training.Model

call (inputs, training=None, mask=None)

dot_prod (layer_0, layer_1)

get_recs (inputs, training=False, **kwargs)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds, train_mask, k=100*)

predict (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

Module contents

elliot.recommender.latent_factor_models.PureSVD package

Submodules

elliot.recommender.latent_factor_models.PureSVD.pure_svd module

Module description:

class `elliot.recommender.latent_factor_models.PureSVD.pure_svd.PureSVD` (*data, config, params, *args, **kwargs*)

Bases: `elliot.recommender.recommender_utils_mixin.RecMixin`,
`elliot.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

predict (*u: int, i: int*)

Get prediction on the user item pair.

Returns:

A single float vaue.

restore_weights ()

train ()

elliot.recommender.latent_factor_models.PureSVD.pure_svd_model module

Module description:

class

`elliot.recommender.latent_factor_models.PureSVD.pure_svd_model.PureSVDModel`
(*factors, data, random_seed*)

Bases: `object`

Simple Matrix Factorization class

get_model_state ()

get_user_recs (*user, k=100*)

predict (*user, item*)

set_model_state (*saving_dict*)

train_step ()

Module contents

elliot.recommender.latent_factor_models.SVDpp package

Submodules

elliot.recommender.latent_factor_models.SVDpp.svdpp module

Module description:

```
class elliot.recommender.latent_factor_models.SVDpp.svdpp.SVDpp ( data, config,
params, *args, **kwargs )
```

```
    Bases: elliot.recommender.recommender_utils_mixin.RecMixin,
           elliot.recommender.base_recommender_model.BaseRecommenderModel
```

```
    get_recommendations ( k: int = 100 )
```

```
    property name
```

```
    predict ( u: int, i: int )
```

```
    train ( )
```

elliot.recommender.latent_factor_models.SVDpp.svdpp_model module

Module description:

```
class elliot.recommender.latent_factor_models.SVDpp.svdpp_model.SVDppModel (
*args, **kwargs )
```

```
    Bases: tensorflow.python.keras.engine.training.Model
```

```
    call ( inputs, training=None, mask=None )
```

```
    get_recs ( inputs, training=False, **kwargs )
```

```
        Get full predictions on the whole users/items matrix.
```

```
    Returns:
```

```
        The matrix of predicted values.
```

```
    get_top_k ( preds, train_mask, k=100 )
```

```
    predict ( inputs, training=False, **kwargs )
```

```
        Get full predictions on the whole users/items matrix.
```

```
    Returns:
```

```
        The matrix of predicted values.
```

```
    train_step ( batch )
```

Module contents

elliot.recommender.latent_factor_models.Slim package

Submodules

elliot.recommender.latent_factor_models.Slim.slim module

Module description:

```
class elliot.recommender.latent_factor_models.Slim.slim.Slim ( data, config, params,
*args, **kwargs )
```

```
    Bases: elliot.recommender.recommender_utils_mixin.RecMixin,
           elliot.recommender.base_recommender_model.BaseRecommenderModel
```

```
    get_recommendations ( k: int = 100 )
```

```
    property name
```

```
    predict ( u: int, i: int )
```

```
        Get prediction on the user item pair.
```

```
    Returns:
```

```
        A single float vaue.
```

restore_weights ()

train ()

elliott.recommender.latent_factor_models.Slim.slim_model module

Module description:

class `elliott.recommender.latent_factor_models.Slim.slim_model.SlimModel (data, num_users, num_items, l1_ratio, alpha, epochs)`

Bases: `object`

get_model_state ()

get_user_recs (user, k=100)

predict (u, i)

set_model_state (saving_dict)

train (verbose)

Module contents

Module description:

elliott.recommender.latent_factor_models.WRMF package

Submodules

elliott.recommender.latent_factor_models.WRMF.wrmf module

Module description:

class `elliott.recommender.latent_factor_models.WRMF.wrmf.WRMF (data, config, params, *args, **kwargs)`

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin, elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (k: int = 100)

property name

predict (u: int, i: int)

Get prediction on the user item pair.

Returns:

A single float vaue.

restore_weights ()

train ()

elliott.recommender.latent_factor_models.WRMF.wrmf_model module

Module description:

class `elliott.recommender.latent_factor_models.WRMF.wrmf_model.WRMFModel (factors, data, random, alpha, reg)`

Bases: `object`

Simple Matrix Factorization class

get_model_state ()

get_user_recs (user, k=100)

predict (user, item)

set_model_state (*saving_dict*)

train_step ()

Module contents

Module contents

Module description:

elliott.recommender.neural package

Subpackages

elliott.recommender.neural.ConvMF package

Submodules

elliott.recommender.neural.ConvMF.convolutional_matrix_factorization module

Module description:

class

`elliott.recommender.neural.ConvMF.convolutional_matrix_factorization.ConvMF (data, config, params, *args, **kwargs)`

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin, elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

restore_weights ()

train ()

elliott.recommender.neural.ConvMF.convolutional_matrix_factorization_model module

Module description:

class

`elliott.recommender.neural.ConvMF.convolutional_matrix_factorization_model.ConvMatrixFactorizationModel (*args, **kwargs)`

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs, training=False, **kwargs*)

Calls the model on new inputs.

In this case *call* just reapplies all ops in the graph to the new inputs (e.g. build a new computational graph from the provided inputs).

Arguments:

inputs: A tensor or list of tensors. *training*: Boolean or boolean scalar tensor, indicating whether to run

the *Network* in training mode or inference mode.

mask: A mask or list of masks. A mask can be

either a tensor or None (no mask).

Returns:

A tensor if there is a single output, or a list of tensors if there are more than one outputs.

get_recs (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds, train_mask, k=100*)

predict (*inputs*, *training=False*, ***kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

The logic for one training step.

This method can be overridden to support custom training logic. This method is called by *Model.make_train_function*.

This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.

Configuration details for *how* this logic is run (e.g. *tf.function* and *tf.distribute.Strategy* settings), should be left to *Model.make_train_function*, which can also be overridden.

Arguments:

data: A nested structure of `Tensor`s.

Returns:

A *dict* containing values that will be passed to *tf.keras.callbacks.CallbackList.on_train_batch_end*. Typically, the values of the *Model*'s metrics are returned. Example: `{'loss': 0.2, 'accuracy': 0.7}`.

class

`elliott.recommender.neural.ConvMF.convolutional_matrix_factorization_model.Convolutional`
(**args*, ***kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs*, ***kwargs*)

Calls the model on new inputs.

In this case *call* just reapplies all ops in the graph to the new inputs (e.g. build a new computational graph from the provided inputs).

Arguments:

inputs: A tensor or list of tensors. *training*: Boolean or boolean scalar tensor, indicating whether to run

the *Network* in training mode or inference mode.

mask: A mask or list of masks. A mask can be

either a tensor or None (no mask).

Returns:

A tensor if there is a single output, or a list of tensors if there are more than one outputs.

class

`elliott.recommender.neural.ConvMF.convolutional_matrix_factorization_model.MLPComponent`
(**args*, ***kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs*, *training=False*, ***kwargs*)

Calls the model on new inputs.

In this case *call* just reapplies all ops in the graph to the new inputs (e.g. build a new computational graph from the provided inputs).

Arguments:

inputs: A tensor or list of tensors. *training*: Boolean or boolean scalar tensor, indicating whether to run

the *Network* in training mode or inference mode.

mask: A mask or list of masks. A mask can be

either a tensor or None (no mask).

Returns:

A tensor if there is a single output, or a list of tensors if there are more than one outputs.

Module contents

elliott.recommender.neural.ConvNeuMF package

Submodules

elliott.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization module

Module description:

class

`elliott.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization.ConvNeuMF`
(*data, config, params, *args, **kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

restore_weights ()

train ()

elliott.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model module

Module description:

class

`elliott.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model.ConvNeuMF`
(**args, **kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs, training=False, **kwargs*)

get_recs (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds, train_mask, k=100*)

predict (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

class

`elliott.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model.ConvNeuMF`
(**args, **kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs, **kwargs*)

class

`elliott.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model.MLP`
(**args, **kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (inputs, training=False, **kwargs)

Module contents

elliott.recommender.neural.DMF package

Submodules

elliott.recommender.neural.DMF.deep_matrix_factorization module

Module description:

class `elliott.recommender.neural.DMF.deep_matrix_factorization.DMF` (data, config, params, *args, **kwargs)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (k: int = 100)

property name

train ()

elliott.recommender.neural.DMF.deep_matrix_factorization_model module

Module description:

class

`elliott.recommender.neural.DMF.deep_matrix_factorization_model.DeepMatrixFactorizationModel` (*args, **kwargs)

Bases: `tensorflow.python.keras.engine.training.Model`

call (inputs, training=None, mask=None)

cosine (layer_0, layer_1)

dot_prod (layer_0, layer_1)

get_recs (inputs, training=False, **kwargs)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (preds, train_mask, k=100)

predict (inputs, training=False, **kwargs)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (batch)

Module contents

elliott.recommender.neural.DeepFM package

Submodules

elliott.recommender.neural.DeepFM.deep_fm module

Module description:

class `elliott.recommender.neural.DeepFM.deep_fm.DeepFM` (data, config, params, *args, **kwargs)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

predict (*u: int, i: int*)

restore_weights ()

train ()

elliot.recommender.neural.DeepFM.deep_fm_model module

Module description:

class *elliot.recommender.neural.DeepFM.deep_fm_model*.**DeepFMModel** (**args, **kwargs*)

Bases: *tensorflow.python.keras.engine.training.Model*

call (*inputs, training=None, mask=None*)

get_recs (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds, train_mask, k=100*)

predict (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

Module contents

elliot.recommender.neural.GeneralizedMF package

Submodules

elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization module

Module description:

class

elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization.**GMF**
(*data, config, params, *args, **kwargs*)

Bases: *elliot.recommender.recommender_utils_mixin.RecMixin*,
elliot.recommender.base_recommender_model.BaseRecommenderModel

get_recommendations (*k: int = 100*)

property name

predict (*u: int, i: int*)

train ()

elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization_model module

Module description:

class

elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization_model.**GeneralizedMatrixFactorizationModel**
(**args, **kwargs*)

Bases: *tensorflow.python.keras.engine.training.Model*

call (*inputs*, *training=None*, *mask=None*)

get_recs (*inputs*, *training=False*, ***kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds*, *train_mask*, *k=100*)

train_step (*batch*)

Module contents

elliott.recommender.neural.ItemAutoRec package

Submodules

elliott.recommender.neural.ItemAutoRec.itemautorec module

Module description:

class `elliott.recommender.neural.ItemAutoRec.itemautorec.ItemAutoRec` (*data*, *config*, *params*, **args*, ***kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

train ()

elliott.recommender.neural.ItemAutoRec.itemautorec_model module

Module description:

class `elliott.recommender.neural.ItemAutoRec.itemautorec_model.Decoder` (**args*, ***kwargs*)

Bases: `tensorflow.python.keras.engine.base_layer.Layer`

call (*inputs*, ***kwargs*)

class `elliott.recommender.neural.ItemAutoRec.itemautorec_model.Encoder` (**args*, ***kwargs*)

Bases: `tensorflow.python.keras.engine.base_layer.Layer`

call (*inputs*, *training=None*)

class

`elliott.recommender.neural.ItemAutoRec.itemautorec_model.ItemAutoRecModel` (**args*, ***kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs*, *training=None*, ***kwargs*)

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_recs (*inputs*, *training=False*, ***kwargs*)

Get full predictions on the whole users/items matrix. [Is Inverted]

Returns:

The matrix of predicted values.

get_top_k (*preds*, *train_mask*, *k=100*)

predict (*inputs*, *training=False*, ***kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

Module contents

Module description:

elliott.recommender.neural.NAIS package

Submodules

elliott.recommender.neural.NAIS.nais module

Module description:

class *elliott.recommender.neural.NAIS.nais.NAIS* (*data*, *config*, *params*, **args*, ***kwargs*)

Bases: *elliott.recommender.recommender_utils_mixin.RecMixin*,
elliott.recommender.base_recommender_model.BaseRecommenderModel

get_recommendations (*k: int = 100*, *auc_compute: bool = False*)

property name

train ()

elliott.recommender.neural.NAIS.nais_model module

Module description:

class *elliott.recommender.neural.NAIS.nais_model.LatentFactor* (**args*, ***kwargs*)

Bases: *tensorflow.python.keras.layers.embeddings.Embedding*

sensor (*sensor_id*)

class *elliott.recommender.neural.NAIS.nais_model.NAIS_model* (**args*, ***kwargs*)

Bases: *tensorflow.python.keras.engine.training.Model*

attention (*user_history*, *target*)

batch_attention (*user_history*, *target*)

batch_predict (*user_start*, *user_stop*)

batch_softmax (*logits*, *item_num*, *similarity*, *user_bias*, *item_bias*)

call (*inputs*, *training=None*)

create_history_item_matrix ()

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

```
get_positions ( predictions, train_mask, items, inner_test_user_true_mask )
get_top_k ( predictions, train_mask, k=100 )
predict ( user, **kwargs )
softmax ( logits, item_num, similarity, user_bias, item_bias, batch_mask_mat=None )
train_step ( batch )
```

Module contents

elliott.recommender.neural.NFM package

Submodules

elliott.recommender.neural.NFM.neural_fm module

Module description:

```
class elliott.recommender.neural.NFM.neural_fm.NFM ( data, config, params, *args, **kwargs )
    Bases: elliott.recommender.recommender_utils_mixin.RecMixin,
            elliott.recommender.base_recommender_model.BaseRecommenderModel
    get_recommendations ( k: int = 100 )
    property name
    predict ( u: int, i: int )
    restore_weights ( )
    train ( )
```

elliott.recommender.neural.NFM.neural_fm_model module

Module description:

```
class
elliott.recommender.neural.NFM.neural_fm_model.NeuralFactorizationMachineModel
( *args, **kwargs )
    Bases: tensorflow.python.keras.engine.training.Model
    call ( inputs, training=None, mask=None )
    get_recs ( inputs, training=False, **kwargs )
        Get full predictions on the whole users/items matrix.
    Returns:
        The matrix of predicted values.
    get_top_k ( preds, train_mask, k=100 )
    predict ( inputs, training=False, **kwargs )
        Get full predictions on the whole users/items matrix.
    Returns:
        The matrix of predicted values.
```

train_step (*batch*)

Module contents

elliott.recommender.neural.NPR package

Submodules

elliott.recommender.neural.NPR.neural_personalized_ranking module

Module description:

class `elliott.recommender.neural.NPR.neural_personalized_ranking.NPR` (*data, config, params, *args, **kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

train ()

elliott.recommender.neural.NPR.neural_personalized_ranking_model module

Module description:

class

`elliott.recommender.neural.NPR.neural_personalized_ranking_model.NPRModel` (**args, **kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs, training=None, mask=None*)

get_recs (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds, train_mask, k=100*)

predict (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

Module contents

elliott.recommender.neural.NeuMF package

Submodules

elliott.recommender.neural.NeuMF.neural_matrix_factorization module

Module description:

class `elliott.recommender.neural.NeuMF.neural_matrix_factorization.NeuMF` (*data, config, params, *args, **kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

```
train ( )
```

elliott.recommender.neural.NeuMF.neural_matrix_factorization_model module

Module description:

class

`elliott.recommender.neural.NeuMF.neural_matrix_factorization_model.NeuralMatrixFactorizationModel`
 (**args, **kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs, training=None, mask=None*)

get_recs (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds, train_mask, k=100*)

predict (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

Module contents

elliott.recommender.neural.UserAutoRec package

Submodules

elliott.recommender.neural.UserAutoRec.userautorec module

Module description:

class `elliott.recommender.neural.UserAutoRec.userautorec.UserAutoRec` (*data, config, params, *args, **kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

property name

train ()

elliott.recommender.neural.UserAutoRec.userautorec_model module

Module description:

class `elliott.recommender.neural.UserAutoRec.userautorec_model.Decoder` (**args, **kwargs*)

Bases: `tensorflow.python.keras.engine.base_layer.Layer`

call (*inputs, **kwargs*)

class `elliott.recommender.neural.UserAutoRec.userautorec_model.Encoder` (**args, **kwargs*)

Bases: `tensorflow.python.keras.engine.base_layer.Layer`

call (*inputs, training=None*)

class

`elliot.recommender.neural.UserAutoRec.userautorec_model.UserAutoRecModel (`
`*args, **kwargs)`

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs*, *training=None*, ***kwargs*)

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_top_k (*preds*, *train_mask*, *k=100*)

predict (*inputs*, *training=False*, ***kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

Module contents

Module description:

elliot.recommender.neural.WideAndDeep package

Submodules

elliot.recommender.neural.WideAndDeep.wide_and_deep module

Module description:

class `elliot.recommender.neural.WideAndDeep.wide_and_deep.WideAndDeep (` *data*,
`config, params, *args, **kwargs)`

Bases: `elliot.recommender.recommender_utils_mixin.RecMixin`,
`elliot.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

train ()

`elliot.recommender.neural.WideAndDeep.wide_and_deep.build_sparse_features (`
`data)`

elliot.recommender.neural.WideAndDeep.wide_and_deep_model module

Module description:

class

`elliot.recommender.neural.WideAndDeep.wide_and_deep_model.WideAndDeepModel (`
`*args, **kwargs)`

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs*, *training=False*, ***kwargs*)

Calls the model on new inputs.

In this case *call* just reapplies all ops in the graph to the new inputs (e.g. build a new compu-

tational graph from the provided inputs).

Arguments:

inputs: A tensor or list of tensors. **training:** Boolean or boolean scalar tensor, indicating whether to run

the *Network* in training mode or inference mode.

mask: A mask or list of masks. A mask can be

either a tensor or None (no mask).

Returns:

A tensor if there is a single output, or a list of tensors if there are more than one outputs.

get_sparse (*u, i*)

get_top_k (*preds, train_mask, k=100*)

get_user_recs (*user, k=100*)

predict (*user, **kwargs*)

Generates output predictions for the input samples.

Computation is done in batches. This method is designed for performance in large scale inputs. For small amount of inputs that fit in one batch, directly using `__call__` is recommended for faster execution, e.g., `model(x)`, or `model(x, training=False)` if you have layers such as `tf.keras.layers.BatchNormalization` that behaves differently during inference. Also, note the fact that test loss is not affected by regularization layers like noise and dropout.

Arguments:

x: Input samples. It could be:

- A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
- A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
- A `tf.data` dataset.
- A generator or `keras.utils.Sequence` instance.

A more detailed description of unpacking behavior for iterator types (Dataset, generator, Sequence) is given in the *Unpacking behavior for iterator-like inputs* section of *Model.fit*.

batch_size: Integer or None.

Number of samples per batch. If unspecified, *batch_size* will default to 32. Do not specify the *batch_size* if your data is in the form of dataset, generators, or `keras.utils.Sequence` instances (since they generate batches).

verbose: Verbosity mode, 0 or 1. **steps:** Total number of steps (batches of samples)

before declaring the prediction round finished. Ignored with the default value of None. If *x* is a `tf.data` dataset and *steps* is None, *predict* will run until the input dataset is exhausted.

callbacks: List of `keras.callbacks.Callback` instances.

List of callbacks to apply during prediction. See `[callbacks](/api-docs/python/tf/keras/callbacks)`.

max_queue_size: Integer. Used for generator or `keras.utils.Sequence`

input only. Maximum size for the generator queue. If unspecified, *max_queue_size* will default to 10.

workers: Integer. Used for generator or `keras.utils.Sequence` input

only. Maximum number of processes to spin up when using process-based threading. If unspecified, *workers* will default to 1. If 0, will execute the generator on the

main thread.

use_multiprocessing: Boolean. Used for generator or

keras.utils.Sequence input only. If *True*, use process-based threading. If unspecified, *use_multiprocessing* will default to *False*. Note that because this implementation relies on multiprocessing, you should not pass non-picklable arguments to the generator as they can't be passed easily to children processes.

See the discussion of *Unpacking behavior for iterator-like inputs* for *Model.fit*. Note that *Model.predict* uses the same interpretation rules as *Model.fit* and *Model.evaluate*, so inputs must be unambiguous for all three methods.

Returns:

Numpy array(s) of predictions.

Raises:

RuntimeError: If *model.predict* is wrapped in *tf.function*. *ValueError*: In case of mismatch between the provided

input data and the model's expectations, or in case a stateful model receives a number of samples that is not a multiple of the batch size.

train_step (*batch*)

The logic for one training step.

This method can be overridden to support custom training logic. This method is called by *Model.make_train_function*.

This method should contain the mathematical logic for one step of training. This typically includes the forward pass, loss calculation, backpropagation, and metric updates.

Configuration details for *how* this logic is run (e.g. *tf.function* and *tf.distribute.Strategy* settings), should be left to *Model.make_train_function*, which can also be overridden.

Arguments:

data: A nested structure of *Tensor*'s.

Returns:

A *dict* containing values that will be passed to *tf.keras.callbacks.CallbackList.on_train_batch_end*. Typically, the values of the *Model*'s metrics are returned. Example: *{'loss': 0.2, 'accuracy': 0.7}*.

Module contents

Module contents

elliot.recommender.unpersonalized package

Subpackages

elliot.recommender.unpersonalized.most_popular package

Submodules

elliot.recommender.unpersonalized.most_popular.most_popular module

Created on April 4, 2020 Tensorflow 2.1.0 implementation of APR. @author Anonymized

```
class elliot.recommender.unpersonalized.most_popular.most_popular.MostPop (
    data, config, params, *args, **kwargs )
```

```
    Bases: elliot.recommender.recommender_utils_mixin.RecMixin,
           elliot.recommender.base_recommender_model.BaseRecommenderModel
```

```
    get_recommendations ( top_k )
```

```
    property name
```

```
    train ( )
```

Module contents

elliot.recommender.unpersonalized.random_recommender package

Submodules

elliott.recommender.unpersonalized.random_recommender.Random module

Created on April 4, 2020 Tensorflow 2.1.0 implementation of APR. @author Anonymized

```
class elliott.recommender.unpersonalized.random_recommender.Random.Random ( data,  
config, params, *args, **kwargs )
```

```
    Bases: elliott.recommender.recommender_utils_mixin.RecMixin,  
           elliott.recommender.base_recommender_model.BaseRecommenderModel
```

```
    get_recommendations ( top_k )
```

```
    property name
```

```
    train ( )
```

Module contents

Module contents

elliott.recommender.visual_recommenders package

Subpackages

elliott.recommender.visual_recommenders.ACF package

Submodules

elliott.recommender.visual_recommenders.ACF.ACF module

Module description:

```
class elliott.recommender.visual_recommenders.ACF.ACF.ACF ( data, config, params, *args,  
**kwargs )
```

```
    Bases: elliott.recommender.recommender_utils_mixin.RecMixin,  
           elliott.recommender.base_recommender_model.BaseRecommenderModel
```

```
    property name
```

```
    train ( )
```

elliott.recommender.visual_recommenders.ACF.ACF_model module

Module description:

```
class elliott.recommender.visual_recommenders.ACF.ACF_model.ACF_model ( *args,  
**kwargs )
```

```
    Bases: tensorflow.python.keras.engine.training.Model
```

```
    call ( inputs, training=None, mask=None )
```

```
    get_config ( )
```

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

```
    get_top_k ( preds, train_mask, k=100 )
```

```
    predict ( start, stop )
```

```
    train_step ( batch )
```

Module contents

Module description:

elliot.recommender.visual_recommenders.DVBPR package

Submodules

elliot.recommender.visual_recommenders.DVBPR.DVBPR module

Module description:

```
class elliot.recommender.visual_recommenders.DVBPR.DVBPR.DVBPR ( data, config,
*args, **kwargs )
    Bases: elliot.recommender.recommender_utils_mixin.RecMixin,
            elliot.recommender.base_recommender_model.BaseRecommenderModel
    get_recommendations ( k: int = 100 )
    property name
    train ( )
```

elliot.recommender.visual_recommenders.DVBPR.DVBPR_model module

Module description:

```
class elliot.recommender.visual_recommenders.DVBPR.DVBPR_model.DVBPR_model (
*args, **kwargs )
    Bases: tensorflow.python.keras.engine.training.Model
    call ( inputs, training=None, mask=None )
    get_config ( )
    get_top_k ( preds, train_mask, k=100 )
    predict_batch ( start, stop, phi )
    train_step ( batch )
```

elliot.recommender.visual_recommenders.DVBPR.FeatureExtractor module

```
class
elliot.recommender.visual_recommenders.DVBPR.FeatureExtractor.FeatureExtractor
( *args, **kwargs )
    Bases: tensorflow.python.keras.engine.training.Model, abc.ABC
    call ( inputs, training=None, mask=None )
        Calls the model on new inputs.
        In this case call just reapplies all ops in the graph to the new inputs (e.g. build a new computational graph from the provided inputs).
    Arguments:
        inputs: A tensor or list of tensors. training: Boolean or boolean scalar tensor, indicating whether to run
                the Network in training mode or inference mode.
        mask: A mask or list of masks. A mask can be
            either a tensor or None (no mask).
    Returns:
        A tensor if there is a single output, or a list of tensors if there are more than one outputs.
```

Module contents

Module description:

elliot.recommender.visual_recommenders.DeepStyle package

Submodules

elliott.recommender.visual_recommenders.DeepStyle.DeepStyle module

Module description:

```
class elliott.recommender.visual_recommenders.DeepStyle.DeepStyle.DeepStyle (
    data, config, params, *args, **kwargs )
    Bases:
        elliott.recommender.recommender_utils_mixin.RecMixin,
        elliott.recommender.base_recommender_model.BaseRecommenderModel
    get_recommendations ( k: int = 100 )

    property name

    train ( )
```

elliott.recommender.visual_recommenders.DeepStyle.DeepStyle_model module

Module description:

```
class
elliott.recommender.visual_recommenders.DeepStyle.DeepStyle_model.DeepStyle_model
( *args, **kwargs )
    Bases: tensorflow.python.keras.engine.training.Model
    call ( inputs, training=None )

    get_config ( )

    get_top_k ( preds, train_mask, k=100 )

    predict ( start, stop, training=False )

    train_step ( batch )
```

Module contents

Module description:

elliott.recommender.visual_recommenders.VBPR package

Submodules

elliott.recommender.visual_recommenders.VBPR.VBPR module

Module description:

```
class elliott.recommender.visual_recommenders.VBPR.VBPR.VBPR ( data, config, params,
    *args, **kwargs )
    Bases:
        elliott.recommender.recommender_utils_mixin.RecMixin,
        elliott.recommender.base_recommender_model.BaseRecommenderModel
    get_recommendations ( k: int = 100 )

    property name

    train ( )
```

elliott.recommender.visual_recommenders.VBPR.VBPR_model module

Module description:

```
class elliott.recommender.visual_recommenders.VBPR.VBPR_model.VBPR_model ( *args,
    **kwargs )
    Bases: tensorflow.python.keras.engine.training.Model
    call ( inputs, training=None )
```

get_config ()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Returns:

Python dictionary.

get_top_k (*preds, train_mask, k=100*)

predict (*start, stop*)

train_step (*batch*)

Module contents

Module description:

elliott.recommender.visual_recommenders.VNPR package

Submodules

elliott.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking module

Module description:

class

`elliott.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking.VNPR`
(*data, config, params, *args, **kwargs*)

Bases: `elliott.recommender.recommender_utils_mixin.RecMixin`,
`elliott.recommender.base_recommender_model.BaseRecommenderModel`

get_recommendations (*k: int = 100*)

property name

train ()

elliott.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking_model module

Module description:

class

`elliott.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking_model.VNPR`
(**args, **kwargs*)

Bases: `tensorflow.python.keras.engine.training.Model`

call (*inputs, training=None, mask=None*)

get_recs (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

get_top_k (*preds, train_mask, k=100*)

predict (*inputs, training=False, **kwargs*)

Get full predictions on the whole users/items matrix.

Returns:

The matrix of predicted values.

train_step (*batch*)

Module contents

Module contents

Module description:

Submodules

elliott.recommender.base_recommender_model module

Module description:

```
class elliott.recommender.base_recommender_model.BaseRecommenderModel ( data,
config, params, *args, **kwargs )
```

Bases: abc.ABC

autoset_params ()

Define Parameters as tuples: (variable_name, public_name, shortcut, default, reading_function, printing_function) Example:

```
self._params_list = [
    ("_similarity", "similarity", "sim", "cosine", None, None), ("_user_profile_type",
    "user_profile", "up", "tfidf", None, None), ("_item_profile_type", "item_profile", "ip",
    "tfidf", None, None), ("_mlpunits", "mlp_units", "mlpunits", "(1,2,3)", lambda x: list(
    make_tuple(x)), lambda x: str(x).replace(",", "-")),
]
```

abstract get_loss ()

abstract get_params ()

get_params_shortcut ()

abstract get_recommendations (**args*)

abstract get_results ()

abstract train ()

```
elliott.recommender.base_recommender_model.init_charger ( init )
```

elliott.recommender.recommender_utils_mixin module

```
class elliott.recommender.recommender_utils_mixin.RecMixin
```

Bases: object

get_loss ()

get_params ()

get_recommendations (*k: int = 100*)

get_results ()

get_train_mask (*start, stop*)

restore_weights ()

train ()

Module contents

Module description:

1.1.7 `elliott.result_handler` package

Submodules

`elliott.result_handler.result_handler` module

Module description:

```
class elliott.result_handler.result_handler.HyperParameterStudy ( rel_threshold=1 )
    Bases: object
        add_trials ( obj )
        save_trials ( output='../results/' )
        save_trials_as_triplets ( output='../results/' )

class elliott.result_handler.result_handler.ResultHandler ( rel_threshold=1 )
    Bases: object
        add_one_shot_recommender ( **kwargs )
        save_best_models ( output='../results/' )
        save_best_results ( output='../results/' )
        save_best_results_as_triplets ( output='../results/' )
        save_best_statistical_results ( stat_test, output='../results/' )

class elliott.result_handler.result_handler.StatTest ( value )
    Bases: enum.Enum
    An enumeration.

    PairedTTest = [<class 'elliott.evaluation.statistical_significance.PairedTTest'>, 'paired_ttest']

    WilcoxonTest = [<class 'elliott.evaluation.statistical_significance.WilcoxonTest'>, 'wilcoxon_test']
```

Module contents

Module description:

1.1.8 `elliott.splitter` package

Submodules

`elliott.splitter.base_splitter` module

```
class elliott.splitter.base_splitter.Splitter ( data: pandas.core.frame.DataFrame,
splitting_ns: types.SimpleNamespace )
    Bases: object
        fold_list_generator ( length, folds=5 )

        generic_split_function ( data: pandas.core.frame.DataFrame, **kwargs ) →
        List[Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]]
```

```

handle_hierarchy ( data: pandas.core.frame.DataFrame, valtest_splitting_ns: types.SimpleNamespace ) → List[Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]]

process_splitting ( )

read_folder ( folder_path )

rearrange_data ( train_test: List[Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]], train_val: List[List[Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]]] )

splitting_best_timestamp ( d: pandas.core.frame.DataFrame, min_below=1, min_over=1 )

splitting_kfolds ( data: pandas.core.frame.DataFrame, folds=5 )

splitting_passed_timestamp ( d: pandas.core.frame.DataFrame, timestamp=1 )

splitting_randomsubsampling_kfolds ( d: pandas.core.frame.DataFrame, folds=5, ratio=0.2 )

splitting_randomsubsampling_kfolds_leavenout ( d: pandas.core.frame.DataFrame, folds=5, n=1 )

splitting_temporal_holdout ( d: pandas.core.frame.DataFrame, ratio=0.2 )

splitting_temporal_leavenout ( d: pandas.core.frame.DataFrame, n=1 )

store_splitting ( tuple_list )

subsampling_leavenout_list_generator ( length, n=1 )

subsampling_list_generator ( length, ratio=0.2 )

```

Module contents

1.1.9 elliot.utils package

Submodules

elliot.utils.folder module

Module description:

```

elliot.utils.folder.build_log_folder ( path_log_folder )

elliot.utils.folder.build_model_folder ( path_output_rec_weight, model )

elliot.utils.folder.create_folder_by_index ( path, index )

elliot.utils.folder.manage_directories ( path_output_rec_result, path_output_rec_weight, path_output_rec_performance )

```

elliot.utils.logger_util module

```

class elliot.utils.logger_util.QueueListenerHandler ( handlers, respect_handler_level=False, auto_run=True, queue=<queue.Queue object> )
    Bases: logging.handlers.QueueHandler

    emit ( record )
        Emit a record.
        Writes the LogRecord to the queue, preparing it for pickling first.

```

start ()

stop ()

elliott.utils.logging module

class `elliott.utils.logging.TimeFilter (name='')`

Bases: `logging.Filter`

filter (*record: logging.LogRecord*) → bool

Determine if the specified record is to be logged.

Is the specified record to be logged? Returns 0 for no, nonzero for yes. If deemed appropriate, the record may be modified in-place.

`elliott.utils.logging.get_logger (name, log_level=10)`

`elliott.utils.logging.init (path_config, folder_log, log_level=30)`

`elliott.utils.logging.prepare_logger (name, path, log_level=10)`

elliott.utils.read module

Module description:

`elliott.utils.read.find_checkpoint (dir, restore_epochs, epochs, rec, best=0)`

- Parameters**
- **dir** – directory of the model where we start from the reading.
 - **restore_epochs** – epoch from which we start from.
 - **epochs** – epochs from which we restore (0 means that we have best)
 - **rec** – recommender model
 - **best** – 0 No Best - 1 Search for the Best

Returns

`elliott.utils.read.load_obj (name)`

Load the pickle object by name :param name: name of file :return:

`elliott.utils.read.read_config (sections_fields)`

Args:

sections_fields (list): list of fields to retrieve from configuration file

Return:

A list of configuration values.

`elliott.utils.read.read_csv (filename)`

Args:

filename (str): csv file path

Return:

A pandas dataframe.

`elliott.utils.read.read_imagenet_classes_txt (filename)`

Args:

filename (str): txt file path

Return:

A list with 1000 imagenet classes as strings.


```
elliott.utils.read.read_multi_config ( )
```

It reads a config file that contains the configuration parameters for the recommendation systems.

Return:

A list of configuration settings.

```
elliott.utils.read.read_np ( filename )
```

Args:

filename (str): filename of numpy to load

Return:

The loaded numpy.

elliott.utils.write module

Module description:

```
elliott.utils.write.save_np ( npy, filename )
```

Store numpy to memory. Args:

npy: numpy to save filename (str): filename

```
elliott.utils.write.save_obj ( obj, name )
```

Store the object in a pkl file :param obj: python object to be stored :param name: file name (Not insert .pkl) :return:

```
elliott.utils.write.store_recommendation ( recommendations, path='' )
```

Store recommendation list (top-k) :return:

Module contents

Module description:

1.2 Submodules

1.3 elliott.run module

Module description:

```
elliott.run.config_test ( builder, base )
```

```
elliott.run.run_experiment ( config_path: str = './config/config.yml' )
```

1.4 Module contents

Module description:

- genindex
- modindex
- search

e

elliott, 93

elliott.dataset, 9

elliott.dataset.abstract_dataset, 8

elliott.dataset.dataloader, 7

elliott.dataset.dataloader.knowledge_aware_chains, 5

elliott.dataset.dataloader.visual_dataloader, 6

elliott.dataset.dataset, 9

elliott.dataset.samplers, 8

elliott.dataset.samplers.custom_pointwise_sparse_sampler, 7

elliott.dataset.samplers.custom_sampler, 7

elliott.dataset.samplers.custom_sparse_sampler, 7

elliott.dataset.samplers.pairwise_sampler, 7

elliott.dataset.samplers.pipeline_sampler, 7

elliott.dataset.samplers.pointwise_cfgan_sampler, 7

elliott.dataset.samplers.pointwise_pos_neg_ratings_sampler, 8

elliott.dataset.samplers.pointwise_pos_neg_ratio_ratings_sampler, 8

elliott.dataset.samplers.pointwise_pos_neg_sampler, 8

elliott.dataset.samplers.pointwise_wide_and_deep_sampler, 8

elliott.dataset.samplers.sparse_sampler, 8

elliott.evaluation, 31

elliott.evaluation.evaluator, 31

elliott.evaluation.metrics, 30

elliott.evaluation.metrics.accuracy, 15

elliott.evaluation.metrics.accuracy.AUC, 11

elliott.evaluation.metrics.accuracy.AUC.auc, 11

9

elliott.evaluation.metrics.accuracy.AUC.gauc, 10

elliott.evaluation.metrics.accuracy.AUC.lauc, 10

elliott.evaluation.metrics.accuracy.DSC, 11

elliott.evaluation.metrics.accuracy.DSC.dsc, 11

elliott.evaluation.metrics.accuracy.f1, 12

elliott.evaluation.metrics.accuracy.f1.extend, 11

elliott.evaluation.metrics.accuracy.f1.f1, 12

elliott.evaluation.metrics.accuracy.hit_rate, 12

elliott.evaluation.metrics.accuracy.hit_rate, 12

elliott.evaluation.metrics.accuracy.map, 13

elliott.evaluation.metrics.accuracy.map.map, 12

elliott.evaluation.metrics.accuracy.mar, 13

elliott.evaluation.metrics.accuracy.mar.mar, 13

elliott.evaluation.metrics.accuracy.mrr, 13

elliott.evaluation.metrics.accuracy.mrr.mrr, 13

elliott.evaluation.metrics.accuracy.ndcg, 14

elliott.evaluation.metrics.accuracy.ndcg.ndcg, 13

elliott.evaluation.metrics.accuracy.precision, 14

elliott.evaluation.metrics.accuracy.precision, 14

elliott.evaluation.metrics.accuracy.recall, 15

elliott.evaluation.metrics.accuracy.recall.re, 15

elliott.evaluation.metrics.base_metric,	elliott.evaluation.metrics.fairness,
29	26
elliott.evaluation.metrics.bias,	elliott.evaluation.metrics.fairness.BiasDispa
18	23
elliott.evaluation.metrics.bias.aclt,	elliott.evaluation.metrics.fairness.BiasDispa
15	21
elliott.evaluation.metrics.bias.aclt.aclt,	elliott.evaluation.metrics.fairness.BiasDispa
15	22
elliott.evaluation.metrics.bias.aplt,	elliott.evaluation.metrics.fairness.BiasDispa
16	22
elliott.evaluation.metrics.bias.aplt.aplt,	elliott.evaluation.metrics.fairness.MAD,
16	24
elliott.evaluation.metrics.bias.arp,	elliott.evaluation.metrics.fairness.MAD.ItemM
16	23
elliott.evaluation.metrics.bias.arp.arp,	elliott.evaluation.metrics.fairness.MAD.ItemM
16	23
elliott.evaluation.metrics.bias.pop_reo,	elliott.evaluation.metrics.fairness.MAD.UserM
17	23
elliott.evaluation.metrics.bias.pop_reo.extendedpop_reo,	elliott.evaluation.metrics.fairness.MAD.UserM
16	24
elliott.evaluation.metrics.bias.pop_reo.pop_reo,	elliott.evaluation.metrics.fairness.reo,
17	25
elliott.evaluation.metrics.bias.pop_rsp,	elliott.evaluation.metrics.fairness.reo.reo,
18	24
elliott.evaluation.metrics.bias.pop_rsp.extendedpop_rsp,	elliott.evaluation.metrics.fairness.rsp,
17	26
elliott.evaluation.metrics.bias.pop_rsp.pop_rsp,	elliott.evaluation.metrics.fairness.rsp.rsp,
18	25
elliott.evaluation.metrics.coverage,	elliott.evaluation.metrics.metrics_utils,
20	29
elliott.evaluation.metrics.coverage.item_coverage,	elliott.evaluation.metrics.novelty,
19	27
elliott.evaluation.metrics.coverage.item_coverage.item_coverage,	elliott.evaluation.metrics.novelty.EFD,
18	26
elliott.evaluation.metrics.coverage.num_retrieved,	elliott.evaluation.metrics.novelty.EFD.efd,
19	26
elliott.evaluation.metrics.coverage.num_retrieved.num_retrieved,	elliott.evaluation.metrics.novelty.EFD.extend
19	26
elliott.evaluation.metrics.coverage.user_coverage,	elliott.evaluation.metrics.novelty.EPC,
20	27
elliott.evaluation.metrics.coverage.user_coverage.user_coverage,	elliott.evaluation.metrics.novelty.EPC.epc,
19	27
elliott.evaluation.metrics.coverage.user_coverage.user_coverage.as_novelty.EPC.extend	
19	27
elliott.evaluation.metrics.diversity,	elliott.evaluation.metrics.rating,
21	29
elliott.evaluation.metrics.diversity.gini_index,	elliott.evaluation.metrics.rating.mae,
21	28
elliott.evaluation.metrics.diversity.gini_index.gini_index,	elliott.evaluation.metrics.rating.mae.mae,
20	27
elliott.evaluation.metrics.diversity.shannon_index,	elliott.evaluation.metrics.rating.mse,
21	28
elliott.evaluation.metrics.diversity.shannon_index.shannon_index,	elliott.evaluation.metrics.rating.mse.mse,
21	28
elliott.evaluation.metrics.diversity.SRecall,	elliott.evaluation.metrics.rating.rmse,
20	29
elliott.evaluation.metrics.diversity.SRecall.SRecall,	elliott.evaluation.metrics.rating.rmse.rmse,
20	28

[elliot.evaluation.metrics.statistical_array_metric](#), 45
[elliot.evaluation.popularity_utils](#), 30
[elliot.evaluation.popularity_utils.popularity](#), 30
[elliot.evaluation.relevance](#), 31
[elliot.evaluation.relevance.relevance](#), 30
[elliot.evaluation.statistical_significance](#), 31
[elliot.hyperoptimization](#), 32
[elliot.hyperoptimization.model_coordinator](#), 32
[elliot.namespace](#), 33
[elliot.namespace.namespace_model](#), 32
[elliot.namespace.namespace_model_builder](#), 32
[elliot.prefiltering](#), 33
[elliot.prefiltering.standard_prefilters](#), 33
[elliot.recommender](#), 90
[elliot.recommender.adversarial](#), 43
[elliot.recommender.adversarial.AMF](#), 41
[elliot.recommender.adversarial.AMF.AMF](#), 38
[elliot.recommender.adversarial.AMF.AMF_model](#), 39
[elliot.recommender.adversarial.AMR](#), 43
[elliot.recommender.adversarial.AMR.AMR](#), 41
[elliot.recommender.adversarial.AMR.AMR_model](#), 41
[elliot.recommender.algebraic](#), 44
[elliot.recommender.algebraic.slope_one](#), 43
[elliot.recommender.algebraic.slope_one.slope_one](#), 43
[elliot.recommender.algebraic.slope_one.slope_one_model](#), 43
[elliot.recommender.attentive](#), 44
[elliot.recommender.attentive.afm](#), 44
[elliot.recommender.attentive.afm.afm](#), 44
[elliot.recommender.attentive.afm.afm_model](#), 44
[elliot.recommender.autoencoders](#), 46
[elliot.recommender.autoencoders.dae](#), 45
[elliot.recommender.autoencoders.dae.multi_dae](#), 44
[elliot.recommender.autoencoders.dae.multi_dae_model](#), 44
[elliot.recommender.autoencoders.vae](#), 46
[elliot.recommender.autoencoders.vae.multi_vae](#), 46
[elliot.recommender.autoencoders.vae.multi_vae_model](#), 46
[elliot.recommender.base_recommender_model](#), 89
[elliot.recommender.content_based](#), 47
[elliot.recommender.content_based.VSM](#), 47
[elliot.recommender.content_based.VSM.tfidf_utils](#), 47
[elliot.recommender.content_based.VSM.vectorizer](#), 47
[elliot.recommender.gan](#), 54
[elliot.recommender.gan.CFGAN](#), 50
[elliot.recommender.gan.CFGAN.cfgan](#), 48
[elliot.recommender.gan.CFGAN.cfgan_model](#), 48
[elliot.recommender.gan.IRGAN](#), 54
[elliot.recommender.gan.IRGAN.irgan](#), 50
[elliot.recommender.gan.IRGAN.irgan_model](#), 50
[elliot.recommender.graph_based](#), 56
[elliot.recommender.graph_based.lightgcn](#), 55
[elliot.recommender.graph_based.lightgcn.LightGCN](#), 54
[elliot.recommender.graph_based.lightgcn.LightGCN_model](#), 54
[elliot.recommender.graph_based.ngcf](#), 55
[elliot.recommender.graph_based.ngcf.NGCF](#), 55
[elliot.recommender.graph_based.ngcf.NGCF_model](#), 55
[elliot.recommender.knowledge_aware](#), 59
[elliot.recommender.knowledge_aware.kaHFM](#), 57
[elliot.recommender.knowledge_aware.kaHFM.kaHFM](#), 56
[elliot.recommender.knowledge_aware.kaHFM.tfidf_utils](#), 57
[elliot.recommender.knowledge_aware.kaHFM_batch](#), 58
[elliot.recommender.knowledge_aware.kaHFM_batch_model](#), 57

74
 elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model,
 74
 elliot.recommender.neural.DeepFM, elliot.recommender.NN.attribute_item_knn,
 76 34
 elliot.recommender.neural.DeepFM.deep_fm, elliot.recommender.NN.attribute_item_knn.att
 75 34
 elliot.recommender.neural.DeepFM.deep_fm_model, elliot.recommender.NN.attribute_item_knn.att
 76 34
 elliot.recommender.neural.DMF, 75 elliot.recommender.NN.attribute_user_knn,
 elliot.recommender.neural.DMF.deep_matrix_factorization, 85
 75 elliot.recommender.NN.attribute_user_knn.att
 elliot.recommender.neural.DMF.deep_matrix_factorization_model, 85
 75 elliot.recommender.NN.attribute_user_knn.att
 elliot.recommender.neural.GeneralizedMF, 35
 77 elliot.recommender.NN.attribute_user_knn.tfi
 elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization, 35
 76 elliot.recommender.NN.item_knn,
 elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization_model, 37
 76 elliot.recommender.NN.item_knn.aiolli_ferrari
 elliot.recommender.neural.ItemAutoRec, 36
 78 elliot.recommender.NN.item_knn.item_knn,
 elliot.recommender.neural.ItemAutoRec.itemauto_rec, 36
 77 elliot.recommender.NN.item_knn.item_knn_simi
 elliot.recommender.neural.ItemAutoRec.itemauto_rec_model, 36
 77 elliot.recommender.NN.user_knn,
 elliot.recommender.neural.NAIS, 38
 79 elliot.recommender.NN.user_knn.aiolli_ferrari
 elliot.recommender.neural.NAIS.nais, 37
 78 elliot.recommender.NN.user_knn.user_knn,
 elliot.recommender.neural.NAIS.nais_model, 38
 78 elliot.recommender.NN.user_knn.user_knn_simi
 elliot.recommender.neural.NeuMF, 38
 81 elliot.recommender.recommender_utils.mixin,
 elliot.recommender.neural.NeuMF.neural_matrix_factorization, 80
 80 elliot.recommender.unpersonalized,
 elliot.recommender.neural.NeuMF.neural_matrix_factorization_model, 85
 81 elliot.recommender.unpersonalized.most_popul
 elliot.recommender.neural.NFM, 80 84
 elliot.recommender.neural.NFM.neural_fm, elliot.recommender.unpersonalized.most_popul
 79 84
 elliot.recommender.neural.NFM.neural_fm_model, elliot.recommender.unpersonalized.random_rec
 79 85
 elliot.recommender.neural.NPR, 80 elliot.recommender.unpersonalized.random_rec
 elliot.recommender.neural.NPR.neural_personalized_ranking, 85
 80 elliot.recommender.visual_recommenders,
 elliot.recommender.neural.NPR.neural_personalized_ranking_model, 89
 80 elliot.recommender.visual_recommenders.ACF,
 elliot.recommender.neural.UserAutoRec, 85
 82 elliot.recommender.visual_recommenders.ACF.A
 elliot.recommender.neural.UserAutoRec.userauto_rec, 85
 81 elliot.recommender.visual_recommenders.ACF.A
 elliot.recommender.neural.UserAutoRec.userauto_rec_model, 85
 81 elliot.recommender.visual_recommenders.DeepS
 elliot.recommender.neural.WideAndDeep, 87
 84 elliot.recommender.visual_recommenders.DeepS
 elliot.recommender.neural.WideAndDeep.wide_and_deep, 87
 82 elliot.recommender.visual_recommenders.DeepS

87
elliot.recommender.visual_recommenders.DVBPR,
86
elliot.recommender.visual_recommenders.DVBPR.DVBPR,
86
elliot.recommender.visual_recommenders.DVBPR.DVBPR_model,
86
elliot.recommender.visual_recommenders.DVBPR.FeatureExtractor,
86
elliot.recommender.visual_recommenders.VBPR,
88
elliot.recommender.visual_recommenders.VBPR.VBPR,
87
elliot.recommender.visual_recommenders.VBPR.VBPR_model,
87
elliot.recommender.visual_recommenders.VNPR,
89
elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking,
88
elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking_model,
88
elliot.result_handler, 90
elliot.result_handler.result_handler,
90
elliot.run, 93
elliot.splitter, 91
elliot.splitter.base_splitter, 90
elliot.utils, 93
elliot.utils.folder, 91
elliot.utils.logger_util, 91
elliot.utils.logging, 92
elliot.utils.read, 92
elliot.utils.write, 93

A

- AbstractDataset (class in `elliott.dataset.abstract_dataset`), 8
- AbstractRelevanceSingleton (class in `elliott.evaluation.relevance.relevance`), 30
- ACF (class in `elliott.recommender.visual_recommenders.ACF.ACF`), 85
- ACF_model (class in `elliott.recommender.visual_recommenders.ACF.ACF_model`), 85
- ACLT (class in `elliott.evaluation.metrics.bias.aclt.aclt`), 15
- `add_oneshot_recommender()` (`elliott.result_handler.result_handler.ResultHandler` method), 90
- `add_trials()` (`elliott.result_handler.result_handler.HyperParameterStudy` method), 90
- AFM (class in `elliott.recommender.attentive.afm.afm`), 44
- AFMModel (class in `elliott.recommender.attentive.afm.afm_model`), 44
- AiolliSimilarity (class in `elliott.recommender.NN.item_knn.aiolli_ferrari`), 36
- AiolliSimilarity (class in `elliott.recommender.NN.user_knn.aiolli_ferrari`), 37
- AMF (class in `elliott.recommender.adversarial.AMF.AMF`), 38
- AMF_model (class in `elliott.recommender.adversarial.AMF.AMF_model`), 39
- AMR (class in `elliott.recommender.adversarial.AMR.AMR`), 41
- AMR_model (class in `elliott.recommender.adversarial.AMR.AMR_model`), 41
- APLT (class in `elliott.evaluation.metrics.bias.aplt.aplt`), 16
- `applyAdjustedCosine()` (`elliott.recommender.NN.item_knn.aiolli_ferrari.Compute_Similarity` method), 36
- `applyAdjustedCosine()` (`elliott.recommender.NN.user_knn.aiolli_ferrari.Compute_Similarity` method), 37
- `applyPearsonCorrelation()` (`elliott.recommender.NN.item_knn.aiolli_ferrari.Compute_Similarity` method), 36
- `applyPearsonCorrelation()` (`elliott.recommender.NN.user_knn.aiolli_ferrari.Compute_Similarity` method), 37
- ARP (class in `elliott.evaluation.metrics.bias.arp.arp`), 16
- `attention()` (`elliott.recommender.neural.NAIS.-nais_model.NAIS_model` method), 78
- AttributeItemKNN (class in `elliott.recommender.NN.attribute_item_knn.attribute_item_knn`), 34
- AttributeUserKNN (class in `elliott.recommender.NN.attribute_user_knn.attribute_user_knn`), 35
- AUC (class in `elliott.evaluation.metrics.accuracy.AUC.auc`), 10
- `autoset_params()` (`elliott.recommender.base_recommender_model.BaseRecommender` method), 89

B

- `base()` (`elliott.namespace.namespace_model_builder.Builder` property), 32
- `base()` (`elliott.namespace.namespace_model_builder.NameSpaceBuilder` property), 33
- BaseMetric (class in `elliott.evaluation.metrics.base_metric`), 29
- BaseRecommenderModel (class in `elliott.recommender.base_recommender_model`), 89
- `batch_attention()` (`elliott.recommender.neural.-NAIS.nais_model.NAIS_model` method), 78
- `batch_predict()` (`elliott.recommender.latent_factor_models.FISM.FISM_model.FISM_model` method), 64
- `batch_predict()` (`elliott.recommender.neural.-NAIS.nais_model.NAIS_model` method), 78

batch_softmax() (elliot.recommender.neural.-
 NAIS.nais_model.NAIS_model
 method), 78
 BiasDisparityBD (class in elliot.evaluation.met-
 rics.fairness.BiasDisparity.BiasDisparityBD),
 21
 BiasDisparityBR (class in elliot.evaluation.met-
 rics.fairness.BiasDisparity.BiasDisparityBR),
 22
 BiasDisparityBS (class in elliot.evaluation.met-
 rics.fairness.BiasDisparity.BiasDisparityBS),
 22
 binary_relevance() (elliot.evaluation.rele-
 vance.relevance.Relevance property),
 31
 BinaryRelevance (class in elliot.evaluation.rele-
 vance.relevance), 30
 BPRMF (class in elliot.recommender.latent_fac-
 tor_models.BPRMF.BPRMF), 59
 BPRMF_batch (class in elliot.recommender.la-
 tent_factor_models.BPRMF_batch.BPRMF_batch),
 60
 BPRMF_batch_model (class in elliot.recom-
 mender.latent_factor_models.BPRMF_batch.BPRMF_batch_model),
 60
 BPRSlim (class in elliot.recommender.latent_-
 factor_models.BPRSlim.bprslim), 61
 BPRSlimModel (class in elliot.recommender.la-
 tent_factor_models.BPRSlim.bprslim_model),
 61
 build_dict() (elliot.dataset.abstract_dataset.Ab-
 stractDataset method), 8
 build_dict() (elliot.dataset.dataloader.knowl-
 edge_aware_chains.KnowledgeChainsDataObject
 method), 5
 build_dict() (elliot.dataset.dataloader.visual_-
 dataloader.VisualDataObject method),
 6
 build_dict() (elliot.dataset.dataset.DataSet
 method), 9
 build_feature_sparse() (elliot.recommender.-
 content_based.VSM.vector_space_model.VSM
 method), 47
 build_feature_sparse() (elliot.recommender.N-
 N.attribute_item_knn.attribute_item_knn.AttributeItemKNN
 method), 34
 build_feature_sparse() (elliot.recommender.N-
 N.attribute_user_knn.attribute_user_knn.AttributeUserKNN
 method), 35
 build_feature_sparse_values() (elliot.recom-
 mender.content_based.VSM.vector_space_model.VSM
 method), 47
 build_feature_sparse_values() (elliot.recom-
 mender.NN.attribute_user_knn.attribute_user_knn.AttributeUserKNN
 method), 35
 build_log_folder() (in module elliot.utils.fold-
 er), 91
 build_model_folder() (in module elliot.utils.-
 folder), 91
 build_perturbation() (elliot.recommender.ad-
 versarial.AMF.AMF_model.AMF_model
 method), 39
 build_perturbation() (elliot.recommender.ad-
 versarial.AMR.AMR_model.AMR_model
 method), 41
 build_sparse() (elliot.dataset.abstract_-
 dataset.AbstractDataset method), 8
 build_sparse() (elliot.dataset.dataloader.knowl-
 edge_aware_chains.KnowledgeChainsDataObject
 method), 5
 build_sparse() (elliot.dataset.dataloader.visu-
 al_dataloader.VisualDataObject
 method), 6
 build_sparse() (elliot.dataset.dataset.DataSet
 method), 9
 build_sparse_features() (in module elliot.rec-
 ommender.neural.WideAndDeep.wide_and_deep),
 82
 build_sparse_ratings() (elliot.dataset.dataload-
 er.knowledge_aware_chains.KnowledgeChainsDataObj
 method), 5
 build_sparse_ratings() (elliot.dataset.dataload-
 er.visual_dataloader.VisualDataObject
 method), 6
 build_sparse_ratings() (elliot.dataset.dataset.-
 DataSet method), 9
 Builder (class in elliot.namespace.namespace_-
 model_builder), 32

C

call() (elliot.recommender.adversarial.AM-
 F.AMF_model.AMF_model method),
 39
 call() (elliot.recommender.adversarial.AM-
 R.AMR_model.AMR_model method),
 41
 call() (elliot.recommender.attentive.afm.afm_-
 model.AFMMModel method), 44
 call() (elliot.recommender.autoencoders.dae.-
 multi_dae_model.Decoder method),
 45
 call() (elliot.recommender.autoencoders.dae.-
 multi_dae_model.DenoisingAutoEncoder
 method), 45
 call() (elliot.recommender.autoencoders.dae.-
 multi_dae_model.Encoder method),
 45
 call() (elliot.recommender.autoencoders.vae.-
 multi_vae_model.Decoder method),
 46
 call() (elliot.recommender.autoencoders.vae.-
 multi_vae_model.Encoder method),
 46
 call() (elliot.recommender.autoencoders.vae.-
 multi_vae_model.Sampling method),

- 46
- call() (elliot.recommender.autoencoders.vae.-multi_vae_model.VariationalAutoEncoder method), 46
- call() (elliot.recommender.gan.IRGAN.irgan_model.Discriminator method), 51
- call() (elliot.recommender.gan.IRGAN.irgan_model.Generator method), 51
- call() (elliot.recommender.gan.IRGAN.irgan_model.IRGAN_model method), 52
- call() (elliot.recommender.graph_based.lightgc_n.LightGCN_model.LightGCNModel method), 54
- call() (elliot.recommender.-graph_based.ngcf.NGCF_model.NGCFModel method), 55
- call() (elliot.recommender.knowledge_aware.kaHFM_batch.kahfm_batch_model.KaHFMBatchModel method), 57
- call() (elliot.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings_model.KaHFMEmbeddingsModel method), 58
- call() (elliot.recommender.latent_factor_model-s.BPRMF_batch.BPRMF_batch_model.BPRMFBatchModel method), 60
- call() (elliot.recommender.latent_factor_model-s.CML.CML_model.CML_model method), 62
- call() (elliot.recommender.latent_factor_model-s.FFM.field_aware_factorization_machine_model.FieldAwareFactorizationMachineModel method), 63
- call() (elliot.recommender.latent_factor_model-s.FISM.FISM_model.FISM_model method), 64
- call() (elliot.recommender.latent_factor_model-s.FM.factorization_machine_model.FactorizationMachineModel method), 65
- call() (elliot.recommender.latent_factor_model-s.FunkSVD.funk_svd_model.FunkSVDModel method), 65
- call() (elliot.recommender.latent_factor_model-s.LogisticMF.logistic_matrix_factorization_model.LogisticMatrixFactorizationModel method), 66
- call() (elliot.recommender.latent_factor_model-s.MF.matrix_factorization_model.MatrixFactorizationModel method), 67
- call() (elliot.recommender.latent_factor_model-s.PMF.probabilistic_matrix_factorization_model.ProbabilisticMatrixFactorizationModel method), 68
- call() (elliot.recommender.latent_factor_model-s.SVDpp.svdpp_model.SVDppModel method), 70
- call() (elliot.recommender.neural.ConvMF.convolutional_matrix_factorization_model.ConvolutionalMatrixFactorizationModel method), 72
- call() (elliot.recommender.neural.ConvMF.convolutional_matrix_factorization_model.ConvolutionalMatrixFactorizationModel method), 73
- call() (elliot.recommender.neural.ConvMF.convolutional_matrix_factorization_model.MLPComponent method), 73
- call() (elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model.ConvolutionalNeuMF method), 74
- call() (elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model.ConvolutionalNeuMF method), 74
- call() (elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model.MLPComponent method), 75
- call() (elliot.recommender.neural.DeepFM.deep_fm_model.DeepFMModel method), 76
- call() (elliot.recommender.neural.DMF.deep_matrix_factorization_model.DeepMatrixFactorizationModel method), 75
- call() (elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization_model.GeneralizedMF method), 75
- call() (elliot.recommender.neural.ItemAutoRec.itemautorec_model.Decoder method), 77
- call() (elliot.recommender.neural.ItemAutoRec.itemautorec_model.Encoder method), 77
- call() (elliot.recommender.neural.ItemAutoRec.itemautorec_model.ItemAutoRecModel method), 77
- call() (elliot.recommender.neural.NAIS.nais_model.NAIS_model method), 78
- call() (elliot.recommender.neural.NeuMF.neural_matrix_factorization_model.NeuralMatrixFactorizationModel method), 81
- call() (elliot.recommender.neural.NFM.neural_fm_model.NeuralFactorizationMachineModel method), 79
- call() (elliot.recommender.neural.NPR.neural_personalized_ranking_model.NPRModel method), 80
- call() (elliot.recommender.neural.ProbabilisticMatrixFactorizationModel.AutoRec.userautorec_model.Decoder method), 81
- call() (elliot.recommender.neural.ProbabilisticMatrixFactorizationModel.AutoRec.userautorec_model.Encoder method), 81
- call() (elliot.recommender.neural.ProbabilisticMatrixFactorizationModel.AutoRec.userautorec_model.UserAutoRecModel method), 82
- call() (elliot.recommender.neural.WideAndDeep.wide_and_deep_model.WideAndDeepModel method), 82
- call() (elliot.recommender.visual_recommenders.ACF.ACF_model.ACF_model method), 85
- call() (elliot.recommender.visual_recommenders.DeepStyle.DeepStyle_model.DeepStyle_model method), 85

- method), 87
- call() (elliot.recommender.visual_recommenders.DVBPR.DVBPR_model.DVBPR_model.method), 86
- call() (elliot.recommender.visual_recommenders.DVBPR.FeatureExtractor.FeatureExtractor.method), 86
- call() (elliot.recommender.visual_recommenders.VBPR.VBPR_model.VBPR_model.method), 87
- call() (elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking_model.VNPR_model.method), 88
- censor() (elliot.recommender.latent_factor_models.CML.CML_model.LatentFactor.method), 62
- censor() (elliot.recommender.latent_factor_models.FISM.FISM_model.LatentFactor.method), 64
- censor() (elliot.recommender.neural.NAIS.-nais_model.LatentFactor.method), 78
- CFGAN (class in elliot.recommender.gan.CFGAN.cfgan), 48
- CFGAN_model (class in elliot.recommender.-gan.CFGAN.cfgan_model), 48
- check_matrix() (in module elliot.recommender.NN.item_knn.aiolli_ferrari), 36
- check_matrix() (in module elliot.recommender.NN.user_knn.aiolli_ferrari), 37
- check_required_attributes() (elliot.dataset.abstract_dataset.ForceRequiredAttributeDefinitionMeta.method), 9
- check_timestamp() (elliot.dataset.dataloader.-knowledge_aware_chains.KnowledgeChainsLoader.method), 5
- check_timestamp() (elliot.dataset.dataloader.visual_dataloader.VisualLoader.method), 6
- check_timestamp() (elliot.dataset.dataset.-DataSetLoader.method), 9
- CML (class in elliot.recommender.latent_factor_models.CML.CML), 62
- CML_model (class in elliot.recommender.latent_factor_models.CML.CML_model), 62
- common_users() (elliot.evaluation.statistical_significance.PairedTTest.static.method), 31
- common_users() (elliot.evaluation.statistical_significance.WilcoxonTest.static.method), 31
- compare() (elliot.evaluation.statistical_significance.PairedTTest.static.method), 31
- compare() (elliot.evaluation.statistical_significance.WilcoxonTest.static.method), 31
- compute_binary_profile() (elliot.recommender.content_based.VSM.vector_space_model.VSM.method), 47
- compute_binary_profile() (elliot.recommender.NN.attribute_user_knn.attribute_user_knn.AttributeUserKnn.method), 35
- compute_cosine() (elliot.recommender.NN.attribute_item_knn.attribute_item_knn_similarity.Similarity.method), 34
- compute_cosine() (elliot.recommender.NN.item_knn.item_knn_similarity.Similarity.method), 37
- compute_idcg() (elliot.evaluation.metrics.fairness.MAD.UserMADranking.UserMADranking.method), 24
- compute_neighbors() (elliot.recommender.NN.attribute_item_knn.attribute_item_knn_similarity.Similarity.method), 34
- compute_neighbors() (elliot.recommender.NN.attribute_user_knn.attribute_user_knn_similarity.Similarity.method), 35
- compute_neighbors() (elliot.recommender.NN.item_knn.item_knn_similarity.Similarity.method), 37
- compute_neighbors() (elliot.recommender.NN.user_knn.user_knn_similarity.Similarity.method), 38
- Compute_Similarity (class in elliot.recommender.NN.item_knn.aiolli_ferrari), 36
- Compute_Similarity (class in elliot.recommender.NN.user_knn.aiolli_ferrari), 37
- compute_similarity() (elliot.recommender.NN.item_knn.aiolli_ferrari.Compute_Similarity.method), 36
- compute_similarity() (elliot.recommender.NN.user_knn.aiolli_ferrari.Compute_Similarity.method), 37
- compute_user_ndcg() (elliot.evaluation.metrics.accuracy.ndcg.ndcg.NDCG.method), 14
- compute_user_ndcg() (elliot.evaluation.metrics.fairness.MAD.UserMADranking.UserMADranking.method), 24
- config_test() (in module elliot.run), 93
- ConvMatrixFactorizationModel (class in elliot.recommender.neural.ConvMF.convolutional_matrix_factorization), 72
- ConvMF (class in elliot.recommender.neural.-ConvMF.convolutional_matrix_factorization), 72
- ConvNeuMF (class in elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization), 74
- ConvNeuralMatrixFactorizationModel (class in elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization), 74

- 74
- ConvolutionalComponent (class in `elliott.recommender.neural.ConvMF.convolutional_matrix_factorization_model`), 73
- ConvolutionalComponent (class in `elliott.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model`), 74
- cosine() (`elliott.recommender.neural.DMF.deep_matrix_factorization_model.DeepMatrixFactorizationModel` method), 75
- create_folder_by_index() (in module `elliott.utils.folder`), 91
- create_history_item_matrix() (`elliott.recommender.latent_factor_models.FISM.FISM_model.FISM_model.deep_matrix_factorization` method), 64
- create_history_item_matrix() (`elliott.recommender.neural.NAIS.nais_model.NAIS_model` method), 78
- ## D
- dataframe_to_dict() (`elliott.dataset.data_loader.knowledge_aware_chains.KnowledgeChainsDataObject` method), 5
- dataframe_to_dict() (`elliott.dataset.data_loader.visual_data_loader.VisualDataObject` method), 6
- dataframe_to_dict() (`elliott.dataset.dataset.DataSet` method), 9
- DataSet (class in `elliott.dataset.dataset`), 9
- DataSetLoader (class in `elliott.dataset.dataset`), 9
- Decoder (class in `elliott.recommender.autoencoders.dae.multi_dae_model`), 45
- Decoder (class in `elliott.recommender.autoencoders.vae.multi_vae_model`), 46
- Decoder (class in `elliott.recommender.neural.ItemAutoRec.itemautorec_model`), 77
- Decoder (class in `elliott.recommender.neural.UserAutoRec.userautorec_model`), 81
- DeepFM (class in `elliott.recommender.neural.DeepFM.deep_fm`), 75
- DeepFMModel (class in `elliott.recommender.neural.DeepFM.deep_fm_model`), 76
- DeepMatrixFactorizationModel (class in `elliott.recommender.neural.DMF.deep_matrix_factorization_model`), 75
- DeepStyle (class in `elliott.recommender.visual_recommenders.DeepStyle.DeepStyle`), 87
- DeepStyle_model (class in `elliott.recommender.visual_recommenders.DeepStyle.DeepStyle_model`), 87
- DenosingAutoEncoder (class in `elliott.recommender.autoencoders.dae.multi_dae_model`), 45
- discounted_relevance() (`elliott.evaluation.relevance.relevance.Relevance` property), 31
- DiscountedRelevance (class in `elliott.evaluation.relevance.relevance`), 31
- discriminate_factorization() (`elliott.recommender.gan.CFGAN.cfgan_model.Discriminator` method), 49
- DiscriminatorModel (class in `elliott.recommender.gan.CFGAN.cfgan_model`), 49
- Discriminator (class in `elliott.recommender.gan.IRGAN.irgan_model`), 51
- DMF (class in `elliott.recommender.neural.DMF.deep_matrix_factorization`), 75
- dot_prod() (`elliott.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization_model` method), 68
- dot_prod() (`elliott.recommender.neural.DMF.deep_matrix_factorization_model.DeepMatrixFactorizationModel` method), 75
- DSC (class in `elliott.evaluation.metrics.accuracy.DSC.dsc`), 11
- DVBPR (class in `elliott.recommender.visual_recommenders.DVBPR.DVBPR`), 86
- DVBPR_model (class in `elliott.recommender.visual_recommenders.DVBPR.DVBPR_model`), 86
- ## E
- EFD (class in `elliott.evaluation.metrics.novelty.EFD.efd`), 26
- elliott
- module, 93
- elliott.dataset
- module, 9
- elliott.dataset.abstract_dataset
- module, 8
- elliott.dataset.data_loader
- module, 7
- elliott.dataset.data_loader.knowledge_aware_chains
- module, 5
- elliott.dataset.data_loader.visual_data_loader
- module, 6
- elliott.dataset.dataset
- module, 9
- elliott.dataset.samplers
- module, 8
- elliott.dataset.samplers.custom_pointwise_sparse_sampler
- module, 7
- elliott.dataset.samplers.custom_sampler
- module, 7
- elliott.dataset.samplers.custom_sparse_sampler
- module, 7
- elliott.dataset.samplers.pairwise_sampler
- module, 7
- elliott.dataset.samplers.pipeline_sampler
- module, 7

elliott.evaluation.metrics.diversity.shannon_entropy	elliott.evaluation.metrics.rating.mse
module, 21	module, 28
elliott.evaluation.metrics.diversity.shannon_entropy.shannon_entropy	elliott.evaluation.metrics.rating.mse.mse
module, 21	module, 28
elliott.evaluation.metrics.diversity.SRecall	elliott.evaluation.metrics.rating.rmse
module, 20	module, 29
elliott.evaluation.metrics.diversity.SRecall.srecall	elliott.evaluation.metrics.rating.rmse.rmse
module, 20	module, 28
elliott.evaluation.metrics.fairness	elliott.evaluation.metrics.statistical_array_metric
module, 26	module, 29
elliott.evaluation.metrics.fairness.BiasDisparity	elliott.evaluation.popularity_utils
module, 23	module, 30
elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparity	elliott.evaluation.popularity_utils.popularity
module, 21	module, 30
elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparity	elliott.evaluation.relevance
module, 22	module, 31
elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparity	elliott.evaluation.relevance.relevance
module, 22	module, 30
elliott.evaluation.metrics.fairness.MAD	elliott.evaluation.statistical_significance
module, 24	module, 31
elliott.evaluation.metrics.fairness.MAD.ItemMADRanking	elliott.hyperoptimization
module, 23	module, 32
elliott.evaluation.metrics.fairness.MAD.ItemMADRating	elliott.hyperoptimization.model_coordinator
module, 23	module, 32
elliott.evaluation.metrics.fairness.MAD.UserMADRanking	elliott.namespace
module, 23	module, 33
elliott.evaluation.metrics.fairness.MAD.UserMADRating	elliott.namespace.namespace_model
module, 24	module, 32
elliott.evaluation.metrics.fairness.reo	elliott.namespace.namespace_model_builder
module, 25	module, 32
elliott.evaluation.metrics.fairness.reo.reo	elliott.prefiltering
module, 24	module, 33
elliott.evaluation.metrics.fairness.rsp	elliott.prefiltering.standard_prefilters
module, 26	module, 33
elliott.evaluation.metrics.fairness.rsp.rsp	elliott.recommender
module, 25	module, 90
elliott.evaluation.metrics.metrics_utils	elliott.recommender.adversarial
module, 29	module, 43
elliott.evaluation.metrics.novelty	elliott.recommender.adversarial.AMF
module, 27	module, 41
elliott.evaluation.metrics.novelty.EFD	elliott.recommender.adversarial.AMF.AMF
module, 26	module, 38
elliott.evaluation.metrics.novelty.EFD.efd	elliott.recommender.adversarial.AMF.AMF_model
module, 26	module, 39
elliott.evaluation.metrics.novelty.EFD.extended_efd	elliott.recommender.adversarial.AMR
module, 26	module, 43
elliott.evaluation.metrics.novelty.EPC	elliott.recommender.adversarial.AMR.AMR
module, 27	module, 41
elliott.evaluation.metrics.novelty.EPC.epc	elliott.recommender.adversarial.AMR.AMR_model
module, 27	module, 41
elliott.evaluation.metrics.novelty.EPC.extended_epc	elliott.recommender.algebraic
module, 27	module, 44
elliott.evaluation.metrics.rating	elliott.recommender.algebraic.slope_one
module, 29	module, 43
elliott.evaluation.metrics.rating.mae	elliott.recommender.algebraic.slope_one.slope_one
module, 28	module, 43
elliott.evaluation.metrics.rating.mae.mae	elliott.recommender.algebraic.slope_one.slope_one_model
module, 27	module, 43

elliot.recommender.attentive
 module, 44
 elliot.recommender.attentive.afm
 module, 44
 elliot.recommender.attentive.afm.afm
 module, 44
 elliot.recommender.attentive.afm.afm_model
 module, 44
 elliot.recommender.autoencoders
 module, 46
 elliot.recommender.autoencoders.dae
 module, 45
 elliot.recommender.autoencoders.dae.multi_dae
 module, 44
 elliot.recommender.autoencoders.dae.multi_dae_model
 module, 45
 elliot.recommender.autoencoders.vae
 module, 46
 elliot.recommender.autoencoders.vae.multi_vae
 module, 45
 elliot.recommender.autoencoders.vae.multi_vae_model
 module, 46
 elliot.recommender.base_recommender_model
 module, 89
 elliot.recommender.content_based
 module, 47
 elliot.recommender.content_based.VSM
 module, 47
 elliot.recommender.content_based.VSM.tfidf_utils
 module, 47
 elliot.recommender.content_based.VSM.vector_space
 module, 47
 elliot.recommender.content_based.VSM.vector_space_similarity
 module, 47
 elliot.recommender.gan
 module, 54
 elliot.recommender.gan.CFGAN
 module, 50
 elliot.recommender.gan.CFGAN.cfgan
 module, 48
 elliot.recommender.gan.CFGAN.cfgan_model
 module, 48
 elliot.recommender.gan.IRGAN
 module, 54
 elliot.recommender.gan.IRGAN.irgan
 module, 50
 elliot.recommender.gan.IRGAN.irgan_model
 module, 50
 elliot.recommender.graph_based
 module, 56
 elliot.recommender.graph_based.lightgcn
 module, 55
 elliot.recommender.graph_based.lightgcn.LightGCN
 module, 54
 elliot.recommender.graph_based.lightgcn.LightGCN_model
 module, 54
 elliot.recommender.graph_based.ngcf
 module, 55
 elliot.recommender.graph_based.ngcf.NGCF
 module, 55
 elliot.recommender.graph_based.ngcf.NGCF_model
 module, 55
 elliot.recommender.knowledge_aware
 module, 59
 elliot.recommender.knowledge_aware.kaHFM
 module, 57
 elliot.recommender.knowledge_aware.kaHFM.ka_hfm
 module, 56
 elliot.recommender.knowledge_aware.kaHFM.tfidf_utils
 module, 57
 elliot.recommender.knowledge_aware.kaHFM_batch
 module, 58
 elliot.recommender.knowledge_aware.kaHFM_batch.kahfm_batch
 module, 57
 elliot.recommender.knowledge_aware.kaHFM_batch.kahfm_batch.tfidf_utils
 module, 57
 elliot.recommender.knowledge_aware.kaHFM_batch.tfidf_utils
 module, 58
 elliot.recommender.knowledge_aware.kahfm_embeddings
 module, 59
 elliot.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings.tfidf_utils
 module, 58
 elliot.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings.tfidf_utils
 module, 59
 elliot.recommender.latent_factor_models
 module, 72
 elliot.recommender.latent_factor_models.BPRMF
 module, 60
 elliot.recommender.latent_factor_models.BPRMF.BPRMF
 module, 59
 elliot.recommender.latent_factor_models.BPRMF_batch
 module, 61
 elliot.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch
 module, 60
 elliot.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch
 module, 60
 elliot.recommender.latent_factor_models.BPRSlim
 module, 61
 elliot.recommender.latent_factor_models.BPRSlim.bprslim
 module, 61
 elliot.recommender.latent_factor_models.BPRSlim.bprslim_model
 module, 61
 elliot.recommender.latent_factor_models.CML
 module, 62
 elliot.recommender.latent_factor_models.CML.CML
 module, 61
 elliot.recommender.latent_factor_models.CML.CML_model
 module, 62
 elliot.recommender.latent_factor_models.FFM
 module, 63
 elliot.recommender.latent_factor_models.FFM.field_aware_factorization
 module, 62
 elliot.recommender.latent_factor_models.FFM.field_aware_factorization
 module, 63

elliott.recommender.latent_factor_models.FISM module, 64	elliott.recommender.latent_factor_models.SVDpp.svdpp_model module, 70
elliott.recommender.latent_factor_models.FISM.FISM module, 63	elliott.recommender.latent_factor_models.WRMF module, 72
elliott.recommender.latent_factor_models.FISM.FISM_model module, 63	elliott.recommender.latent_factor_models.WRMF.wrmf module, 71
elliott.recommender.latent_factor_models.FM module, 65	elliott.recommender.latent_factor_models.WRMF.wrmf_model module, 71
elliott.recommender.latent_factor_models.FM.factorization module, 64	elliott.recommender.neural module, 84
elliott.recommender.latent_factor_models.FM.factorization_model module, 64	elliott.recommender.neural.ConvMF module, 74
elliott.recommender.latent_factor_models.FunkSVD module, 66	elliott.recommender.neural.ConvMF.convolutional_matrix_factor module, 72
elliott.recommender.latent_factor_models.FunkSVD.funksvd module, 65	elliott.recommender.neural.ConvMF.convolutional_matrix_factor module, 72
elliott.recommender.latent_factor_models.FunkSVD.funksvd_model module, 65	elliott.recommender.neural.ConvNeuMF module, 75
elliott.recommender.latent_factor_models.LogisticMF module, 66	elliott.recommender.neural.ConvNeuMF.convolutional_neural_m module, 74
elliott.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization module, 66	elliott.recommender.neural.ConvNeuMF.convolutional_neural_m module, 74
elliott.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization_model module, 66	elliott.recommender.neural.DeepFM module, 76
elliott.recommender.latent_factor_models.MF module, 67	elliott.recommender.neural.DeepFM.deep_fm module, 75
elliott.recommender.latent_factor_models.MF.matrix_factorization module, 66	elliott.recommender.neural.DeepFM.deep_fm_model module, 76
elliott.recommender.latent_factor_models.MF.matrix_factorization_model module, 67	elliott.recommender.neural.DMF module, 75
elliott.recommender.latent_factor_models.NonNegMF module, 68	elliott.recommender.neural.DMF.deep_matrix_factorization module, 75
elliott.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization module, 67	elliott.recommender.neural.DMF.deep_matrix_factorization_mod module, 75
elliott.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization_model module, 67	elliott.recommender.neural.GeneralizedMF module, 77
elliott.recommender.latent_factor_models.PMF module, 69	elliott.recommender.neural.GeneralizedMF.generalized_matrix_f module, 76
elliott.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization module, 68	elliott.recommender.neural.GeneralizedMF.generalized_matrix_f module, 76
elliott.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization_model module, 68	elliott.recommender.neural.ItemAutoRec module, 78
elliott.recommender.latent_factor_models.PureSVD module, 69	elliott.recommender.neural.ItemAutoRec.itemautorec module, 77
elliott.recommender.latent_factor_models.PureSVD.pure_svd module, 69	elliott.recommender.neural.ItemAutoRec.itemautorec_model module, 77
elliott.recommender.latent_factor_models.PureSVD.pure_svd_model module, 69	elliott.recommender.neural.NAIS module, 79
elliott.recommender.latent_factor_models.Slim module, 71	elliott.recommender.neural.NAIS.nais module, 78
elliott.recommender.latent_factor_models.Slim.slim module, 70	elliott.recommender.neural.NAIS.nais_model module, 78
elliott.recommender.latent_factor_models.Slim.slim_model module, 71	elliott.recommender.neural.NeuMF module, 81
elliott.recommender.latent_factor_models.SVDpp module, 70	elliott.recommender.neural.NeuMF.neural_matrix_factorization module, 80
elliott.recommender.latent_factor_models.SVDpp.svdpp_model module, 70	elliott.recommender.neural.NeuMF.neural_matrix_factorization_ module, 81

elliott.recommender.neural.NFM	elliott.recommender.unpersonalized
module, 80	module, 85
elliott.recommender.neural.NFM.neural_fm	elliott.recommender.unpersonalized.most_popular
module, 79	module, 84
elliott.recommender.neural.NFM.neural_fm_model	elliott.recommender.unpersonalized.most_popular.most_popular_model
module, 79	module, 84
elliott.recommender.neural.NPR	elliott.recommender.unpersonalized.random_recommender
module, 80	module, 85
elliott.recommender.neural.NPR.neural_personalized_recommender	elliott.recommender.unpersonalized.random_recommender.RandomRecommender
module, 80	module, 85
elliott.recommender.neural.NPR.neural_personalized_recommender_model	elliott.recommender.visual_recommenders
module, 80	module, 89
elliott.recommender.neural.UserAutoRec	elliott.recommender.visual_recommenders.ACF
module, 82	module, 85
elliott.recommender.neural.UserAutoRec.userautorec	elliott.recommender.visual_recommenders.ACF.ACF
module, 81	module, 85
elliott.recommender.neural.UserAutoRec.userautorec_model	elliott.recommender.visual_recommenders.ACF.ACF_model
module, 81	module, 85
elliott.recommender.neural.WideAndDeep	elliott.recommender.visual_recommenders.DeepStyle
module, 84	module, 87
elliott.recommender.neural.WideAndDeep.wide_and_deep	elliott.recommender.visual_recommenders.DeepStyle.DeepStyle
module, 82	module, 87
elliott.recommender.neural.WideAndDeep.wide_and_deep_model	elliott.recommender.visual_recommenders.DeepStyle.DeepStyle_model
module, 82	module, 87
elliott.recommender.NN	elliott.recommender.visual_recommenders.DVBPR
module, 38	module, 86
elliott.recommender.NN.attribute_item_knn	elliott.recommender.visual_recommenders.DVBPR.DVBPR
module, 34	module, 86
elliott.recommender.NN.attribute_item_knn.attribute_item_knn	elliott.recommender.visual_recommenders.DVBPR.DVBPR_model
module, 34	module, 86
elliott.recommender.NN.attribute_item_knn.attribute_item_knn_similarity	elliott.recommender.visual_recommenders.DVBPR.FeatureExtractor
module, 34	module, 86
elliott.recommender.NN.attribute_user_knn	elliott.recommender.visual_recommenders.VBPR
module, 35	module, 88
elliott.recommender.NN.attribute_user_knn.attribute_user_knn	elliott.recommender.visual_recommenders.VBPR.VBPR
module, 35	module, 87
elliott.recommender.NN.attribute_user_knn.attribute_user_knn_similarity	elliott.recommender.visual_recommenders.VBPR.VBPR_model
module, 35	module, 87
elliott.recommender.NN.attribute_user_knn.tfidf_utils	elliott.recommender.visual_recommenders.VNPR
module, 35	module, 89
elliott.recommender.NN.item_knn	elliott.recommender.visual_recommenders.VNPR.visual_neural
module, 37	module, 88
elliott.recommender.NN.item_knn.aiolli_ferrari	elliott.recommender.visual_recommenders.VNPR.visual_neural_model
module, 36	module, 88
elliott.recommender.NN.item_knn.item_knn	elliott.result_handler
module, 36	module, 90
elliott.recommender.NN.item_knn.item_knn_similarity	elliott.result_handler.result_handler
module, 36	module, 90
elliott.recommender.NN.user_knn	elliott.run
module, 38	module, 93
elliott.recommender.NN.user_knn.aiolli_ferrari	elliott.splitter
module, 37	module, 91
elliott.recommender.NN.user_knn.user_knn	elliott.splitter.base_splitter
module, 38	module, 90
elliott.recommender.NN.user_knn.user_knn_similarity	elliott.utils
module, 38	module, 93
elliott.recommender.recommender_utils_mixin	elliott.utils.folder
module, 89	module, 91

elliot.utils.logger_util
 module, 91
 elliot.utils.logging
 module, 92
 elliot.utils.read
 module, 92
 elliot.utils.write
 module, 93
 emit() (elliot.utils.logger_util.QueueListener-
 Handler method), 91
 Encoder (class in elliot.recommender.autoen-
 coders.dae.multi_dae_model), 45
 Encoder (class in elliot.recommender.autoen-
 coders.vae.multi_vae_model), 46
 Encoder (class in elliot.recommender.neu-
 ral.ItemAutoRec.itemautorec_model),
 77
 Encoder (class in elliot.recommender.neu-
 ral.UserAutoRec.userautorec_model),
 81
 EPC (class in elliot.evaluation.metrics.novel-
 ty.EPC.epc), 27
 eval() (elliot.evaluation.evaluator.Evaluator
 method), 31
 eval() (elliot.evaluation.metrics.accuracy.AUC.auc.AUC method), 10
 eval() (elliot.evaluation.metrics.accuracy.AUC.-
 gauc.GAUC method), 10
 eval() (elliot.evaluation.metrics.base_met-
 ric.BaseMetric method), 29
 eval() (elliot.evaluation.metrics.bias.pop_re-
 o.extended_pop_reo.ExtendedPopREO
 method), 17
 eval() (elliot.evaluation.metrics.bias.pop_re-
 o.pop_reo.PopREO method), 17
 eval() (elliot.evaluation.metrics.bias.pop_r-
 sp.extended_pop_rsp.ExtendedPopRSP
 method), 18
 eval() (elliot.evaluation.metrics.bias.pop_r-
 sp.pop_rsp.PopRSP method), 18
 eval() (elliot.evaluation.metrics.coverage.item_-
 coverage.item_coverage.ItemCoverage
 method), 19
 eval() (elliot.evaluation.metrics.coverage.user_-
 coverage.user_coverage.UserCoverage
 method), 19
 eval() (elliot.evaluation.metrics.coverage.user_-
 coverage.user_coverage_at_n.UserCoverageAtN
 method), 20
 eval() (elliot.evaluation.metrics.diversi-
 ty.gini_index.gini_index.GiniIndex
 method), 21
 eval() (elliot.evaluation.metrics.diversity.shan-
 non_entropy.shannon_entropy.ShannonEntropy
 method), 21
 eval() (elliot.evaluation.metrics.fairness.Bias-
 Disparity.BiasDisparityBD.BiasDisparityBD
 method), 21
 eval() (elliot.evaluation.metrics.fairness.Bias-
 Disparity.BiasDisparityBR.BiasDisparityBR
 method), 22
 eval() (elliot.evaluation.metrics.fairness.Bias-
 Disparity.BiasDisparityBS.BiasDisparityBS
 method), 22
 eval() (elliot.evaluation.metrics.fairness.-
 MAD.ItemMADranking.ItemMADranking
 method), 23
 eval() (elliot.evaluation.metrics.fairness.-
 MAD.ItemMADrating.ItemMADrating
 method), 23
 eval() (elliot.evaluation.metrics.fairness.-
 MAD.UserMADranking.UserMADranking
 method), 24
 eval() (elliot.evaluation.metrics.fairness.-
 MAD.UserMADrating.UserMADrating
 method), 24
 eval() (elliot.evaluation.metrics.fairness.reo.re-
 o.REO method), 25
 eval() (elliot.evaluation.metrics.fairness.rsp.r-
 sp.RSP method), 26
 eval() (elliot.evaluation.metrics.metrics_util-
 s.ProxyMetric method), 29
 eval() (elliot.evaluation.metrics.metrics_util-
 s.ProxyStatisticalMetric method), 29
 eval() (elliot.evaluation.metrics.rating.mae.-
 mae.MAE method), 28
 eval() (elliot.evaluation.metrics.rat-
 ing.mse.mse.MSE method), 28
 eval() (elliot.evaluation.metrics.rat-
 ing.rmse.rmse.RMSE method), 29
 eval_at_k() (elliot.evaluation.evaluator.Evalu-
 ator method), 31
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.AUC.gauc.GAUC method), 10
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.AUC.lauc.LAUC method), 11
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.DSC.dsc.DSC method), 11
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.f1.extended_f1.ExtendedF1
 method), 11
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.f1.f1.F1 method), 12
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.hit_rate.hit_rate.HR method),
 12
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.map.map.MAP method), 13
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.mar.mar.MAR method), 13
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.mrr.mrr.MRR method), 13
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.ndcg.ndcg.NDCG method), 14
 eval_user_metric() (elliot.evaluation.metrics.ac-
 curacy.precision.precision.Precision

method), 14

eval_user_metric() (elliot.evaluation.metrics.accuracy.recall.recall.Recall method), 15

eval_user_metric() (elliot.evaluation.metrics.bias.aclt.aclt.ACLT method), 15

eval_user_metric() (elliot.evaluation.metrics.bias.aplt.aplt.APLT method), 16

eval_user_metric() (elliot.evaluation.metrics.bias.arp.arp.ARP method), 16

eval_user_metric() (elliot.evaluation.metrics.coverage.num_retrieved.num_retrieved.NumRetrieved method), 19

eval_user_metric() (elliot.evaluation.metrics.diversity.SRecall.srecall.SRecall method), 20

eval_user_metric() (elliot.evaluation.metrics.metrics_utils.ProxyStatisticalMetric method), 29

eval_user_metric() (elliot.evaluation.metrics.novelty.EFD.efd.EFD method), 26

eval_user_metric() (elliot.evaluation.metrics.novelty.EFD.extended_efd.ExtendedEFD method), 26

eval_user_metric() (elliot.evaluation.metrics.novelty.EPC.epc.EPC method), 27

eval_user_metric() (elliot.evaluation.metrics.novelty.EPC.extended_epc.ExtendedEPC method), 27

eval_user_metric() (elliot.evaluation.metrics.rating.mae.mae.MAE method), 28

eval_user_metric() (elliot.evaluation.metrics.rating.mse.mse.MSE method), 28

eval_user_metric() (elliot.evaluation.metrics.rating.rmse.rmse.RMSE method), 29

eval_user_metric() (elliot.evaluation.metrics.statistical_array_metric.StatisticalMetric method), 30

Evaluator (class in elliot.evaluation.evaluator), 31

ExtendedEFD (class in elliot.evaluation.metrics.novelty.EFD.extended_efd), 26

ExtendedEPC (class in elliot.evaluation.metrics.novelty.EPC.extended_epc), 27

ExtendedF1 (class in elliot.evaluation.metrics.accuracy.f1.extended_f1), 11

ExtendedPopREO (class in elliot.evaluation.metrics.bias.pop_reo.extended_pop_reo), 16

ExtendedPopRSP (class in elliot.evaluation.metrics.bias.pop_rsp.extended_pop_rsp), 17

F

F1 (class in elliot.evaluation.metrics.accuracy.f1.f1), 12

FactorizationMachineModel (class in elliot.recommender.latent_factor_models.FM.factorization_machine_model), 64

FeatureExtractor (class in elliot.recommender.visual_recommenders.DVBPR.FeatureExtractor), 86

FFM (class in elliot.recommender.latent_factor_models.FFM.field_aware_factorization_machine), 62

FieldAwareFactorizationMachineModel (class in elliot.recommender.latent_factor_models.FFM.field_aware_factorization_machine_model), 63

fill_base() (elliot.namespace.namespace_model.NamespaceModel method), 32

fill_model() (elliot.namespace.namespace_model.NamespaceModel method), 32

filter() (elliot.prefiltering.standard_prefilters.PreFilter static method), 33

filter() (elliot.utils.logging.TimeFilter method), 92

filter_items_by_popularity() (elliot.prefiltering.standard_prefilters.PreFilter static method), 33

filter_iterative_k_core() (elliot.prefiltering.standard_prefilters.PreFilter static method), 33

filter_ratings_by_global_average() (elliot.prefiltering.standard_prefilters.PreFilter static method), 33

filter_ratings_by_threshold() (elliot.prefiltering.standard_prefilters.PreFilter static method), 33

filter_ratings_by_user_average() (elliot.prefiltering.standard_prefilters.PreFilter static method), 33

filter_retain_cold_users() (elliot.prefiltering.standard_prefilters.PreFilter static method), 33

filter_rounds_k_core() (elliot.prefiltering.standard_prefilters.PreFilter static method), 33

filter_users_by_profile_size() (elliot.prefiltering.standard_prefilters.PreFilter static method), 33

find_checkpoint() (in module elliot.utils.read), 92

FISM (class in elliot.recommender.latent_factor_models.FISM.FISM), 63

FISM_model (class in elliot.recommender.latent_factor_models.FISM.FISM_model), 63

FM (class in elliot.recommender.latent_factor_models.FM.factorization_machine), 64

fold_list_generator() (elliot.splitter.base_splitter.Splitter method), 90

ForceRequiredAttributeDefinitionMeta (class in elliot.dataset.abstract_dataset), 9

FunkSVD (class in elliot.recommender.latent_factor_models.FunkSVD.funk_svd), 65

- FunkSVDModel (class in `elliott.recommender.latent_factor_models.FunkSVD.funk_svd_model`), 65
- ## G
- GAUC (class in `elliott.evaluation.metrics.accuracy.AUC.gauc`), 10
- GeneralizedMatrixFactorizationModel (class in `elliott.recommender.neural.GeneralizedMF.generalized_matrix_factorization_model`), 76
- `generate_dataobjects()` (`elliott.dataset.data_loader.knowledge_aware_chains.KnowledgeChainsLoader` method), 5
- `generate_dataobjects()` (`elliott.dataset.data_loader.visual_data_loader.VisualLoader` method), 6
- `generate_dataobjects()` (`elliott.dataset.dataset.DataSetLoader` method), 9
- `generate_dataobjects_mock()` (`elliott.dataset.data_loader.knowledge_aware_chains.KnowledgeChainsLoader` method), 5
- `generate_dataobjects_mock()` (`elliott.dataset.data_loader.visual_data_loader.VisualLoader` method), 6
- `generate_dataobjects_mock()` (`elliott.dataset.dataset.DataSetLoader` method), 9
- `generate_fake_data()` (`elliott.recommender.gan.CFGAN.cfgan_model.Generator` method), 50
- Generator (class in `elliott.recommender.gan.CFGAN.cfgan_model`), 50
- Generator (class in `elliott.recommender.gan.IRGAN.irgan_model`), 51
- `generic_split_function()` (`elliott.splitter.base_splitter.Splitter` method), 90
- `get()` (`elliott.evaluation.metrics.accuracy.f1.extended_f1.ExtendedF1` method), 11
- `get()` (`elliott.evaluation.metrics.base_metric.BaseMetric` method), 29
- `get()` (`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBD.BiasDisparityBD` method), 21
- `get()` (`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBR.BiasDisparityBR` method), 22
- `get()` (`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBS.BiasDisparityBS` method), 22
- `get()` (`elliott.evaluation.metrics.fairness.MAD.ItemMADranking.ItemMADranking` method), 23
- `get()` (`elliott.evaluation.metrics.fairness.MAD.ItemMADrating.ItemMADrating` method), 23
- `get()` (`elliott.evaluation.metrics.fairness.MAD.UserMADranking.UserMADranking` method), 24
- `get()` (`elliott.evaluation.metrics.fairness.MAD.UserMADrating.UserMADrating` method), 24
- `get()` (`elliott.evaluation.metrics.fairness.reo.REO` method), 25
- `get()` (`elliott.evaluation.metrics.fairness.rsp.rsp.RSP` method), 26
- `get_BR()` (`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBR.BiasDisparityBR` method), 22
- `get_BS()` (`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBS.BiasDisparityBS` method), 22
- `get_config()` (`elliott.recommender.adversarial.AMF.AMF_model.AMF_model` method), 39
- `get_config()` (`elliott.recommender.adversarial.AMR.AMR_model.AMR_model` method), 41
- `get_config()` (`elliott.recommender.autoencoders.dae_multi_dae_model.DenoisingAutoEncoder` method), 45
- `get_config()` (`elliott.recommender.autoencoders.vae_multi_vae_model.VariationalAutoEncoder` method), 46
- `get_config()` (`elliott.recommender.gan.CFGAN.cfgan_model.CFGAN_model` method), 48
- `get_config()` (`elliott.recommender.gan.IRGAN.irgan_model.IRGAN_model` method), 52
- `get_config()` (`elliott.recommender.graph_based.lightgcn.LightGCN_model.LightGCNModel` method), 54
- `get_config()` (`elliott.recommender.graph_based.ngcf.NGCF_model.NGCFModel` method), 55
- `get_config()` (`elliott.recommender.knowledge_aware.kaHFM_batch.kahfm_batch_model.KaHFM` method), 57
- `get_config()` (`elliott.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings_model` method), 58
- `get_config()` (`elliott.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch_model.BPRMF_batch_model` method), 60
- `get_config()` (`elliott.recommender.latent_factor_models.CML.CML_model.CML_model` method), 62
- `get_config()` (`elliott.recommender.latent_factor_models.FISM.FISM_model.FISM_model` method), 64
- `get_config()` (`elliott.recommender.neural.ItemAutoRec.itemautorec_model.ItemAutoRecModel` method), 77
- `get_config()` (`elliott.recommender.neural.NAIS.nais_model.NAIS_model` method), 78
- `get_config()` (`elliott.recommender.neural.User`

AutoRec.userautorec_model.UserAutoRecModel method), 59

method), 82

get_config() (elliot.recommender.visual_recommenders.ACF.ACF_model.ACF_model method), 85

get_config() (elliot.recommender.visual_recommenders.DeepStyle.DeepStyle_model.DeepStyle_model method), 87

get_config() (elliot.recommender.visual_recommenders.DVBPR.DVBPR_model.DVBPR_model method), 86

get_config() (elliot.recommender.visual_recommenders.VBPR.VBPR_model.VBPR_model method), 88

get_custom_pop_obj() (elliot.evaluation.popularity_utils.popularity.Popularity method), 30

get_factors() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56

get_item_bias() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56

get_item_bias() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.MF method), 59

get_item_factors() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56

get_item_factors() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.MF method), 59

get_item_neighbors() (elliot.recommender.N-N.attribute_item_knn.attribute_item_knn_similarity.Similarity method), 34

get_item_neighbors() (elliot.recommender.N-N.item_knn.item_knn_similarity.Similarity method), 37

get_logger() (in module elliot.utils.logging), 92

get_long_tail() (elliot.evaluation.popularity_utils.popularity.Popularity method), 30

get_loss() (elliot.recommender.base_recommender_model.BaseRecommenderModel method), 89

get_loss() (elliot.recommender.recommender_utils_mixin.RecMixin method), 89

get_model_state() (elliot.recommender.algebraic.slope_one.slope_one_model.SlopeOneModel method), 43

get_model_state() (elliot.recommender.content_based.VSM.vector_space_model_similarity.Similarity method), 47

get_model_state() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56

get_model_state() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.MF method), 59

get_model_state() (elliot.recommender.latent_factor_models.BPRSlim.bprslim_model.BPRSlimModel method), 61

get_model_state() (elliot.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization method), 68

get_model_state() (elliot.recommender.latent_factor_models.PureSVD.pure_svd_model.PureSVDModel method), 69

get_model_state() (elliot.recommender.latent_factor_models.Slim.slim_model.SlimModel method), 71

get_model_state() (elliot.recommender.latent_factor_models.WRMF.wrmf_model.WRMFModel method), 71

get_model_state() (elliot.recommender.NN.attribute_item_knn.attribute_item_knn_similarity.Similarity method), 34

get_model_state() (elliot.recommender.NN.attribute_user_knn.attribute_user_knn_similarity.Similarity method), 35

get_model_state() (elliot.recommender.N-N.item_knn.item_knn_similarity.Similarity method), 37

get_model_state() (elliot.recommender.N-N.user_knn.user_knn_similarity.Similarity method), 38

get_needed_recommendations() (elliot.evaluation.evaluator.Evaluator method), 31

get_params() (elliot.recommender.base_recommender_model.BaseRecommenderModel method), 89

get_params() (elliot.recommender.recommender_utils_mixin.RecMixin method), 89

get_params_shortcut() (elliot.recommender.base_recommender_model.BaseRecommenderModel method), 89

get_pop_items() (elliot.evaluation.popularity_utils.popularity.Popularity method), 30

get_positions() (elliot.recommender.adversarial.AMF.AMF_model.AMF_model method), 39

get_positions() (elliot.recommender.gan.IR-GAN.irgan_model.IRGAN_model method), 52

get_positions() (elliot.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch_model.BPRMF_batch_model method), 61

get_positions() (elliot.recommender.latent_factor_models.CML.CML_model.CML_model method), 62

get_positions() (elliot.recommender.latent_factor_models.FISM.FISM_model.FISM_model method), 64

get_positions() (elliot.recommender.neural.-

- NAIS.nais_model.NAIS_model method), 79
- get_profiles() (elliot.recommender.content_based.VSM.tfidf_utils.TFIDF method), 47
- get_profiles() (elliot.recommender.knowledge_aware.kaHFM.tfidf_utils.TFIDF method), 57
- get_profiles() (elliot.recommender.knowledge_aware.kaHFM_batch.tfidf_utils.TFIDF method), 58
- get_profiles() (elliot.recommender.knowledge_aware.kahfm_embeddings.tfidf_utils.TFIDF method), 59
- get_profiles() (elliot.recommender.NN.attribute_user_knn.tfidf_utils.TFIDF method), 35
- get_recommendations() (elliot.recommender.adversarial.AMF.AMF.AMF method), 38
- get_recommendations() (elliot.recommender.adversarial.AMR.AMR.AMR method), 41
- get_recommendations() (elliot.recommender.algebraic.slope_one.slope_one.SlopeOne method), 43
- get_recommendations() (elliot.recommender.base_recommender_model.BaseRecommenderModel method), 89
- get_recommendations() (elliot.recommender.-content_based.VSM.vector_space_model.VSM method), 47
- get_recommendations() (elliot.recommender.-gan.CFGAN.cfgan.CFGAN method), 48
- get_recommendations() (elliot.recommender.-gan.IRGAN.irgan.IRGAN method), 50
- get_recommendations() (elliot.recommender.-graph_based.lightgcn.LightGCN.LightGCN method), 54
- get_recommendations() (elliot.recommender.-graph_based.ngcf.NGCF.NGCF method), 55
- get_recommendations() (elliot.recommender.-knowledge_aware.kaHFM.ka_hfm.KaHFM method), 56
- get_recommendations() (elliot.recommender.-knowledge_aware.kaHFM_batch.kahfm_batch.KaHFMBatch method), 57
- get_recommendations() (elliot.recommender.-knowledge_aware.kahfm_embeddings.kahfm_embeddings.KaHFMEmbeddings method), 58
- get_recommendations() (elliot.recommender.la-
tent_factor_models.BPRMF.BPRMF.BPRMF method), 59
- get_recommendations() (elliot.recommender.la-
tent_factor_models.BPRMF_batch.BPRMF_batch.BPRMFBatch method), 60
- get_recommendations() (elliot.recommender.la-
tent_factor_models.BPRSlim.bprslim.BPRSlim method), 61
- get_recommendations() (elliot.recommender.la-
tent_factor_models.CML.CML.CML method), 62
- get_recommendations() (elliot.recommender.la-
tent_factor_models.FFM.field_aware_factorization_machine.FFM method), 63
- get_recommendations() (elliot.recommender.la-
tent_factor_models.FISM.FISM.FISM method), 63
- get_recommendations() (elliot.recommender.la-
tent_factor_models.FM.factorization_machine.FM method), 64
- get_recommendations() (elliot.recommender.la-
tent_factor_models.FunkSVD.funk_svd.FunkSVD method), 65
- get_recommendations() (elliot.recommender.la-
tent_factor_models.LogisticMF.logistic_matrix_factorization.LogisticMF method), 66
- get_recommendations() (elliot.recommender.la-
tent_factor_models.MF.matrix_factorization.MF method), 66
- get_recommendations() (elliot.recommender.la-
tent_factor_models.NonNegMF.non_negative_matrix_factorization.NonNegMF method), 67
- get_recommendations() (elliot.recommender.la-
tent_factor_models.PMF.probabilistic_matrix_factorization.PMF method), 68
- get_recommendations() (elliot.recommender.la-
tent_factor_models.PureSVD.pure_svd.PureSVD method), 69
- get_recommendations() (elliot.recommender.la-
tent_factor_models.Slim.slim.Slim method), 70
- get_recommendations() (elliot.recommender.la-
tent_factor_models.SVDpp.svdpp.SVDpp method), 70
- get_recommendations() (elliot.recommender.la-
tent_factor_models.WRMF.wrmf.WRMF method), 71
- get_recommendations() (elliot.recommender.neural.ConvMF.convolutional_matrix_factorization.ConvMF method), 72
- get_recommendations() (elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization.ConvNeuMF method), 74
- get_recommendations() (elliot.recommender.neural.DeepFM.deep_fm.DeepFM method), 75
- get_recommendations() (elliot.recommender.neural.DMF.deep_matrix_factorization.DMF method), 75
- get_recommendations() (elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization.GeneralizedMF method), 76
- get_recommendations() (elliot.recommender.neural.KaHFM.kahfm.KaHFM method), 76

mender.neural.ItemAutoRec.itemautorec.ItemAutoRecmethod), 65
method), 77
get_recommendations() (elliott.recommender.neural.NAIS.nais.NAIS
method), 78
get_recommendations() (elliott.recommender.neural.NeuMF.neural_matrix_factorization.NeuMFmethod), 80
get_recommendations() (elliott.recommender.neural.NFM.neural_fm.NFM
method), 79
get_recommendations() (elliott.recommender.neural.NPR.neural_personalized_ranking.NPRmethod), 80
get_recommendations() (elliott.recommender.neural.WideAndDeep.wide_and_deep.WideAndDeepmethod), 82
get_recommendations() (elliott.recommender.N-
N.attribute_item_knn.attribute_item_knn.AttributeItemKNNmethod), 34
get_recommendations() (elliott.recommender.N-
N.attribute_user_knn.attribute_user_knn.AttributeUserKNNmethod), 35
get_recommendations() (elliott.recommender.N-
N.item_knn.item_knn.ItemKNNmethod), 36
get_recommendations() (elliott.recommender.N-
N.user_knn.user_knn.UserKNNmethod), 38
get_recommendations() (elliott.recommender.recommender_utils.mixin.RecMixin
method), 89
get_recommendations() (elliott.recommender.unpersonalized.most_popular.most_popular.MostPopmethod), 84
get_recommendations() (elliott.recommender.unpersonalized.random_recommender.RandomRecommendermethod), 85
get_recommendations() (elliott.recommender.visual_recommenders.DeepStyle.DeepStyle.DeepStyle
method), 87
get_recommendations() (elliott.recommender.visual_recommenders.DVBPR.DVBPR.DVBPR
method), 86
get_recommendations() (elliott.recommender.visual_recommenders.VBPR.VBPR.VBPR
method), 87
get_recommendations() (elliott.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking.VNPRmethod), 88
get_recs() (elliott.recommender.attentive.afm.afm_model.AFMMModel
method), 44
get_recs() (elliott.recommender.latent_factor_ -
models.FFM.field_aware_factorization_machine_model.FieldAwareFactorizationMachineModel
method), 63
get_recs() (elliott.recommender.latent_factor_ -
models.FM.factorization_machine_model.FactorizationMachineModelmethod), 63
get_recs() (elliott.recommender.latent_factor_ -
models.FunkSVD.funk_svd_model.FunkSVDModel
method), 65
get_recs() (elliott.recommender.latent_factor_ -
models.MF.matrix_factorization_model.MatrixFactorizationModelmethod), 67
get_recs() (elliott.recommender.latent_factor_ -
models.PMF.probabilistic_matrix_factorization_model.PMFmethod), 68
get_recs() (elliott.recommender.latent_factor_ -
models.SVDpp.svdpp_model.SVDppModel
method), 70
get_recs() (elliott.recommender.neural.ConvMF -
convolutional_matrix_factorization_model.ConvMatrixFactorizationModelmethod), 72
get_recs() (elliott.recommender.neural.Conv-
vNeuMF.convolutional_neural_matrix_factorization_model.ConvNeuMFmethod), 74
get_recs() (elliott.recommender.neural.DeepFM.deep_fm_model.DeepFMModel
method), 76
get_recs() (elliott.recommender.neural.DM-
F.deep_matrix_factorization_model.DeepMatrixFactorizationModelmethod), 75
get_recs() (elliott.recommender.neural.General-
izedMF.generalized_matrix_factorization_model.GeneralizedMFmethod), 77
get_recs() (elliott.recommender.neural.ItemAu-
toRec.itemautorec_model.ItemAutoRecModel
method), 78
get_recs() (elliott.recommender.neural.NeuM-
F.neural_matrix_factorization_model.NeuralMatrixFactorizationModelmethod), 81
get_recs() (elliott.recommender.neural.NFM.neural_fm_model.NeuralFactorizationMachineModelmethod), 79
get_recs() (elliott.recommender.neural.NPR.neu-
ral_personalized_ranking_model.NPRModel
method), 80
get_recs() (elliott.recommender.visual_recom-
menders.VNPR.visual_neural_personalized_ranking_model.VNPRmethod), 88
get_rel() (elliott.evaluation.relevance.rele-
vance.AbstractRelevanceSingleton
method), 30
get_rel() (elliott.evaluation.relevance.rele-
vance.BinaryRelevanceSingletonmethod), 30
get_relevance() (elliott.evaluation.relevance.rele-
vance.DiscountedRelevanceSingletonmethod), 31
get_results() (elliott.recommender.base_recom-
mender_model.BaseRecommenderModel
method), 89
get_results() (elliott.recommender.base_recom-
mender_utils.mixin.RecMixin
method), 89
get_results() (elliott.evaluation.populari-
ty_evaluation_model.PopularityEvaluationModelmethod), 89

ty_utils.popularity.Popularity method), 30

get_sorted_pop_items() (elliot.evaluation.popularity_utils.popularity.Popularity method), 30

get_sparse() (elliot.recommender.neural.WideAndDeep.wide_and_deep_model.WideAndDeepModel method), 83

get_test() (elliot.dataset.abstract_dataset.AbstractDataset method), 9

get_test() (elliot.dataset.data_loader.knowledge_aware_chains.KnowledgeChainsDataObject method), 5

get_test() (elliot.dataset.data_loader.visual_data_loader.VisualDataObject method), 6

get_test() (elliot.dataset.dataset.DataSet method), 9

get_test() (elliot.evaluation.relevance.relevance.Relevance method), 31

get_top_k() (elliot.recommender.adversarial.AMF.AMF_model.AMF_model method), 39

get_top_k() (elliot.recommender.adversarial.AMR.AMR_model.AMR_model method), 41

get_top_k() (elliot.recommender.attentive.afm.afm_model.AFMMModel method), 44

get_top_k() (elliot.recommender.autoencoders.-dae.multi_dae_model.DenoisingAutoEncoder method), 45

get_top_k() (elliot.recommender.autoencoders.-vae.multi_vae_model.VariationalAutoEncoder method), 46

get_top_k() (elliot.recommender.gan.CF-GAN.cfgan_model.CFGAN_model method), 48

get_top_k() (elliot.recommender.gan.IRGAN.irgan_model.IRGAN_model method), 52

get_top_k() (elliot.recommender.graph_based.-lightgcn.LightGCN_model.LightGCNModel method), 54

get_top_k() (elliot.recommender.-graph_based.ngcf.NGCF_model.NGCFModel method), 55

get_top_k() (elliot.recommender.knowledge_aware.kaHFM_batch.kahfm_batch_model.KaHFMBatchModel method), 57

get_top_k() (elliot.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings_model.KaHFMEmbeddingsModel method), 58

get_top_k() (elliot.recommender.latent_factor_-models.BPRMF_batch.BPRMF_batch_model.BPRMFBatchModel method), 61

get_top_k() (elliot.recommender.latent_factor_-models.CML.CML_model.CML_model method), 62

get_top_k() (elliot.recommender.latent_factor_-models.FFM.field_aware_factorization_machine_model.FFMFieldAwareFactorizationMachineModel method), 63

get_top_k() (elliot.recommender.latent_factor_-models.FISM.FISM_model.FISM_model method), 64

get_top_k() (elliot.recommender.latent_factor_-models.FM.factorization_machine_model.FactorizationMachineModel method), 65

get_top_k() (elliot.recommender.latent_factor_-models.FunkSVD.funk_svd_model.FunkSVDModel method), 65

get_top_k() (elliot.recommender.latent_factor_-models.LogisticMF.logistic_matrix_factorization_model.LogisticMatrixFactorizationModel method), 66

get_top_k() (elliot.recommender.latent_factor_-models.MF.matrix_factorization_model.MatrixFactorizationModel method), 67

get_top_k() (elliot.recommender.latent_factor_-models.PMF.probabilistic_matrix_factorization_model.PMFProbabilisticMatrixFactorizationModel method), 69

get_top_k() (elliot.recommender.latent_factor_-models.SVDpp.svdpp_model.SVDppModel method), 70

get_top_k() (elliot.recommender.neural.ConvMF.convolutional_matrix_factorization_model.ConvMatrixFactorizationModel method), 72

get_top_k() (elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model.ConvNeuMatrixFactorizationModel method), 74

get_top_k() (elliot.recommender.neural.DeepFM.deep_fm_model.DeepFMModel method), 76

get_top_k() (elliot.recommender.neural.DMF.deep_matrix_factorization_model.DeepMatrixFactorizationModel method), 75

get_top_k() (elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization_model.GeneralizedMatrixFactorizationModel method), 77

get_top_k() (elliot.recommender.neural.ItemAutoRec.itemautorec_model.ItemAutoRecModel method), 78

get_top_k() (elliot.recommender.neural.NAIS.-nais_model.NAIS_model method), 79

get_top_k() (elliot.recommender.neural.NeuMF.neural_matrix_factorization_model.NeuralMatrixFactorizationModel method), 81

get_top_k() (elliot.recommender.neural.NFM.neural_fm_model.NeuralFactorizationMachineModel method), 79

get_top_k() (elliot.recommender.neural.NPR.neural_personalized_ranking_model.NPRModel method), 80

get_top_k() (elliot.recommender.neural.BPRMF_batch.BPRMF_batch_model.BPRMFBatchModel method), 82

get_top_k() (elliot.recommender.neural.UserAutoRec.userautorec_model.UserAutoRecModel method), 82

get_top_k() (elliot.recommender.neural.WideAndDeep.wide_and_deep_model.WideAndDeepModel method), 83

method), 83

get_top_k() (elliot.recommender.visual_recommenders.ACF.ACF_model.ACF_model method), 85

get_top_k() (elliot.recommender.visual_recommenders.DeepStyle.DeepStyle_model.DeepStyle_model method), 87

get_top_k() (elliot.recommender.visual_recommenders.DVBPR.DVBPR_model.DVBPR_model method), 86

get_top_k() (elliot.recommender.visual_recommenders.VBPR.VBPR_model.VBPR_model method), 88

get_top_k() (elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking_model.VNPR_model method), 88

get_train_mask() (elliot.recommender.recommender_utils_mixin.RecMixin method), 89

get_transactions() (elliot.recommender.content_based.VSM.vector_space_model_similarity.Similarity method), 47

get_transactions() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56

get_transactions() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.MF method), 59

get_transactions() (elliot.recommender.NN.attribute_item_knn.attribute_item_knn_similarity.Similarity method), 34

get_transactions() (elliot.recommender.NN.attribute_user_knn.attribute_user_knn_similarity.Similarity method), 35

get_transactions() (elliot.recommender.NN.item_knn.item_knn_similarity.Similarity method), 37

get_transactions() (elliot.recommender.NN.user_knn.user_knn_similarity.Similarity method), 38

get_user_bias() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56

get_user_bias() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.MF method), 59

get_user_factors() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56

get_user_factors() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.MF method), 59

get_user_neighbors() (elliot.recommender.NN.attribute_user_knn.attribute_user_knn_similarity.Similarity method), 35

get_user_neighbors() (elliot.recommender.NN.user_knn.user_knn_similarity.Similarity method), 38

get_user_recs() (elliot.recommender.algebraic.slope_one.slope_one_model.SlopeOneModel method), 43

get_user_recs() (elliot.recommender.content_based.VSM.vector_space_model_similarity.Similarity method), 47

get_user_recs() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56

get_user_recs() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.MF method), 59

get_user_recs() (elliot.recommender.latent_factor_models.BPRSlim.bprslim_model.BPRSlimModel method), 59

get_user_recs() (elliot.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization.NonNegMF method), 68

get_user_recs() (elliot.recommender.latent_factor_models.PureSVD.pure_svd_model.PureSVDModel method), 69

get_user_recs() (elliot.recommender.latent_factor_models.Slim.slim_model.SlimModel method), 71

get_user_recs() (elliot.recommender.latent_factor_models.WRMF.wrmf_model.WRMFModel method), 71

get_user_recs() (elliot.recommender.neural.WideAndDeep.wide_and_deep_model.WideAndDeep method), 83

get_user_recs() (elliot.recommender.NN.attribute_item_knn.attribute_item_knn_similarity.Similarity method), 34

get_user_recs() (elliot.recommender.NN.attribute_user_knn.attribute_user_knn_similarity.Similarity method), 35

get_user_recs() (elliot.recommender.NN.item_knn.item_knn_similarity.Similarity method), 37

get_user_recs() (elliot.recommender.NN.user_knn.aiolli_ferrari.AiolliSimilarity method), 36

get_user_recs() (elliot.recommender.NN.item_knn.item_knn_similarity.Similarity method), 37

get_user_recs() (elliot.recommender.NN.user_knn.aiolli_ferrari.AiolliSimilarity method), 37

get_user_recs() (elliot.recommender.NN.user_knn.user_knn_similarity.Similarity method), 38

get_user_recs_argpartition() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56

get_user_recs_argpartition() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.MF method), 60

get_user_rel() (elliot.evaluation.relevance.relevance.BinaryRelevance method), 30

get_user_rel() (elliot.evaluation.relevance.relevance.DiscountedRelevance method),

31
 get_user_rel_gains() (elliot.evaluation.relevance.relevance.BinaryRelevance method), 31
 get_user_rel_gains() (elliot.evaluation.relevance.relevance.DiscountedRelevance method), 31
 get_validation() (elliot.dataset.data_loader.knowledge_aware_chains.KnowledgeChainsDataObject method), 5
 get_validation() (elliot.dataset.data_loader.visual_data_loader.VisualDataObject method), 6
 get_validation() (elliot.dataset.dataset.DataSet method), 9
 GiniIndex (class in elliot.evaluation.metrics.diversity.gini_index.gini_index), 20
 GMF (class in elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization), 76

H

handle_hierarchy() (elliot.splitter.base_splitter.Splitter method), 91
 HR (class in elliot.evaluation.metrics.accuracy.hit_rate.hit_rate), 12
 HyperParameterStudy (class in elliot.result_handler.result_handler), 90

I

infer() (elliot.recommender.gan.CFGAN.cfgan_model.Generator method), 50
 init() (in module elliot.utils.logging), 92
 init_charger() (in module elliot.recommender.base_recommender_model), 89
 initialize() (elliot.recommender.algebraic.slope_one.slope_one_model.SlopeOneModel method), 43
 initialize() (elliot.recommender.content_based.VSM.vector_space_model_similarity.Similarity method), 47
 initialize() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.MF method), 56
 initialize() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.MF method), 60
 initialize() (elliot.recommender.NN.attribute_item_knn.attribute_item_knn_similarity.Similarity method), 34
 initialize() (elliot.recommender.NN.attribute_user_knn.attribute_user_knn_similarity.Similarity method), 35
 initialize() (elliot.recommender.NN.item_knn.aiolli_ferrari.AiolliSimilarity method), 36

initialize() (elliot.recommender.NN.item_knn.item_knn_similarity.Similarity method), 37
 initialize() (elliot.recommender.NN.user_knn.aiolli_ferrari.AiolliSimilarity method), 37
 initialize() (elliot.recommender.NN.user_knn.user_knn_similarity.Similarity method), 38
 IRGAN (class in elliot.recommender.gan.IRGAN.ircgan), 50
 IRGAN_model (class in elliot.recommender.gan.IRGAN.ircgan_model), 52
 ItemAutoRec (class in elliot.recommender.neural.ItemAutoRec.itemautorec), 77
 ItemAutoRecModel (class in elliot.recommender.neural.ItemAutoRec.itemautorec_model), 77
 ItemCoverage (class in elliot.evaluation.metrics.coverage.item_coverage.item_coverage), 18
 ItemKNN (class in elliot.recommender.NN.item_knn.item_knn), 36
 ItemMADranking (class in elliot.evaluation.metrics.fairness.MAD.ItemMADranking), 23
 ItemMADrating (class in elliot.evaluation.metrics.fairness.MAD.ItemMADrating), 23

K

KaHFM (class in elliot.recommender.knowledge_aware.kaHFM.ka_hfm), 56
 KaHFM_model (class in elliot.recommender.knowledge_aware.kaHFM_batch.ka_hfm_batch_model), 57
 KaHFMBatch (class in elliot.recommender.knowledge_aware.kaHFM_batch.ka_hfm_batch), 57
 KaHFMEmbeddings (class in elliot.recommender.knowledge_aware.ka_hfm_embeddings.ka_hfm_embeddings), 58
 KaHFMEmbeddingsModel (class in elliot.recommender.knowledge_aware.ka_hfm_embeddings.ka_hfm_embeddings_model), 58
 KnowledgeChainsDataObject (class in elliot.dataset.data_loader.knowledge_aware_chains), 5
 KnowledgeChainsLoader (class in elliot.dataset.data_loader.knowledge_aware_chains_loader), 5
 LatentFactor (class in elliot.recommender.latent_factor_models.CML.CML_model), 62
 LatentFactor (class in elliot.recommender.latent_factor_models.CML.CML_model), 62

t_factor_models.FISM.FISM_model), 64	MF (class in elliot.recommender.latent_factor_ - models.BPRMF.BPRMF), 59
LatentFactor (class in elliot.recommender.neu- ral.NAIS.nais_model), 78	MF (class in elliot.recommender.latent_factor_ - models.MF.matrix_factorization), 66
LAUC (class in elliot.evaluation.metrics.accuracy.AUC.lauc), 11	MLPComponent (class in elliot.recom- mender.neural.ConvMF.convolutional_matrix_factoriza- 73
LightGCN (class in elliot.recommender.- graph_based.lightgcn.LightGCN), 54	MLPComponent (class in elliot.recom- mender.neural.ConvNeuMF.convolutional_neural_mat- 74
LightGCNModel (class in elliot.recommender.- graph_based.lightgcn.LightGCN_model), 54	ModelCoordinator (class in elliot.hyperopti- mization.model_coordinator), 32
load_attribute_file() (elliot.dataset.dataloader.- knowledge_aware_chains.KnowledgeChainsLoad- method), 5	models() (elliot.namespace.namespace_mod- el_builder.Builder method), 33
load_dataset_dataframe() (elliot.dataset.dat- aloader.knowledge_aware_chains.KnowledgeChainsLoad- method), 6	models() (elliot.namespace.namespace_mod- el_builder.NameSpaceBuilder method), 33
load_dataset_dataframe() (elliot.dataset.dat- aloader.visual_data_loader.VisualLoader method), 6	module elliot, 93 elliot.dataset, 9 elliot.dataset.abstract_dataset, 8 elliot.dataset.data_loader, 7 elliot.dataset.data_loader.knowledge_aware_chains, 5 elliot.dataset.data_loader.visual_data_loader, 6
load_dataset_dict() (elliot.dataset.data_loader.- knowledge_aware_chains.KnowledgeChainsLoad- method), 6	elliot.dataset.dataset, 9 elliot.dataset.samplers, 8 elliot.dataset.samplers.custom_pointwise_sparse_sampler, 7 elliot.dataset.samplers.custom_sampler, 7 elliot.dataset.samplers.custom_sparse_sampler, 7 elliot.dataset.samplers.pairwise_sampler, 7 elliot.dataset.samplers.pipeline_sampler, 7 elliot.dataset.samplers.pointwise_cfgan_sampler, 7 elliot.dataset.samplers.pointwise_pos_neg_ratings_sampler, 8 elliot.dataset.samplers.pointwise_pos_neg_ratio_ratings_sampler, 8 elliot.dataset.samplers.pointwise_pos_neg_sampler, 8 elliot.dataset.samplers.pointwise_wide_and_deep_sampler, 8 elliot.dataset.samplers.sparse_sampler, 8 elliot.evaluation, 31 elliot.evaluation.evaluator, 31 elliot.evaluation.metrics, 30 elliot.evaluation.metrics.accuracy, 15 elliot.evaluation.metrics.accuracy.AUC, 11 elliot.evaluation.metrics.accuracy.AUC.auc, 9 elliot.evaluation.metrics.accuracy.AUC.gauc, 10 elliot.evaluation.metrics.accuracy.AUC.lauc, 10 elliot.evaluation.metrics.accuracy.DSC, 11
load_feature_names() (elliot.dataset.data_loader.- knowledge_aware_chains.KnowledgeChainsLoad- method), 6	
load_item_set() (elliot.dataset.data_loader.- knowledge_aware_chains.KnowledgeChainsLoad- method), 6	
load_obj() (in module elliot.utils.read), 92	
load_properties() (elliot.dataset.data_loader.- knowledge_aware_chains.KnowledgeChainsLoad- method), 6	
logarithmic_ranking_discount() (elliot.evaluation.relevance.relevance.AbstractRelevanceSingleItemRelevanceDiscounter static method), 30	
LogisticMatrixFactorization (class in elliot.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization_model), 66	
LogisticMatrixFactorizationModel (class in elliot.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization_model), 66	

M

MAE (class in elliot.evaluation.metrics.rating.- mae.mae), 27
manage_directories() (in module elliot.utils.- folder), 91
MAP (class in elliot.evaluation.metrics.accuracy.map.map), 12
MAR (class in elliot.evaluation.metrics.accuracy.mar.mar), 13
MatrixFactorizationModel (class in elliot.recommender.latent_factor_models.MF.matrix_factorization_model), 67
MF (class in elliot.recommender.knowledge_aware.kaHFM.ka_hfm), 56

elliot.evaluation.metrics.accuracy.DSC.dsc, 11
 elliot.evaluation.metrics.accuracy.f1, 12
 elliot.evaluation.metrics.accuracy.f1.extended_f1, 11
 elliot.evaluation.metrics.accuracy.f1.f1, 12
 elliot.evaluation.metrics.accuracy.hit_rate, 12
 elliot.evaluation.metrics.accuracy.hit_rate.hit_rate, 12
 elliot.evaluation.metrics.accuracy.map, 13
 elliot.evaluation.metrics.accuracy.map.map, 12
 elliot.evaluation.metrics.accuracy.mar, 13
 elliot.evaluation.metrics.accuracy.mar.mar, 13
 elliot.evaluation.metrics.accuracy.mrr, 13
 elliot.evaluation.metrics.accuracy.mrr.mrr, 13
 elliot.evaluation.metrics.accuracy.ndcg, 14
 elliot.evaluation.metrics.accuracy.ndcg.ndcg, 13
 elliot.evaluation.metrics.accuracy.precision, 14
 elliot.evaluation.metrics.accuracy.precision.precision, 14
 elliot.evaluation.metrics.accuracy.recall, 15
 elliot.evaluation.metrics.accuracy.recall.recall, 15
 elliot.evaluation.metrics.base_metric, 29
 elliot.evaluation.metrics.bias, 18
 elliot.evaluation.metrics.bias.aclt, 15
 elliot.evaluation.metrics.bias.aclt.aclt, 15
 elliot.evaluation.metrics.bias.aplt, 16
 elliot.evaluation.metrics.bias.aplt.aplt, 16
 elliot.evaluation.metrics.bias.arp, 16
 elliot.evaluation.metrics.bias.arp.arp, 16
 elliot.evaluation.metrics.bias.pop_reo, 17
 elliot.evaluation.metrics.bias.pop_reo.extended_pop_reo, 16
 elliot.evaluation.metrics.bias.pop_reo.pop_reo, 17
 elliot.evaluation.metrics.bias.pop_rsp, 18
 elliot.evaluation.metrics.bias.pop_rsp.extended_pop_rsp, 17
 elliot.evaluation.metrics.bias.pop_rsp.pop_rsp, 18
 elliot.evaluation.metrics.coverage, 20
 elliot.evaluation.metrics.coverage.item_coverage, 19
 elliot.evaluation.metrics.coverage.item_coverage.item_coverage, 18
 elliot.evaluation.metrics.coverage.num_retrieved, 19
 elliot.evaluation.metrics.coverage.num_retrieved.num_retrieved, 19
 elliot.evaluation.metrics.coverage.user_coverage, 20
 elliot.evaluation.metrics.coverage.user_coverage.user_coverage, 19
 elliot.evaluation.metrics.coverage.user_coverage.user_coverage, 19
 elliot.evaluation.metrics.diversity, 21
 elliot.evaluation.metrics.diversity.gini_index, 21
 elliot.evaluation.metrics.diversity.gini_index.gini_index, 20
 elliot.evaluation.metrics.diversity.shannon_entropy, 21
 elliot.evaluation.metrics.diversity.shannon_entropy.shannon_entropy, 21
 elliot.evaluation.metrics.diversity.SRecall, 20
 elliot.evaluation.metrics.diversity.SRecall.srecall, 20
 elliot.evaluation.metrics.fairness, 26
 elliot.evaluation.metrics.fairness.BiasDisparity, 23
 elliot.evaluation.metrics.fairness.BiasDisparity.BiasDisparity, 21
 elliot.evaluation.metrics.fairness.BiasDisparity.BiasDisparity, 22
 elliot.evaluation.metrics.fairness.BiasDisparity.BiasDisparity, 22
 elliot.evaluation.metrics.fairness.MAD, 24
 elliot.evaluation.metrics.fairness.MAD.ItemMADRanking, 23
 elliot.evaluation.metrics.fairness.MAD.ItemMADRating, 23
 elliot.evaluation.metrics.fairness.MAD.UserMADRanking, 23
 elliot.evaluation.metrics.fairness.MAD.UserMADRating, 24
 elliot.evaluation.metrics.fairness.reo, 25
 elliot.evaluation.metrics.fairness.reo.reo, 24
 elliot.evaluation.metrics.fairness.rsp, 26
 elliot.evaluation.metrics.fairness.rsp.rsp, 25
 elliot.evaluation.metrics.metrics_utils, 29
 elliot.evaluation.metrics.novelty, 27
 elliot.evaluation.metrics.novelty.EFD, 26
 elliot.evaluation.metrics.novelty.EFD.efd, 26
 elliot.evaluation.metrics.novelty.EFD.extended_efd, 26
 elliot.evaluation.metrics.novelty.EPC, 27
 elliot.evaluation.metrics.novelty.EPC.epc, 27
 elliot.evaluation.metrics.novelty.EPC.extended_epc, 27
 elliot.evaluation.metrics.rating, 29
 elliot.evaluation.metrics.rating.mae, 28
 elliot.evaluation.metrics.rating.mae.mae, 27
 elliot.evaluation.metrics.rating.mse, 28
 elliot.evaluation.metrics.rating.mse.mse, 28
 elliot.evaluation.metrics.rating.rmse, 29

elliot.evaluation.metrics.rating.rmse.rmse, 28
 elliot.evaluation.metrics.statistical_array_metric, 29
 elliot.evaluation.popularity_utils, 30
 elliot.evaluation.popularity_utils.popularity, 30
 elliot.evaluation.relevance, 31
 elliot.evaluation.relevance.relevance, 30
 elliot.evaluation.statistical_significance, 31
 elliot.hyperoptimization, 32
 elliot.hyperoptimization.model_coordinator, 32
 elliot.namespace, 33
 elliot.namespace.namespace_model, 32
 elliot.namespace.namespace_model_builder, 32
 elliot.prefiltering, 33
 elliot.prefiltering.standard_prefilters, 33
 elliot.recommender, 90
 elliot.recommender.adversarial, 43
 elliot.recommender.adversarial.AMF, 41
 elliot.recommender.adversarial.AMF.AMF, 38
 elliot.recommender.adversarial.AMF.AMF_model, 39
 elliot.recommender.adversarial.AMR, 43
 elliot.recommender.adversarial.AMR.AMR, 41
 elliot.recommender.adversarial.AMR.AMR_model, 41
 elliot.recommender.algebraic, 44
 elliot.recommender.algebraic.slope_one, 43
 elliot.recommender.algebraic.slope_one.slope_one, 43
 elliot.recommender.algebraic.slope_one.slope_one_model, 43
 elliot.recommender.attentive, 44
 elliot.recommender.attentive.afm, 44
 elliot.recommender.attentive.afm.afm, 44
 elliot.recommender.attentive.afm.afm_model, 44
 elliot.recommender.autoencoders, 46
 elliot.recommender.autoencoders.dae, 45
 elliot.recommender.autoencoders.dae.multi_dae, 44
 elliot.recommender.autoencoders.dae.multi_dae_model, 45
 elliot.recommender.autoencoders.vae, 46
 elliot.recommender.autoencoders.vae.multi_vae, 45
 elliot.recommender.autoencoders.vae.multi_vae_model, 46
 elliot.recommender.base_recommender_model, 89
 elliot.recommender.content_based, 47
 elliot.recommender.content_based.VSM, 47
 elliot.recommender.content_based.VSM.tfidf_utils, 47
 elliot.recommender.content_based.VSM.vector_space_model, 47
 elliot.recommender.content_based.VSM.vector_space_model_builder, 47
 elliot.recommender.gan, 54
 elliot.recommender.gan.CFGAN, 50
 elliot.recommender.gan.CFGAN.cfgan, 48
 elliot.recommender.gan.CFGAN.cfgan_model, 48
 elliot.recommender.gan.IRGAN, 54
 elliot.recommender.gan.IRGAN.irgan, 50
 elliot.recommender.gan.IRGAN.irgan_model, 50
 elliot.recommender.graph_based, 56
 elliot.recommender.graph_based.lightgcn, 55
 elliot.recommender.graph_based.lightgcn.LightGCN, 54
 elliot.recommender.graph_based.lightgcn.LightGCN_model, 54
 elliot.recommender.graph_based.ngcf, 55
 elliot.recommender.graph_based.ngcf.NGCF, 55
 elliot.recommender.graph_based.ngcf.NGCF_model, 55
 elliot.recommender.knowledge_aware, 59
 elliot.recommender.knowledge_aware.kaHFM, 57
 elliot.recommender.knowledge_aware.kaHFM.ka_hfm, 56
 elliot.recommender.knowledge_aware.kaHFM.tfidf_utils, 57
 elliot.recommender.knowledge_aware.kaHFM_batch, 58
 elliot.recommender.knowledge_aware.kaHFM_batch.ka_hfm, 57
 elliot.recommender.knowledge_aware.kaHFM_batch.ka_hfm_model, 57
 elliot.recommender.knowledge_aware.kaHFM_batch.tfidf_utils, 58
 elliot.recommender.knowledge_aware.kahfm_embeddings, 59
 elliot.recommender.knowledge_aware.kahfm_embeddings.L, 58
 elliot.recommender.knowledge_aware.kahfm_embeddings.L_batch, 58
 elliot.recommender.knowledge_aware.kahfm_embeddings.L_batch.L, 59
 elliot.recommender.latent_factor_models, 72
 elliot.recommender.latent_factor_models.BPRMF, 60
 elliot.recommender.latent_factor_models.BPRMF.BPRMF, 59
 elliot.recommender.latent_factor_models.BPRMF_batch, 61
 elliot.recommender.latent_factor_models.BPRMF_batch.BPRMF, 61

60	69
60	68
61	68
61	69
61	69
62	69
61	71
62	70
63	71
62	70
63	71
62	70
63	72
63	71
65	71
64	72
64	72
66	72
65	74
65	74
66	75
66	76
66	76
67	75
66	75
67	77
68	75
67	78
67	
67	
67	

elliott.recommender.neural.ItemAutoRec.itemautorec,	elliott.recommender.NN.user_knn.aiolli_ferrari,
77	37
elliott.recommender.neural.ItemAutoRec.itemautorec_model,	elliott.recommender.NN.user_knn.user_knn,
77	38
elliott.recommender.neural.NAIS,	elliott.recommender.NN.user_knn.user_knn_similarity,
79	38
elliott.recommender.neural.NAIS.nais,	elliott.recommender.recommender_utils_mixin,
78	89
elliott.recommender.neural.NAIS.nais_model,	elliott.recommender.unpersonalized,
81	85
elliott.recommender.neural.NeuMF,	elliott.recommender.unpersonalized.most_popular,
80	84
elliott.recommender.neural.NeuMF.neural_matrix_factorization,	elliott.recommender.unpersonalized.most_popular.most_popular,
81	84
elliott.recommender.neural.NFM,	elliott.recommender.unpersonalized.random_recommender,
80	85
elliott.recommender.neural.NFM.neural_fm,	elliott.recommender.unpersonalized.random_recommender_model,
79	85
elliott.recommender.neural.NFM.neural_fm_model,	elliott.recommender.visual_recommenders,
79	89
elliott.recommender.neural.NPR,	elliott.recommender.visual_recommenders.ACF,
80	85
elliott.recommender.neural.NPR.neural_personalized_recommender,	elliott.recommender.visual_recommenders.ACF.ACF,
80	85
elliott.recommender.neural.NPR.neural_personalized_recommender_model,	elliott.recommender.visual_recommenders.ACF.ACF_model,
82	85
elliott.recommender.neural.UserAutoRec,	elliott.recommender.visual_recommenders.DeepStyle,
81	87
elliott.recommender.neural.UserAutoRec.userautorec,	elliott.recommender.visual_recommenders.DeepStyle.DeepStyle,
81	87
elliott.recommender.neural.UserAutoRec.userautorec_model,	elliott.recommender.visual_recommenders.DeepStyle.DeepStyle_model,
81	87
elliott.recommender.neural.WideAndDeep,	elliott.recommender.visual_recommenders.DVBPR,
84	86
elliott.recommender.neural.WideAndDeep.wide_and_deep,	elliott.recommender.visual_recommenders.DVBPR.DVBPR,
82	86
elliott.recommender.neural.WideAndDeep.wide_and_deep_model,	elliott.recommender.visual_recommenders.DVBPR.DVBPR_model,
82	86
elliott.recommender.NN,	elliott.recommender.visual_recommenders.VBPR,
38	88
elliott.recommender.NN.attribute_item_knn,	elliott.recommender.visual_recommenders.VBPR.VBPR,
34	87
elliott.recommender.NN.attribute_item_knn.attribute_item_knn,	elliott.recommender.visual_recommenders.VBPR.VBPR_model,
34	87
elliott.recommender.NN.attribute_item_knn.attribute_item_knn_similarity,	elliott.recommender.visual_recommenders.VNPR,
34	88
elliott.recommender.NN.attribute_user_knn,	elliott.recommender.visual_recommenders.VNPR.visual_neural,
35	88
elliott.recommender.NN.attribute_user_knn.attribute_user_knn,	elliott.recommender.visual_recommenders.VNPR.visual_neural_model,
35	88
elliott.recommender.NN.attribute_user_knn.attribute_user_knn_similarity,	elliott.result_handler,
35	90
elliott.recommender.NN.attribute_user_knn.tfidf_utils,	elliott.result_handler.result_handler,
35	90
elliott.recommender.NN.item_knn,	elliott.run,
37	93
elliott.recommender.NN.item_knn.aiolli_ferrari,	elliott.splitter,
36	91
elliott.recommender.NN.item_knn.item_knn,	elliott.splitter.base_splitter,
36	90
elliott.recommender.NN.item_knn.item_knn_similarity,	elliott.utils,
36	93
elliott.recommender.NN.user_knn,	elliott.utils.folder,
38	91

[elliot.utils.logger_util](#), 91
[elliot.utils.logging](#), 92
[elliot.utils.read](#), 92
[elliot.utils.write](#), 93
[MostPop](#) (class in [elliot.recommender.unpersonalized.most_popular.most_popular](#)), 84
[MRR](#) (class in [elliot.evaluation.metrics.accuracy.mrr.mrr](#)), 13
[MSE](#) (class in [elliot.evaluation.metrics.rating.mse.mse](#)), 28
[MultiDAE](#) (class in [elliot.recommender.autoencoders.dae.multi_dae](#)), 44
[MultiVAE](#) (class in [elliot.recommender.autoencoders.vae.multi_vae](#)), 45

N

[NAIS](#) (class in [elliot.recommender.neural.NAIS.nais](#)), 78
[NAIS_model](#) (class in [elliot.recommender.neural.NAIS.nais_model](#)), 78
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.AUC.auc.AUC](#) static method), 10
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.AUC.gauc.GAUC](#) static method), 10
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.AUC.lauc.LAUC](#) static method), 11
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.DSC.dsc.DSC](#) static method), 11
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.f1.extended_f1.ExtendedF1](#) static method), 11
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.f1.f1.F1](#) static method), 12
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.hit_rate.hit_rate.HR](#) static method), 12
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.map.map.MAP](#) static method), 13
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.mar.mar.MAR](#) static method), 13
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.mrr.mrr.MRR](#) static method), 13
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.ndcg.ndcg.NDCG](#) static method), 14
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.precision.precision.Precision](#) static method), 14
[name\(\)](#) ([elliot.evaluation.metrics.accuracy.recall.recall.Recall](#) static method), 15
[name\(\)](#) ([elliot.evaluation.metrics.base_metric.BaseMetric](#) method), 29
[name\(\)](#) ([elliot.evaluation.metrics.bias.aclt.aclt.ACLT](#) static method), 15
[name\(\)](#) ([elliot.evaluation.metrics.s.bias.aplt.aplt.APLT](#) static method), 16

[name\(\)](#) ([elliot.evaluation.metrics.s.bias.arp.arp.ARP](#) static method), 16
[name\(\)](#) ([elliot.evaluation.metrics.bias.pop_reo.extended_pop_reo.ExtendedPopREO](#) static method), 17
[name\(\)](#) ([elliot.evaluation.metrics.bias.pop_reo.pop_reo.PopREO](#) static method), 17
[name\(\)](#) ([elliot.evaluation.metrics.bias.pop_rsp.extended_pop_rsp.ExtendedPopRSP](#) static method), 18
[name\(\)](#) ([elliot.evaluation.metrics.bias.pop_rsp.pop_rsp.PopRSP](#) static method), 18
[name\(\)](#) ([elliot.evaluation.metrics.coverage.item_coverage.item_coverage.ItemCoverage](#) static method), 19
[name\(\)](#) ([elliot.evaluation.metrics.coverage.num_retrieved.num_retrieved.NumRetrieved](#) static method), 19
[name\(\)](#) ([elliot.evaluation.metrics.coverage.user_coverage.user_coverage.UserCoverage](#) static method), 19
[name\(\)](#) ([elliot.evaluation.metrics.coverage.user_coverage.user_coverage_at_n.UserCoverageAtN](#) static method), 20
[name\(\)](#) ([elliot.evaluation.metrics.diversity.gini_index.gini_index.GiniIndex](#) static method), 21
[name\(\)](#) ([elliot.evaluation.metrics.diversity.shannon_entropy.shannon_entropy.ShannonEntropy](#) static method), 21
[name\(\)](#) ([elliot.evaluation.metrics.diversity.srecall.srecall.SRecall](#) static method), 20
[name\(\)](#) ([elliot.evaluation.metrics.fairness.bias_disparity.bias_disparity.BD.BiasDisparityBD](#) method), 21
[name\(\)](#) ([elliot.evaluation.metrics.fairness.bias_disparity.bias_disparity.BR.BiasDisparityBR](#) method), 22
[name\(\)](#) ([elliot.evaluation.metrics.fairness.bias_disparity.bias_disparity.BS.BiasDisparityBS](#) method), 23
[name\(\)](#) ([elliot.evaluation.metrics.fairness.item_madranking.item_madranking.ItemMADRanking](#) method), 23
[name\(\)](#) ([elliot.evaluation.metrics.fairness.item_madrating.item_madrating.ItemMADRating](#) method), 23
[name\(\)](#) ([elliot.evaluation.metrics.fairness.item_madranking.item_madranking.UserMADRanking](#) method), 24
[name\(\)](#) ([elliot.evaluation.metrics.fairness.item_madrating.item_madrating.UserMADRating](#) method), 24
[name\(\)](#) ([elliot.evaluation.metrics.fairness.reo.reo.REO](#) method), 25
[name\(\)](#) ([elliot.evaluation.metrics.fairness.rsp.rsp.RSP](#) method), 26
[name\(\)](#) ([elliot.evaluation.metrics.metrics_util](#)

s.ProxyMetric method), 29

name() (elliot.evaluation.metrics.metrics_util-
s.ProxyStatisticalMetric method), 29

name() (elliot.evaluation.metrics.novel-
ty.EFD.efd.EFD static method), 26

name() (elliot.evaluation.metrics.novel-
ty.EFD.extended_efd.ExtendedEFD
static method), 26

name() (elliot.evaluation.metrics.novel-
ty.EPC.epc.EPC static method), 27

name() (elliot.evaluation.metrics.novel-
ty.EPC.extended_epc.ExtendedEPC
static method), 27

name() (elliot.evaluation.metrics.rating.mae.-
mae.MAE static method), 28

name() (elliot.evaluation.metrics.rat-
ing.mse.mse.MSE static method), 28

name() (elliot.evaluation.metrics.rat-
ing.rmse.rmse.RMSE static method),
29

name() (elliot.recommender.adversarial.AM-
F.AMF.AMF property), 39

name() (elliot.recommender.adversarial.AM-
R.AMR.AMR property), 41

name() (elliot.recommender.algebraic.s-
lope_one.slope_one.SlopeOne proper-
ty), 43

name() (elliot.recommender.atten-
tive.afm.afm.AFM property), 44

name() (elliot.recommender.autoencoders.dae.-
multi_dae.MultiDAE property), 44

name() (elliot.recommender.autoencoders.vae.-
multi_vae.MultiVAE property), 45

name() (elliot.recommender.content_based.VS-
M.vector_space_model.VSM proper-
ty), 47

name() (elliot.recommender.gan.CFGAN.cf-
gan.CFGAN property), 48

name() (elliot.recommender.gan.IRGAN.ir-
gan.IRGAN property), 50

name() (elliot.recommender.graph_based.light-
gcn.LightGCN.LightGCN property),
54

name() (elliot.recommender.-
graph_based.ngcf.NGCF.NGCF prop-
erty), 55

name() (elliot.recommender.knowl-
edge_aware.kaHFM.ka_hfm.KaHFM
property), 56

name() (elliot.recommender.knowl-
edge_aware.kaHFM.ka_hfm.MF prop-
erty), 56

name() (elliot.recommender.knowl-
edge_aware.kaHFM_batch.kahfm_batch.KaHFMBatchy), 57

name() (elliot.recommender.knowl-
edge_aware.kahfm_embeddings.kahfm_embeddings.KaHFMEmbeddings
property), 58

name() (elliot.recommender.latent_factor_mod-
els.BPRMF.BPRMF.BPRMF property),
59

name() (elliot.recommender.latent_factor_mod-
els.BPRMF.BPRMF.MF property), 60

name() (elliot.recommender.latent_factor_mod-
els.BPRMF_batch.BPRMF_batch.BPRMF_batch
property), 60

name() (elliot.recommender.latent_factor_mod-
els.BPRSlim.bprslim.BPRSlim proper-
ty), 61

name() (elliot.recommender.latent_factor_mod-
els.CML.CML.CML property), 62

name() (elliot.recommender.latent_factor_mod-
els.FFM.field_aware_factorization_machine.FFM
property), 63

name() (elliot.recommender.latent_factor_mod-
els.FISM.FISM.FISM property), 63

name() (elliot.recommender.latent_factor_mod-
els.FM.factorization_machine.FM
property), 64

name() (elliot.recommender.latent_factor_mod-
els.FunkSVD.funk_svd.FunkSVD
property), 65

name() (elliot.recommender.latent_factor_mod-
els.LogisticMF.logistic_matrix_factorization.LogisticMa
property), 66

name() (elliot.recommender.latent_factor_mod-
els.MF.matrix_factorization.MF prop-
erty), 66

name() (elliot.recommender.latent_factor_mod-
els.NonNegMF.non_negative_matrix_factorization.Non
property), 67

name() (elliot.recommender.latent_factor_mod-
els.PMF.probabilistic_matrix_factorization.PMF
property), 68

name() (elliot.recommender.latent_factor_mod-
els.PureSVD.pure_svd.PureSVD prop-
erty), 69

name() (elliot.recommender.latent_factor_mod-
els.Slim.slim.Slim property), 70

name() (elliot.recommender.latent_factor_mod-
els.SVDpp.svdpp.SVDpp property), 70

name() (elliot.recommender.latent_factor_mod-
els.WRMF.wrmf.WRMF property), 71

name() (elliot.recommender.neural.ConvMF.-
convolutional_matrix_factorization.ConvMF
property), 72

name() (elliot.recommender.neural.ConvNeuM-
F.convolutional_neural_matrix_factorization.ConvNeu
property), 74

name() (elliot.recommender.neu-
ral.DeepFM.deep_fm.DeepFM proper-
ty), 75

name() (elliot.recommender.neural.DMF.deep_-
matrix_factorization.DMF property),
75

name() (elliot.recommender.neural.General-

izedMF.generalized_matrix_factorization.GMF property), 76

name() (elliot.recommender.neural.ItemAutoRec.itemautorec.ItemAutoRec property), 77

name() (elliot.recommender.neural.NAIS.nais.-NAIS property), 78

name() (elliot.recommender.neural.NeuMF.neural_matrix_factorization.NeuMF property), 80

name() (elliot.recommender.neural.NFM.neural_fm.NFM property), 79

name() (elliot.recommender.neural.NPR.neural_personalized_ranking.NPR property), 80

name() (elliot.recommender.neural.UserAutoRec.userautorec.UserAutoRec property), 81

name() (elliot.recommender.neural.WideAndDeep.wide_and_deep.WideAndDeep property), 82

name() (elliot.recommender.NN.attribute_item_knn.attribute_item_knn.AttributeItemKNN property), 34

name() (elliot.recommender.NN.attribute_user_knn.attribute_user_knn.AttributeUserKNN property), 35

name() (elliot.recommender.NN.item_knn.item_knn.ItemKNN property), 36

name() (elliot.recommender.NN.user_knn.user_knn.UserKNN property), 38

name() (elliot.recommender.unpersonalized.-most_popular.most_popular.MostPop property), 84

name() (elliot.recommender.unpersonalized.random_recommender.Random.Random property), 85

name() (elliot.recommender.visual_recommenders.ACF.ACF.ACF property), 85

name() (elliot.recommender.visual_recommenders.DeepStyle.DeepStyle.DeepStyle property), 87

name() (elliot.recommender.visual_recommenders.DVBPR.DVBPR.DVBPR property), 86

name() (elliot.recommender.visual_recommenders.VBPR.VBPR.VBPR property), 87

name() (elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking.VNPR property), 88

NameSpaceBuilder (class in elliot.namespace.-namespace_model_builder), 33

NameSpaceModel (class in elliot.namespace.-namespace_model), 32

NDCG (class in elliot.evaluation.metrics.accuracy.ndcg.ndcg), 14

needs_full_recommendations() (elliot.evaluation.metrics.accuracy.AUC.auc.AUC static method), 10

needs_full_recommendations() (elliot.evaluation.metrics.accuracy.AUC.gauc.GAUC static method), 10

needs_full_recommendations() (elliot.evaluation.metrics.base_metric.BaseMetric static method), 29

needs_full_recommendations() (elliot.evaluation.metrics.metrics_utils.ProxyMetric method), 29

needs_full_recommendations() (elliot.evaluation.metrics.metrics_utils.ProxyStatisticalMetric method), 29

needs_full_recommendations() (elliot.evaluation.metrics.rating.mae.mae.MAE static method), 28

needs_full_recommendations() (elliot.evaluation.metrics.rating.mse.mse.MSE static method), 28

needs_full_recommendations() (elliot.evaluation.metrics.rating.rmse.rmse.RMSE static method), 29

NeuMF (class in elliot.recommender.neural.NeuMF.neural_matrix_factorization), 80

NeuralFactorizationMachineModel (class in elliot.recommender.neural.NFM.neural_fm_model), 79

NeuralMatrixFactorizationModel (class in elliot.recommender.neural.NeuMF.neural_matrix_factorization), 81

NFM (class in elliot.recommender.neural.NFM.neural_fm), 79

NGCF (class in elliot.recommender.-graph_based.ngcf.NGCF), 55

NGCFModel (class in elliot.recommender.-graph_based.ngcf.NGCF_model), 55

NonNegMF (class in elliot.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization), 67

NonNegMFModel (class in elliot.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization), 68

NPR (class in elliot.recommender.neural.NPR.neural_personalized_ranking), 80

NPRModel (class in elliot.recommender.neural.NPR.neural_personalized_ranking_model), 80

NonNegVNPR (class in elliot.evaluation.metrics.coverage.num_retrieved.num_retrieved), 19

O

objective() (elliot.hyperoptimization.model_coordinator.ModelCoordinator method), 32

P

PairedTTest (class in `elliott.evaluation.statistical_significance`), 31

PairedTTest (`elliott.result_handler.result_handler.StatTest` attribute), 90

`parse_algorithms()` (in module `elliott.hyperoptimization`), 32

`parse_metric()` (in module `elliott.evaluation.metrics`), 30

`parse_metrics()` (in module `elliott.evaluation.metrics`), 30

`pipeline()` (`elliott.dataset.sampler.s.pipeline_sampler.Sampler` method), 7

`pipeline_eval()` (`elliott.dataset.sampler.s.pipeline_sampler.Sampler` method), 7

PMF (class in `elliott.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization`), 68

PopREO (class in `elliott.evaluation.metrics.bias.pop_reo.pop_reo`), 17

PopRSP (class in `elliott.evaluation.metrics.bias.pop_rsp.pop_rsp`), 18

Popularity (class in `elliott.evaluation.popularity_utils.popularity`), 30

`pre_train_discriminator()` (`elliott.recommender.gan.IRGAN.iran_model.IRGAN_model` method), 52

`pre_train_generator()` (`elliott.recommender.gan.IRGAN.iran_model.IRGAN_model` method), 52

Precision (class in `elliott.evaluation.metrics.accuracy.precision.precision`), 14

`predict()` (`elliott.recommender.adversarial.AMF.AMF_model.AMF_model` method), 39

`predict()` (`elliott.recommender.adversarial.AMR.AMR_model.AMR_model` method), 41

`predict()` (`elliott.recommender.algebraic.slope_one.slope_one_model.SlopeOneModel` method), 43

`predict()` (`elliott.recommender.attentive.afm.afm_model.AFMMModel` method), 44

`predict()` (`elliott.recommender.autoencoders.dae.multi_dae_model.DenoisingAutoEncoder` method), 45

`predict()` (`elliott.recommender.autoencoders.vae.multi_vae_model.VariationalAutoEncoder` method), 46

`predict()` (`elliott.recommender.gan.CFGAN.cf_gan_model.CFGAN_model` method), 48

`predict()` (`elliott.recommender.gan.IRGAN.iran_model.IRGAN_model` method),

52

`predict()` (`elliott.recommender.graph_based.lightgcn.LightGCN_model.LightGCNModel` method), 54

`predict()` (`elliott.recommender.graph_based.ngcf.NGCF_model.NGCFModel` method), 55

`predict()` (`elliott.recommender.knowledge_aware.kaHFM.ka_hfm.KaHFM` method), 56

`predict()` (`elliott.recommender.knowledge_aware.kaHFM.ka_hfm.MF` method), 56

`predict()` (`elliott.recommender.knowledge_aware.kaHFM_batch.kahfm_batch_model.KaHFM` method), 57

`predict()` (`elliott.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings_m` method), 58

`predict()` (`elliott.recommender.latent_factor_models.BPRMF.BPRMF.BPRMF` method), 59

`predict()` (`elliott.recommender.latent_factor_models.BPRMF.BPRMF.MF` method), 60

`predict()` (`elliott.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch_model.BPRMF_b` method), 61

`predict()` (`elliott.recommender.latent_factor_models.BPRSlim.bprslim.BPRSlim` method), 61

`predict()` (`elliott.recommender.latent_factor_models.BPRSlim.bprslim_model.BPRSlimModel` method), 61

`predict()` (`elliott.recommender.latent_factor_models.CML.CML_model.CML_model` method), 62

`predict()` (`elliott.recommender.latent_factor_models.FFM.field_aware_factorization_machine.FFM` method), 63

`predict()` (`elliott.recommender.latent_factor_models.FFM.field_aware_factorization_machine_model` method), 63

`predict()` (`elliott.recommender.latent_factor_models.FISM.FISM_model.FISM_model` method), 64

`predict()` (`elliott.recommender.latent_factor_models.FM.factorization_machine.FM` method), 64

`predict()` (`elliott.recommender.latent_factor_models.FM.factorization_machine_model.Factorization` method), 65

`predict()` (`elliott.recommender.latent_factor_models.FunkSVD.funk_svd.FunkSVD` method), 65

`predict()` (`elliott.recommender.latent_factor_models.FunkSVD.funk_svd_model.FunkSVDModel` method), 65

predict() (elliot.recommender.latent_factor_ - method), 75
models.LogisticMF.logistic_matrix_factorization_model.LogisticMatrixFactorizationModel method), 66
predict() (elliot.recommender.latent_factor_ - models.MF.matrix_factorization.MF method), 66
predict() (elliot.recommender.latent_factor_ - models.MF.matrix_factorization_model.MatrixFactorizationModel method), 67
predict() (elliot.recommender.latent_factor_ - models.NonNegMF.non_negative_matrix_factorization_model.NonNegMF method), 67
predict() (elliot.recommender.latent_factor_ - models.NonNegMF.non_negative_matrix_factorization_model.NonNegMF method), 68
predict() (elliot.recommender.latent_factor_ - models.PMF.probabilistic_matrix_factorization.PMF method), 68
predict() (elliot.recommender.latent_factor_ - models.PMF.probabilistic_matrix_factorization_model.PMF method), 69
predict() (elliot.recommender.latent_factor_ - models.PureSVD.pure_svd.PureSVD method), 69
predict() (elliot.recommender.latent_factor_ - models.PureSVD.pure_svd_model.PureSVDModel method), 69
predict() (elliot.recommender.latent_factor_ - models.Slim.slim.Slim method), 70
predict() (elliot.recommender.latent_factor_ - models.Slim.slim_model.SlimModel method), 71
predict() (elliot.recommender.latent_factor_ - models.SVDpp.svdpp.SVDpp method), 70
predict() (elliot.recommender.latent_factor_ - models.SVDpp.svdpp_model.SVDppModel method), 70
predict() (elliot.recommender.latent_factor_ - models.WRMF.wrmf.WRMF method), 71
predict() (elliot.recommender.latent_factor_ - models.WRMF.wrmf_model.WRMFModel method), 71
predict() (elliot.recommender.neural.ConvMF-convolutional_matrix_factorization_model.ConvMatrixFactorizationModel method), 73
predict() (elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model.ConvNeuMF method), 74
predict() (elliot.recommender.neural.DeepFM.deep_fm.DeepFM method), 76
predict() (elliot.recommender.neural.DeepFM.deep_fm_model.DeepFMModel method), 76
predict() (elliot.recommender.neural.DM-F.deep_matrix_factorization_model.DeepMatrixFactorizationModel method), 75
predict() (elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization.GMF method), 76
predict() (elliot.recommender.neural.ItemAutoRec.itemautorec_model.ItemAutoRecModel method), 78
predict() (elliot.recommender.neural.NAIS-nais_model.NAIS_model method), 79
predict() (elliot.recommender.neural.NeuMF-NeuMF method), 81
predict() (elliot.recommender.neural.NFM.neural_fm_model.NFM method), 79
predict() (elliot.recommender.neural.NFM.neural_fm_model.NeuralFactorizationMachineModel method), 79
predict() (elliot.recommender.neural.NPR.neural_personalized_ranking_model.NPRModel method), 80
predict() (elliot.recommender.neural.NPR.neural_personalized_ranking_model.RecommBLSMF method), 80
predict() (elliot.recommender.neural.UserAutoRec.userautorec_model.UserAutoRecModel method), 82
predict() (elliot.recommender.neural.WideAndDeep.wide_and_deep_model.WideAndDeepModel method), 83
predict() (elliot.recommender.NN.item_kn-n.aiolli_ferrari.AiolliSimilarity method), 36
predict() (elliot.recommender.NN.user_kn-n.aiolli_ferrari.AiolliSimilarity method), 37
predict() (elliot.recommender.visual_recommenders.ACF.ACF_model.ACF_model method), 85
predict() (elliot.recommender.visual_recommenders.DeepStyle.DeepStyle_model.DeepStyle_model method), 87
predict() (elliot.recommender.visual_recommenders.VBPR.VBPR_model.VBPR_model method), 88
predict() (elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking_model.VNPR method), 88
predict_all() (elliot.recommender.knowledge_aware_kahfm_batch.kahfm_batch_model.KaHFM method), 57
predict_batch() (elliot.recommender.knowledge_aware_kahfm_batch.kahfm_batch_model.KaHFM method), 58
predict_batch() (elliot.recommender.knowledge_aware_kahfm_embeddings.kahfm_embeddings_model.kahfm_embeddings method), 58
predict_batch() (elliot.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization_model.LogisticMatrixFactorizationModel method), 66
predict_batch() (elliot.recommender.visual_recommenders.DVBPR.DVBPR_model.DVBPR_model method), 89

method), 86

PreFilter (class in `elliott.prefiltering.standard_prefilters`), 33

`prepare_logger()` (in module `elliott.utils.logging`), 92

ProbabilisticMatrixFactorizationModel (class in `elliott.recommender.latent_factor_models.PMF`), 68

`process()` (`elliott.evaluation.metrics.accuracy.f1.extended_f1.ExtendedF1` method), 11

`process()` (`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBD.BiasDisparityBD` method), 22

`process()` (`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBR.BiasDisparityBR` method), 22

`process()` (`elliott.evaluation.metrics.fairness.BiasDisparity.BiasDisparityBS.BiasDisparityBS` method), 23

`process()` (`elliott.evaluation.metrics.fairness.reo.reo.REO` method), 25

`process()` (`elliott.evaluation.metrics.fairness.rsp.rsp.RSP` method), 26

`process_cosine()` (`elliott.recommender.N-N.item_knn.item_knn_similarity.Similarity` method), 37

`process_similarity()` (`elliott.recommender.content_based.VSM.vector_space_model_similarity.Similarity` method), 47

`process_similarity()` (`elliott.recommender.N-N.attribute_item_knn.attribute_item_knn_similarity.Similarity` method), 34

`process_similarity()` (`elliott.recommender.N-N.attribute_user_knn.attribute_user_knn_similarity.Similarity` method), 35

`process_similarity()` (`elliott.recommender.N-N.item_knn.item_knn_similarity.Similarity` method), 37

`process_similarity()` (`elliott.recommender.N-N.user_knn.user_knn_similarity.Similarity` method), 38

`process_splitting()` (`elliott.splitter.base_splitter.Splitter` method), 91

ProxyMetric (class in `elliott.evaluation.metrics.metrics_utils`), 29

ProxyStatisticalMetric (class in `elliott.evaluation.metrics.metrics_utils`), 29

PureSVD (class in `elliott.recommender.latent_factor_models.PureSVD.pure_svd`), 69

PureSVDModel (class in `elliott.recommender.latent_factor_models.PureSVD.pure_svd_model`), 69

Q

QueueListenerHandler (class in `elliott.utils.logger_util`), 91

R

Random (class in `elliott.recommender.unpersonalized.random_recommender.Random`), 85

`read_config()` (in module `elliott.utils.read`), 92

`read_csv()` (in module `elliott.utils.read`), 92

`read_folder()` (`elliott.splitter.base_splitter.Splitter` method), 91

`read_image()` (`elliott.dataset.sampler.s.pipeline_sampler.Sampler` method), 7

`read_imagenet_classes_txt()` (in module `elliott.utils.read`), 92

`read_images()` (`elliott.dataset.dataloader.visual_dataloader.VisualDataObject` method), 6

`read_images_multiprocessing()` (`elliott.dataset.dataloader.visual_dataloader.VisualDataObject` method), 6

`read_images_triple()` (`elliott.dataset.sampler.s.pipeline_sampler.Sampler` method), 7

`read_multi_config()` (in module `elliott.utils.read`), 93

`read_np()` (in module `elliott.utils.read`), 93

`read_single_image()` (`elliott.dataset.dataloader.visual_dataloader.VisualDataObject` static method), 6

`read_splitting()` (`elliott.dataset.dataloader.knowledge_aware_chains.KnowledgeChainsLoader` method), 6

`read_splitting()` (`elliott.dataset.dataloader.visual_dataloader.VisualLoader` method), 8

`read_splitting()` (`elliott.dataset.dataset.DataSetLoader` method), 9

`rearrange_data()` (`elliott.splitter.base_splitter.Splitter` method), 91

Recall (class in `elliott.evaluation.metrics.accuracy.recall.recall`), 15

RecMixin (class in `elliott.recommender.recommender_utils.mixin`), 89

`reduce_attribute_map_property_selection()` (`elliott.dataset.dataloader.knowledge_aware_chains.KnowledgeChainsLoader` method), 6

`reduce_dataset_by_item_list()` (`elliott.dataset.dataloader.knowledge_aware_chains.KnowledgeChainsLoader` method), 6

`reduce_dataset_by_item_list()` (`elliott.dataset.dataloader.visual_dataloader.VisualLoader` method), 7

Relevance (class in `elliott.evaluation.relevance.relevance`), 31

REO (class in `elliott.evaluation.metrics.fairness.reo.reo`), 25

`required_attributes` (`elliott.dataset.abstract_dataset.AbstractDataset` attribute), 9

[restore_weights\(\) \(elliot.recommender.algebraic.slope_one.slope_one.SlopeOne method\), 43](#)
[restore_weights\(\) \(elliot.recommender.content_based.VSM.vector_space_model.VSM method\), 47](#)
[restore_weights\(\) \(elliot.recommender.knowledge_aware.kaHFM.ka_hfm.KaHFM method\), 56](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.BPRMF.BPRMF.BPRMF method\), 59](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch.BPRMF_batch method\), 60](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.BPRSlim.bprslim.BPRSlim method\), 61](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.CML.CML.CML method\), 62](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.FFM.field_aware_factorization.ResultHandler.FFM method\), 63](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.FISM.FISM.FISM method\), 63](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.FM.factorization_machine.FM method\), 64](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.FunkSVD.funk_svd.FunkSVD method\), 65](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization.LogisticMatrixFactorization method\), 66](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.MF.matrix_factorization.MF method\), 66](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization.NonNegMF method\), 67](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization.PMF method\), 68](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.PureSVD.pure_svd.PureSVD method\), 69](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.Slim.slim.Slim method\), 71](#)
[restore_weights\(\) \(elliot.recommender.latent_factor_models.WRMF.wrmf.WRMF method\), 71](#)
[restore_weights\(\) \(elliot.recommender.neural.ConvMF.convolutional_matrix_factorization.ConvMF method\), 72](#)
[restore_weights\(\) \(elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization.ConvNeuMF method\), 74](#)
[restore_weights\(\) \(elliot.recommender.neural.DeepFM.deep_fm.DeepFM method\), 76](#)
[restore_weights\(\) \(elliot.recommender.neural.NFM.neural_fm.NFM method\), 79](#)
[restore_weights\(\) \(elliot.recommender.NN.attribute_item_knn.attribute_item_knn.AttributeItemKNN method\), 34](#)
[restore_weights\(\) \(elliot.recommender.NN.attribute_user_knn.attribute_user_knn.AttributeUserKNN method\), 35](#)
[restore_weights\(\) \(elliot.recommender.NN.item_knn.item_knn.ItemKNN method\), 36](#)
[restore_weights\(\) \(elliot.recommender.NN.user_knn.user_knn.UserKNN method\), 38](#)
[restore_weights\(\) \(elliot.recommender.recommender_utils_mixin.RecMixin method\), 89](#)
[ResultHandler.FFM \(class in elliot.result_handler.result_handler\), 90](#)
[RMSE \(class in elliot.evaluation.metrics.rating.rmse.rmse\), 28](#)
[RSP \(class in elliot.evaluation.metrics.fairness.rsp.rsp\), 25](#)
[run_experiment\(\) \(in module elliot.run\), 93](#)

S

[Sampler \(class in elliot.dataset.samplers.custom_pointwise_sparse_sampler\), 7](#)
[Sampler \(class in elliot.dataset.samplers.custom_pointwise_sparse_sampler.LogisticMatrixFactorization\), 7](#)
[Sampler \(class in elliot.dataset.samplers.custom_sparse_sampler\), 7](#)
[Sampler \(class in elliot.dataset.samplers.pairwise_sampler\), 7](#)
[Sampler \(class in elliot.dataset.sampler.pipeline_sampler\), 7](#)
[Sampler \(class in elliot.dataset.samplers.pointwise_cfgan_sampler\), 8](#)
[Sampler \(class in elliot.dataset.samplers.pointwise_pos_neg_ratings_sampler\), 8](#)
[Sampler \(class in elliot.dataset.samplers.pointwise_pos_neg_ratio_ratings_sampler\), 8](#)
[Sampler \(class in elliot.dataset.samplers.pointwise_pos_neg_sampler\), 8](#)
[Sampler \(class in elliot.dataset.samplers.pointwise_wide_and_deep_sampler\), 8](#)
[Sampler \(class in elliot.dataset.samplers.sparse_sampler\), 8](#)
[Sampling \(class in elliot.recommender.autoencoders.vae.multi_vae_model\), 46](#)
[save_best_models\(\) \(elliot.result_handler.recommender_utils_mixin.RecMixin method\), 89](#)

sult_handler.ResultHandler method),
 90
 save_best_results() (elliot.result_handler.re-
 sult_handler.ResultHandler method),
 90
 save_best_results_as_triplets() (elliot.re-
 sult_handler.result_handler.ResultHandler
 method), 90
 save_best_statistical_results() (elliot.result_han-
 dler.result_handler.ResultHandler
 method), 90
 save_np() (in module elliot.utils.write), 93
 save_obj() (in module elliot.utils.write), 93
 save_trials() (elliot.result_handler.result_han-
 dler.HyperParameterStudy method),
 90
 save_trials_as_triplets() (elliot.result_han-
 dler.result_handler.HyperParameterStudy
 method), 90
 score_item() (elliot.recommender.NN.at-
 tribute_item_knn.attribute_item_knn_similarity.Similar-
 ity static method), 34
 score_item() (elliot.recommender.NN.at-
 tribute_user_knn.attribute_user_knn_similarity.Similar-
 ity static method), 35
 score_item() (elliot.recommender.NN.item_kn-
 n.item_knn_similarity.Similarity static
 method), 37
 score_item() (elliot.recommender.NN.user_kn-
 n.user_knn_similarity.Similarity static
 method), 38
 set_item_bias() (elliot.recommender.knowl-
 edge_aware.kaHFM.ka_hfm.MF
 method), 56
 set_item_bias() (elliot.recommender.latent_fac-
 tor_models.BPRMF.BPRMF.MF
 method), 60
 set_item_factors() (elliot.recommender.knowl-
 edge_aware.kaHFM.ka_hfm.MF
 method), 57
 set_item_factors() (elliot.recommender.latent_-
 factor_models.BPRMF.BPRMF.MF
 method), 60
 set_model_state() (elliot.recommender.alge-
 bric.slope_one.slope_one_model.SlopeOneModel
 method), 43
 set_model_state() (elliot.recommender.con-
 tent_based.VSM.vector_space_model_similarity.Similar-
 ity method), 47
 set_model_state() (elliot.recommender.knowl-
 edge_aware.kaHFM.ka_hfm.MF
 method), 57
 set_model_state() (elliot.recommender.latent_-
 factor_models.BPRMF.BPRMF.MF
 method), 60
 set_model_state() (elliot.recommender.latent_-
 factor_models.BPRSlim.bprslim_model.BPRSlimModel
 method), 61
 set_model_state() (elliot.recommender.latent_-
 factor_models.NonNegMF.non_negative_matrix_factor-
 method), 68
 set_model_state() (elliot.recommender.latent_-
 factor_models.PureSVD.pure_svd_model.PureSVDModel
 method), 69
 set_model_state() (elliot.recommender.latent_-
 factor_models.Slim.slim_model.SlimModel
 method), 71
 set_model_state() (elliot.recommender.latent_-
 factor_models.WRMF.wrmf_model.WRMFModel
 method), 72
 set_model_state() (elliot.recommender.NN.at-
 tribute_item_knn.attribute_item_knn_similarity.Similar-
 ity method), 34
 set_model_state() (elliot.recommender.NN.at-
 tribute_user_knn.attribute_user_knn_similarity.Similar-
 ity method), 35
 set_model_state() (elliot.recommender.N-
 N.item_knn.item_knn_similarity.Similarity
 method), 37
 set_model_state() (elliot.recommender.N-
 N.user_knn.user_knn_similarity.Similarity
 method), 38
 set_update_user() (elliot.recommender.latent_-
 factor_models.LogisticMF.logistic_matrix_factorization-
 method), 66
 set_user_bias() (elliot.recommender.knowl-
 edge_aware.kaHFM.ka_hfm.MF
 method), 57
 set_user_bias() (elliot.recommender.latent_fac-
 tor_models.BPRMF.BPRMF.MF
 method), 60
 set_user_factors() (elliot.recommender.knowl-
 edge_aware.kaHFM.ka_hfm.MF
 method), 57
 set_user_factors() (elliot.recommender.latent_-
 factor_models.BPRMF.BPRMF.MF
 method), 60
 ShannonEntropy (class in elliot.evaluation.met-
 rics.diversity.shannon_entropy.shannon_entropy),
 21
 Similarity (class in elliot.recommender.con-
 tent_based.VSM.vector_space_model_similarity),
 47
 Similarity (class in elliot.recommender.NN.at-
 tribute_item_knn.attribute_item_knn_similarity),
 34
 Similarity (class in elliot.recommender.NN.at-
 tribute_user_knn.attribute_user_knn_similarity),
 35
 Similarity (class in elliot.recommender.N-
 N.item_knn.item_knn_similarity), 36
 Similarity (class in elliot.recommender.N-
 N.user_knn.user_knn_similarity), 38
 single() (elliot.hyperoptimization.model_coor-
 dinator.ModelCoordinator method),
 32

- Slim (class in `elliott.recommender.latent_factor_models.Slim.slim`), 70
- SlimModel (class in `elliott.recommender.latent_factor_models.Slim.slim_model`), 71
- SlopeOne (class in `elliott.recommender.algebraic.slope_one.slope_one`), 43
- SlopeOneModel (class in `elliott.recommender.algebraic.slope_one.slope_one_model`), 43
- `softmax()` (`elliott.recommender.neural.NAIS_nais_model.NAIS_model` method), 79
- Splitter (class in `elliott.splitter.base_splitter`), 90
- `splitting_best_timestamp()` (`elliott.splitter.base_splitter.Splitter` method), 91
- `splitting_kfolds()` (`elliott.splitter.base_splitter.Splitter` method), 91
- `splitting_passed_timestamp()` (`elliott.splitter.base_splitter.Splitter` method), 91
- `splitting_randomsubsampling_kfolds()` (`elliott.splitter.base_splitter.Splitter` method), 91
- `splitting_randomsubsampling_kfolds_leavenout()` (`elliott.splitter.base_splitter.Splitter` method), 91
- `splitting_temporal_holdout()` (`elliott.splitter.base_splitter.Splitter` method), 91
- `splitting_temporal_leavenout()` (`elliott.splitter.base_splitter.Splitter` method), 91
- SRecall (class in `elliott.evaluation.metrics.diversity.SRecall.srecall`), 20
- `start()` (`elliott.utils.logger_util.QueueListenerHandler` method), 92
- StatisticalMetric (class in `elliott.evaluation.metrics.statistical_array_metric`), 30
- StatTest (class in `elliott.result_handler.result_handler`), 90
- `step()` (`elliott.dataset.samplers.custom_pointwise_sparse_sampler.Sampler` method), 7
- `step()` (`elliott.dataset.samplers.custom_sampler.Sampler` method), 7
- `step()` (`elliott.dataset.samplers.custom_sparse_sampler.Sampler` method), 7
- `step()` (`elliott.dataset.samplers.pairwise_sampler.Sampler` method), 7
- `step()` (`elliott.dataset.samplers.pipeline_sampler.Sampler` method), 7
- `step()` (`elliott.dataset.samplers.pointwise_cf_gan_sampler.Sampler` method), 8
- `step()` (`elliott.dataset.samplers.pointwise_pos_neg_ratings_sampler.Sampler` method), 8
- `step()` (`elliott.dataset.samplers.pointwise_pos_neg_ratio_ratings_sampler.Sampler` method), 8
- `step()` (`elliott.dataset.samplers.pointwise_pos_neg_sampler.Sampler` method), 8
- `step()` (`elliott.dataset.samplers.pointwise_wise_wide_and_deep_sampler.Sampler` method), 8
- `step()` (`elliott.dataset.samplers.sparse_sampler.Sampler` method), 8
- `stop()` (`elliott.utils.logger_util.QueueListenerHandler` method), 92
- `store_recommendation()` (in module `elliott.utils.write`), 93
- `store_splitting()` (`elliott.splitter.base_splitter.Splitter` method), 91
- `subsampling_leavenout_list_generator()` (`elliott.splitter.base_splitter.Splitter` method), 91
- `subsampling_list_generator()` (`elliott.splitter.base_splitter.Splitter` method), 91
- `suggest()` (in module `elliott.hyperoptimization`), 32
- SVDpp (class in `elliott.recommender.latent_factor_models.SVDpp.svdpp`), 70
- SVDppModel (class in `elliott.recommender.latent_factor_models.SVDpp.svdpp_model`), 70
- ## T
- TFIDF (class in `elliott.recommender.content_based.VSM.tfidf_utils`), 47
- TFIDF (class in `elliott.recommender.knowledge_aware.kaHFM.tfidf_utils`), 57
- TFIDF (class in `elliott.recommender.knowledge_aware.kaHFM_batch.tfidf_utils`), 58
- TFIDF (class in `elliott.recommender.knowledge_aware.kahfm_embeddings.tfidf_utils`), 59
- TFIDF (class in `elliott.recommender.NN.attribute_user_knn.tfidf_utils`), 35
- `tfidf()` (`elliott.recommender.content_based.VSM.tfidf_utils.TFIDF` method), 47
- `tfidf()` (`elliott.recommender.knowledge_aware.kaHFM.tfidf_utils.TFIDF` method), 57
- `tfidf()` (`elliott.recommender.knowledge_aware.kaHFM_batch.tfidf_utils.TFIDF` method), 58
- `tfidf()` (`elliott.recommender.knowledge_aware.kahfm_embeddings.tfidf_utils.TFIDF` method), 59
- `tfidf()` (`elliott.recommender.NN.attribute_user_knn.tfidf_utils.TFIDF` method), 35
- TimeFilter (class in `elliott.utils.logging`), 92
- `train()` (`elliott.recommender.adversarial.AMF.AMF` method), 39
- `train()` (`elliott.recommender.adversarial.AMR.AMR` method), 41
- `train()` (`elliott.recommender.algebraic.slope_one.slope_one.SlopeOne`

method), 43

train() (elliot.recommender.attentive.afm.afm.AFM method), 44

train() (elliot.recommender.autoencoders.dae.-multi_dae.MultiDAE method), 44

train() (elliot.recommender.autoencoders.vae.-multi_vae.MultiVAE method), 46

train() (elliot.recommender.base_recommender_model.BaseRecommenderModel method), 89

train() (elliot.recommender.content_based.VS-M.vector_space_model.VSM method), 47

train() (elliot.recommender.gan.CFGAN.cf-gan.CFGAN method), 48

train() (elliot.recommender.gan.IRGAN.ir-gan.IRGAN method), 50

train() (elliot.recommender.graph_based.light-gcn.LightGCN.LightGCN method), 54

train() (elliot.recommender.-graph_based.ngcf.NGCF.NGCF method), 55

train() (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.KaHFM method), 56

train() (elliot.recommender.knowledge_aware.kaHFM_batch.kahfm_batch.KaHFMBatchral.DeepFM.deep_fm.DeepFM method), 57

train() (elliot.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings.KaHFMEmbeddings method), 58

train() (elliot.recommender.latent_factor_models.BPRMF.BPRMF.BPRMF method), 59

train() (elliot.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch.BPRMF_batch method), 60

train() (elliot.recommender.latent_factor_models.BPRSlim.bprslim.BPRSlim method), 61

train() (elliot.recommender.latent_factor_models.CML.CML.CML method), 62

train() (elliot.recommender.latent_factor_models.FFM.field_aware_factorization_machine.FFM method), 63

train() (elliot.recommender.latent_factor_models.FISM.FISM.FISM method), 63

train() (elliot.recommender.latent_factor_models.FM.factorization_machine.FM method), 64

train() (elliot.recommender.latent_factor_models.FunkSVD.funk_svd.FunkSVD method), 65

train() (elliot.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization.LogisticMatrixFactorization method), 66

train() (elliot.recommender.latent_factor_models.MF.matrix_factorization.MF method), 67

train() (elliot.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization.NonNegMF method), 67

train() (elliot.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization.PMF method), 68

train() (elliot.recommender.latent_factor_models.PureSVD.pure_svd.PureSVD method), 69

train() (elliot.recommender.latent_factor_models.Slim.slim.Slim method), 71

train() (elliot.recommender.latent_factor_models.Slim.slim_model.SlimModel method), 71

train() (elliot.recommender.latent_factor_models.SVDpp.svdpp.SVDpp method), 70

train() (elliot.recommender.latent_factor_models.WRMF.wrmf.WRMF method), 71

train() (elliot.recommender.neural.ConvMF.-convolutional_matrix_factorization.ConvMF method), 72

train() (elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization.ConvNeuMF method), 74

train() (elliot.recommender.neural.DMF.deep_-KahFMEmbeddings.DMF method), 75

train() (elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization.GMF method), 76

train() (elliot.recommender.neural.ItemAutoRec.itemautorec.ItemAutoRec method), 77

train() (elliot.recommender.neural.NAIS.nais.-NAIS method), 78

train() (elliot.recommender.neural.NeuMF.neural_matrix_factorization.NeuMF method), 81

train() (elliot.recommender.neural.NFM.neural_fm.NFM method), 79

train() (elliot.recommender.neural.NPR.neural_personalized_ranking.NPR method), 80

train() (elliot.recommender.neural.UserAutoRec.userautorec.UserAutoRec method), 81

train() (elliot.recommender.neural.WideAndDeep.wide_and_deep.WideAndDeep method), 82

train() (elliot.recommender.NN.attribute_item_knn.attribute_item_knn.AttributeItemKNN method), 83

train() (elliot.recommender.NN.attribute_user_knn.attribute_user_knn.AttributeUserKNN method), 35

`train()` (elliot.recommender.NN.item_knn.item_knn.ItemKNN method), 36
`train()` (elliot.recommender.NN.user_knn.user_knn.UserKNN method), 38
`train()` (elliot.recommender.recommender_utils.mixin.RecMixin method), 89
`train()` (elliot.recommender.unpersonalized.-most_popular.most_popular.MostPop method), 84
`train()` (elliot.recommender.unpersonalized.random_recommender.Random.Random method), 85
`train()` (elliot.recommender.visual_recommenders.ACF.ACF.ACF method), 85
`train()` (elliot.recommender.visual_recommenders.DeepStyle.DeepStyle.DeepStyle method), 87
`train()` (elliot.recommender.visual_recommenders.DVBPR.DVBPR.DVBPR method), 86
`train()` (elliot.recommender.visual_recommenders.VBPR.VBPR.VBPR method), 87
`train()` (elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking.VNPR.VNPR method), 88
`train_step()` (elliot.recommender.adversarial.AMF.AMF_model.AMF_model method), 40
`train_step()` (elliot.recommender.adversarial.AMR.AMR_model.AMR_model method), 42
`train_step()` (elliot.recommender.attentive.afm.afm_model.AFMMModel method), 44
`train_step()` (elliot.recommender.autoencoders.-dae.multi_dae_model.DenoisingAutoEncoder method), 45
`train_step()` (elliot.recommender.autoencoders.-vae.multi_vae_model.VariationalAutoEncoder method), 46
`train_step()` (elliot.recommender.gan.CF-GAN.cfgan_model.CFGAN_model method), 49
`train_step()` (elliot.recommender.gan.CF-GAN.cfgan_model.Discriminator method), 50
`train_step()` (elliot.recommender.gan.CF-GAN.cfgan_model.Generator method), 50
`train_step()` (elliot.recommender.gan.IRGAN.irgan_model.Discriminator method), 51
`train_step()` (elliot.recommender.gan.IRGAN.irgan_model.Generator method), 51
`train_step()` (elliot.recommender.gan.IRGAN.irgan_model.IRGAN_model method), 53
`train_step()` (elliot.recommender.graph_based.-lightgcn.LightGCN_model.LightGCNModel method), 54
`train_step()` (elliot.recommender.-graph_based.ngcf.NGCF_model.NGCFModel method), 55
`train_step()` (elliot.recommender.knowledge_aware.kaHFM.ka_hfm.KaHFM method), 56
`train_step()` (elliot.recommender.knowledge_aware.kaHFM_batch.kahfm_batch_model.KaHFM_batch_model method), 58
`train_step()` (elliot.recommender.knowledge_aware.kahfm_embeddings.kahfm_embeddings_model.KahfmEmbeddingsModel method), 58
`train_step()` (elliot.recommender.latent_factor_models.BPRMF.BPRMF.BPRMF method), 59
`train_step()` (elliot.recommender.latent_factor_models.BPRMF_batch.BPRMF_batch_model.BPRMF_batch_model method), 61
`train_step()` (elliot.recommender.latent_factor_models.BPRSlim.bprslim_model.BPRSlimModel method), 61
`train_step()` (elliot.recommender.latent_factor_models.CML.CML_model.CML_model method), 62
`train_step()` (elliot.recommender.latent_factor_models.FFM.field_aware_factorization_machine_model.FieldAwareFactorizationMachineModel method), 63
`train_step()` (elliot.recommender.latent_factor_models.FISM.FISM_model.FISM_model method), 64
`train_step()` (elliot.recommender.latent_factor_models.FM.factorization_machine_model.FactorizationMachineModel method), 65
`train_step()` (elliot.recommender.latent_factor_models.FunkSVD.funk_svd_model.FunkSVDModel method), 66
`train_step()` (elliot.recommender.latent_factor_models.LogisticMF.logistic_matrix_factorization_model.LogisticMatrixFactorizationModel method), 66
`train_step()` (elliot.recommender.latent_factor_models.MF.matrix_factorization_model.MatrixFactorizationModel method), 67
`train_step()` (elliot.recommender.latent_factor_models.NonNegMF.non_negative_matrix_factorization_model.NonNegativeMatrixFactorizationModel method), 68
`train_step()` (elliot.recommender.latent_factor_models.PMF.probabilistic_matrix_factorization_model.ProbabilisticMatrixFactorizationModel method), 69
`train_step()` (elliot.recommender.latent_factor_models.PureSVD.pure_svd_model.PureSVDModel method), 69
`train_step()` (elliot.recommender.latent_factor_models.SVDpp.svdpp_model.SVDppModel method), 70
`train_step()` (elliot.recommender.latent_factor_models.WRMF.wrmf_model.WRMFModel method), 70

method), 72

train_step() (elliot.recommender.neural.ConvMF.convolutional_matrix_factorization_model.method), 73

train_step() (elliot.recommender.neural.ConvNeuMF.convolutional_neural_matrix_factorization_model.method), 74

train_step() (elliot.recommender.neural.DeepFM.deep_fm_model.DeepFMModel.useOnlyBooleanInteractions() (elliot.recommender.NN.user_knn.aiolli_ferrari.Compute_Similarity method), 36

train_step() (elliot.recommender.neural.DMF.deep_matrix_factorization_model.DeepMatrixFactorizationModel.method), 75

train_step() (elliot.recommender.neural.GeneralizedMF.generalized_matrix_factorization_model.GeneralizedMatrixFactorizationModel.method), 77

train_step() (elliot.recommender.neural.ItemAutoRec.itemautorec_model.ItemAutoRecModel.method), 78

train_step() (elliot.recommender.neural.NAIS.-nais_model.NAIS_model.method), 79

train_step() (elliot.recommender.neural.NeuMF.neural_matrix_factorization_model.NeuMFModel.method), 81

train_step() (elliot.recommender.neural.NFM.neural_fm_model.NeuralFactorizationMachineModel.method), 80

train_step() (elliot.recommender.neural.NPR.neural_personalized_ranking_model.NPRModel.method), 80

train_step() (elliot.recommender.neural.UserAutoRec.userautorec_model.UserAutoRecModel.method), 82

train_step() (elliot.recommender.neural.WideAndDeep.wide_and_deep_model.WideAndDeepModel.method), 84

train_step() (elliot.recommender.visual_recommenders.ACF.ACF_model.ACF_model.method), 85

train_step() (elliot.recommender.visual_recommenders.DeepStyle.DeepStyle_model.DeepStyle_model.method), 87

train_step() (elliot.recommender.visual_recommenders.DVBPR.DVBPR_model.DVBPR_model.method), 86

train_step() (elliot.recommender.visual_recommenders.VBPR.VBPR_model.VBPR_model.method), 88

train_step() (elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking_model.VNPRModel.method), 89

train_step_with_reward() (elliot.recommender.-gan.IRGAN.irgan_model.Generator.method), 52

edge_aware.kaHFM.ka_hfm.KaHFM.method), 56

latent_factor_models.BPRMF.BPRMF.BPRMF.method), 59

OnlyBooleanInteractions() (elliot.recommender.NN.item_knn.aiolli_ferrari.Compute_Similarity method), 36

UserAutoRecRatio (class in elliot.recommender.NN.user_knn.aiolli_ferrari.Compute_Similarity method), 37

UserAutoRecModel (class in elliot.recommender.NN.user_knn.aiolli_ferrari.Compute_Similarity method), 81

UserCoverage (class in elliot.evaluation.metrics.coverage.user_coverage.user_coverage), 19

UserCoverageAtN (class in elliot.evaluation.-metrics.coverage.user_coverage.user_coverage_at_n), 20

UserKFM (class in elliot.recommender.NN.user_knn.user_knn), 38

UserMADranking (class in elliot.evaluation.-metrics.fairness.MAD.UserMADranking), 24

UserMADrating (class in elliot.evaluation.metrics.fairness.MAD.UserMADrating), 24

VariationalAutoEncoder (class in elliot.recommender.autoencoders.vae.multi_vae_model), 46

VBPR (class in elliot.recommender.visual_recommenders.VBPR.VBPR), 87

VBPR_model (class in elliot.recommender.visual_recommenders.VBPR.VBPR_model), 87

VisualDataObject (class in elliot.dataset.dataloader.visual_dataloader), 6

VisualLoader (class in elliot.dataset.dataloader.visual_dataloader), 6

VNPR (class in elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking), 88

VNPRModel (class in elliot.recommender.visual_recommenders.VNPR.visual_neural_personalized_ranking_model.VNPRModel), 88

VSM (class in elliot.recommender.content_based.VSM.vector_space_model), 47

U

update_factors() (elliot.recommender.knowl-

W

WideAndDeep (class in elliot.recommender.neural.WideAndDeep.wide_and_deep),

82
WideAndDeepModel (class in `elliott.recommender.neural.WideAndDeep.wide_and_deep_model`),
82
WilcoxonTest (class in `elliott.evaluation.statistical_significance`), 31
WilcoxonTest (`elliott.result_handler.result_handler.StatTest` attribute), 90
WRMF (class in `elliott.recommender.latent_factor_models.WRMF.wrmf`), 71
WRMFModel (class in `elliott.recommender.latent_factor_models.WRMF.wrmf_model`),
71

