Package 'casimir'

November 17, 2025

Title Comparing Automated Subject Indexing Methods in R

Version 0.3.3

Description Perform evaluation of automatic subject
indexing methods. The main focus of the package is to enable efficient
computation of set retrieval and ranked retrieval metrics across multiple
dimensions of a dataset, e.g. document strata or subsets of the label set.
The package also provides the possibility of computing bootstrap confidence
intervals for all major metrics, with seamless integration of parallel
computation and propensity scored variants of standard metrics.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Imports dplyr (>= 1.1.1), furrr, purrr, rsample, tidyr, rlang,
collapse (>= 2.1.0), stringr, options, withr

Suggests testthat (>= 3.0.0), tibble, tidyverse, ggplot2, future

Config/testthat/edition 3

Depends R (>= 4.1.0)

LazyData true

URL https://deutsche-nationalbibliothek.github.io/casimir/

NeedsCompilation no

Author Maximilian Kähler [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4695-0565>),
Markus Schumacher [aut],
Deutsche Nationalbibliothek [cph]

Maintainer Maximilian Kähler <m.kaehler@dnb.de>

Repository CRAN

Date/Publication 2025-11-17 21:30:07 UTC

Contents

`apply_threshold` *Filter predictions based on score and rank*

Description

Helper function for filtering predictions with score above a certain threshold or rank below some limit rank.

Usage

```
apply_threshold(threshold, limit = NA_real_, base_compare)
```

Arguments

`threshold`	A numeric threshold between 0 and 1.
`limit`	An integer cutoff >= 1 for rank-based thresholding. Requires a column `"rank"` in input `base_compare`.
`base_compare`	A data.frame as created by `create_comparison`, containing columns `"gold"`, `"score"`.

Value

A data.frame with observations that satisfy (`score` >= `threshold` AND (if applicable) `rank` <= `limit`) OR `gold` == `TRUE`. A new logical column `suggested` indicates TRUE if `score` >= `threshold` AND (if applicable) `rank` <= `limit`, and FALSE for false negative observations (that may have no score, a score below the threshold or rank above the limit).

Examples

```
library(casimir)

gold <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "b",
  "A", "c",
  "B", "a",
  "B", "d",
  "C", "a",
  "C", "b",
  "C", "d",
  "C", "f"
)

pred <- tibble::tribble(
  ~doc_id, ~label_id, ~score,
  "A", "a", 0.9,
  "A", "d", 0.7,
```

```
  "A", "f", 0.3,
  "A", "c", 0.1,
  "B", "a", 0.8,
  "B", "e", 0.6,
  "B", "d", 0.1,
  "C", "f", 0.1,
  "C", "c", 0.2,
  "C", "e", 0.2
)

base_compare <- create_comparison(gold, pred)

res_0 <- apply_threshold(
  threshold = 0.3,
  base_compare = base_compare
)
```

`boot_worker_fn` *Compute bootstrap replica of pr auc*

Description

A wrapper for use within bootstrap computation of pr auc which covers the repeated application of:

1. join with resampled doc_ids
2. summarise_intermediate_results
3. postprocessing of curve data
4. auc computation

Usage

```
boot_worker_fn(
  sampled_id_list,
  intermed_res,
  propensity_scored,
  replace_zero_division_with
)
```

Arguments

`sampled_id_list`	A list of all doc_ids of the examples drawn in each bootstrap iteration.
`intermed_res`	Intermediate results as produced by `compute_intermediate_results`, with a column `"searchspace_id"` as grouping variable.
`propensity_scored`	Logical, whether to use propensity scores as weights.

replace_zero_division_with
: In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1. (Defaults to NULL, overwritable using option 'casimir.replace_zero_division_with' or environment variable 'R_CASIMIR_REPLACE_ZERO_DIVISION_WITH')

Value

A data.frame with a column "pr_auc" and optional grouping_vars.

---

check_id_vars *Coerce id columns to character*

Description

Internal helper function designed to ensure that id columns are not passed as factor variables. Factor variables in id columns may cause undesired behaviour with the drop_empty_group argument.

Usage

```
check_id_vars(df)
```

Arguments

df	An input data.frame.

Value

The input data.frame df with the id columns being no longer factor variables.

---

check_id_vars_col *Coerce column to character*

Description

Check an arbitrary column in a data.frame for factor type and coerce to character.

Usage

```
check_id_vars_col(df, col)
```

Arguments

df	An input data.frame.
col	The name of the column to check.

Value

The input data.frame df with the specified column being no longer a factor variable.

---

check_repair_relevance_compare

Check for inconsistent relevance values

Description

Internal helper function to check a comparison matrix for inconsistent relevance values of gold standard and predicted labels.

Usage

```
check_repair_relevance_compare(
  gold_vs_pred,
  ignore_inconsistencies = options::opt("ignore_inconsistencies")
)
```

Arguments

gold_vs_pred	As created by create_comparison.
ignore_inconsistencies	Warnings about data inconsistencies will be silenced. (Defaults to FALSE, overwritable using option 'casimir.ignore_inconsistencies' or environment variable 'R_CASIMIR_IGNORE_INCONSISTENCIES')

Value

A valid comparison matrix with possibly corrected relevance values, being compatible with compute_intermediate_result

check_repair_relevance_pred *Check for inconsistent relevance values*

Description

Internal helper function to check a data.frame with predicted labels for a valid relevance column.

Usage

```
check_repair_relevance_pred(
  predicted,
  ignore_inconsistencies = options::opt("ignore_inconsistencies")
)
```

Arguments

predicted	Multi-label prediction results. Expects a data.frame with columns "label_id", "doc_id", "relevance".
ignore_inconsistencies	Warnings about data inconsistencies will be silenced. (Defaults to FALSE, overwritable using option 'casimir.ignore_inconsistencies' or environment variable 'R_CASIMIR_IGNORE_INCONSISTENCIES')

Value

A valid predicted data.frame with possibly eliminated missing values.

compute_intermediate_results *Compute intermediate set retrieval results per group*

Description

Compute intermediate set retrieval results per group such as number of gold standard and predicted labels, number of true positives, false positives and false negatives, precision, R-precision, recall and F1 score.

Usage

```
compute_intermediate_results(
  gold_vs_pred,
  grouping_var,
  propensity_scored = FALSE,
  cost_fp = NULL,
```

```
  drop_empty_groups = options::opt("drop_empty_groups"),
  check_group_names = options::opt("check_group_names")
)

compute_intermediate_results_dplyr(
  gold_vs_pred,
  grouping_var,
  propensity_scored = FALSE,
  cost_fp = NULL
)
```

Arguments

gold_vs_pred	A data.frame with logical columns "suggested", "gold" as produced by create_comparison.
grouping_var	A character vector of grouping variables that must be present in gold_vs_pred (dplyr version requires rlang symbols).
propensity_scored	Logical, whether to use propensity scores as weights.
cost_fp	A numeric value > 0, defaults to NULL.
drop_empty_groups	Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to TRUE, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')
check_group_names	Perform replacement of dots in grouping columns. Disable for faster computation if you can make sure that all columns used for grouping ("doc_id", "label_id", "doc_groups", "label_groups") do not contain dots. (Defaults to TRUE, overwritable using option 'casimir.check_group_names' or environment variable 'R_CASIMIR_CHECK_GROUP_NAMES')

Value

A list of two elements:

- results_table A data.frame with columns "n_gold", "n_suggested", "tp", "fp", "fn", "prec", "rprec", "rec", "f1".
- grouping_var The input vector grouping_var.

Functions

- compute_intermediate_results_dplyr(): Variant with dplyr based internals rather than collapse internals.

Examples

```
library(casimir)

gold <- tibble::tribble(
  ~doc_id, ~label_id,
```

```
  "A", "a",
  "A", "b",
  "A", "c",
  "B", "a",
  "B", "d",
  "C", "a",
  "C", "b",
  "C", "d",
  "C", "f"
)

pred <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "d",
  "A", "f",
  "B", "a",
  "B", "e",
  "C", "f"
)

gold_vs_pred <- create_comparison(gold, pred)

compute_intermediate_results(gold_vs_pred, "doc_id")
```

`compute_intermediate_results_rr` *Compute intermediate ranked retrieval results per group*

Description

Compute intermediate ranked retrieval results per group such as Discounted Cumulative Gain (DCG), Ideal Discounted Cumulative Gain (IDCG), Normalised Discounted Cumulative Gain (NDCG) and Label Ranking Average Precision (LRAP).

Usage

```
compute_intermediate_results_rr(
  gold_vs_pred,
  grouping_var,
  drop_empty_groups = options::opt("drop_empty_groups")
)
```

Arguments

`gold_vs_pred`	A data.frame as generated by `create_comparison`, additionally containing a column `"score"`.
`grouping_var`	A character vector of grouping variables that must be present in `gold_vs_pred`.

drop_empty_groups
: Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to TRUE, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')

Value

A data.frame with columns "dcg", "idcg", "ndcg", "lrap".

Examples

```
library(casimir)

gold <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "b",
  "A", "c",
  "A", "d",
  "A", "e",
)

pred <- tibble::tribble(
  ~doc_id, ~label_id, ~score,
  "A", "f", 0.3277,
  "A", "e", 0.32172,
  "A", "b", 0.13517,
  "A", "g", 0.10134,
  "A", "h", 0.09152,
  "A", "a", 0.07483,
  "A", "i", 0.03649,
  "A", "j", 0.03551,
  "A", "k", 0.03397,
  "A", "c", 0.03364
)

gold_vs_pred <- create_comparison(gold, pred)

compute_intermediate_results_rr(
  gold_vs_pred,
  rlang::syms(c("doc_id"))
)
```

---

compute_propensity_scores

Compute inverse propensity scores

Description

Compute inverse propensity scores based on a label distribution. Propensity scores for extreme multi-label learning are proposed in Jain, H., Prabhu, Y., & Varma, M. (2016). Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking and Other Missing Label Applications. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-Aug, 935–944. doi:10.1145/2939672.2939756.

Usage

```
compute_propensity_scores(label_distribution, a = 0.55, b = 1.5)
```

Arguments

label_distribution	Expects a data.frame with columns "label_id", "label_freq", "n_docs". label_freq corresponds to the number of occurences a label has in the gold standard. n_docs corresponds to the total number of documents in the gold standard.
a	A numeric parameter for the propensity score calculation, defaults to 0.55.
b	A numeric parameter for the propensity score calculation, defaults to 1.5.

Value

A data.frame with columns "label_id", "label_weight".

Examples

```
library(tidyverse)
library(casimir)

label_distribution <- dnb_label_distribution

compute_propensity_scores(label_distribution)
```

---

compute_pr_auc *Compute area under precision-recall curve*

Description

Compute the area under the precision-recall curve with support for bootstrap-based confidence intervals and different stratification and aggregation modes for the underlying precision and recall aggregation. Precision is calculated as the best value at a given level of recall for all possible thresholds on score and limits on rank. In essence, compute_pr_auc performs a two-dimensional optimisation over thresholds and limits applying both threshold-based cutoff as well as rank-based cutoff.

Usage

```
compute_pr_auc(
  predicted,
  gold_standard,
  doc_groups = NULL,
  label_groups = NULL,
  mode = "doc-avg",
  steps = 100,
  thresholds = NULL,
  limit_range = NA_real_,
  compute_bootstrap_ci = FALSE,
  n_bt = 10L,
  seed = NULL,
  graded_relevance = FALSE,
  rename_metrics = FALSE,
  propensity_scored = FALSE,
  label_distribution = NULL,
  cost_fp_constant = NULL,
  replace_zero_division_with = options::opt("replace_zero_division_with"),
  drop_empty_groups = options::opt("drop_empty_groups"),
  ignore_inconsistencies = options::opt("ignore_inconsistencies"),
  verbose = options::opt("verbose"),
  progress = options::opt("progress")
)
```

Arguments

predicted	Multi-label prediction results. Expects a data.frame with columns "label_id", "doc_id", "score".
gold_standard	Expects a data.frame with columns "label_id","doc_id".
doc_groups	A two-column data.frame with a column "doc_id" and a second column defining groups of documents to stratify results by. It is recommended that groups are of type factor so that levels are not implicitly dropped during bootstrap replications.
label_groups	A two-column data.frame with a column "label_id" and a second column defining groups of labels to stratify results by. Results in each stratum will restrict gold standard and predictions to the specified label groups as if the vocabulary was consisting of the label group only. All modes "doc-avg", "subj-avg", "micro" are supported within label strata. Nevertheless, mixing mode = "doc-avg" with fine-grained label strata can result in many missing values on document-level results. Also rank-based thresholding (e.g. top 5) will result in inhomogeneous numbers of labels per document within the defined label strata. mode = "subj-avg" or mode = "micro" can be more appropriate in these circumstances.
mode	One of the following aggregation modes: "doc-avg","subj-avg", "micro".
steps	Number of breaks to divide the interval $[0, 1]$ into. These breaks will be used as quantiles to be calculated on the true positive suggestions' score distribution

and therefore build one axis of the grid for computing the pr curve.

`thresholds` Alternatively to steps, one can manually set the thresholds to be used to build the pr curve. Defaults to the quantiles of the true positive suggestions' score distribution to be obtained from `steps`.

`limit_range` A vector of limit values to apply on the rank column. Defaults to NA, applying no cutoff on the predictions' label rank.

`compute_bootstrap_ci`
A logical indicator for computing bootstrap CIs.

`n_bt` An integer number of resamples to be used for bootstrapping.

`seed` Pass a seed to make bootstrap replication reproducible.

`graded_relevance`
A logical indicator for graded relevance. Defaults to `FALSE` for binary relevance. If set to `TRUE`, the `predicted` data.frame should contain a numeric column `"relevance"` with values in the range of $[0, 1]$.

`rename_metrics` If set to `TRUE`, the metric names in the output will be renamed if:

`graded_relevance == TRUE` prefixed with *"g-"* to indicate that metrics are computed with graded relevance.

`propensity_scored == TRUE` prefixed with *"ps-"* to indicate that metrics are computed with propensity scores.

`!is.null(k)` suffixed with *"@k"* to indicate that metrics are limited to top k predictions.

`propensity_scored`
Logical, whether to use propensity scores as weights.

`label_distribution`
Expects a data.frame with columns `"label_id"`, `"label_freq"`, `"n_docs"`. `label_freq` corresponds to the number of occurences a label has in the gold standard. `n_docs` corresponds to the total number of documents in the gold standard.

`cost_fp_constant`
Constant cost assigned to false positives. `cost_fp_constant` must be a numeric value > 0 or one of 'max', 'min', 'mean' (computed with reference to the `gold_standard` label distribution). Defaults to NULL, i.e. label weights are applied to false positives in the same way as to false negatives and true positives.

`replace_zero_division_with`
In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1. (Defaults to `NULL`, overwritable using option 'casimir.replace_zero_division_with' or environment variable 'R_CASIMIR_REPLACE_ZERO_DIVISION_WITH')

`drop_empty_groups`
Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to `TRUE`, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')

ignore_inconsistencies
: Warnings about data inconsistencies will be silenced. (Defaults to FALSE, overwritable using option 'casimir.ignore_inconsistencies' or environment variable 'R_CASIMIR_IGNORE_INCONSISTENCIES')

verbose
: Verbose reporting of computation steps for debugging. (Defaults to FALSE, overwritable using option 'casimir.verbose' or environment variable 'R_CASIMIR_VERBOSE')

progress
: Display progress bars for iterated computations (like bootstrap CI or pr curves). (Defaults to FALSE, overwritable using option 'casimir.progress' or environment variable 'R_CASIMIR_PROGRESS')

Value

A data.frame with columns "pr_auc" and (if applicable) "ci_lower", "ci_upper" and additional stratification variables.

See Also

compute_set_retrieval_scores, compute_pr_auc_from_curve

Examples

```
library(ggplot2)
library(casimir)

gold <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "b",
  "A", "c",
  "B", "a",
  "B", "d",
  "C", "a",
  "C", "b",
  "C", "d",
  "C", "f"
)

pred <- tibble::tribble(
  ~doc_id, ~label_id, ~score, ~rank,
  "A", "a", 0.9, 1,
  "A", "d", 0.7, 2,
  "A", "f", 0.3, 3,
  "A", "c", 0.1, 4,
  "B", "a", 0.8, 1,
  "B", "e", 0.6, 2,
  "B", "d", 0.1, 3,
  "C", "f", 0.1, 3,
  "C", "c", 0.2, 1,
  "C", "e", 0.2, 1
)

auc <- compute_pr_auc(pred, gold, mode = "doc-avg")
```

compute_pr_auc_from_curve *Compute area under precision-recall curve*

Description

Compute the area under the precision-recall curve given pr curve data. This function is mainly intended for generating plot data. For computation of the area under the curve, use `compute_pr_auc`. The function uses a simple trapezoidal rule approximation along the steps of the generated curve data.

Usage

```
compute_pr_auc_from_curve(
  pr_curve_data,
  grouping_vars = NULL,
  drop_empty_groups = options::opt("drop_empty_groups")
)
```

Arguments

`pr_curve_data`	A data.frame as produced by `compute_pr_curve`, containing columns `"searchspace_id"`, `"prec"`, `"rec"`, `"prec_cummax"`, `"mode"`.
`grouping_vars`	Additional columns of the input data to group by.
`drop_empty_groups`	Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to `TRUE`, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')

Value

A data.frame with a column `"pr_auc"` and optional `grouping_vars`.

See Also

compute_pr_curve

Examples

```
library(ggplot2)
library(casimir)

gold <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "b",
  "A", "c",
  "B", "a",
```

```
  "B", "d",
  "C", "a",
  "C", "b",
  "C", "d",
  "C", "f"
)

pred <- tibble::tribble(
  ~doc_id, ~label_id, ~score, ~rank,
  "A", "a", 0.9, 1,
  "A", "d", 0.7, 2,
  "A", "f", 0.3, 3,
  "A", "c", 0.1, 4,
  "B", "a", 0.8, 1,
  "B", "e", 0.6, 2,
  "B", "d", 0.1, 3,
  "C", "f", 0.1, 3,
  "C", "c", 0.2, 1,
  "C", "e", 0.2, 1
)

pr_curve <- compute_pr_curve(
  gold,
  pred,
  mode = "doc-avg",
  optimize_cutoff = TRUE
)

auc <- compute_pr_auc_from_curve(pr_curve)

# note that pr curves take the cummax(prec), not the precision
ggplot(pr_curve$plot_data, aes(x = rec, y = prec_cummax)) +
  geom_point(
    data = pr_curve$opt_cutoff,
    aes(x = rec, y = prec_cummax),
    color = "red",
    shape = "star"
  ) +
  geom_text(
    data = pr_curve$opt_cutoff,
    aes(
      x = rec + 0.2, y = prec_cummax,
      label = paste("f1_opt =", round(f1_max, 3))
    ),
    color = "red"
  ) +
  geom_path() +
  coord_cartesian(xlim = c(0, 1), ylim = c(0, 1))
```

`compute_pr_curve` *Compute precision-recall curve*

Description

Compute the precision-recall curve for a given step size and limit range.

Usage

```
compute_pr_curve(
  predicted,
  gold_standard,
  doc_groups = NULL,
  label_groups = NULL,
  mode = "doc-avg",
  steps = 100,
  thresholds = NULL,
  limit_range = NA_real_,
  optimize_cutoff = FALSE,
  graded_relevance = FALSE,
  propensity_scored = FALSE,
  label_distribution = NULL,
  cost_fp_constant = NULL,
  replace_zero_division_with = options::opt("replace_zero_division_with"),
  drop_empty_groups = options::opt("drop_empty_groups"),
  ignore_inconsistencies = options::opt("ignore_inconsistencies"),
  verbose = options::opt("verbose"),
  progress = options::opt("progress")
)
```

Arguments

predicted	Multi-label prediction results. Expects a data.frame with columns "label_id", "doc_id", "score".
gold_standard	Expects a data.frame with columns "label_id","doc_id".
doc_groups	A two-column data.frame with a column "doc_id" and a second column defining groups of documents to stratify results by. It is recommended that groups are of type factor so that levels are not implicitly dropped during bootstrap replications.
label_groups	A two-column data.frame with a column "label_id" and a second column defining groups of labels to stratify results by. Results in each stratum will restrict gold standard and predictions to the specified label groups as if the vocabulary was consisting of the label group only. All modes "doc-avg", "subj-avg", "micro" are supported within label strata. Nevertheless, mixing mode = "doc-avg" with fine-grained label strata can result in many missing values on document-level results. Also rank-based thresholding (e.g. top 5) will result in inhomogeneous numbers of labels per document within the defined label strata. mode = "subj-avg" or mode = "micro" can be more appropriate in these circumstances.
mode	One of the following aggregation modes: "doc-avg","subj-avg", "micro".

steps
: Number of breaks to divide the interval $[0, 1]$ into. These breaks will be used as quantiles to be calculated on the true positive suggestions' score distribution and therefore build one axis of the grid for computing the pr curve.

thresholds
: Alternatively to steps, one can manually set the thresholds to be used to build the pr curve. Defaults to the quantiles of the true positive suggestions' score distribution to be obtained from `steps`.

limit_range
: A vector of limit values to apply on the rank column. Defaults to NA, applying no cutoff on the predictions' label rank.

optimize_cutoff
: Logical. If `TRUE`, a grid search in the search space specified by `limit_range` and `steps` or `thresholds` is performed to find optimal limit and threshold with respect to F1 measure.

graded_relevance
: A logical indicator for graded relevance. Defaults to `FALSE` for binary relevance. If set to `TRUE`, the `predicted` data.frame should contain a numeric column `"relevance"` with values in the range of $[0, 1]$.

propensity_scored
: Logical, whether to use propensity scores as weights.

label_distribution
: Expects a data.frame with columns `"label_id"`, `"label_freq"`, `"n_docs"`. `label_freq` corresponds to the number of occurences a label has in the gold standard. `n_docs` corresponds to the total number of documents in the gold standard.

cost_fp_constant
: Constant cost assigned to false positives. `cost_fp_constant` must be a numeric value > 0 or one of 'max', 'min', 'mean' (computed with reference to the `gold_standard` label distribution). Defaults to NULL, i.e. label weights are applied to false positives in the same way as to false negatives and true positives.

replace_zero_division_with
: In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1. (Defaults to `NULL`, overwritable using option 'casimir.replace_zero_division_with' or environment variable 'R_CASIMIR_REPLACE_ZERO_DIVISION_WITH')

drop_empty_groups
: Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to `TRUE`, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')

ignore_inconsistencies
: Warnings about data inconsistencies will be silenced. (Defaults to `FALSE`, overwritable using option 'casimir.ignore_inconsistencies' or environment variable 'R_CASIMIR_IGNORE_INCONSISTENCIES')

verbose	Verbose reporting of computation steps for debugging. (Defaults to FALSE, overwritable using option 'casimir.verbose' or environment variable 'R_CASIMIR_VERBOSE')
progress	Display progress bars for iterated computations (like bootstrap CI or pr curves). (Defaults to FALSE, overwritable using option 'casimir.progress' or environment variable 'R_CASIMIR_PROGRESS')

Value

A list of three elements:

- plot_data A data.frame with full pr curves and columns "searchspace_id", "prec", "rec", "prec_cummax", "mode".
- opt_cutoff A data.frame with optimal cutoffs and columns "thresholds", "limits", "searchspace_id", "f1_max", "prec", "rec", "prec_cummax", "mode".
- all_cutoffs A data.frame with all cutoffs and columns "thresholds", "limits", "searchspace_id", "metric", "value", "support", "f1_max", "prec", "rec", "prec_cummax", "mode".

All three data.frames may contain additional stratification variables passed with doc_groups and label_groups. The latter two data.frames are non-empty only if optimize_cutoff == TRUE.

Examples

```
library(ggplot2)
library(casimir)

gold <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "b",
  "A", "c",
  "B", "a",
  "B", "d",
  "C", "a",
  "C", "b",
  "C", "d",
  "C", "f"
)

pred <- tibble::tribble(
  ~doc_id, ~label_id, ~score, ~rank,
  "A", "a", 0.9, 1,
  "A", "d", 0.7, 2,
  "A", "f", 0.3, 3,
  "A", "c", 0.1, 4,
  "B", "a", 0.8, 1,
  "B", "e", 0.6, 2,
  "B", "d", 0.1, 3,
  "C", "f", 0.1, 1,
  "C", "c", 0.2, 2,
  "C", "e", 0.2, 2
)
```

```
pr_curve <- compute_pr_curve(
  pred,
  gold,
  mode = "doc-avg",
  optimize_cutoff = TRUE
)

auc <- compute_pr_auc_from_curve(pr_curve$plot_data)

# note that pr curves take the cummax(prec), not the precision
ggplot(pr_curve$plot_data, aes(x = rec, y = prec_cummax)) +
  geom_point(
    data = pr_curve$opt_cutoff,
    aes(x = rec, y = prec_cummax),
    color = "red",
    shape = "star"
  ) +
  geom_text(
    data = pr_curve$opt_cutoff,
    aes(
      x = rec + 0.2, y = prec_cummax,
      label = paste("f1_opt =", round(f1_max, 3))
    ),
    color = "red"
  ) +
  geom_path() +
  coord_cartesian(xlim = c(0, 1), ylim = c(0, 1))
```

`compute_ranked_retrieval_scores` *Compute ranked retrieval scores*

Description

This function computes the ranked retrieval scores Discounted Cumulative Gain (DCG), Ideal Discounted Cumulative Gain (IDCG), Normalised Discounted Cumulative Gain (NDCG) and Label Ranking Average Precision (LRAP). Ranked retrieval, unlike set retrieval, assumes ordered predictions. Unlike set retrieval metrics, ranked retrieval metrics are logically bound to a document-wise evaluation. Thus, only the aggregation mode "doc-avg" is available for these scores.

Usage

```
compute_ranked_retrieval_scores(
  predicted,
  gold_standard,
  doc_groups = NULL,
  drop_empty_groups = options::opt("drop_empty_groups"),
  progress = options::opt("progress")
)
```

Arguments

predicted	Multi-label prediction results. Expects a data.frame with columns "label_id", "doc_id", "score".
gold_standard	Expects a data.frame with columns "label_id","doc_id".
doc_groups	A two-column data.frame with a column "doc_id" and a second column defining groups of documents to stratify results by. It is recommended that groups are of type factor so that levels are not implicitly dropped during bootstrap replications.
drop_empty_groups	Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to TRUE, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')
progress	Display progress bars for iterated computations (like bootstrap CI or pr curves). (Defaults to FALSE, overwritable using option 'casimir.progress' or environment variable 'R_CASIMIR_PROGRESS')

Value

A data.frame with columns "metric", "mode", "value", "support" and optional grouping variables supplied in doc_groups. Here, support is defined as number of documents that contribute to the document average in aggregation of the overall result.

Examples

```
# some dummy results
gold <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "b",
  "A", "c",
  "A", "d",
  "A", "e",
)

pred <- tibble::tribble(
  ~doc_id, ~label_id, ~score,
  "A", "f", 0.3277,
  "A", "e", 0.32172,
  "A", "b", 0.13517,
  "A", "g", 0.10134,
  "A", "h", 0.09152,
  "A", "a", 0.07483,
  "A", "i", 0.03649,
  "A", "j", 0.03551,
  "A", "k", 0.03397,
  "A", "c", 0.03364
)

results <- compute_ranked_retrieval_scores(
  pred,
```

```
  gold
)
```

`compute_set_retrieval_scores` *Compute multi-label metrics*

Description

Compute multi-label metrics precision, recall, F1 and R-precision for subject indexing results.

Usage

```
compute_set_retrieval_scores(
  predicted,
  gold_standard,
  k = NULL,
  mode = "doc-avg",
  compute_bootstrap_ci = FALSE,
  n_bt = 10L,
  doc_groups = NULL,
  label_groups = NULL,
  graded_relevance = FALSE,
  rename_metrics = FALSE,
  seed = NULL,
  propensity_scored = FALSE,
  label_distribution = NULL,
  cost_fp_constant = NULL,
  replace_zero_division_with = options::opt("replace_zero_division_with"),
  drop_empty_groups = options::opt("drop_empty_groups"),
  ignore_inconsistencies = options::opt("ignore_inconsistencies"),
  verbose = options::opt("verbose"),
  progress = options::opt("progress")
)

compute_set_retrieval_scores_dplyr(
  predicted,
  gold_standard,
  k = NULL,
  mode = "doc-avg",
  compute_bootstrap_ci = FALSE,
  n_bt = 10L,
  doc_groups = NULL,
  label_groups = NULL,
  graded_relevance = FALSE,
  rename_metrics = FALSE,
```

```
  seed = NULL,
  propensity_scored = FALSE,
  label_distribution = NULL,
  cost_fp_constant = NULL,
  ignore_inconsistencies = FALSE,
  verbose = FALSE,
  progress = FALSE
)
```

Arguments

predicted	Multi-label prediction results. Expects a data.frame with columns "label_id", "doc_id".
gold_standard	Expects a data.frame with columns "label_id","doc_id".
k	An integer limit on the number of predictions per document to consider. Requires a column "score" in input predicted.
mode	One of the following aggregation modes: "doc-avg","subj-avg", "micro".
compute_bootstrap_ci	A logical indicator for computing bootstrap CIs.
n_bt	An integer number of resamples to be used for bootstrapping.
doc_groups	A two-column data.frame with a column "doc_id" and a second column defining groups of documents to stratify results by. It is recommended that groups are of type factor so that levels are not implicitly dropped during bootstrap replications.
label_groups	A two-column data.frame with a column "label_id" and a second column defining groups of labels to stratify results by. Results in each stratum will restrict gold standard and predictions to the specified label groups as if the vocabulary was consisting of the label group only. All modes "doc-avg", "subj-avg", "micro" are supported within label strata. Nevertheless, mixing mode = "doc-avg" with fine-grained label strata can result in many missing values on document-level results. Also rank-based thresholding (e.g. top 5) will result in inhomogeneous numbers of labels per document within the defined label strata. mode = "subj-avg" or mode = "micro" can be more appropriate in these circumstances.
graded_relevance	A logical indicator for graded relevance. Defaults to FALSE for binary relevance. If set to TRUE, the predicted data.frame should contain a numeric column "relevance" with values in the range of $[0, 1]$.
rename_metrics	If set to TRUE, the metric names in the output will be renamed if: graded_relevance == TRUE prefixed with *"g-"* to indicate that metrics are computed with graded relevance. propensity_scored == TRUE prefixed with *"ps-"* to indicate that metrics are computed with propensity scores. !is.null(k) suffixed with *"@k"* to indicate that metrics are limited to top k predictions.
seed	Pass a seed to make bootstrap replication reproducible.

`propensity_scored`
: Logical, whether to use propensity scores as weights.

`label_distribution`
: Expects a data.frame with columns `"label_id"`, `"label_freq"`, `"n_docs"`. `label_freq` corresponds to the number of occurences a label has in the gold standard. `n_docs` corresponds to the total number of documents in the gold standard.

`cost_fp_constant`
: Constant cost assigned to false positives. `cost_fp_constant` must be a numeric value > 0 or one of 'max', 'min', 'mean' (computed with reference to the `gold_standard` label distribution). Defaults to NULL, i.e. label weights are applied to false positives in the same way as to false negatives and true positives.

`replace_zero_division_with`
: In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1. (Defaults to `NULL`, overwritable using option 'casimir.replace_zero_division_with' or environment variable 'R_CASIMIR_REPLACE_ZERO_DIVISION_WITH')

`drop_empty_groups`
: Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to `TRUE`, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')

`ignore_inconsistencies`
: Warnings about data inconsistencies will be silenced. (Defaults to `FALSE`, overwritable using option 'casimir.ignore_inconsistencies' or environment variable 'R_CASIMIR_IGNORE_INCONSISTENCIES')

`verbose`
: Verbose reporting of computation steps for debugging. (Defaults to `FALSE`, overwritable using option 'casimir.verbose' or environment variable 'R_CASIMIR_VERBOSE')

`progress`
: Display progress bars for iterated computations (like bootstrap CI or pr curves). (Defaults to `FALSE`, overwritable using option 'casimir.progress' or environment variable 'R_CASIMIR_PROGRESS')

Value

A data.frame with columns `"metric"`, `"mode"`, `"value"`, `"support"` and optional grouping variables supplied in `doc_groups` or `label_groups`. Here, `support` is defined for each `mode` as:

`mode == "doc-avg"` The number of tested documents.

`mode == "subj-avg"` The number of labels contributing to the subj-average.

`mode == "micro"` The number of doc-label pairs contributing to the denominator of the respective metric, e.g. $tp + fp$ for precision, $tp + fn$ for recall, $tp + (fp + fn)/2$ for F1 and $min(tp + fp, tp + fn)$ for R-precision.

Functions

- `compute_set_retrieval_scores_dplyr()`: Variant with internal usage of dplyr rather than collapse library. Tends to be slower, but more stable.

Examples

```
library(tidyverse)
library(casimir)
library(furrr)

gold <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "b",
  "A", "c",
  "B", "a",
  "B", "d",
  "C", "a",
  "C", "b",
  "C", "d",
  "C", "f",
)

pred <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "d",
  "A", "f",
  "B", "a",
  "B", "e",
  "C", "f",
)

plan(sequential) # or whatever resources you have

a <- compute_set_retrieval_scores(
  pred, gold,
  mode = "doc-avg",
  compute_bootstrap_ci = TRUE,
  n_bt = 100L
)

ggplot(a, aes(x = metric, y = value)) +
  geom_bar(stat = "identity") +
  geom_errorbar(aes(ymin = ci_lower, ymax = ci_upper)) +
  facet_wrap(vars(metric), scales = "free")

# example with graded relevance
pred_w_relevance <- tibble::tribble(
  ~doc_id, ~label_id, ~relevance,
  "A", "a", 1.0,
  "A", "d", 0.0,
```

```
  "A", "f", 0.0,
  "B", "a", 1.0,
  "B", "e", 1 / 3,
  "C", "f", 1.0,
)

b <- compute_set_retrieval_scores(
  pred_w_relevance, gold,
  mode = "doc-avg",
  graded_relevance = TRUE
)
```

`create_comparison` *Join gold standard and predicted results*

Description

Join the gold standard and the predicted results in one table based on the document id and the label id.

Usage

```
create_comparison(
  predicted,
  gold_standard,
  doc_groups = NULL,
  label_groups = NULL,
  graded_relevance = FALSE,
  propensity_scored = FALSE,
  label_distribution = NULL,
  ignore_inconsistencies = options::opt("ignore_inconsistencies")
)
```

Arguments

`predicted`	Multi-label prediction results. Expects a data.frame with columns `"label_id"`, `"doc_id"`.
`gold_standard`	Expects a data.frame with columns `"label_id"`,`"doc_id"`.
`doc_groups`	A two-column data.frame with a column `"doc_id"` and a second column defining groups of documents to stratify results by. It is recommended that groups are of type factor so that levels are not implicitly dropped during bootstrap replications.
`label_groups`	A two-column data.frame with a column `"label_id"` and a second column defining groups of labels to stratify results by. Results in each stratum will restrict gold standard and predictions to the specified label groups as if the vocabulary was consisting of the label group only. All modes `"doc-avg"`,

"subj-avg", "micro" are supported within label strata. Nevertheless, mixing mode = "doc-avg" with fine-grained label strata can result in many missing values on document-level results. Also rank-based thresholding (e.g. top 5) will result in inhomogeneous numbers of labels per document within the defined label strata. mode = "subj-avg" or mode = "micro" can be more appropriate in these circumstances.

graded_relevance
: A logical indicator for graded relevance. Defaults to FALSE for binary relevance. If set to TRUE, the predicted data.frame should contain a numeric column "relevance" with values in the range of $[0, 1]$.

propensity_scored
: Logical, whether to use propensity scores as weights.

label_distribution
: Expects a data.frame with columns "label_id", "label_freq", "n_docs". label_freq corresponds to the number of occurences a label has in the gold standard. n_docs corresponds to the total number of documents in the gold standard.

ignore_inconsistencies
: Warnings about data inconsistencies will be silenced. (Defaults to FALSE, overwritable using option 'casimir.ignore_inconsistencies' or environment variable 'R_CASIMIR_IGNORE_INCONSISTENCIES')

Value

A data.frame with columns "label_id", "doc_id", "suggested", "gold".

Examples

```
library(casimir)

gold <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "b",
  "A", "c",
  "B", "a",
  "B", "d",
  "C", "a",
  "C", "b",
  "C", "d",
  "C", "f"
)

pred <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "d",
  "A", "f",
  "B", "a",
  "B", "e",
```

```
  "C", "f"
)

create_comparison(pred, gold)
```

`create_rank_col` *Create a rank column*

Description

Create a rank per document id based on score.

Usage

```
create_rank_col(df)

create_rank_col_dplyr(df)
```

Arguments

`df`	A data.frame with columns `"doc_id"`, `"score"`.

Value

The input data.frame `df` with an additional column `"rank"`.

Functions

- `create_rank_col_dplyr()`: Variant with internal usage of dplyr rather than collapse library.

`dcg_score` *Helper function for document-wise computation of ranked retrieval scores*

Description

Helper function for document-wise computation of ranked retrieval scores DCG, NDCG and LRAP. Implemented as in Annif https://github.com/NatLibFi/Annif/blob/master/annif/eval.py. Reference implementation of DCG to test against.

Usage

```
dcg_score(gold_vs_pred, limit = NULL)
```

Arguments

`gold_vs_pred`	A data.frame as generated by `create_comparison`.
`limit`	An integer cutoff value for DCG@N.

Value

The numeric value of DCG.

`dnb_gold_standard` *DNB gold standard data for computing evaluation metrics*

Description

A subset of documents found in the catalogue of the DNB with intellectually assigned subject labels from the GND subject vocabulary. The document ids match those in the `dnb_test_predictions` dataset.

Usage

```
dnb_gold_standard
```

Format

`dnb_gold_standard`**:**

A data.frame with 337 rows and 2 columns:

`doc_id` DNB identifier of a document in the catalogue.

`label_id` DNB identifier of a concept in the GND subject vocabulary.

`dnb_label_distribution`

DNB label distribution for computing propensity scored metrics

---

Description

A subset of labels used in the catalogue of the DNB along with their frequencies of occurrence. The label_ids match those in the `dnb_gold_standard` and `dnb_test_predictions` datasets.

Usage

```
dnb_label_distribution
```

Format

`dnb_label_distribution`**:**

A data frame with 7,772 rows and 3 columns:

`label_id` DNB identifier of a concept in the GND subject vocabulary.

`label_freq` Number of occurences of the specified label in the overall catalogue.

`n_docs` Overall number of documents in the ground truth dataset.

dnb_test_predictions *DNB test predictions for computing evaluation metrics*

Description

A subset of documents found in the catalogue of the DNB with predictions generated with some arbitrary indexing method. The document ids match those in the dnb_gold_standard dataset.

Usage

```
dnb_test_predictions
```

Format

dnb_test_predictions:

A data frame with 100,000 rows and 3 columns:

doc_id DNB identifier of a document in the catalogue.

label_id DNB identifier of a concept in the GND subject vocabulary.

score A confidence score in $[0, 1]$ generated by the indexing method.

find_ps_rprec_deno *Compute the denominator for R-precision*

Description

Compute the denominator for R-precision based on propensity scored ranking of gold standard labels.

Usage

```
find_ps_rprec_deno(gold_vs_pred, grouping_var, cost_fp)

find_ps_rprec_deno_dplyr(gold_vs_pred, grouping_var, cost_fp)
```

Arguments

gold_vs_pred	A data.frame with logical columns "suggested", "gold" as produced by create_comparison.
grouping_var	A character vector of grouping variables that must be present in gold_vs_pred (dplyr version requires rlang symbols).
cost_fp	A numeric value > 0, defaults to NULL.

Value

A data.frame with columns "n_gold", "n_suggested", "tp", "fp", "fn", "delta_relevance", "rprec_deno".

Functions

- `find_ps_rprec_deno_dplyr()`: Variant with dplyr based internals rather than collapse internals.

`generate_pr_auc_replica` *Compute bootstrap replica of pr auc*

Description

Helper function which performs the major bootstrap operation and wraps the repeated application of `summarise_intermediate_results` and `compute_pr_auc_from_curve` for each bootstrap run.

Usage

```
generate_pr_auc_replica(
  intermed_res_all_thrsld,
  seed,
  n_bt,
  propensity_scored,
  replace_zero_division_with = options::opt("replace_zero_division_with"),
  progress = options::opt("progress")
)
```

Arguments

`intermed_res_all_thrsld`	Intermediate results as produced by `compute_intermediate_results`, with a column `"searchspace_id"` as grouping variable.
`seed`	Pass a seed to make bootstrap replication reproducible.
`n_bt`	An integer number of resamples to be used for bootstrapping.
`propensity_scored`	Logical, whether to use propensity scores as weights.
`replace_zero_division_with`	In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1. (Defaults to `NULL`, overwritable using option 'casimir.replace_zero_division_with' or environment variable 'R_CASIMIR_REPLACE_ZERO_DIVISION_WITH')
`progress`	Display progress bars for iterated computations (like bootstrap CI or pr curves). (Defaults to `FALSE`, overwritable using option 'casimir.progress' or environment variable 'R_CASIMIR_PROGRESS')

Value

A data.frame with columns "boot_replicate", "pr_auc".

---

generate_replicate_results
Compute bootstrapping results

Description

Wrapper for computing n_bt bootstrap replica, combining the functionality of compute_intermediate_results and summarise_intermediate_results.

Usage

```
generate_replicate_results(
  base_compare,
  n_bt,
  grouping_var,
  seed = NULL,
  ps_flags = list(intermed = FALSE, summarise = FALSE),
  label_distribution = NULL,
  cost_fp = NULL,
  replace_zero_division_with = options::opt("replace_zero_division_with"),
  drop_empty_groups = options::opt("drop_empty_groups"),
  progress = options::opt("progress")
)

generate_replicate_results_dplyr(
  base_compare,
  n_bt,
  grouping_var,
  seed = NULL,
  label_distribution = NULL,
  ps_flags = list(intermed = FALSE, summarise = FALSE),
  cost_fp = NULL,
  progress = FALSE
)
```

Arguments

base_compare	A data.frame as generated by create_comparison.
n_bt	An integer number of resamples to be used for bootstrapping.
grouping_var	A character vector of variables that must be present in base_compare.
seed	A seed passed to resampling step for reproducibility.
ps_flags	A list as returned by set_ps_flags.

label_distribution
: Expects a data.frame with columns "label_id", "label_freq", "n_docs". label_freq corresponds to the number of occurences a label has in the gold standard. n_docs corresponds to the total number of documents in the gold standard.

cost_fp
: A numeric value > 0, defaults to NULL.

replace_zero_division_with
: In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1. (Defaults to NULL, overwritable using option 'casimir.replace_zero_division_with' or environment variable 'R_CASIMIR_REPLACE_ZERO_DIVISION_WITH')

drop_empty_groups
: Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to TRUE, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')

progress
: Display progress bars for iterated computations (like bootstrap CI or pr curves). (Defaults to FALSE, overwritable using option 'casimir.progress' or environment variable 'R_CASIMIR_PROGRESS')

Value

A data.frame containing n_bt boot replica of results as returned by compute_intermediate_results and summarise_intermediate_results.

Functions

- generate_replicate_results_dplyr(): Variant with dplyr based internals rather than collapse internals.

---

helper_f *Calculate bootstrapping results for one sample*

Description

Internal wrapper for computing bootstrapping results on one sample, combining the functionality of compute_intermediate_results and summarise_intermediate_results.

Usage

```
helper_f(
  sampled_id_list,
  compare_cpy,
  grouping_var,
  label_distribution = NULL,
  ps_flags = list(intermed = FALSE, summarise = FALSE),
  cost_fp = NULL,
  replace_zero_division_with = options::opt("replace_zero_division_with"),
  drop_empty_groups = options::opt("drop_empty_groups")
)
```

Arguments

`sampled_id_list`
: A list of all doc_ids of this bootstrap.

`compare_cpy`
: As created by `create_comparison`.

`grouping_var`
: A vector of variables to be used for aggregation.

`label_distribution`
: Expects a data.frame with columns `"label_id"`, `"label_freq"`, `"n_docs"`. `label_freq` corresponds to the number of occurences a label has in the gold standard. `n_docs` corresponds to the total number of documents in the gold standard.

`ps_flags`
: A list as returned by `set_ps_flags`.

`cost_fp`
: A numeric value > 0, defaults to NULL.

`replace_zero_division_with`
: In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1. (Defaults to `NULL`, overwritable using option 'casimir.replace_zero_division_with' or environment variable 'R_CASIMIR_REPLACE_ZERO_DIVISION_WITH')

`drop_empty_groups`
: Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to `TRUE`, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')

Value

A data.frame as returned by `summarise_intermediate_results`.

helper_f_dplyr *Calculate bootstrapping results for one sample*

Description

Internal wrapper for computing bootstrapping results on one sample, combining the functionality of `compute_intermediate_results` and `summarise_intermediate_results`.

Usage

```
helper_f_dplyr(
  sampled_id_list,
  compare_cpy,
  grouping_var,
  ps_flags = list(intermed = FALSE, summarise = FALSE),
  label_distribution = NULL,
  cost_fp = NULL
)
```

Arguments

`sampled_id_list`	A list of all doc_ids of this bootstrap.
`compare_cpy`	As created by `create_comparison`.
`grouping_var`	A vector of variables to be used for aggregation.
`ps_flags`	A list with logicals `"intermed"` and `"summarise"`.
`label_distribution`	Expects a data.frame with columns `"label_id"`, `"label_freq"`, `"n_docs"`. `label_freq` corresponds to the number of occurences a label has in the gold standard. `n_docs` corresponds to the total number of documents in the gold standard.
`cost_fp`	A numeric value > 0, defaults to NULL.

Value

A data.frame as returned by `summarise_intermediate_results_dplyr`.

`join_propensity_scores` *Join propensity scores*

Description

Helper function to perform a secure join of a comparison matrix with propensity scores.

Usage

```
join_propensity_scores(input_data, label_weights)

join_propensity_scores_dplyr(input_data, label_weights)
```

Arguments

`input_data`	A data.frame containing at least the column `"label_id"`.
`label_weights`	Expects a data.frame with columns `"label_id"`,`"label_weight"`.

Value

The input data.frame `input_data` with an additional column `"label_weight"`.

Functions

- `join_propensity_scores_dplyr()`: Variant with dplyr based internals rather than collapse internals.

Examples

```
library(casimir)

gold <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "b",
  "A", "c",
  "B", "a",
  "B", "d",
  "C", "a",
  "C", "b",
  "C", "d",
  "C", "f"
)

pred <- tibble::tribble(
  ~doc_id, ~label_id,
  "A", "a",
  "A", "d",
```

```
  "A", "f",
  "B", "a",
  "B", "e",
  "C", "f"
)

label_distribution <- tibble::tribble(
  ~label_id, ~label_freq, ~n_docs,
  "a", 10000, 10100,
  "b", 1000, 10100,
  "c", 100, 10100,
  "d", 1, 10100,
  "e", 1, 10100,
  "f", 2, 10100,
  "g", 0, 10100
)

comp <- create_comparison(gold, pred)
label_weights <- compute_propensity_scores(label_distribution)
comp_w_label_weights <- join_propensity_scores(
  input_data = comp,
  label_weights = label_weights
)
```

`lrap_score` *Helper function for document-wise computation of ranked retrieval scores*

Description

Helper function for document-wise computation of ranked retrieval scores DCG, NDCG and LRAP. Implemented as in Annif https://github.com/NatLibFi/Annif/blob/master/annif/eval.py. Reference implementation for Label Ranking Average Precision.

Usage

```
lrap_score(gold_vs_pred)
```

Arguments

`gold_vs_pred`	A data.frame as generated by `create_comparison`.

Value

The numeric value of LRAP.

ndcg_score — *Helper function for document-wise computation of ranked retrieval scores*

Description

Helper function for document-wise computation of ranked retrieval scores DCG, NDCG and LRAP. Implemented as in Annif https://github.com/NatLibFi/Annif/blob/master/annif/eval.py. Reference implementation for NDCG to test against.

Usage

```
ndcg_score(gold_vs_pred, limit = NULL)
```

Arguments

gold_vs_pred	A data.frame as generated by create_comparison.
limit	An integer cutoff value for NDCG@N.

Value

The numeric value of NDCG.

options — *casimir Options*

Description

Internally used, package-specific options. All options will prioritize R options() values, and fall back to environment variables if undefined. If neither the option nor the environment variable is set, a default value is used.

Checking Option Values

Option values specific to casimir can be accessed by passing the package name to env.

```
options::opts(env = "casimir")

options::opt(x, default, env = "casimir")
```

Options

ignore_inconsistencies Warnings about data inconsistencies will be silenced.

default: FALSE

option: casimir.ignore_inconsistencies

envvar: R_CASIMIR_IGNORE_INCONSISTENCIES (evaluated if possible, raw string otherwise)

progress Display progress bars for iterated computations (like bootstrap CI or pr curves).

default: FALSE

option: casimir.progress

envvar: R_CASIMIR_PROGRESS (evaluated if possible, raw string otherwise)

verbose Verbose reporting of computation steps for debugging.

default: FALSE

option: casimir.verbose

envvar: R_CASIMIR_VERBOSE (evaluated if possible, raw string otherwise)

check_group_names Perform replacement of dots in grouping columns. Disable for faster computation if you can make sure that all columns used for grouping ("doc_id", "label_id", "doc_groups", "label_groups") do not contain dots.

default: TRUE

option: casimir.check_group_names

envvar: R_CASIMIR_CHECK_GROUP_NAMES (evaluated if possible, raw string otherwise)

drop_empty_groups Should empty levels of factor variables be dropped in grouped set retrieval computation?

default: TRUE

option: casimir.drop_empty_groups

envvar: R_CASIMIR_DROP_EMPTY_GROUPS (evaluated if possible, raw string otherwise)

replace_zero_division_with In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1.

default: NULL

option: casimir.replace_zero_division_with

envvar: R_CASIMIR_REPLACE_ZERO_DIVISION_WITH (evaluated if possible, raw string otherwise)

See Also

options getOption Sys.setenv Sys.getenv

option_params *Declaration of options to be used as identical function arguments*

Description

Declaration of options to be used as identical function arguments

Arguments

check_group_names
: Perform replacement of dots in grouping columns. Disable for faster computation if you can make sure that all columns used for grouping ("doc_id", "label_id", "doc_groups", "label_groups") do not contain dots. (Defaults to TRUE, overwritable using option 'casimir.check_group_names' or environment variable 'R_CASIMIR_CHECK_GROUP_NAMES')

ignore_inconsistencies
: Warnings about data inconsistencies will be silenced. (Defaults to FALSE, overwritable using option 'casimir.ignore_inconsistencies' or environment variable 'R_CASIMIR_IGNORE_INCONSISTENCIES')

drop_empty_groups
: Should empty levels of factor variables be dropped in grouped set retrieval computation? (Defaults to TRUE, overwritable using option 'casimir.drop_empty_groups' or environment variable 'R_CASIMIR_DROP_EMPTY_GROUPS')

replace_zero_division_with
: In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1. (Defaults to NULL, overwritable using option 'casimir.replace_zero_division_with' or environment variable 'R_CASIMIR_REPLACE_ZERO_DIVISION_WITH')

progress
: Display progress bars for iterated computations (like bootstrap CI or pr curves). (Defaults to FALSE, overwritable using option 'casimir.progress' or environment variable 'R_CASIMIR_PROGRESS')

verbose
: Verbose reporting of computation steps for debugging. (Defaults to FALSE, overwritable using option 'casimir.verbose' or environment variable 'R_CASIMIR_VERBOSE')

process_cost_fp *Process cost for false positives*

Description

Calculate the cost for false positives depending on the chosen `cost_fp_constant`.

Usage

```
process_cost_fp(cost_fp_constant, gold_vs_pred)
```

Arguments

`cost_fp_constant`	Constant cost assigned to false positives. `cost_fp_constant` must be a numeric value > 0 or one of 'max', 'min', 'mean' (computed with reference to the `gold_standard` label distribution). Defaults to NULL, i.e. label weights are applied to false positives in the same way as to false negatives and true positives.
`gold_vs_pred`	A data.frame with logical columns `"suggested"`, `"gold"` as produced by `create_comparison`.

Value

A numeric value > 0.

---

pr_curve_post_processing *Postprocessing of pr curve data*

Description

Reshape pr curve data to a format that is easier for plotting.

Usage

```
pr_curve_post_processing(results_summary)
```

Arguments

`results_summary`	As produced by `summarise_intermediate_results`.

Value

A data.frame with columns `"searchspace_id"`, `"prec"`, `"rec"`, `"prec_cummax"` and possible additional stratification variables.

`rename_metrics` *Rename metrics*

Description

Rename metric names for generalised precision etc. The output will be renamed if:

`graded_relevance == TRUE` prefixed with *"g-"* to indicate that metrics are computed with graded relevance.

`propensity_scored == TRUE` prefixed with *"ps-"* to indicate that metrics are computed with propensity scores.

`!is.null(k)` suffixed with *"@k"* to indicate that metrics are limited to top k predictions.

Usage

```
rename_metrics(
  res_df,
  k = NULL,
  propensity_scored = FALSE,
  graded_relevance = FALSE
)
```

Arguments

`res_df`	A data.frame with a column `"metric"` containing metric names `"f1"`, `"prec"`, `"rec"`, `"rprec"`.
`k`	An integer limit on the number of predictions per document to consider. Requires a column `"score"` in input `predicted`.
`propensity_scored`	Logical, whether to use propensity scores as weights.
`graded_relevance`	A logical indicator for graded relevance. Defaults to `FALSE` for binary relevance. If set to `TRUE`, the `predicted` data.frame should contain a numeric column `"relevance"` with values in the range of $[0, 1]$.

Value

The input data.frame `res_df` with renamed metrics for generalised precision etc.

set_grouping_var *Set grouping variables*

Description

Determine the appropriate grouping variables for each aggregation mode.

Usage

```
set_grouping_var(mode, doc_groups, label_groups, var = NULL)
```

Arguments

mode	One of the following aggregation modes: "doc-avg","subj-avg", "micro".
doc_groups	A two-column data.frame with a column "doc_id" and a second column defining groups of documents to stratify results by. It is recommended that groups are of type factor so that levels are not implicitly dropped during bootstrap replications.
label_groups	A two-column data.frame with a column "label_id" and a second column defining groups of labels to stratify results by. Results in each stratum will restrict gold standard and predictions to the specified label groups as if the vocabulary was consisting of the label group only. All modes "doc-avg", "subj-avg", "micro" are supported within label strata. Nevertheless, mixing mode = "doc-avg" with fine-grained label strata can result in many missing values on document-level results. Also rank-based thresholding (e.g. top 5) will result in inhomogeneous numbers of labels per document within the defined label strata. mode = "subj-avg" or mode = "micro" can be more appropriate in these circumstances.
var	Additional variables to include.

Value

A character vector of variables determining the grouping structure.

set_ps_flags *Set flags for propensity scores*

Description

Generate flags if propensity scores should be applied to intermediate results or summarised results.

Usage

```
set_ps_flags(mode, propensity_scored)
```

Arguments

mode	One of the following aggregation modes: "doc-avg","subj-avg", "micro".
propensity_scored	Logical, whether to use propensity scores as weights.

Value

A list containing logical flags "intermed" and "summarise".

---

summarise_intermediate_results

Compute the mean of intermediate results

Description

Compute the mean of intermediate results created by compute_intermediate_results.

Usage

```
summarise_intermediate_results(
  intermediate_results,
  propensity_scored = FALSE,
  label_distribution = NULL,
  set = FALSE,
  replace_zero_division_with = options::opt("replace_zero_division_with")
)
```

Arguments

intermediate_results	As produced by compute_intermediate_results. This requires a list containing: • results_table A data.frame with columns "prec", "rprec", "rec", "f1". • grouping_var A character vector of variables to group by.
propensity_scored	Logical, whether to use propensity scores as weights.
label_distribution	Expects a data.frame with columns "label_id", "label_freq", "n_docs". label_freq corresponds to the number of occurences a label has in the gold standard. n_docs corresponds to the total number of documents in the gold standard.
set	Logical. Allow in-place modification of intermediate_results. Only recommended for internal package usage.

replace_zero_division_with
: In macro averaged results (doc-avg, subj-avg), it may occur that some instances have no predictions or no gold standard. In these cases, calculating precision and recall may lead to division by zero. CASIMiR standardly removes these missing values from macro averages, leading to a smaller support (count of instances that were averaged). Other implementations of macro averaged precision and recall default to 0 in these cases. This option allows to control the default. Set any value between 0 and 1. (Defaults to NULL, overwritable using option 'casimir.replace_zero_division_with' or environment variable 'R_CASIMIR_REPLACE_ZERO_DIVISION_WITH')

Value

A data.frame with columns "metric", "value".

---

summarise_intermediate_results_dplyr

Compute the mean of intermediate results

Description

Compute the mean of intermediate results created by compute_intermediate_results. Variant with dplyr based internals rather than collapse internals.

Usage

```
summarise_intermediate_results_dplyr(
  intermediate_results,
  propensity_scored = FALSE,
  label_distribution = NULL
)
```

Arguments

intermediate_results
: As produced by compute_intermediate_results. This requires a list containing:
  - results_table A data.frame with columns "prec", "rprec", "rec", "f1".
  - grouping_var A character vector of variables to group by.

propensity_scored
: Logical, whether to use propensity scores as weights.

label_distribution
: Expects a data.frame with columns "label_id", "label_freq", "n_docs". label_freq corresponds to the number of occurences a label has in the gold standard. n_docs corresponds to the total number of documents in the gold standard.

Value

A data.frame with columns `"metric"`, `"value"`.

Index