ir-measures

Sean MacAvaney

Jan 03, 2022

CONTENTS

ir-measures is a Python package that interfaces with several information retrieval (IR) evaluation tools, including pytrec_eval, gdeval, trectools, and others.

This package aims to simplify IR evaluation by providing an easy and flexible evaluation interface and by standardizing measure names (and their parameters).

CHAPTER ONE

QUICK START

Install ir-measures from pip:

```
$ pip install ir-measures
```

Compute measures from the command line:

```
$ ir_measures path/to/qrels path/to/run nDCG@10 P@5 'P(rel=2)@5' Judged@10
nDCG@10  0.6251
P@5   0.7486
P(rel=2)@5  0.6000
Judged@10   0.9486
```

You can alternatively use a dataset ID from ir_datasets in place of `path/to/qrels`.

Compute measures from python:

```
>>> import ir_measures
>>> from ir_measures import *
>>> qrels = ir_measures.read_trec_qrels('path/to/qrels')
>>> run = ir_measures.read_trec_run('path/to/run')
>>> ir_measures.calc_aggregate([nDCG@10, P@5, P(rel=2)@5, Judged@10], qrels, run)
{
    nDCG@10: 0.6251,
    P@5: 0.7486,
    P(rel=2)@5: 0.6000,
    Judged@10: 0.9486
}
```

CHAPTER TWO

PYTERRIER INTEGRATION

ir_measures is used by the PyTerrier platform to evaluate ranking pipelines. In the following example, BM25 is evaluated using the standard measures for the TREC Deep Learning benchmark, provided by ir_measures:

```
import pyterrier as pt
from pyterrier.measures import *
dataset = pt.get_dataset("trec-deep-learning-passages")
bm25 = pt.BatchRetrieve(index, wmodel="BM25")
pt.Experiment(
    [bm25],
    dataset.get_topics("test-2019"),
    dataset.get_qrels("test-2019"),
    eval_metrics=[RR(rel=2), nDCG@10, nDCG@100, AP(rel=2)],
#                 ^ using ir_measures
)
```

CHAPTER

THREE

TABLE OF CONTENTS

3.1 Getting Started

3.1.1 Installation

You can install ir-measures from pip:

```
$ pip install ir-measures
```

You can also install from current development version:

```
$ git clone git@github.com:terrierteam/ir_measures.git
$ cd ir_measures
$ python setup.py install
```

3.1.2 Command Line Interface

`ir_measures` can be used on the command line with an interface similar to trec_eval:

```
$ ir_measures path/to/qrels path/to/run nDCG@10 P@5 'P(rel=2)@5' Judged@10
nDCG@10  0.6251
P@5   0.7486
P(rel=2)@5  0.6000
Judged@10   0.9486
```

You can alternatively use a dataset ID from ir_datasets in place of `path/to/qrels`.

You can see per-topic results using the `-q` flag (similar to trec_eval):

```
$ ir_measures -q path/to/qrels path/to/run nDCG@10 P@5 'P(rel=2)@5' Judged@10
1   P@5 0.6000
1   nDCG@10 0.5134
2   P@5 1.0000
2   nDCG@10 0.8522
3   P@5 1.0000
...
34  Judged@10   1.0000
35  Judged@10   1.0000
all nDCG@10 0.6251
all P@5 0.7486
```

(continues on next page)

(continued from previous page)

```
all	P(rel=2)@5	0.6000
all	Judged@10	0.9486
```

The first column in the output is the query ID (or `all` for the aggregated results, which can be disabled with the `-n` flag). Results are written to the output stream as they are calculated by `iter_calc`. Thus, they may not be in a predictable order[1].

Full list of command line arguments:

- `-h` (`--help`): print information about running the command
- `-p X` (`--places X`): number of decimal places to use when writing the output. Default: `4`.
- `-q` (`--by_query`): print the results by query (topic), as shown above.
- `-n` (`--no_summary`): when used with `-q`, does not print aggregated (`all`) values.
- `--provider X`: forces the use of a particular provider, rather than using the default fallback approach. Possible values are: `pytrec_eval`, `judged`, `gdeval`, `trectools`, and `msmarco`.

3.1.3 Python Interface

Compute measures from python:

```
>>> import ir_measures
>>> from ir_measures import *
>>> qrels = ir_measures.read_trec_qrels('path/to/qrels')
>>> run = ir_measures.read_trec_run('path/to/run')
>>> ir_measures.calc_aggregate([nDCG@10, P@5, P(rel=2)@5, Judged@10], qrels, run)
{
    nDCG@10: 0.6251,
    P@5: 0.7486,
    P(rel=2)@5: 0.6000,
    Judged@10: 0.9486
}
```

Per-topic results can be calculated using `iter_calc`:

```
>>> for metric in ir_measures.iter_calc([nDCG@10, P@5, P(rel=2)@5, Judged@10], qrels,␣
↪run):
...     print(metric)
Metric(query_id='1', measure=P@5, value=0.6)
Metric(query_id='1', measure=nDCG@10, value=0.5134306625775544)
Metric(query_id='2', measure=P@5, value=1.0)
Metric(query_id='2', measure=nDCG@10, value=0.8521705090845474)
Metric(query_id='3', measure=P@5, value=1.0)
...
Metric(query_id='33', measure=Judged@10, value=1.0)
Metric(query_id='34', measure=Judged@10, value=1.0)
```

Here again, the results from `iter_calc` may not be returned in a predictable order[1].

[1] In the examples, `P@5` and `nDCG@10` are returned first, as they are both calculated in one invocation of `pytrec_eval`. Then, results for `P(rel=2)@5` are returned (as a second invocation of `pytrec_eval` because it only supports one relevance level at a time). Finally, results for `Judged@10` are returned, as these are calculated by the `judged` provider.

3.1.4 Qrels formats

Query relevance assessments can be provided in a variety of formats.

namedtuple iterable: Any iterable of named tuples. You can use `ir_measures.Qrel`, or any other NamedTuple with the fields `query_id`, `doc_id`, and `relevance` (order and additional fields do not matter if another type of NamedTuple; the field names just need to match):

```
qrels = [
    ir_measures.Qrel("Q0", "D0", 0),
    ir_measures.Qrel("Q0", "D1", 1),
    ir_measures.Qrel("Q1", "D0", 0),
    ir_measures.Qrel("Q1", "D3", 2),
]
```

Note that if the results are an iterator (such as the result of a generator), `ir_measures` will consume the entire sequence.

`ir_measures.Qrel` support an optional fourth parameter, `iteration`. This is the source of the subtopic ID used for diversity measures (name matches TREC conventions). Note that unlike TREC-formatted qrels, this parameter is the last element, since this is required for optional parameters in namedtuples.

Pandas dataframe: A pandas dataframe with the columns `query_id`, `doc_id`, and `relevance`:

```
import pandas as pd
qrels = pd.DataFrame([
    {'query_id': "Q0", 'doc_id': "D0", 'relevance': 0},
    {'query_id': "Q0", 'doc_id': "D1", 'relevance': 1},
    {'query_id': "Q1", 'doc_id': "D0", 'relevance': 0},
    {'query_id': "Q1", 'doc_id': "D3", 'relevance': 2},
])
```

Dataframes support an optional fourth parameter, `iteration`. This is the source of the subtopic ID used for diversity measures (name matches TREC conventions).

If your dataframe has columns named something else, you can always map them with the `rename` function. For instance, if your dataframe has the columns `qid`, `docno`, and `label`, you can easily make a qrels dataframe that is compatible with ir-measures like so:

```
qrels = df.rename(columns={'qid': 'query_id', 'docno': 'doc_id', 'label': 'relevance'})
```

TREC-formatted qrels file: You can read a TREC-formatted qrels file:

```
# a file path:
qrels = ir_measures.read_trec_qrels('path/to/qrels')
# raw qrels file contents:
qrels = ir_measures.read_trec_qrels('''
Q0 0 D0 0
Q0 0 D1 1
Q1 0 D0 0
Q1 0 D3 2
''')
# TREC qrels format: "query_id iteration doc_id relevance".
```

Note that `read_trec_qrels` returns a generator. If you need to use the qrels multiple times, wrap it in the `list` constructor to read the all qrels into memory.

ir_datasets qrels: Qrels from the ir_datasets package. This mode simply adheres to the **namedtuple iterable** specification above:

```
import ir_datasets
qrels = ir_datasets.load('trec-robust04').qrels_iter()
```

dict-of-dict: Qrels structured in a hierarchy. At the first level, query IDs map to another dictionary. At the second level, document IDs map to (integer) relevance scores:

```
qrels = {
    'Q0': {
        "D0": 0,
        "D1": 1,
    },
    "Q1": {
        "D0": 0,
        "D3": 2
    }
}
```

Note that this format does not support the iteration field, so it should not be used with diversity measures.

3.1.5 Run formats

System outputs can be provided in a variety of formats.

namedtuple iterable: Any iterable of named tuples. You can use `ir_measures.ScoredDoc`, or any other NamedTuple with the fields `query_id`, `doc_id`, and `score`:

```
run = [
    ir_measures.ScoredDoc("Q0", "D0", 1.2),
    ir_measures.ScoredDoc("Q0", "D1", 1.0),
    ir_measures.ScoredDoc("Q1", "D0", 2.4),
    ir_measures.ScoredDoc("Q1", "D3", 3.6),
]
```

Note that if the results are an iterator (such as the result of a generator), `ir_measures` will consume the entire sequence.

Pandas dataframe: A pandas dataframe with the columns `query_id`, `doc_id`, and `score`:

```
import pandas as pd
run = pd.DataFrame([
    {'query_id': "Q0", 'doc_id': "D0", 'score': 1.2},
    {'query_id': "Q0", 'doc_id': "D1", 'score': 1.0},
    {'query_id': "Q1", 'doc_id': "D0", 'score': 2.4},
    {'query_id': "Q1", 'doc_id': "D3", 'score': 3.6},
])
```

If your dataframe has columns named something else, you can always map them with the `rename` function. For instance, if your dataframe has the columns `qid`, `docno`, and `output`, you can easily make a qrels dataframe that is compatible with ir-measures like so:

```
run = df.rename(columns={'qid': 'query_id', 'docno': 'doc_id', 'output': 'score'})
```

TREC-formatted run file: You can read a TREC-formatted run file:

```
# a file path:
run = ir_measures.read_trec_run('path/to/run')
# raw run file contents:
run = ir_measures.read_trec_run('''
Q0 0 D0 0 1.2 runid
Q0 0 D1 1 1.0 runid
Q1 0 D3 0 3.6 runid
Q1 0 D0 1 2.4 runid
''')
# TREC run format: "query_id ignored doc_id rank score runid". This parser ignores
↪"ignored", "rank", and "runid".
```

Note that `read_trec_run` returns a generator. If you need to use the qrels multiple times, wrap it in the `list` constructor to read the all qrels into memory.

dict-of-dict: Run structured in a hierarchy. At the first level, query IDs map to another dictionary. At the second level, document IDs map to (float) ranking scores:

```
run = {
    'Q0': {
        "D0": 1.2,
        "D1": 1.0,
    },
    "Q1": {
        "D0": 2.4,
        "D3": 3.6
    }
}
```

3.1.6 Measure Objects

Measure objects speficy the measure you want to calculate, along with any parameters they may have. There are several ways to create them. The easiest is to specify them directly in code:

```
>>> from ir_measures import * # imports all measure names
>>> AP
AP
>>> AP(rel=2)
AP(rel=2)
>>> nDCG@20
nDCG@20
>>> P(rel=2)@10
P(rel=2)@10
```

Notice that measures can include parameters. For instance, `AP(rel=2)` is the average precision measure with a minimum relevance level of 2 (i.e., documents need to be scored at least 2 to count as relevant.) Or `nDCG@20`, which specifies a ranking cutoff threshold of 20. See the measure's documentation for full details of available parameters.

If you need to get a measure object from a string (e.g., if specified by the user as a command line argument), use the `ir_measures.parse_measure` function:

```
>>> ir_measures.parse_measure('AP')
AP
```

(continues on next page)

(continued from previous page)

```
>>> ir_measures.parse_measure('AP(rel=2)')
AP(rel=2)
>>> ir_measures.parse_measure('nDCG@20')
nDCG@20
>>> ir_measures.parse_measure('P(rel=2)@10')
P(rel=2)@10
```

If you are familiar with the measure and family names from `trec_eval`, you can map them to measure objects using `ir_measures.parse_trec_measure()`:

```
>>> ir_measures.parse_trec_measure('map')
[AP]
>>> ir_measures.parse_trec_measure('P') # expands to multiple levels
[P@5, P@10, P@15, P@20, P@30, P@100, P@200, P@500, P@1000]
>>> ir_measures.parse_trec_measure('P_3,8') # or 'P.3,8'
[P@3, P@8]
>>> ir_measures.parse_trec_measure('ndcg')
[nDCG]
>>> ir_measures.parse_trec_measure('ndcg_cut_10')
[nDCG@10]
>>> ir_measures.parse_trec_measure('official')
[P@5, P@10, P@15, P@20, P@30, P@100, P@200, P@500, P@1000, Rprec, Bpref, IPrec@0.0,␣
↪IPrec@0.1, IPrec@0.2, IPrec@0.3, IPrec@0.4, IPrec@0.5, IPrec@0.6, IPrec@0.7, IPrec@0.8,
↪ IPrec@0.9, IPrec@1.0, AP, NumQ, NumRel, NumRet(rel=1), NumRet, RR]
```

Note that a single `trec_eval` measure name can map to multiple measures, so measures are returned as a list.

Measures are be passed into methods like `ir_measures.calc_aggregate`, `ir_measures.iter_calc`, and `ir_measures.evaluator`. You can also calculate values from the measure object itself:

```
>>> AP.calc_aggregate(qrels, run)
0.2842120439595336
>>> (nDCG@10).calc_aggregate(qrels, run) # parens needed when @cutoff is used
0.6250748053944134
>>> for metric in (P(rel=2)@10).iter_calc(qrels, run):
...     print(metric)
Metric(query_id='1', measure=P(rel=2)@10, value=0.5)
Metric(query_id='2', measure=P(rel=2)@10, value=0.8)
...
Metric(query_id='35', measure=P(rel=2)@10, value=0.9)
```

3.1.7 Scoring multiple runs

Sometimes you need to evaluate several different systems using the same benchmark. To avoid redundant work for every run (such as processing qrels), you can create an `evaluator(measures, qrels)` object that can be re-used on multiple runs. An evaluator object has `calc_aggregate(run)` and `calc_iter(run)` methods.

```
>>> evaluator = ir_measures.evaluator([nDCG@10, P@5, P(rel=2)@5, Judged@10], qrels)
>>> evaluator.calc_aggregate(run1)
{nDCG@10: 0.6250, P@5: 0.7485, P(rel=2)@5: 0.6000, Judged@10: 0.9485}
>>> evaluator.calc_aggregate(run2)
{nDCG@10: 0.6285, P@5: 0.7771, P(rel=2)@5: 0.6285, Judged@10: 0.9400}
```

(continues on next page)

(continued from previous page)

```
>>> evaluator.calc_aggregate(run3)
{nDCG@10: 0.5286, P@5: 0.6228, P(rel=2)@5: 0.4628, Judged@10: 0.8485}
```

3.1.8 Empty Set Behaviour

ir-measures normalizes the behavior across tools by always returning results based on all queries that appear in the provided qrels, regardless of what appears in the run. This corresponds with the `-c` flag in `trec_eval`. Queries that appear in the run but not the qrels are ignored, and queries that appear in the qrels but not the run are given a score of 0.

This behaviour is based on the following reasoning:

1. Queries that do not appear in the qrels were not judged, and therefore cannot be properly scored if returned in the run.
2. Queries that do not appear in the run may have returned no results, and therefore be scored as such.

We believe that these are the proper settings, so there is currently no way to change this behaviour directly in the software. If you wish to only score some of the queries provided in the qrels, you may of course filter down the qrels provided to ir-measures to only those queries.

3.2 Measures

3.2.1 `alpha_DCG`

A version of DCG that accounts for multiple possible query intents.

```
@inproceedings{Clarke2008NoveltyAD,
```

title={Novelty and diversity in information retrieval evaluation}, author={Charles L. A. Clarke and Maheedhar Kolla and Gordon V. Cormack and Olga Vechtomova and Azin Ashkan and Stefan B{"u}ttcher and Ian MacKinnon}, booktitle={SIGIR}, year={2008}

}

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)
- `alpha` (float) - Redundancy intolerance
- `judged_only` (bool) - calculate measure using only judged documents (i.e., discard unjudged documents)

Provided by:

- `pyndeval`: `alpha_DCG(alpha=ANY, rel=ANY, judged_only=ANY)@ANY`

3.2.2 alpha_nDCG

A version of nDCG that accounts for multiple possible query intents.

```
@inproceedings{Clarke2008NoveltyAD,
```

title={Novelty and diversity in information retrieval evaluation}, author={Charles L. A. Clarke and Maheedhar Kolla and Gordon V. Cormack and Olga Vechtomova and Azin Ashkan and Stefan B{"u}ttcher and Ian MacKinnon}, booktitle={SIGIR}, year={2008}

}

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)
- `alpha` (float) - Redundancy intolerance
- `judged_only` (bool) - calculate measure using only judged documents (i.e., discard unjudged documents)

Provided by:

- `pyndeval`: `alpha_nDCG(alpha=ANY, rel=ANY, judged_only=ANY)@ANY`

3.2.3 Accuracy

The accuracy metric corresponds to the probability that a relevant document is ranked before a non relevant one. This metric is intended to be used as a diagnostic metric (checking discrepancies between train, validation and test with pairwise costs). As such, runs with no relevant document are discarded.

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `accuracy`: `Accuracy(cutoff=ANY, rel=ANY)`

3.2.4 AP

The [Mean] Average Precision ([M]AP). The average precision of a single query is the mean of the precision scores at each relevant item returned in a search results list.

AP is typically used for adhoc ranking tasks where getting as many relevant items as possible is. It is commonly referred to as MAP, by taking the mean of AP over the query set.

```
@article{Harman:1992:ESIR,
```

author = {Donna Harman}, title = {Evaluation Issues in Information Retrieval}, journal = {Information Processing and Management}, volume = {28}, number = {4}, pages = {439 - -440}, year = {1992},

}

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- cwl_eval: AP(rel=ANY)@NOT_PROVIDED
- pytrec_eval: AP(rel=ANY)@ANY
- trectools: AP(rel=1)@ANY

3.2.5 AP_IA

Intent-aware (Mean) Average Precision

Parameters:

- rel (int) - minimum relevance score to be considered relevant (inclusive)
- judged_only (bool) - calculate measure using only judged documents (i.e., discard unjudged documents)

Provided by:

- pyndeval: AP_IA(rel=ANY, judged_only=ANY)

3.2.6 BPM

The Bejeweled Player Model (BPM).

```
@inproceedings{Zhang:2017:EWS:3077136.3080841,
  author = {Zhang, Fan and Liu, Yiqun and Li, Xin and Zhang, Min and Xu, Yinghui and Ma,␣
↪Shaoping},
  title = {Evaluating Web Search with a Bejeweled Player Model},
  booktitle = {SIGIR},
  year = {2017},
  url = {http://doi.acm.org/10.1145/3077136.3080841}
}
```

Parameters:

- cutoff (int) - ranking cutoff threshold
- T (float) - total desired gain (normalized)
- min_rel (int) - minimum relevance score
- max_rel (int) - maximum relevance score

Provided by:

- cwl_eval: BPM(T=ANY, min_rel=ANY, max_rel=REQUIRED)@ANY

3.2.7 Bpref

Binary Preference (Bpref). This measure examines the relative ranks of judged relevant and non-relevant documents. Non-judged documents are not considered.

```
@inproceedings{Buckley2004RetrievalEW,
```

title={Retrieval evaluation with incomplete information}, author={Chris Buckley and Ellen M. Voorhees}, booktitle={SIGIR}, year={2004}

}

Parameters:

- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `pytrec_eval`: `Bpref(rel=ANY)`
- `trectools`: `Bpref(rel=1)`

3.2.8 Compat

Compatibility measure desribed in:

```
@article{10.1145/3451161,
```

author = {Clarke, Charles L. A. and Vtyurina, Alexandra and Smucker, Mark D.}, title = {Assessing Top-k Preferences}, journal = {ACM Transactions on Information Systems}, volume = {39}, number = {3}, articleno = {33}, numpages = {21}, year = {2021}, url = {https://doi.org/10.1145/3451161},

}

Parameters:

- `p` (float) - persistence
- `normalize` (bool) - apply normalization for finite ideal rankings

Provided by:

- `compat`: `Compat(p=ANY, normalize=ANY)`

3.2.9 ERR

The Expected Reciprocal Rank (ERR) is a precision-focused measure. In essence, an extension of reciprocal rank that encapsulates both graded relevance and a more realistic cascade-based user model of how users brwose a ranking.

Parameters:

- `cutoff` (int) - ranking cutoff threshold

Provided by:

- `gdeval`: `ERR@REQUIRED`

3.2.10 `ERR_IA`

Intent-Aware Expected Reciprocal Rank with collection-independent normalisation.

```
@inproceedings{10.1145/1645953.1646033,
```

author = {Chapelle, Olivier and Metlzer, Donald and Zhang, Ya and Grinspan, Pierre}, title = {Expected Reciprocal Rank for Graded Relevance}, booktitle = {CIKM}, year = {2009}

}

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)
- `judged_only` (bool) - calculate measure using only judged documents (i.e., discard unjudged documents)

Provided by:

- `pyndeval`: `ERR_IA(rel=ANY, judged_only=ANY)@ANY`

3.2.11 `infAP`

Inferred AP. AP implementation that accounts for pooled-but-unjudged documents by assuming that they are relevant at the same proportion as other judged documents. Essentially, skips documents that were pooled-but-not-judged, and assumes unjudged are non-relevant.

Pooled-but-unjudged indicated by a score of -1, by convention. Note that not all qrels use this convention.

Parameters:

- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `pytrec_eval`: `infAP(rel=ANY)`

3.2.12 `INSQ`

INSQ, a variant of INST

```
@inproceedings{Moffat:2015:IAM:2838931.2838938,
  author = {Moffat, Alistair and Bailey, Peter and Scholer, Falk and Thomas, Paul},
  title = {INST: An Adaptive Metric for Information Retrieval Evaluation},
  booktitle = {Proceedings of the 20th Australasian Document Computing Symposium},
  year = {2015},
  url = {http://doi.acm.org/10.1145/2838931.2838938}
}
```

Parameters:

- `T` (float) - TODO
- `min_rel` (int) - minimum relevance score
- `max_rel` (int) - maximum relevance score

Provided by:

- `cwl_eval`: `INSQ(T=ANY, min_rel=ANY, max_rel=REQUIRED)`

3.2.13 INST

INST

```
@inproceedings{Moffat:2012:MMI:2407085.2407092,
  author = {Moffat, Alistair and Scholer, Falk and Thomas, Paul},
  title = {Models and Metrics: IR Evaluation As a User Process},
  booktitle = {Proceedings of the Seventeenth Australasian Document Computing Symposium},
  year = {2012},
  url = {http://doi.acm.org/10.1145/2407085.2407092}
}
```

Parameters:

- `T` (float) - TODO
- `min_rel` (int) - minimum relevance score
- `max_rel` (int) - maximum relevance score

Provided by:

- `cwl_eval`: `INST(T=ANY, min_rel=ANY, max_rel=REQUIRED)`

3.2.14 IPrec

Interpolated Precision at a given recall cutoff. Used for building precision-recall graphs. Unlike most measures, where @ indicates an absolute cutoff threshold, here @ sets the recall cutoff.

Parameters:

- `recall` (float) - recall threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `pytrec_eval`: `IPrec@ANY`

3.2.15 Judged

Percentage of results in the top k (cutoff) results that have relevance judgments. Equivalent to P@k with a rel lower than any judgment.

Parameters:

- `cutoff` (int) - ranking cutoff threshold

Provided by:

- `judged`: `Judged@ANY`

3.2.16 nDCG

The normalized Discounted Cumulative Gain (nDCG). Uses graded labels - systems that put the highest graded documents at the top of the ranking. It is normalized wrt. the Ideal NDCG, i.e. documents ranked in descending order of graded label.

```
@article{Jarvelin:2002:CGE:582415.582418,
```

author = {J"{a}rvelin, Kalervo and Kek"{a}l"{a}inen, Jaana}, title = {Cumulated Gain-based Evaluation of IR Techniques}, journal = {ACM Trans. Inf. Syst.}, volume = {20}, number = {4}, year = {2002}, pages = {422–446}, numpages = {25}, url = {http://doi.acm.org/10.1145/582415.582418},

}

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `dcg` (str) - DCG formulation

Provided by:

- `pytrec_eval`: `nDCG(dcg='log2')@ANY`
- `gdeval`: `nDCG(dcg='exp-log2')@REQUIRED`
- `trectools`: `nDCG(dcg=ANY)@ANY`

3.2.17 NERR10

Version of the Not (but Nearly) Expected Reciprocal Rank (NERR) measure, version from Equation (10) of the the following paper.

```
@inproceedings{Azzopardi:2021:ECE:3471158.3472239,
  author = {Azzopardi, Leif and Mackenzie, Joel and Moffat, Alistair},
  title = {{ERR} is not {C/W/L}: Exploring the Relationship Between Expected Reciprocal␣
↪Rank and Other Metrics},
  booktitle = {ICTIR},
  year = {2021},
  url = {https://doi.org/10.1145/3471158.3472239}
}
```

Parameters:

- `p` (float) - persistence
- `min_rel` (int) - minimum relevance score
- `max_rel` (int) - maximum relevance score

Provided by:

- `cwl_eval`: `NERR10(p=ANY, min_rel=ANY, max_rel=REQUIRED)`

3.2.18 NERR11

Version of the Not (but Nearly) Expected Reciprocal Rank (NERR) measure, version from Equation (12) of the the following paper.

```
@inproceedings{Azzopardi:2021:ECE:3471158.3472239,
  author = {Azzopardi, Leif and Mackenzie, Joel and Moffat, Alistair},
  title = {{ERR} is not {C/W/L}: Exploring the Relationship Between Expected Reciprocal␣
↪Rank and Other Metrics},
  booktitle = {ICTIR},
  year = {2021},
  url = {https://doi.org/10.1145/3471158.3472239}
}
```

Parameters:

- `T` (float) - total desired gain (normalized)
- `min_rel` (int) - minimum relevance score
- `max_rel` (int) - maximum relevance score

Provided by:

- cwl_eval: `NERR11(T=ANY, min_rel=ANY, max_rel=REQUIRED)`

3.2.19 NERR8

Version of the Not (but Nearly) Expected Reciprocal Rank (NERR) measure, version from Equation (8) of the the following paper.

```
@inproceedings{Azzopardi:2021:ECE:3471158.3472239,
  author = {Azzopardi, Leif and Mackenzie, Joel and Moffat, Alistair},
  title = {{ERR} is not {C/W/L}: Exploring the Relationship Between Expected Reciprocal␣
↪Rank and Other Metrics},
  booktitle = {ICTIR},
  year = {2021},
  url = {https://doi.org/10.1145/3471158.3472239}
}
```

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `min_rel` (int) - minimum relevance score
- `max_rel` (int) - maximum relevance score

Provided by:

- cwl_eval: `NERR8(min_rel=ANY, max_rel=REQUIRED)@REQUIRED`

3.2.20 NERR9

Version of the Not (but Nearly) Expected Reciprocal Rank (NERR) measure, version from Equation (9) of the the following paper.

```
@inproceedings{Azzopardi:2021:ECE:3471158.3472239,
  author = {Azzopardi, Leif and Mackenzie, Joel and Moffat, Alistair},
  title = {{ERR} is not {C/W/L}: Exploring the Relationship Between Expected Reciprocal␣
↪Rank and Other Metrics},
  booktitle = {ICTIR},
  year = {2021},
  url = {https://doi.org/10.1145/3471158.3472239}
}
```

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `min_rel` (int) - minimum relevance score
- `max_rel` (int) - maximum relevance score

Provided by:

- `cwl_eval`: `NERR9(min_rel=ANY, max_rel=REQUIRED)@REQUIRED`

3.2.21 nERR_IA

Intent-Aware Expected Reciprocal Rank with collection-dependent normalisation.

```
@inproceedings{10.1145/1645953.1646033,
```

author = {Chapelle, Olivier and Metlzer, Donald and Zhang, Ya and Grinspan, Pierre}, title = {Expected Reciprocal Rank for Graded Relevance}, booktitle = {CIKM}, year = {2009}

}

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)
- `judged_only` (bool) - calculate measure using only judged documents (i.e., discard unjudged documents)

Provided by:

- `pyndeval`: `nERR_IA(rel=ANY, judged_only=ANY)@ANY`

3.2.22 nNRBP

Novelty- and Rank-Biased Precision with collection-dependent normalisation.

```
@InProceedings{10.1007/978-3-642-04417-5_17,
```

author="Clarke, Charles L. A. and Kolla, Maheedhar and Vechtomova, Olga", title="An Effectiveness Measure for Ambiguous and Underspecified Queries ", booktitle="ICTIR", year="2009"

}

Parameters:

- `rel` (int) - minimum relevance score to be considered relevant (inclusive)
- `alpha` (float) - Redundancy intolerance
- `beta` (float) - Patience

Provided by:

- `pyndeval`: `nNRBP(alpha=ANY, beta=ANY, rel=ANY)`

3.2.23 NRBP

Novelty- and Rank-Biased Precision with collection-independent normalisation.

```
@InProceedings{10.1007/978-3-642-04417-5_17,
```

author="Clarke, Charles L. A. and Kolla, Maheedhar and Vechtomova, Olga", title="An Effectiveness Measure for Ambiguous and Underspecified Queries ", booktitle="ICTIR", year="2009"

}

Parameters:

- `rel` (int) - minimum relevance score to be considered relevant (inclusive)
- `alpha` (float) - Redundancy intolerance
- `beta` (float) - Patience

Provided by:

- `pyndeval`: `NRBP(alpha=ANY, beta=ANY, rel=ANY)`

3.2.24 NumQ

The total number of queries.

Provided by:

- `pytrec_eval`: `NumQ`

3.2.25 NumRel

The number of relevant documents the query has (independent of what the system retrieved).

Parameters:

- `rel` (int) - minimum relevance score to be counted (inclusive)

Provided by:

- `pytrec_eval`: `NumRel(rel=1)`

3.2.26 NumRet

The number of results returned. When rel is provided, counts the number of documents returned with at least that relevance score (inclusive).

Parameters:

- `rel` (int) - minimum relevance score to be counted (inclusive), or all documents returned if NOT_PROVIDED

Provided by:

- `pytrec_eval`: `NumRet(rel=ANY)`

3.2.27 P

Basic measure for that computes the percentage of documents in the top cutoff results that are labeled as relevant. cutoff is a required parameter, and can be provided as P@cutoff.

```
@misc{rijsbergen:1979:ir,
```

title={Information Retrieval.}, author={Van Rijsbergen, Cornelis J}, year={1979}, publisher={USA: Butterworth-Heinemann}

}

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `cwl_eval`: `P(rel=ANY)@ANY`
- `pytrec_eval`: `P(rel=ANY)@ANY`
- `trectools`: `P(rel=1)@ANY`

3.2.28 `P_IA`

Intent-aware Precision@k.

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)
- `judged_only` (bool) - calculate measure using only judged documents (i.e., discard unjudged documents)

Provided by:

- `pyndeval`: `P_IA(rel=ANY, judged_only=ANY)@ANY`

3.2.29 `R`

Recall@k (R@k). The fraction of relevant documents for a query that have been retrieved by rank k.

NOTE: Some tasks define Recall@k as whether any relevant documents are found in the top k results. This software follows the TREC convention and refers to that measure as Success@k.

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `pytrec_eval`: `R@ANY`

3.2.30 `RBP`

The Rank-Biased Precision (RBP).

```
@article{Moffat:2008:RPM:1416950.1416952,
  author = {Moffat, Alistair and Zobel, Justin},
  title = {Rank-biased Precision for Measurement of Retrieval Effectiveness},
  journal = {ACM Trans. Inf. Syst.},
  year = {2008},
  url = {http://doi.acm.org/10.1145/1416950.1416952}
}
```

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- p (float) - persistence
- `rel` (int) - minimum relevance score to be considered relevant (inclusive), or NOT_PROVIDED to use graded relevance

Provided by:

- `cwl_eval`: `RBP(rel=REQUIRED, p=ANY)@NOT_PROVIDED`
- `trectools`: `RBP(p=ANY, rel=ANY)@ANY`

3.2.31 Rprec

The precision of at R, where R is the number of relevant documents for a given query. Has the cute property that it is also the recall at R.

```
@misc{Buckley2005RetrievalSE,
```

title={Retrieval System Evaluation}, author={Chris Buckley and Ellen M. Voorhees}, annote={Chapter 3 in TREC: Experiment and Evaluation in Information Retrieval}, howpublished={MIT Press}, year={2005}

}

Parameters:

- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `pytrec_eval`: `Rprec(rel=ANY)`
- `trectools`: `Rprec(rel=1)`

3.2.32 RR

The [Mean] Reciprocal Rank ([M]RR) is a precision-focused measure that scores based on the reciprocal of the rank of the highest-scoring relevance document. An optional cutoff can be provided to limit the depth explored. rel (default 1) controls which relevance level is considered relevant.

```
@article{kantor2000trec,
```

title={The TREC-5 Confusion Track}, author={Kantor, Paul and Voorhees, Ellen}, journal={Information Retrieval}, volume={2}, number={2-3}, pages={165–176}, year={2000}

}

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `cwl_eval`: `RR(rel=ANY)@NOT_PROVIDED`
- `pytrec_eval`: `RR(rel=ANY)@NOT_PROVIDED`
- `trectools`: `RR(rel=1)@NOT_PROVIDED`
- `msmarco`: `RR(rel=ANY)@ANY`

3.2.33 `SDCG`

The Scaled Discounted Cumulative Gain (SDCG), a variant of nDCG that assumes more fully-relevant documents exist but are not labeled.

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `dcg` (str) - DCG formulation
- `min_rel` (int) - minimum relevance score
- `max_rel` (int) - maximum relevance score

Provided by:

- `cwl_eval`: `SDCG(dcg='log2', min_rel=ANY, max_rel=REQUIRED)@REQUIRED`

3.2.34 `SetAP`

The unranked Set AP (SetAP); i.e., SetP * SetR

Parameters:

- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `pytrec_eval`: `SetAP(rel=ANY)`

3.2.35 `SetF`

The Set F measure (SetF); i.e., the harmonic mean of SetP and SetR

Parameters:

- `rel` (int) - minimum relevance score to be considered relevant (inclusive)
- `beta` (float) - relative importance of R to P in the harmonic mean

Provided by:

- `pytrec_eval`: `SetF(rel=ANY, beta=ANY)`

3.2.36 `SetP`

The Set Precision (SetP); i.e., the number of relevant docs divided by the total number retrieved

Parameters:

- `rel` (int) - minimum relevance score to be considered relevant (inclusive)
- `relative` (bool) - calculate the measure using the maximum possible SetP for the provided result size

Provided by:

- `pytrec_eval`: `SetP(rel=ANY, relative=ANY)`

3.2.37 `SetR`

The Set Recall (SetR); i.e., the number of relevant docs divided by the total number of relevant documents

Parameters:

- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `pytrec_eval`: `SetR(rel=ANY)`

3.2.38 `StRecall`

Subtopic recall (the number of subtopics covered by the top k docs)

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `pyndeval`: `StRecall(rel=ANY)@ANY`

3.2.39 `Success`

1 if a document with at least rel relevance is found in the first cutoff documents, else 0.

NOTE: Some refer to this measure as Recall@k. This software follows the TREC convention, where Recall@k is defined as the proportion of known relevant documents retrieved in the top k results.

Parameters:

- `cutoff` (int) - ranking cutoff threshold
- `rel` (int) - minimum relevance score to be considered relevant (inclusive)

Provided by:

- `pytrec_eval`: `Success(rel=ANY)@ANY`

3.2.40 Aliases

These provide shortcuts to "canonical" measures, and are typically used when multiple names or casings for the same measure exist. You can use them just like any other measure and the identifiers are equal (e.g., `AP == MAP`) but the names will appear in the canonical form when printed.

- `BPref` → `Bpref`
- `MAP` → `AP`
- `MAP_IA` → `AP_IA`
- `MRR` → `RR`
- `NDCG` → `nDCG`
- `NumRelRet` → `NumRet(rel=1)`
- `Precision` → `P`

- `Recall` → `R`
- `RPrec` → `Rprec`
- `SetRelP` → `SetP(relative=True)`
- `_DCG` → `alpha_DCG`
- `_nDCG` → `alpha_nDCG`

3.3 Providers

3.3.1 `accuracy`

Provider for the accuracy metric

3.3.2 `compat`

Version of the compatibility measure desribed in

@article{10.1145/3451161, author = {Clarke, Charles L. A. and Vtyurina, Alexandra and Smucker, Mark D.}, title = {Assessing Top-k Preferences}, journal = {ACM Transactions on Information Systems}, volume = {39}, number = {3}, articleno = {33}, numpages = {21}, year = {2021}, url = {https://doi.org/10.1145/3451161},

}

Supported Measures:

- `Compat(p=ANY, normalize=ANY)`

3.3.3 `cwl_eval`

cwl_eval, providing C/W/L ("cool") framework measures.

https://github.com/ireval/cwl

```
@inproceedings{azzopardi2019cwl,
  author = {Azzopardi, Leif and Thomas, Paul and Moffat, Alistair},
  title = {cwl\_eval: An Evaluation Tool for Information Retrieval},
  booktitle = {SIGIR},
  year = {2019}
}
```

Supported Measures:

- `P(rel=ANY)@ANY`
- `RR(rel=ANY)@NOT_PROVIDED`
- `AP(rel=ANY)@NOT_PROVIDED`
- `RBP(rel=REQUIRED, p=ANY)@NOT_PROVIDED`
- `BPM(T=ANY, min_rel=ANY, max_rel=REQUIRED)@ANY`
- `SDCG(dcg='log2', min_rel=ANY, max_rel=REQUIRED)@REQUIRED`

- `NERR8(min_rel=ANY, max_rel=REQUIRED)@REQUIRED`
- `NERR9(min_rel=ANY, max_rel=REQUIRED)@REQUIRED`
- `NERR10(p=ANY, min_rel=ANY, max_rel=REQUIRED)`
- `NERR11(T=ANY, min_rel=ANY, max_rel=REQUIRED)`
- `INST(T=ANY, min_rel=ANY, max_rel=REQUIRED)`
- `INSQ(T=ANY, min_rel=ANY, max_rel=REQUIRED)`

3.3.4 `gdeval`

gdeval

Supported Measures:

- `nDCG(dcg='exp-log2')@REQUIRED`
- `ERR@REQUIRED`

3.3.5 `judged`

python implementation of judgment rate

Adapted from OpenNIR's implementation: https://github.com/Georgetown-IR-Lab/OpenNIR/blob/master/onir/metrics/judged.py

Supported Measures:

- `Judged@ANY`

3.3.6 `msmarco`

MS MARCO's implementation of RR

Supported Measures:

- `RR(rel=ANY)@ANY`

3.3.7 `pyndeval`

pyndeval

Supported Measures:

- `ERR_IA(rel=ANY, judged_only=ANY)@ANY`
- `nERR_IA(rel=ANY, judged_only=ANY)@ANY`
- `alpha_DCG(alpha=ANY, rel=ANY, judged_only=ANY)@ANY`
- `alpha_nDCG(alpha=ANY, rel=ANY, judged_only=ANY)@ANY`
- `NRBP(alpha=ANY, beta=ANY, rel=ANY)`
- `nNRBP(alpha=ANY, beta=ANY, rel=ANY)`
- `AP_IA(rel=ANY, judged_only=ANY)`

- `P_IA(rel=ANY, judged_only=ANY)@ANY`
- `StRecall(rel=ANY)@ANY`

3.3.8 `pytrec_eval`

pytrec_eval

https://github.com/cvangysel/pytrec_eval

```
@inproceedings{VanGysel2018pytreceval,
 title={Pytrec\_eval: An Extremely Fast Python Interface to trec\_eval},
 author={Van Gysel, Christophe and de Rijke, Maarten},
 publisher={ACM},
 booktitle={SIGIR},
 year={2018},
}
```

Supported Measures:

- `P(rel=ANY)@ANY`
- `RR(rel=ANY)@NOT_PROVIDED`
- `Rprec(rel=ANY)`
- `AP(rel=ANY)@ANY`
- `nDCG(dcg='log2')@ANY`
- `R@ANY`
- `Bpref(rel=ANY)`
- `NumRet(rel=ANY)`
- `NumQ`
- `NumRel(rel=1)`
- `SetAP(rel=ANY)`
- `SetF(rel=ANY, beta=ANY)`
- `SetP(rel=ANY, relative=ANY)`
- `SetR(rel=ANY)`
- `Success(rel=ANY)@ANY`
- `IPrec@ANY`
- `infAP(rel=ANY)`

3.3.9 trectools

trectools

https://github.com/joaopalotti/trectools

```
@inproceedings{palotti2019,
   author = {Palotti, Joao and Scells, Harrisen and Zuccon, Guido},
   title = {TrecTools: an open-source Python library for Information Retrieval␣
↪practitioners involved in TREC-like campaigns},
   series = {SIGIR'19},
   year = {2019},
   location = {Paris, France},
   publisher = {ACM}
}
```

Supported Measures:

- `P(rel=1)@ANY`
- `RR(rel=1)@NOT_PROVIDED`
- `Rprec(rel=1)`
- `AP(rel=1)@ANY`
- `nDCG(dcg=ANY)@ANY`
- `Bpref(rel=1)`
- `RBP(p=ANY, rel=ANY)@ANY`

3.4 API Reference

3.4.1 Metric Calculation

3.4.2 Parsing

3.4.3 Data Classes