Streamlining Evaluation with `ir-measures`

Sean MacAvaney, Craig Macdonald, and Iadh Ounis

University of Glasgow, United Kingdom
{first.last}@glasgow.ac.uk

Abstract. We present `ir-measures`, a new tool that makes it convenient to calculate a diverse set of evaluation measures used in information retrieval. Rather than implementing its own measure calculations, `ir-measures` provides a common interface to a handful of evaluation tools. The necessary tools are automatically invoked (potentially multiple times) to calculate all the desired metrics, simplifying the evaluation process for the user. The tool also makes it easier for researchers to use recently-proposed measures (such as those from the C/W/L framework) alongside traditional measures, potentially encouraging their adoption.

1 Introduction

The field of Information Retrieval (IR) is fortunate to have a vibrant and diverse ecosystem of tools and resources. This is particularly true for evaluation tools; there exists a variety of fully-fledged evaluation suites capable of calculating a wide array of measures (e.g., `trec_eval` [24], `cwl_eval` [2], `trectools` [25], and RankEval [17]) as well as single-purpose scripts that are usually designed for the evaluation of specific tasks or datasets.[1] However, none of these tools themselves provide comprehensive coverage of evaluation metrics, so researchers often need to run multiple tools to get all the desired results. Even when a single tool can provide the desired measures, it can sometimes require multiple invocations with different settings to get all desired results (e.g., the TREC Deep Learning passage ranking task [10] requires multiple invocations of `trec_eval` with different relevance cutoff thresholds).

In this demonstration, we present a new evaluation tool: `ir-measures`.[2] Unlike prior tools, which provide their own measure implementations, `ir-measures` operates as an abstraction over multiple evaluation tools. Researchers are able to simply describe *what* evaluation measures they want in natural syntax (e.g., `nDCG@20` for nDCG [13] with a rank cutoff of 20, or `AP(rel=2)` for Average Precision [12] with a binary relevance cutoff of 2), without necessarily needing to concern themselves with which specific tools provide the functionality they are looking for or what settings would give the desired results. By providing both a Python and command line interface and accepting multiple input and output formats, the tool is convenient to use in a variety of environments (e.g., both as

[1] For instance, the MSMARCO MRR evaluation script: https://git.io/JKG1S
[2] Docs: https://ir-measur.es/, Source: https://github.com/terrierteam/ir_measures

a component of larger IR toolkits, or for simply doing *ad hoc* evaluation). By interfacing with existing evaluation toolkits, `ir-measures` is more sustainable than efforts that re-implement evaluation measures, especially given the ongoing debate over the suitability of some measures (e.g., [11, 27]) and the proliferation of new measures (e.g., [1, 9]). An interactive demonstration of the software is available at: https://git.io/JMt6G.

2 Background

Recently, there have been several efforts to resolve incompatibilities for other IR tools. `trectools` [25] provides Python implementations of numerous IR-related functions including pooling, fusion, and evaluation techniques (including a handful of evaluation measures). PyTerrier [20] provides a Python interface to a myriad of retrieval, rewriting, learning-to-rank, and neural re-ranking techniques as well as an infrastructure for conducting IR experiments. CIFF [16] defines a common index interchange format, for compatibility between search engines. `ir_datasets` [19] provides a common interface to access and work with document corpora and test collections. `ir-measures` is complementary to efforts like these. In Section 4, we show that `ir-measures` can easily be integrated into other tools, bolstering their evaluation capacity.

3 `ir-measures`

`ir-measures` provides access to over 30 evaluation measures. Table 1 provides a summary of the supported measures, which span a variety of categories and applications (e.g., intent-aware measures, set measures, etc.) Measures are referenced by name and a measure-dependent set of parameters (e.g., `AP(rel=2)` specifies a minimum relevance level and `nDCG@10` specifies a rank cutoff). We refer the reader to the measure documentation[3] for further details.

Table 1. Measures provided by `ir-measures`, along with their providers.

Measure	Provided by...	Measure	Provided by...
`alpha_nDCG` [7]	`ndeval`	`NERR` [1]	`cwl_eval`
`(M)AP(@k)` [12]	`cwl_eval`, `trec_eval`, `trectools`	`NRBP` [8]	`ndeval`
`(M)AP_IA`	`ndeval`	`NumQ`, `NumRel`, `NumRet`	`trec_eval`
`BPM` [32]	`cwl_eval`	`P(recision)@k` [29]	`trec_eval`, `cwl_eval`, `trectools`
`Bpref` [4]	`trec_eval`, `trectools`	`P_IA@k`	`ndeval`
`Compat` [9]	Compatibility script	`R(ecall)@k`	`trec_eval`
`ERR@k` [6]	`gdeval`	`RBP` [8]	`cwl_eval`, `trectools`
`ERR_IA` [6]	`ndeval`	`Rprec` [5]	`trec_eval`, `trectools`
`infAP` [31]	`trec_eval`	`(M)RR` [15]	`trec_eval`, `cwl_eval`, `trectools`, MSMARCO
`INSQ` [21], `INST` [23]	`cwl_eval`	`SDCG@k`	`cwl_eval`
`IPrec@i`	`trec_eval`	`SetAP`, `SetF`, `SetP`, `SetR`	`trec_eval`
`Judged@k`	OpenNIR script	`STREC`	`ndeval`
`nDCG(@k)` [13]	`trec_eval`, `gdeval`, `trectools`	`Success@k`	`trec_eval`

[3] https://ir-measur.es/en/latest/measures.html

Providers. The calculation of measure values themselves are implemented by *providers*. Not all providers are able to calculate all measures (or all parameters of a measure). The current version of `ir-measures` includes eight providers:

`trec_eval` [24] is a well-known IR evaluation tool that is used for calculating a variety of measures for TREC tasks. We use Python bindings adapted from the `pytrec_eval` [28] package.

`cwl_eval` [2] provides an implementation of a variety of measures that adhere to the C/W/L framework [22], such as `BPM` and `RBP`.

`ndeval`[4] enables the calculation of measures that consider multiple possible query intents (i.e., diversity measures), such as `alpha_nDCG` and `ERR_IA`.

The `trectools` [25] tookit includes Python implementations of a variety of evaluation measures, including `AP`, `Bpref`, and others. `gdeval`[5] includes an implementation of `ERR@k` and an alternative formulation of `nDCG@k` that places additional weight on high relevance.

The **OpenNIR `Judged@k` script**[6] was adapted from the OpenNIR toolkit [18] to calculate `Judged@k`, a measure of the proportion of top-ranked documents that have relevance assessments.

The **MSMARCO `RR@k` script**[7] is an interface to the official evaluation script for the MSMARCO dataset [3], with minor adjustments to allow for the configuration of the measure parameters and handling of edge cases.

The **Compatibility script**[8] provides `Compat` [9], a recently-proposed measure that calculates the Rank Biased Overlap of a result set compared to the closest ideal ranking of qrels, which can consider preference judgments.

Interfaces. `ir-measures` can be installed using `pip install ir-measures`, which provides both a command line interface and a Python package. The command line interface is similar to that of `trec_eval`, accepting a TREC-formatted relevance judgments (qrels) file, a run file, and the desired measures:

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

Command line arguments allow the user to get results by query, use a particular provider, and control the output format. If the `ir-datasets` [19] package is installed, a dataset identifier can be used in place of the qrels path.

The Python API makes it simple to calculate measures from a larger toolkit. A variety of input formats are accepted, including TREC-formatted files, dictionaries, Pandas dataframes, and `ir-datasets` iterators. The Python API also can provide results by query and can reuse evaluation objects for improved efficiency over multiple runs. Here is a simple example that calculates four measures:

[4] https://git.io/JKG94, https://git.io/JKCTo [5] https://git.io/JKCT1
[6] https://git.io/JKG9O [7] https://git.io/JKG1S [8] https://git.io/JKCT5

```
> import ir_measures
> from ir_measures import * # import natural measure names
> 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(rel=2)@5, Judged@10], qrels, run)
{nDCG@10: 0.6251, P(rel=2)@5: 0.6000, Judged@10: 0.9486}
```

4 Adoption of `ir-measures`

`ir-measures` is already in use by several tools, demonstrating its utility. It recently replaced `pytrec_eval` in PyTerrier [20], allowing retrieval pipelines to easily be evaluated on a variety of measures. For instance, the following example show an experiment on the TREC COVID [30] dataset. `ir-measures` allows the evaluation measures to be expressed clearly and concisely, and automatically invokes the necessary tools to compute the desired metrics:

```
import pyterrier as pt
dataset = pt.get_dataset('trec-covid')
pt.Experiment(
  [pt.TerrierRetrieve.from_dataset(dataset, "terrier_stemmed")],
  dataset.get_topics("round5"),
  dataset.get_qrels("round5"),
  eval_metrics=[nDCG@10, P(rel=2)@5, Judged@10])
```

It is also used by OpenNIR [18], Experimaestro [26], and DiffIR [14]. The `ir-datasets` [19] package uses `ir-measures` notation to provide documentation of the official evaluation measures for test collections.

A core design decision of `ir-measures` is to limit the required dependencies to the Python Standard Library and the packages for the measure providers (which can be omitted, but will degrade functionality). This should encourage the adoption of the tool by reducing the chance of package incompatibilities.

5 Conclusion

We demonstrated the new `ir-measures` package, which simplifies the computation of a variety of evaluation measures for IR researchers. We believe that by leveraging a variety of established tools (rather than providing its own implementations), `ir-measures` can be a salable and appealing choice for evaluation. We expect that our tool will also encourage the adoption of new evaluation measures, since they can be easily computed alongside long-established measures.

Acknowledgements. We thank the contributors to the `ir-measures` repository. We acknowledge EPSRC grant EP/R018634/1: Closed-Loop Data Science for Complex, Computationally- & Data-Intensive Analytics.

References

1. Azzopardi, L., Mackenzie, J., Moffat, A.: ERR is not C/W/L: Exploring the relationship between expected reciprocal rank and other metrics. In: ICTIR (2021)
2. Azzopardi, L., Thomas, P., Moffat, A.: Cwl_eval: An evaluation tool for information retrieval. In: SIGIR (2019)
3. Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., Rosenberg, M., Song, X., Stoica, A., Tiwary, S., Wang, T.: MS MARCO: A human generated machine reading comprehension dataset. In: CoCo@NIPS (2016)
4. Buckley, C., Voorhees, E.M.: Retrieval evaluation with incomplete information. In: SIGIR (2004)
5. Buckley, C., Voorhees, E.M.: Retrieval system evaluation. MIT Press (2005)
6. Chapelle, O., Metlzer, D., Zhang, Y., Grinspan, P.: Expected reciprocal rank for graded relevance. In: CIKM (2009)
7. Clarke, C.L.A., Kolla, M., Cormack, G.V., Vechtomova, O., Ashkan, A., Büttcher, S., MacKinnon, I.: Novelty and diversity in information retrieval evaluation. In: SIGIR (2008)
8. Clarke, C.L.A., Kolla, M., Vechtomova, O.: An effectiveness measure for ambiguous and underspecified queries. In: ICTIR (2009)
9. Clarke, C.L.A., Vtyurina, A., Smucker, M.D.: Assessing top-k preferences. TOIS **39**(3) (2021)
10. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.: Overview of the TREC 2019 deep learning track. In: TREC (2019)
11. Fuhr, N.: Some common mistakes in ir evaluation, and how they can be avoided. SIGIR Forum **51**, 32–41 (2018)
12. Harman, D.: Evaluation issues in information retrieval. IPM **28**(4), 439–440 (1992)
13. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. TOIS **20**(4), 422–446 (2002)
14. Jose, K.M., Nguyen, T., MacAvaney, S., Dalton, J., Yates, A.: Diffir: Exploring differences in ranking models' behavior. In: SIGIR (2021)
15. Kantor, P., Voorhees, E.: The TREC-5 confusion track. Information Retrieval **2**(2-3), 165–176 (2000)
16. Lin, J., Mackenzie, J., Kamphuis, C., Macdonald, C., Mallia, A., Siedlaczek, M., Trotman, A., de Vries, A.P.: Supporting interoperability between open-source search engines with the common index file format. SIGIR (2020)
17. Lucchese, C., Muntean, C.I., Nardini, F.M., Perego, R., Trani, S.: Rankeval: An evaluation and analysis framework for learning-to-rank solutions. In: SIGIR (2017)
18. MacAvaney, S.: OpenNIR: A complete neural ad-hoc ranking pipeline. In: WSDM (2020)
19. MacAvaney, S., Yates, A., Feldman, S., Downey, D., Cohan, A., Goharian, N.: Simplified data wrangling with ir_datasets. In: SIGIR (2021)

20. Macdonald, C., Tonellotto, N.: Declarative experimentation ininformation retrieval using PyTerrier. In: Proceedings of ICTIR 2020 (2020)
21. Moffat, A., Bailey, P., Scholer, F., Thomas, P.: Inst: An adaptive metric for information retrieval evaluation. In: Australasian Document Computing Symposium (2015)
22. Moffat, A., Bailey, P., Scholer, F., Thomas, P.: Incorporating user expectations and behavior into the measurement of search effectiveness. TOIS **35**(3) (Jun 2017)
23. Moffat, A., Scholer, F., Thomas, P.: Models and metrics: IR evaluation as a user process. In: Australasian Document Computing Symposium (2012)
24. National Institute of Standards and Technology: trec_eval. https://github.com/usnistgov/trec_eval (1993–2021)
25. Palotti, J., Scells, H., Zuccon, G.: TrecTools: an open-source python library for information retrieval practitioners involved in TREC-like campaigns. In: SIGIR (2019)
26. Piwowarski, B.: Experimaestro and Datamaestro: Experiment and dataset managers (for IR). In: SIGIR (2020)
27. Sakai, T.: On Fuhr's guideline for IR evaluation. SIGIR Forum **54**, 1 – 8 (2020)
28. Van Gysel, C., de Rijke, M.: Pytrec_eval: An extremely fast python interface to trec_eval. In: SIGIR (2018)
29. Van Rijsbergen, C.J.: Information retrieval. (1979)
30. Voorhees, E., Alam, T., Bedrick, S., Demner-Fushman, D., Hersh, W., Lo, K., Roberts, K., Soboroff, I., Wang, L.L.: Trec-covid: Constructing a pandemic information retrieval test collection. ArXiv **abs/2005.04474** (2020)
31. Yilmaz, E., Aslam, J.A.: Estimating average precision with incomplete and imperfect judgments. In: CIKM (2006)
32. Zhang, F., Liu, Y., Li, X., Zhang, M., Xu, Y., Ma, S.: Evaluating web search with a bejeweled player model. In: SIGIR (2017)