cwl_eval: An Evaluation Tool for Information Retrieval

Leif Azzopardi
University of Strathclyde
Glasgow, UK
leifos@acm.org

Paul Thomas
Microsoft
Canberra, Australia
pathom@microsoft.com

Alistair Moffat
The University of Melbourne
Melbourne, Australia
ammoffat@unimelb.edu.au

ABSTRACT

We present a tool ("cwl_eval") which unifies many metrics typically used to evaluate information retrieval systems using test collections. In the **C**/**W**/**L** framework metrics are specified via a single function which can be used to derive a number of related measurements: *Expected Utility* per item, *Expected Total Utility*, *Expected Cost* per item, *Expected Total Cost*, and *Expected Depth*. The **C**/**W**/**L** framework brings together several independent approaches for measuring the quality of a ranked list, and provides a coherent user model-based framework for developing measures based on utility (gain) and cost. Here we outline the **C**/**W**/**L** measurement framework; describe the cwl_eval architecture; and provide examples of how to use it. We provide implementations of a number of recent metrics, including Time Biased Gain, U-Measure, Bejewelled Measure, and the Information Foraging Based Measure, as well as previous metrics such as Precision, Average Precision, Discounted Cumulative Gain, Rank-Biased Precision, and INST. By providing state-of-the-art and traditional metrics within the same framework, we promote a standardised approach to evaluating search effectiveness.

ACM Reference Format:
Leif Azzopardi, Paul Thomas, and Alistair Moffat. 2019. cwl_eval: An Evaluation Tool for Information Retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331398

1 INTRODUCTION

Effectiveness evaluation has played a central role in the development of information retrieval systems [11]. Over the years many metrics have been proposed, with the more recent ones employing multi-valued and/or discounted relevance values (Discounted Cumulative Gain (DCG) [3] and Rank Biased Precision (RBP) [5]); cost (or time) associated with viewing result items (Time Biased Gain (TBG) [12]); and the way in which users adapt their interactions according to their goals and constraints (INST [7], Bejewelled Player Model (BPM) [14], and Information Foraging Theory (IFT) [1]). With each metric proposed a new evaluation script is typically introduced (or sometimes not), with the explanation in part that the metrics are more sophisticated and go beyond the typical assumptions made by trec_eval, and in part because of the complexity already present in trec_eval. The growth in the diversity of tools means that researchers wanting to use "state-of-the-art" metrics in their work need to obtain a variety of independent scripts (if they exist), and manage their differing input formats [9]. As a consequence there has been slow uptake of newer metrics despite evidence that metrics like INST, TBG, BPM, and IFT are able, in various ways, to make more accurate predictions of performance, and/or are more correlated with user satisfaction than traditional approaches; and despite the standard set of trec_eval metrics making questionable modelling assumptions (Average Precision (AP) [8]), or having mathematical infelicities (Reciprocal Rank (RR) [2]).

We describe a common, extensible, open-source resource for evaluation metrics, ensuring back-compatibility with trec_eval, so that previous measurements of performance remain available in a single tool, and allowing the community to also employ these newer measures. In its simplest form, cwl_eval takes the same input as trec_eval, but also provides additional options.

To provide the theoretical underpinnings we draw upon the **C**/**W**/**L** framework proposed by Moffat et al. [8]. This framework provides a standardised and intuitive way to encode a wide range of metrics, and enables reporting of not just *Expected Utility* per item inspected, the rate at which gain is acquired, but also the related measurements of *Expected Total Utility*, the gain accumulated from the whole list; *Expected Cost* per item inspected; *Expected Total Cost*, the cost incurred in examining the results list; and *Expected Depth*, the number of items a searcher examines given the user model encoded within the metric.

The **C**/**W**/**L** framework is described in detail elsewhere [1, 7, 8]. In brief, it models the manner in which probabilistic users examine a ranked list, assuming that each user reads top down and is governed by a conditional continuation probability $0 \le C_i \le 1$ that indicates the fraction that proceed to examine the item at depth $i+1$, given that they have examined the item at depth i. Different choices of $\mathbf{C} = \langle C_i \rangle$ then reflect different beliefs about user behaviour and hence define different metrics. A vector of weights $\mathbf{W} = \langle W_i \rangle$ can be derived from **C**, where W_i is an assessment of the expected proportion of user attention directed at the item at rank i. Using these weights the expected utility (EU) of a ranking is computed as the dot product between the relevance (that is, gain) vector and **W**; and the expected total utility (ETU), how much the user takes from the interaction as a whole, via a similar computation. We can also calculate the expected cost (EC) of examining an item, the dot product between the weights and a cost vector [1, 12], and the expected total cost (ETC). The costs used for the latter two computations can be in units of documents, characters, reading seconds, and so on. For example, the U-Measure [10] calculates cost in characters, while TBG and IFT use seconds. Finally, we can calculate the expected depth (ED), that is, how far down the ranked list a randomly-selected user will go.

Fundamentally, the **C**/**W**/**L** framework generates a wide range of measurements regarding the predicted user interactions with the ranked list of search results; and hence, conversely, allows for a wide range of observational information to be used to derive metrics that relate to user behaviour, and thus to useful measurement.

2 TOOLKIT, APP AND NOTEBOOKS

Toolkit. `cwl_eval` defines three core classes. First, the **Ranking** class is minimally composed of two ordered lists representing the gains and costs of the items returned from the search. Attributes such as total gain and total relevant items are calculated from these two lists. The ranking is then passed to the **CWLRuler**, which is composed of a number of **CWLMetric**s; or directly to a single **CWLMetric**. The latter produces the measurements.

The heart of the toolkit is the **CWLMetric** class, which is used as the basis of different measures. It requires the definition of the **C**, **W** and/or **L** vectors. (Moffat et al. [6] define the connections between them.) Typically, **C** is the easiest commencement point, and **W** and **L** are computed from it by the base class. By creating a Ranking object, and a Metric object, it is possible to ask the Metric for **C**, **W**, or **L** for that Ranking. The examples below show how this is useful in exploring how metrics behave, and the different kinds of user models that they encode. Adding a new metric to the toolkit is simple: first inherit from the **CWLMetric** class, and then describe the **C** vector.

Graded Gain Values. Järvelin and Kekäläinen [3] make use of *graded gains* and non-binary relevance judgements; Moffat and Zobel [5] similarly allow fractional utility. `cwl_eval` also allows fractional gains to be input as part of a standard TREC qrels file.

Residuals. In their description of RBP Moffat and Zobel [5] introduce the notion of *residuals*, the score uncertainty introduced by unjudged documents. In an ideal evaluation, all documents in the ranking would have associated gain values, and metric score measurements would be precise. Howevere, in most practical experimentation only a small subset of the documents in the ranking have been judged. The standard (and `trec_eval`) approach is to regard unjudged documents as being non-relevant, with a gain of zero. The same convention is adopted in `cwl_eval`, and as a result, most (but not all) **C**/**W**/**L** metrics are computed as lower bounds on the "true" score. The residual is then the difference between that lower score and a matching upper value that arises from supposing that every single unjudged document—right through to the end of the collection—is fully relevant.

In particular, high residual values indicate an experimental context in which there might be non-trivial imprecision in the measured values, with further relevance judgements being the only way to be sure whether that is in fact the case. Note that residuals should *not* be thought of as being confidence intervals, or as having any statistical basis. Their sole purpose is to provide a bounding range on the eventual score, based on the partial evidence supplied by incomplete relevance judgements.

Application. The program `cwl_eval.py` provides a similar interface to `trec_eval`, and in simplest form is used via

```
python cwl_eval.py <TREC-QRELs> <TREC-RESULTS>
```

where `<TREC-QRELS>` is the name of a standard four-column TREC-formatted relevance file, and `<TREC-RESULTS>` a standard six-column TREC-formatted result file. When no cost file is specified, `cwl_eval` assumes the cost of each item to be 1.0. If a cost file is specified via the flag "`-c <COSTS>`", then document-specific costs are used to calculate expected (and total) costs.

Note the relationship between the measurements: EU (the expected rate at which gain is acquired per item viewed, averaged over all users) multiplied by ED yields ETU. Similarly, EC (the expected cost per item examined, in the cost units supplied), multiplied by ED, is equal to ETC. Reporting one measurement (EU, the per-item-viewed rate at which gain is accumulated, as is typically the case with Precision and RBP, for example) provides part of the picture; reporting both the expected number of items per unit of cost, and the expected cost, provides more information. Further, note that the rate of gain per unit of cost can be calculated by taking ETU and dividing it by ETC. As already noted, costs can be specified in any units, for example, documents, seconds, characters, kilobytes, etc. If seconds are used, then EC and ETC are seconds per document, and total seconds, respectively.

By default `cwl_eval` outputs a list of metric scores (Precision, RR, AP, RBP, INST, and so on). It can also be readily configured to report other sets of metrics, using the flag "`-m <METRICS>`" and listing the metrics to be reported in a file, one per line, for example:

```
PrecisionCWLMetrics(k=10)
INSTCWLMetric(T=2.5)
APCWLMetric()
RBPCWLMetric(theta=0.3)
TBGCWLMetric(h=20)
```

It is straightforward to select and configure metrics. For example, `cwl_bpm.py` provides an implementation of the Bejewelled Player measure [14], which minimally takes two parameters: T, the total amount of benefit desired, and K, the total amount of cost available to be spent. Adding `BPMCWLMetric(T=4, K=10)` to the list of metrics means that the static Bejewelled Player Model will be computed.

For convenience, three other flags can be set: "`-b <BIBFILE>`", to save the BibTeX associated with the metrics specified/used; "`-n`" to include the column names of each measurement in the output; and "`-r`" to compute and report residuals where it is appropriate to do so. Note that all output is provided on a per query/topic basis, the presumption being that a subsequent processing phase will compute average scores over topics and also (if required) carry out statistical tests.

Tests. As part of the development, we checked `cwl_eval`'s output of a range of traditional metrics (P@k, AP, NDCG@k) for consistency with `trec_eval`. For this we used standard TREC test collections, and then ranked the documents for the corresponding topics using BM25. Those runs were then fed into both evaluation tools, and the scores for the metrics were compared.

One issue that needed to be addressed was the internal sorting performed by `trec_eval`, which ignores the provided ranking, and sorts the list by score, and then reverse document identifier (see Yang et al. [13] for further discussion). When we re-sorted the runs according to this criteria, `cwl_eval` provided the same scores as `trec_eval`. Scripts for all of these validation tests are provided in the `cwl_eval` GitHub repository. We also checked the results against

those of `inst_eval`[1] [4], and the internal tests of the `irmetrics` package[2], to verify that the scores were equivalent; scores agreed to within rounding error.

Implementation. The C/W/L Evaluation Toolkit, and `cwl_eval`, is available at https://github.com/ireval/cwl. A demonstration of using the toolkit in Jupyter Notebooks is provided at https://github.com/ireval/cwl-examples.

3 C/W/L EXAMPLES

To provide examples of how the C/W/L framework can be used to see the inner workings of the different metrics, we created a number of Jupyter Notebooks, which show: (i) how the different metrics can be instantiated; (ii) how to access the different vectors; and (iii) how to plot them.

Example Topics. For the examples below, we have assumed that we have two topics (T1 and T2). The table below shows the corresponding gain vectors (in the range 0–1) and cost vectors (item inspection times, in seconds), to depth ten in each case.

rank i	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15
T1 Gain	0.0,	0.0,	0.2,	0.4,	1.0,	0.2,	0.0,	0.0,	1.0,	0.0,	0.0,	0.4,	0.0,	0.0,	0.0
Cost	1.2,	0.6,	0.4,	0.6,	3.6,	1.6,	0.6,	2.6,	0.6,	0.6,	0.6,	0.6,	0.6,	0.6,	1.8
T2 Gain	1.0,	0.0,	1.0,	0.4,	0.0,	0.2,	0.0,	0.0,	1.0,	0.2,	0.0,	0.4,	0.0,	0.0,	0.0
Cost	3.2,	1.6,	1.4,	0.6,	3.6,	1.6,	0.6,	1.6,	2.6,	0.2,	1.2,	0.2,	0.2,	0.6,	1.8

Sample Output. Below we have included two examples of the output from `cwl_eval`, where: (i) no cost information is included, and (ii) cost information is included. Table 1 shows the resulting output when costs are not provided (only T1 is reported). Here, the default cost per item is assumed to be one (a unit cost), and so the Expected Cost per item viewed is one, and the Expected Total Cost is the same as Expected Depth.

Table 2 shows the output when the document costs are provided. Now the expected cost per item depends on how much time it takes to process each item, and the proportion of attention allocated to each item. In this example the expected total cost is now different from the expected depth, by a factor related to the weighted cost of processing the inspected items. Costs are in the units provided, so here the output units are seconds/document (EC) or seconds (ETC).

Inside Metrics. The repository includes a Jupyter Notebook to show how we can visualise and inspect the models within the metrics. Figure 1 shows the C/W/L functions for a subset of the metrics on the two example topics. For RBP the plots are the same for topics T1 and T2—this is because RBP is not sensitive to either the cost values or the gain values. Both TBG and BPM consider both the gain and cost vectors to determine how likely a user is to continue down the ranked list (C), and hence the proportion of attention that is paid to each result (W) and how likely the user is to stop at a given rank (L).

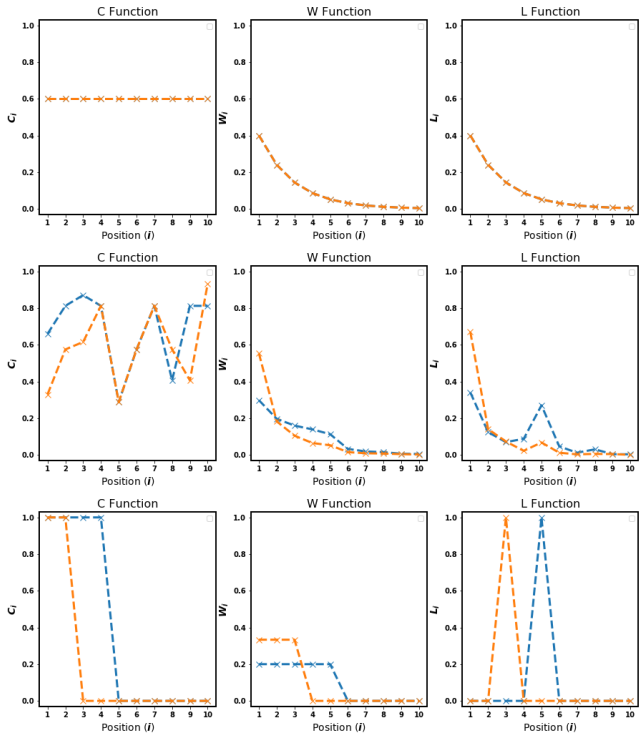

Figure 1: C/W/L function plots for three metrics for Topics T1 and T2. Top row: RBP, $\phi = 0.8$; middle row, TBG, $H = 2$; bottom row, BPM, $T = 2$, $K = 10.0$, $hc = 0.5$, $hb = 0.5$

Visualising the Measurements. Since `cwl_eval` outputs a series of measurements given the metric, it is now possible to contextualise the usual Expected Utility measurement with respect to the Expected Depth, and to visualise how the EU and ED change when

[1] https://github.com/ielab/inst_eval
[2] https://github.com/Microsoft/irmetrics-r

Table 1: `cwl_eval` output without cost information.

Topic	Metric	EU	ETU	EC	ETC	ED
T1	AP	0.2722	1.6000	1.0000	5.8776	5.8776
T1	RR	0.0667	0.2000	1.0000	3.0000	3.0000
T1	P@5	0.3200	1.6000	1.0000	5.0000	5.0000
T1	NDCG-k@10	0.2270	1.0314	1.0000	4.5436	4.5436
T1	INST-T=2	0.1545	0.6069	1.0000	3.9220	3.9292
T1	TBG-H@2	0.1752	0.5981	1.0000	3.4142	3.4142
T1	BPM-Dynamic-...	0.3200	1.6000	1.0000	5.0000	5.0000
T1	IFT-...	0.0659	0.1097	1.0000	1.6649	1.6649

Table 2: `cwl_eval` output with costs information.

Topic	Metric	EU	ETU	EC	ETC	ED
T1	AP	0.2722	1.6000	1.1681	6.8653	5.8776
T1	RR	0.0667	0.2000	0.7333	2.2000	3.0000
T1	P@5	0.3200	1.6000	1.2800	6.4000	5.0000
T1	NDCG-k@10	0.2270	1.0314	1.1827	5.3738	4.5436
T1	RBP@0.6	0.1287	0.3218	1.0208	2.5520	2.5000
T1	TBG-H@2	0.2143	0.7195	1.1513	3.8663	3.3582
T1	BPM-Dynamic...	0.3200	1.6000	1.2800	6.4000	5.0000
T1	IFT...	0.0748	0.1269	1.0857	1.8412	1.6959

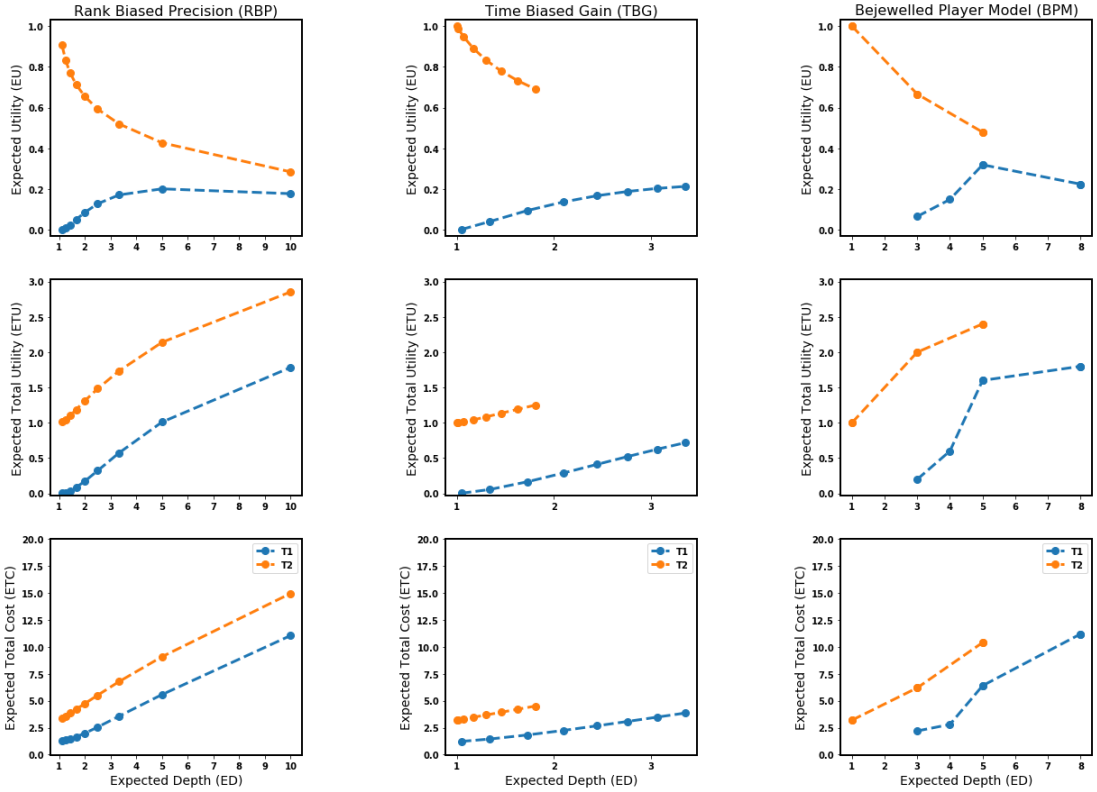

Figure 2: Example of how measurements change over metric parameters. Left column: RBP (where θ was varied from 0.1 to 0.9); middle: TBG (where the half-life parameter is varied from 0.25 to 2); Right: BPM (where T was varied from 0.5 to 4.0). Top row: Expected Utility (EU) per item inspected vs Expected Depth (ED); middle: Expected Total Utility (ETU) vs ED; bottom: Expected Total Cost (ETC) vs ED.

the parameters of the metrics are varied. Figure 2 shows how Expected Utility and Expected Total Utility, and Expected Total Cost, vary as a function of ED for three metrics. Each plotted point represents a particular parameter setting combination for the metric and its corresponding predictions.

4 SUMMARY

We have described the C/W/L evaluation framework, a toolkit and an application for evaluating information retrieval systems. This work represents the unification of various metrics within one package, enabling direct comparison between the estimates of such metrics. It also provides the foundations for the development of new utility- and cost-based metrics.

REFERENCES

[1] Leif Azzopardi, Paul Thomas, and Nick Craswell. Measuring the utility of search engine result pages: An information foraging based measure. In *Proc. SIGIR*, pages 605–614, 2018.

[2] Norbert Fuhr. Some common mistakes in IR evaluation, and how they can be avoided. *SIGIR Forum*, 52(2):32–41, 2017.

[3] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[4] Bevan Koopman and Guido Zuccon. A test collection for matching patient trials. In *Proc. SIGIR*, pages 669–672, 2016.

[5] Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Inf. Syst.*, 27(1):2:1–2:27, 2008.

[6] Alistair Moffat, Paul Thomas, and Falk Scholer. Users versus models: What observation tells us about effectiveness metrics. In *Proc. CIKM*, pages 659–668, 2013.

[7] Alistair Moffat, Peter Bailey, Falk Scholer, and Paul Thomas. INST: An adaptive metric for information retrieval evaluation. In *Proc. Aust. Doc. Comp. Symp.*, pages 5:1–5:4, 2015.

[8] Alistair Moffat, Peter Bailey, Falk Scholer, and Paul Thomas. Incorporating user expectations and behavior into the measurement of search effectiveness. *ACM Trans. Inf. Syst.*, 35(3):24:1–24:38, 2017.

[9] Joao Palotti, Harrisen Scells, and Guido Zuccon. Trectools: an open-source python library for information retrieval practitioners involved in trec-like campaigns. In *Proc. SIGIR*, 2019.

[10] Tetsuya Sakai and Zhicheng Dou. Summaries, ranked retrieval and sessions: A unified framework for information access evaluation. In *Proc. SIGIR*, pages 473–482, 2013.

[11] Mark Sanderson. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375, 2010.

[12] Mark D. Smucker and Charles L. A. Clarke. Time-based calibration of effectiveness measures. In *Proc. SIGIR*, pages 95–104, 2012.

[13] Ziying Yang, Alistair Moffat, and Andrew Turpin. How precise does document scoring need to be? In *Proc. AIRS*, pages 279–291, 2016.

[14] Fan Zhang, Yiqun Liu, Xin Li, Min Zhang, Yinghui Xu, and Shaoping Ma. Evaluating web search with a bejeweled player model. In *Proc. SIGIR*, pages 425–434, 2017.