Quit While Ahead: Evaluating Truncated Rankings

Fei Liu, Alistair Moffat, Timothy Baldwin
The University of Melbourne
Melbourne, Australia
fliu3@student.unimelb.edu.au,
ammoffat@unimelb.edu.au, tb@ldwin.net

Xiuzhen Zhang
RMIT University
Melbourne, Australia
xiuzhen.zhang@rmit.edu.au

ABSTRACT

Many types of search tasks are answered through the computation of a ranked list of suggested answers. We re-examine the usual assumption that answer lists should be as long as possible, and suggest that when the number of matching items is potentially small – perhaps even zero – it may be more helpful to "quit while ahead", that is, to truncate the answer ranking earlier rather than later. To capture this effect, metrics are required which are attuned to the length of the ranking, and can handle cases in which there are no relevant documents. In this work we explore a generalized approach for representing truncated result sets, and propose modifications to a number of popular evaluation metrics.

1. INTRODUCTION AND BACKGROUND

Ranked answer lists are a staple of search; and mechanisms for generating and evaluating them are widely known [1]. In most experimentation, ranked lists are taken to be of arbitrary length, that is, potentially spanning every item in the underlying collection; or to be of some fixed but large length, perhaps to depth $d = 1{,}000$. But there are also situations in which there is only a small number of relevant answers ("find the home page of ...") or no relevant answers to date ("how do I get LaTeX to ..."), for which generating a long list of unhelpful results is counter-productive. When confronted with such questions, an effective retrieval system might truncate its ranking after just a few suggestions, or even offer no answers at all, choosing to "quit while ahead"; assuming, of course, that the user understands the message being conveyed when a truncated ranking is generated by a system. Here we consider how to compute an effectiveness score for rankings that are of variable – and possibly zero – length, based on which we propose modifications to a range of popular evaluation metrics.

Effectiveness Metrics for Extended Rankings A large number of effectiveness metrics for ranked lists have been described, covering both binary relevance (the gain r_i associated with position i in the ranking is either zero or one), and graded relevance (r_i may take on arbitrary non-negative values). These include precision-focused metrics such as Precision@k and Reciprocal Rank (RR), which is the precision at the first relevant document in the ranking. Other metrics add a recall component, such as Average Precision (AP).

Järvelin and Kekäläinen [2] describe a top-weighted evaluation metric they call *discounted cumulative gain* (DCG). A key development in this metric is that items near the top of the ranking are explicitly given a greater influence on the final score than are items later in the ranking. The formulation usually used is given by DCG@$d = \sum_{i=1}^{d}(r_i/\log_2(1+i))$, where d is the chosen evaluation depth. An issue with DCG is that the values generated are unbounded; to address this, Järvelin and Kekäläinen also introduce a normalized version (NDCG), defined as the DCG score at that depth divided by the DCG of a permuted ideal ranking in which all relevant documents are returned at the head of the answer list: NDCG@d = DCG@d / DCG$_I$@d. An NDCG@d score of 1.0 indicates that, down to depth d, the ranking is as good as would have been attained by an omniscient system. Note, however, that the DCG score of a ranking in which there are no relevant answers is zero; and hence that NDCG is undefined on nil-answer queries. Other recall-based metrics, including Average Precision and the Q-measure [5], face the same challenge.

Moffat and Zobel [3] proposed an alternative top-weighted approach that avoids the need for the normalizing step. Their *Rank-Biased Precision* (RBP) metric is based on a simple user model, assuming that the user always looks at the first returned document, and then continues from one depth i in the ranking to the next depth $i+1$ with a fixed probability p, their *persistence*. The expected per-document rate at which gain is accrued is then given by RBP $= (1-p)\cdot\sum_{i=1}^{\infty} r_i \cdot p^{i-1}$. Rank-biased precision assigns a score of zero to an empty ranking list, regardless of whether the query that led to the ranking has answers or not.

Effectiveness Metrics for Truncated Rankings Peñas and Rodrigo [4] note that in some question-answering (QA) scenarios, not responding is preferable to responding incorrectly, and propose a metric they denote c@1. Scores are based on having correct answers at the head of the ranked list, together with a component that is extrapolated for empty lists: $c@1 = n_{ac}/n + (n_{ac}/n)\cdot(n_u/n)$, where n_{ac} is the number of correctly answers across a set of n questions, and n_u is the number of unanswered questions. However, c@1 is only applicable in cases where each question has a single correct answer, such as reading comprehension tests.

Another option for adding nil-answer assessment to an evaluation is to treat questions for which there are no answers differently from the *has-answer* queries. This may be appropriate if the distribution for the two classes of questions is imbalanced and nil-answer questions account for a small fraction of queries; the evaluation can then be one of correct classification between the two classes, followed by a standard evaluation within the has-answer class. For example, in the TREC 2001 QA track, there are 49 nil-answer ques-

SIGIR '16, July 17–21, 2016, Pisa, Italy

ISBN 978-1-4503-4069-4/16/07...$15.00

DOI: http://dx.doi.org/10.1145/2911451.2914737

Liu, Fei, Alistair Moffat, Timothy Baldwin and Xiuzhen Zhang (to appear) Quit While Ahead: Evaluating Truncated Rankings. In Proceedings of 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016), Pisa, Italy.

tions, out of 492 test questions. Similar statistics arise in the TREC 2002–2007 QA tracks. But note also that there are cases where nil-answer queries dominate. For example, in duplicate question detection for community question answering, the expectation is that most new questions will not have previously been asked.

Sakai [5] proposed that NIL be regarded as a valid answer list of length one with positive gain, and showed that under this interpretation the Q-Measure (and other recall-based approaches) can be used to evaluate nil-answer questions. A similar approach was also used in the 2001 TREC QA track [6], where systems were permitted to return NIL in their answer lists. Any NIL's that appeared were assigned a gain of $r_i = 1.0$ if and only if there were no "actual" answers to that query, and a gain of $r_i = 0$ otherwise. Systems were free to continue listing documents after the NIL, meaning that a simple hedging strategy is to prefix NIL to every returned list; another, to insert NIL part way through every answer list. We explore the implications of this approach in more detail in Section 3.

2. EVALUATING ARBITRARY RANKINGS

All Rankings Are Different We propose that a system always be viewed as returning a ranking of documents, and that the length of that ranking always be regarded as having been determined by the system in response to the query. We then require that the evaluation process employed should be applicable to all rankings, including those of zero length.

As a motivating example, consider the case of a query for which there are known to be $R = 3$ relevant answers. For this query the five-document ranking (reading r_i values from left to right, with "1" representing relevant, and "0" denoting non-relevant) "10100" is almost certainly superior to the ranking "01001", a relativity supported by all of RR, AP, NDCG, and RBP. Now consider the three-element ranking "101". It seems clear that "101" must be regarded as superior (or, at the very least, not inferior) to "10100", since it has the relevant documents in the same positions, and fewer non-relevant documents. Next, consider the ranking "011". Where does it fit in relation to the other three rankings? Most metrics would assess it as being inferior to "101" and better than "01001", but what about in comparison to "10100"? That is, is: "101" > "10100" > "011" > "01001" the preferred ordering from a user's point of view, where > is used as an abbreviation for numeric order, based on score? Or is: "101" > "011" > "10100" > "01001" the preferred relationship? And, what about the ranking "1" – is one correct answer and no non-relevant answers better, or worse, than the rankings shown, all of which contain two correct answers? Finally, do any of these relativities change if instead of $R = 3$ relevant documents, there are known to be $R = 5$, or $R = 10$?

In the proposed new framework, in which ranking length is also regarded as being a factor that affects the score, dealing with nil-answer queries becomes a natural extension. If a query has no answers, then we would expect the evaluation metric to tell us that "" > "0" > "00" > "000", and so on. Indeed, if a query has no answers, and a system returns a ranking containing no documents, would we not wish the score of that ranking to be 1.0, representing "fully correct system response, and cannot be improved on"?

Depth-Sensitive Evaluation To allow ranking length to influence assessed effectiveness, we modify every ranking to add a nominal *terminal document* at the first rank position after the last one supplied by the retrieval system. For example the ranking "011" is extended to make a new ranking "011t", where "t" represents the terminal document, and reflects that the system declined to provide an answer document in that or any subsequent position. Provided that a corresponding gain value $r_{\mathtt{t}}$ is also assumed, any weighted-precision effectiveness metric, such as RR, Precision@k, or RBP, can then be used to score the ranking.

The key to making this approach work is selecting a value for $r_{\mathtt{t}}$, the gain value associated with the terminal document. In the 2001 TREC QA Track, and in the example presented by Sakai [5], $r_{\mathtt{t}} = 1.0$ iff the question is a nil-answer one, and $r_{\mathtt{t}} = 0.0$ if not. We propose a more gradual approach. Suppose that the total gain pool for the query is $R \geq 0$. Then at depth $d \geq 0$ in any given ranking the fraction of the available gain that has been accrued is given by $\sum_{i=1}^{d} r_i/R$. On this basis, we define:

$$r_{\mathtt{t}} = \begin{cases} 1 & \text{if } R = 0 \\ \sum_{i=1}^{d} r_i/R & \text{if } R > 0\,. \end{cases} \tag{1}$$

To understand the implications of this definition, consider the metric RR, defined for binary gain values as the reciprocal of the first rank at which a relevant document appears. If a ranking of length d contains a relevant answer, then RR has the same value as it always does, since the terminal document at depth $d + 1$ has no bearing. If a ranking of length d does not contain a relevant answer, and if $R > 0$, then $r_{\mathtt{t}} = 0$ and hence the value of RR is zero, as it should be – the system failed to return an answer that exists. But if $R = 0$, then $r_{\mathtt{t}} = 1$, and the value of RR is given by $1/(d+1)$. That is, an empty ranking will be given a score of 1.0 if there are no relevant documents in the collection; the ranking "0" will be given a score of 0.5 when $R = 0$, and so on. Overall, the adjusted RR′ computation that takes the terminal document into account smoothly adapts its score on nil-answer queries, as required; and has its previous behavior on has-answer queries.

In the case of RBP, $r_{\mathtt{t}}$ is used in a slightly different way. Since RBP computes an infinite weighted sum over a geometric sequence of weights, it is appropriate to presume an arbitrary number of answers past the dth one, all with gain $r_{\mathtt{t}}$. That is, the finite truncated gain vector $\langle r_1, r_2, \cdots, r_d\rangle$ is treated as an infinite one, $\langle r_1, r_2, \cdots, r_d, r_{\mathtt{t}}, r_{\mathtt{t}}, r_{\mathtt{t}}, \cdots\rangle$, and the RBP score computed as normal. This has the same effect as taking the RBP residual at depth d, which is given by p^d, and multiplying it by $r_{\mathtt{t}}$. That is, we define the adjusted RBP as

$$\text{RBP}' = \text{RBP}'@d = (1-p)\cdot\sum_{i=1}^{d} r_i \cdot p^{i-1} + r_{\mathtt{t}} \cdot p^d\,. \tag{2}$$

As a third example, consider NDCG. To adjust this metric to handle truncated lists, we add $r_{\mathtt{t}}$ as a $(d+1)$ th gain value, as for RR, and then use the usual scoring approach to depth $d + 1$ rather than to depth d:

$$\text{NDCG}' = \text{NDCG}'@d = \frac{\text{DCG}@(d+1)\langle r_1, r_2, \cdots, r_d, r_{\mathtt{t}}\rangle}{\text{DCG}_I@(d+1)}\,. \tag{3}$$

Note that this approach also means that d is no longer a parameter of the metric and is instead the length of the ranking supplied by the system; note also that the ideal $(d+1)$-element ranking used in the denominator includes an extra gain of 1.0 in the first zero-gain position only if there are fewer than $d+1$ full- or part-gain answers for the query. For example, if $R = 3$, and all gain values are binary, then the ranking "101" leads to $r_{\mathtt{t}} = 2/3$, and is scored as:

$$\text{NDCG}' = \frac{1/\log 2 + 1/\log 4 + (2/3)/\log 5}{1/\log 2 + 1/\log 3 + 1/\log 4 + 1/\log 5} = 0.698\,,$$

where the final term in the denominator arises because in an ideal ranking of $d = 3$ documents, the corresponding ideal $r_{\mathtt{t}}$ value placed in the fourth position of the ranking would be 1.0.

Ranking	R	$r_{\mathtt{t}}$	RR′	RBP′	NDCG′	AP′
"00"	$R=0$	1.000	0.333	0.250	0.500	0.333
"000"	$R=0$	1.000	0.250	0.125	0.431	0.250
"111"	$R=3$	1.000	1.000	1.000	1.000	1.000
"11"	$R=3$	0.667	1.000	0.917	0.922	0.648
"11100"	$R=3$	1.000	1.000	0.906	0.971	0.917
"101"	$R=3$	0.667	1.000	0.708	0.698	0.528
"1"	$R=3$	0.333	1.000	0.667	0.742	0.306
"10100"	$R=3$	0.667	1.000	0.646	0.678	0.491
"011"	$R=3$	0.667	0.500	0.458	0.554	0.403
"01001"	$R=3$	0.667	0.500	0.302	0.490	0.299

Table 1: Example truncated answer rankings and their modified scores, for two different queries, one with $R = 0$ and one with $R = 3$. The parameter $p = 0.5$ is assumed for the RBP computation. Within each group, the results are sorted by RBP′, which (by chance, for these examples) also corresponds to RR′-order.

Average precision (AP) is handled similarly, by defining $r_{d+1} = r_{\mathtt{t}}$, and then scoring the resulting extended-by-one ranking:

$$\text{AP}' = \frac{1}{R+1} \sum_{i=1}^{d+1} r_i \frac{\sum_{j=1}^{i} r_j}{i} . \quad (4)$$

As is also the case with NDCG′, the reference ranking used by AP′ contains R instances of $r_i = 1$, followed by a nominal terminating document with a gain of 1.0, that is, $R + 1$ values in total.

Table 1 shows scores computed for a range of rankings using the modified versions of RR, RBP, NDCG, and AP. The different adjusted metrics place different emphases on the tradeoff between recall and precision. All of the metrics respect the strict pairwise orderings noted earlier, for example, that "101" $\geq$ "10100"; but they vary in their response to other relativities, such as the question as to whether "1" is better or worse than "101". Note how the different metrics place different emphases on the rankings, resulting in variations in their score orderings.

3. EXPERIMENTS AND RESULTS

Tasks and Test Collections To explore the ramifications of the proposed approach, we employ the runs submitted to the main task of the TREC 2001 QA track. Participants were invited to submit a ranked list of [*doc-id*, *answer-str*] pairs of length up to five for each question; and for questions deemed to have no answer, were permitted to return "NIL" rather than one of the pairs. Overall, 36 groups contributed a total of 67 submissions to the QA main task; 47 of them are available for download.[1] The question set consists of 492 queries, 49 of which are nil-answer queries. The 443 has-answer questions have on average 25.7 relevant answers each.

Interpretation of Truncation To evaluate the proposed approach, we transform each individual run using the rules shown in Table 2, so that we accurately capture any evidence of deliberate truncation. The first two rules, covering cases where fewer than five results are provided, or where an explicit "NIL" is provided, are evidence of system-initiated truncation, and are processed as such in our comparison; in the third case we cannot infer truncation, and those runs are retained intact and scored in the original manner by the unmodified metrics throughout our experimentation.

[1] http://trec.nist.gov/results/trec10/qa_main_input.html

$a_{\mathtt{NIL}}$	n	modified run
i	≤ 5	$a_1, \cdots, a_{i-1}, \mathtt{t}$
-1	< 5	$a_1, \cdots, a_n, \mathtt{t}$
-1	$= 5$	$a_1, \cdots, a_5$

Table 2: Transformation of a run $\langle a_1, \cdots, a_{\mathtt{NIL}}, \cdots, a_n \rangle$, where $a_{\mathtt{NIL}}$ is the rank of an explicit NIL document (either rank $i \in [1, n]$, or -1 indicating not present) to a new ranked list.

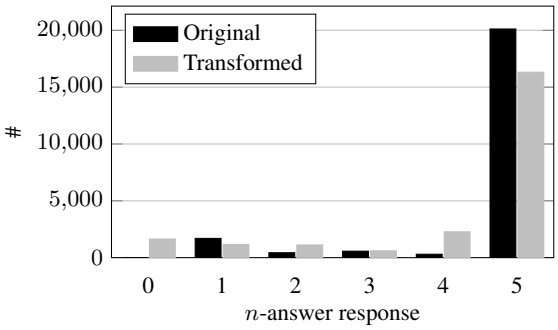

Figure 1: Distribution of lengths of 23,124 query responses.

Figure 1 shows the distribution before and after transformation of the 23,124 runs submitted for the 492 queries by the 47 participants. The number of five-answer lists is reduced from around 20k to 16k, generating a total of approximately 7k truncated answer lists post-transformation. The number of zero-answer lists is zero before the transformation, because even when a system believes a query is a nil-answer question, it must return a "NIL" to indicate so. This also accounts for the decline in the number of single-answer responses post-transformation.

Results and Analysis We first compare the TREC QA systems against each other using the TREC methodology (that is, with NIL in runs given a gain of 1.0 iff a query is nil-answer and otherwise given a gain of 0.0, and with metrics then applied in their standard form), and using our proposed modified approach applied to the transformed version of each run. Four different effectiveness metrics were explored, with the goal of determining the extent to which systems are affected by the proposed alteration in methodology. Each run for each system was scored using the two different approaches, and then system averages computed. In all of these evaluations, a [*doc-id*, *answer-str*] pair is considered correct iff the *answer-str* contains an answer to the question and is supported by the document specified by the *doc-id*.

Table 3 compares the system orderings generated by the four pairs of original/modified metrics using Kendall's τ, which computes a correlation coefficient between pairs of ordered lists over the same domain. Three evaluation metric pairs give rise to τ scores greater than 0.9, indicating strong agreement between the system ordering induced by the original metric and the system ordering generated by its modified version. The strong agreement between RR and RR′ was expected, because scores are primarily derived from just one relevant document, and because only a minority of the runs had explicit NIL markers. The similarly strong agreement between NDCG and NDCG′ was more surprising. At the other end of the scale, the pair AP/AP′ has the lowest τ among the four metrics, but they are still strongly correlated.

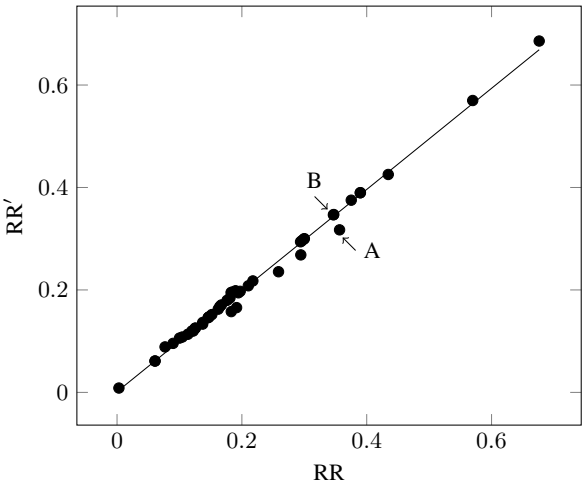

Figure 2: Relationship between RR and RR′ scores for 47 systems, with each system's score the mean over 492 queries.

Metric Pair	Kendall's τ
RR/RR′	0.960
NDCG/NDCG′	0.958
RBP@0.5/RBP′@0.5	0.916
AP/AP′	0.870

Table 3: Kendall's τ correlation coefficient calculated from the system orderings generated by pairs of original and modified metrics.

Figure 2 provides details of the relationship between the RR and RR′ scores for the set of systems. Overall, RR and RR′ are in high agreement in regard to both system ordering (Table 3) and in terms of the actual scores assigned. However, there are also inverted pairs, where a system is ranked higher by the original metric but has inferior score in the modified. For example, the system marked with "A" has a slightly higher score than does "B" for RR, but is ranked lower than "B" by RR′ because of "B"s aggressive (and effective) truncation strategy.

We also investigated the impact of truncation on performance of individual systems. The horizontal axis in Figure 3 (% truncation) is the fraction of answer lists of length less than five, including NIL, but excluding terminal documents. Both of the top two systems receive a boost in score when truncation is taken into consideration. In the $[0.15, 0.3]$ score range, despite the aggressive truncation, there are systems that obtain little improvement, in part due to their placement of a NIL at the end of every run. In addition, much of the truncation is a consequence of the system's inability to find a correct answer, rather than intentionally terminating the answer list. In such cases, even though there is explicit truncation, the system is not rewarded as there is no relevant document in the truncated answer list. Some systems sometimes prematurely truncate an answer list by placing a NIL before relevant documents. This causes the performance to drop when the modified metrics are employed. Two of the systems generated a NIL in the fifth position of all of their answer lists.

4. CONCLUSIONS AND FUTURE WORK

We have identified an opportunity to refine the way in which truncated rankings are evaluated, and at the same time deal seamlessly with a well-known shortcoming of recall-based evaluation metrics, namely, their inability to cope with queries with no relevant documents. By providing modified effectiveness approaches that provide subtle differentiation between runs of different lengths (for example, because "110" < "11" in our mechanism, but not in previous approaches to the problem) we are better able to nuance system evaluations. The approach we employ – the appending of a terminal document to every ranking, to indicate the truncation point, and modifications to a range of standard evaluation metrics including RR, RBP, NDCG and AP – is both intuitive, and also easy to implement and apply. In retrieval experiments over a large QA dataset, containing a non-trivial fraction of nil-answer queries, we illustrated the effectiveness of the modified metrics, and demonstrated that a refined evaluation of truncated document rankings can help differentiate system orderings.

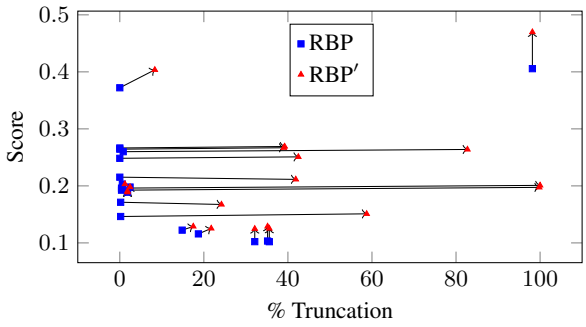

Figure 3: Impact of proposed methodology on effectiveness scores of the top 20 systems. Percentage truncation (horizontal axis) is the fraction of truncated answers (length of answer list < 5, excluding the terminal document), with the two points marking pre- and post transformation scores. The RBP parameter is 0.5 throughout.

The obvious next step in our project is the development of methods for taking long document rankings and identifying, relative to the truncation-sensitive metrics, the point in each at which truncation is appropriate. One possible way of approaching this problem would be through analysis of the distribution of document scores in the ranking, in both relative and absolute terms. Query analysis could also be performed to predict the R value for a given query, for incorporation into the truncation process. We leave this exploration to future work.

Acknowledgments The authors thank MACE Engineering Group for their early support of this work. The third author was supported by ARC grant FT120100658.

References

[1] C. Buckley and E. M. Voorhees. Retrieval system evaluation. In E. M. Voorhees and D. K. Harman, editors, *TREC: Experiment and Evaluation in Information Retrieval*, chapter 3, pages 53–75. MIT Press, 2005.

[2] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Sys.*, 20(4):422–446, 2002.

[3] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Inf. Sys.*, 27(1):2:1–2:27, 2008.

[4] A. Peñas and A. Rodrigo. A simple measure to assess non-response. In *Proc. ACL/HLT*, pages 1415–1424, 2011.

[5] T. Sakai. New performance metrics based on multigrade relevance: Their application to question answering. In *Proc. NTCIR*, 2004.

[6] E. M. Voorhees. Overview of the TREC 2001 question answering track. In *Proc. TREC*, pages 42–51, 2002.