RD-004

NTCIREVAL: A Generic Toolkit for Information Access Evaluation

Tetsuya Sakai†

1. Introduction

Over the past decades, *Information Access* (IA) tasks have evolved and diversified. For example, in the mid-20th century, *Information Retrieval* (IR) was about *set retrieval* for libraries; then with the advent of the digital information overload era, *ranked retrieval* became a necessity; now in the 21st century, we are experiencing richer forms of IR such as *diversified Web search* in order to satisfy *ambiguous* and *underspecified* queries [25]. Moreover, with the progress in natural language processing, automatic *Question Answering* (QA) [10, 15, 20] and leveraging *Community QA* (CQA) data [24] have become feasible. Many of these IA tasks involve automatic ranking of items (e.g. documents or answer strings).

To ensure progress in IA research, reliable evaluation metrics are an absolute necessity. Given an IA task definition, an evaluation metric should be designed so that it can guide the system towards the right goal of that particular task. Hence, together with IA tasks, IA evaluation methods and metrics have also evolved and diversified.

This paper introduces a tookit for evaluating a variety of IA tasks, called NTCIREVAL, designed primarily for tasks that involve ranking of items. NTCIREVAL is available at `http://research.nii.ac.jp/ntcir/tools/ntcireval-en.html`. (This paper discusses the version released in April 2011.) It works on UNIX/Linux platforms. While NTCIREVAL can handle some of the ongoing and past IA tasks of *NTCIR*, the sesquiannual IA evaluation workshop run by National Institute of Informatics, it is a generic toolkit that can be used for other IA tasks. The main objective of this paper is to provide an overview of the philosophy behind and functionalities of NTCIREVAL, so that IA researchers can quickly understand and utilise it whenever appropriate. Because IA research relies much on experimentation, sharing such an evaluation toolkit among the IA researchers should help enhance the reproducibility of experiments, and also foster discussions on how to better evaluate IA tasks. This paper should also serve as a noncomprehensive survey of recent developments in the field of IA evaluation metrics.

The remainder of this paper is organised as follows. Section 2 discusses the design philosophy of NTCIREVAL. Section 3 explains how NTCIREVAL can be used for traditional ranked retrieval evaluation and its extensions. Section 4 explains how it can be used for diversified search evaluation. Finally, Section 5 summarises this paper and provides some general recommendations for IA researchers.

2. Design Philosophy

2.1 Overview

NTCIREVAL consists of a simple C program called `ntcir_eval` and some shell scripts. `ntcir_eval` has been designed to work for a single *topic* (i.e. *search request*): it basically compares a system output with a gold standard (i.e. "right answers") for that particular topic. It has several subcommands for different IA tasks, some of which are shown in Table 1. As this paper is designed to introduce only the general principles of NTCIREVAL, we refer the reader to the README files for details on the arguments and options that can be used with `ntcir_eval` and with the shell scripts.

Compared to `trec_eval`‡, a C program that has been widely used at the Text Retrieval Conference (TREC) and other IR evaluation workshops for over a decade (with numerous updates), NTCIREVAL has several characteristics. For the purpose of discussing these characteristics in Sections 2.2-2.4, Figure 1 shows a very simple example of how `ntcir_eval` can be used on a command line:

```
% cat example.rel
a L1
b L0
% cat example.res
c
b
a
%  cat example.res | ntcir_eval label -r example.rel
c
b L0
a L1
% cat example.res | ntcir_eval label -r example.rel |
ntcir_eval compute -r example.rel -g 1:2
 # syslen=3 jrel=1 jnonrel=1
 # r1=3 rp=3
 RR=                0.3333
 O-measure=         0.5000
 P-measure=         0.5000
 P-plus=            0.5000
 AP=                0.3333
 Q-measure=         0.5000
 NCUgu,P=           0.3333
 NCUgu,BR=          0.5000
 NCUrb,P=           0.3333
 NCUrb,BR=          0.5000
 RBP=               0.0226
 ERR=               0.1111
 AP@1000=           0.3333
 Q@1000=            0.5000
 nDCG@1000=         0.6309
 MSnDCG@1000=       0.5000
 P@1000=            0.0010
 nERR@1000=         0.3333
 Hit@1000=          1.0000
```

Figure 1: An example usage of `ntcir_eval`.

Here, `example.rel` is a relevance assessment file and `example.res` is a system's ranked list for a particular topic. `a`, `b` and `c` represent retrieved items (e.g. document IDs), and L0 and L1 are *relevance labels*. L0 represents an item that was explicitly judged to be nonrelevant, while L1 represents an item that was judged to be relevant with a *relevance level* of 1. The rest of the information shown in Figure 1 will be discussed later.

2.2 Per-topic Execution

IA evaluation often relies on a test collection with a *set* of topics with relevance assessments. Thus, at TREC, for example, a program (like `trec_eval`) typically reads a *qrels* file (a single file containing the relevance assessments for *all* topics) and a *run* file (a single file containing the ranked list of documents for *all* topics), and output evaluation metrics for each topic as well as summary statistics such as the mean of each metric over the topic set.

In contrast, as Figure 1 shows, `ntcir_eval` works only for a single topic: it reads a gold standard file, reads a system output and then computes metrics for a

†Microsoft Research Asia tetsuyasakai@acm.org

‡`http://trec.nist.gov/trec_eval/`

Copyright © 2011 by Information Processing Society of Japan and
The Instiute of Electronics, Information and Communication Engineers
All rights reserved.

Table 1: ntcir_eval subcommands.

Section 3. (traditional ranked retrieval)	`label`	adds relevance labels to a ranked list
	`compute`	computes evaluation metrics for an output from `label`
Section 4. (diversified search)	`glabel`	adds global gain values to a ranked list
	`gcompute`	computes evaluation metrics for an output from `glabel`
	`irec`	computes intent recall for a ranked list

particular topic. The shell scripts take care of running `ntcir_eval` for every topic, computing means, and so on. This reflects the view that what is happening *per topic* is central to IA evaluation. Moreover, this design allows researchers to flexibly use different evaluation settings for different topics, as illustrated below.

Let us go back to the last command in Figure 1. This command first uses the `label` subcommand to add relevance labels to the system's ranked list, and then feeds the output to the `compute` subcommand to compute evaluation metrics. The `compute` subcommand specifies a relevance assessment file by a `-r` option, and also specifies the *gain values* [9] for computing graded relevance metrics by a `-g` option. Here, `-g` serves the following two purposes: (1) declare that the highest relevance level is 2 (by specifying two values separated by a colon); and (2) set the gain value for L1 to be 1 and that for L2 to be 2. If the NTCIREVAL user wants to use exactly the same options for every topic, he can hard-code them within a shell script that runs `ntcir_eval` for every topic. (A sample script is included in NTCIREVAL.) Alternatively, if he wants to set the options *per topic*, he can write a shell script for that purpose. For example, he may choose to write a script that first examines the highest relevance label within the `rel` file (which in the case of `example.rel` is L1), and then dynamically set the `-g` option accordingly (e.g. `-g 1` for some topics and `-g 1:2` for other topics). Dynamically changing the highest relevance level $h(\geq 1)$ across topics will affect metrics that directly relies on h, such as *Expected Reciprocal Rank* [6] and *Rank-Biased Precision* [11]: Section 3.1 discusses these metrics.

NTCIREVAL also contains some scripts for splitting "TREC-style" qrels files and run files into per-topic files[§]. Thus, with NTCIREVAL, a directory is created for every topic, and all *per-topic* gold standards, system outputs and intermediate results are stored under each topic directory. This facilitates per-topic failure analysis, which is vital for advancing the state-of-the-art of IA technologies.

Furthermore, while NTCIREVAL contains some scripts for computing *arithmetic* means of per-topic metric values output by `ntcir_eval`, note that the arithmetic mean is not the only possible way to summarise a system's performance. For example, one can easily write a shell script that computes *geometric* means in order to pay more attention to "hard" topics (i.e. those for which the system performs poorly) [12, 16]. This is another benefit of separating the computation of per-topic performances from that of overall summary statistics.

[§]The scripts for splitting the run files take the original rankings in the run files "as is," not allowing any weak ordering. Thus it is the system's responsibility to break ties. This is in contrast to `trec_eval` which, for historical reasons, reranks documentIDs internally based on the *scores* given within the run files.

2.3 Labelling/Computation Separation

The C program `ntcir_eval` itself has a few unique features. One of them is related to the aforementioned per-topic analysis of experimental results. It can be observed in Figure 1 that `ntcir_eval` isolates the process of *labelling* the system output from that of metrics computation, by means of the two subcommands `label` and `compute`. Here, *labelling* refers to the process of determining which items in the system output should be *considered* relevant.

Figure 1 includes a very simple example of labelling: `ntcir_eval` compares the `res` (system's result) file with the `rel` (relevance assessment) file, and adds appropriate relevance labels to the system output. The first advantage of isolating the labelling process from metrics computation is that it helps the NTCIREVAL user manually examine the quality of each system output: he can easily see which item is relevant, as well as how relevant. But there are more advantages, as discussed below.

Figure 2 shows what happens when the `-j` ("judged") option is used with the `label` subcommand. In IR evaluation based on *pooling* where relevance assessments are performed only for items that have been retrieved by at least one participating system [13], items can be categorised into the following three classes: (i) *judged relevant* items (i.e. items included in the pool and judged to be relevant); (ii) *judged nonrelevant* items (i.e. items included in the pool and judged to be nonrelevant, represented as L0); and (iii) *unjudged items* (i.e. items that were not in the pool and therefore we do not know whether they are relevant or not). It can be observed that, while all three items are output by the `label` command in Figure 1, the unjudged item `c` is not output by the `label` command in Figure 2, because of the `-j` option. A ranked list whose unjudged documents have been removed in this way is called a *condensed list*. It has been shown that if IR metrics are computed based on a condensed list instead of the original ranked list, they can provide more reliable results when the relevance assessments are *incomplete* (i.e. there are many relevant documents that have not been identified) [4, 19, 21]. Thus, the metrics shown in Figure 2 are condensed-list versions of the original metrics. We shall discuss them again in Section 3.2.

Figure 3 shows another example of utilising the fact that `ntcir_eval` isolates labelling from metrics computation. Here, instead of `example.rel` which we used in Figure 1, a slightly modified gold-standard file called `example.erel` is used. This file has a third field, which represents the ID of an *equivalence class* to which each item belongs. Note that a `-ec` option is used with both `label` and `compute` in order to declare that the `erel` file contains equivalence class information. Evaluation based on equivalence classes is useful, for example, for evaluating ranked lists of answer strings in factoid QA [15, 20]. For example, in response to a question: "Who wrote songs for The Beatles with John

Copyright © 2011 by Information Processing Society of Japan and
The Instiute of Electronics, Information and Communication Engineers
All rights reserved.

```
% cat example.res | ntcir_eval label -j -r example.rel
b L0
a L1
% cat example.res | ntcir_eval label -j -r example.rel |
ntcir_eval compute -r example.rel -g 1:2
 # syslen=2 jrel=1 jnonrel=1
 # r1=2 rp=2
 RR=                0.5000
 O-measure=         0.6667
 P-measure=         0.6667
 P-plus=            0.6667
 AP=                0.5000
 Q-measure=         0.6667
 NCUgu,P=           0.5000
 NCUgu,BR=          0.6667
 NCUrb,P=           0.5000
 NCUrb,BR=          0.6667
 RBP=               0.0238
 ERR=               0.1667
 AP@1000=           0.5000
 Q@1000=            0.6667
 nDCG@1000=         1.0000
 MSnDCG@1000=       0.6309
 P@1000=            0.0010
 nERR@1000=         0.5000
 Hit@1000=          1.0000
```

Figure 2: Using `ntcir_eval label` with `-j`.

Lennon?” suppose that a system returned “Paul McCartney” at rank 2 and “McCartney” at rank 3, and that these two answer strings form an equivalence class (i.e. they are interchangeable). Then, EC-based evaluation can penalise this redundancy by treating the correct answer at rank 3 as if it is nonrelevant. The `elabel` command in Figure 3 does exactly this: although `a` in `example.res` is a correct item according to `example.erel`, it is not marked as relevant because a correct answer from the same equivalence class has already been found at rank 2, namely, `b`.

```
% cat example.erel
a L1 1
b L2 1
% cat example.res | ntcir_eval label -ec -r example.erel
c
b L2 1
a
% cat example.res | ntcir_eval label -ec -r example.erel |
ntcir_eval compute -ec -r example.erel -g 1:2
 # syslen=3 jrel=1 jnonrel=0
 # r1=2 rp=2
 RR=                0.5000
 O-measure=         0.7500
 P-measure=         0.7500
 P-plus=            0.7500
 AP=                0.5000
 Q-measure=         0.7500
 NCUgu,P=           0.5000
 NCUgu,BR=          0.7500
 NCUrb,P=           0.5000
 NCUrb,BR=          0.7500
 RBP=               0.0475
 ERR=               0.3333
 AP@1000=           0.5000
 Q@1000=            0.7500
 nDCG@1000=         1.0000
 MSnDCG@1000=       0.6309
 P@1000=            0.0010
 nERR@1000=         0.5000
 Hit@1000=          1.0000
```

Figure 3: Using `ntcir_eval label` and `compute` with `-ec`.

In Figure 3, the `compute` subcommand with `-ec` takes the result of `label` (also with `-ec`) and then computes IR metrics just as in Figure 1. The only difference here is that now the total number of equivalence classes is taken as the number of relevant items. Section 3.3 will discuss more on these equivalence-class versions of the original metrics.

Because of this separation between labelling and metrics computation, other labelling strategies can easily be implemented if required. For example, as Sakai [18] suggested, it would be easy to implement and experiment with IR metrics based on *combinatorial relevance*: suppose that, in a patent search task, patents `a` and `b` can invalidate a new patent application *only if the two are used together*. Then, suppose that a patent search system returned `a` at rank 1 and `b` at rank 3. By assuming that the patent searcher needs to scan the ranked list down to rank 3 in order to obtain both of these “pieces of” relevant items, we may choose to skip `a`, and label only `b` as relevant. Then standard IR metrics may be computed using `compute`.

In summary, the isolation of labelling from metrics computation makes `ntcir_eval` quite flexible.

2.4 Graded Relevance

For over a decade after 1992, TREC used *binary* relevance assessments for IR evaluation. Reflecting this history, `trec_eval` is basically a tool for computing *binary-relevance* metrics such as *Average Precision* [4]. It is only recently that a patch was added to `trec_eval` (in version 8) so that it can compute *normalised Discoundted Cumulative Gain* (nDCG) [9], a graded relevance metric¶.

In contrast, NTCIR has used graded relevance assessments from the very beginning (i.e. since 1999). Somewhat reflecting this history, NTCIREVAL has been designed from the very beginning as a toolkit for evaluation with graded-relevance evaluation metrics. It can compute a variety of graded-relevance metrics which `trec_eval` does not cover. Furthermore, `ntcir_eval` can compute equivalence-class versions of different metrics, as well as a variety of diversity search metrics. Details are provided below.

3. Traditional Ranked Retrieval

This section discusses the metrics for traditional ranked retrieval, as well as their condensed-list and equivalence-class versions, that `ntcir_eval` supports.

3.1 Basic Metrics

Let us go back to Figure 1, which shows all the metrics computed by the `compute` subcommand by default. In this figure, `syslen` is the size of the system output, `jrel` is the number of judged relevant documents, and `jnonrel` is the number of judged nonrelevant (i.e. L0) documents. The rest of the output are various evaluation metric values.

We first define binary-relevance metrics. *RR* is the *Reciprocal Rank*: let r_1 denote the rank of the first relevant document in the ranked list; then $RR = 1/r_1$. If there is no relevant document in the list, RR is defined to be zero. RR can be interpreted as a binary-relevance evaluation metric for *navigational* queries [3], where the user typically requires exactly one relevant document.

Let $I(r)$ be a flag s.t. $I(r) = 1$ if the document at rank r is relevant and 0 otherwise, and let $C(r) = \sum_{k=1}^{r} I(k)$, i.e. number of relevant documents between ranks 1 and r. Then *Hit* at document cutoff l (where $l = 1000$ by default) is defined as $Hit@l = 1$ if $C(l) > 0$ and 0 otherwise; *Precision* at l is defined as $P@l = C(l)/l$. Furthermore, let R denote the total number of known relevant documents. Then *Average Precision* is given by:

$$AP = \frac{1}{R} \sum_{r} I(r) \frac{C(r)}{r} \ . \tag{1}$$

AP is a popular binary-relevance evaluation metric suitable for *informational* queries [3] where the user

¶I thank Ian Soboroff for the information on his `trec_eval` patch.

Copyright © 2011 by Information Processing Society of Japan and
The Instiute of Electronics, Information and Communication Engineers
All rights reserved.

typically requires as many relevant documents as possible.

Figure 1 also shows a document cutoff-based variant of AP (`AP@l`), which replaces the R in Eq. 1 with $\min(l, R)$ to ensure that the highest possible value is 1 even if $l < R$.

Next, we define graded-relevance metrics, which can distinguish between (say) highly relevant and partially relevant documents. Let $g(r)$ denote the *gain value* at rank r. For example, suppose we have L2 (relevant) and L1 (partially relevant) documents. Then, the gain value setting shown in Figure 1 means that $g(r) = 2$ if the document at rank r is L2, and $g(r) = 1$ if the document is L1, and $g(r) = 0$ if the document is either L0 (judged nonrelevant) or unjudged. Let the *cumulative gain* at rank r be $cg(r) = \sum_{k=1}^{r} g(k)$.

Many graded-relevance metrics rely on the notion of the *ideal* ranked list [9]. This can be obtained by sorting all known relevant documents in decreasing order of the relevance levels. Let $g^*(r)$ denote the gain value at rank r in an ideal list, and let $cg^*(r) = \sum_{k=1}^{r} g^*(k)$. Furthermore, for a given positive parameter β (which defaults to 1), let $BR(r) = (C(r) + \beta cg(r))/(r + \beta cg^*(r))$. This is a graded-relevance extension of precision called the *blended ratio* [15].

O-measure, a graded-relevance version of RR, is defined as: $O\text{-}measure = BR(r_1)$ if there is at least one relevant document in the ranked list, and zero otherwise. Note that the relevance level of the document at rank r_1 does not matter as long as it is at least somewhat relevant. Thus, RR and O-measure assume that the search engine user stops scanning the ranked list as soon as he finds one (somewhat) relevant document. In contrast, *P-measure* and P^+ (`P-plus` in Figure 1) assume that the user goes down as far as the *preferred rank* r_p, which is the highest rank that has one of the most relevant documents within the ranked list [17]. Thus, for a ranked list that contains at least one relevant document, $P\text{-}measure = BR(r_p)$, and

$$P^+ = \frac{1}{C(r_p)} \sum_{r=1}^{r_p} I(r) BR(r) \ . \qquad (2)$$

Again, for a ranked list that does not contain a relevant document, we define: $P\text{-}measure = P^+ = 0$.

Q-measure is a graded-relevance extension of AP, defined as [15]:

$$Q\text{-}measure = \frac{1}{R} \sum_{r} I(r) BR(r) \ . \qquad (3)$$

Also, `ntcir_eval` computes a cutoff-based variant of Q-measure (`Q@l`) [26] by replacing the R in Eq. 3 by $\min(l, R)$.

Expected Reciprocal Rank (ERR) [6] and *nERR@l (normalised ERR at cutoff l)* [26] are defined as follows. Let $Pr(r)$ denote the *relevance probability* of the document at rank r. (In our implementation, we let $Pr(r) = g(r)/(g_h + 1)$, where g_h is the gain value for the highest relevance level.) ERR interprets this as the probability that the user is satisfied with the document at rank r. Thus the probability that the user is dissatisfied with documents from ranks 1 to r is given by $dsat(r) = \prod_{k=1}^{r}(1 - Pr(k))$. Let $Pr^*(r)$ and $dsat^*(r)$ denote the corresponding probabilities for the ideal ranked list. Then ERR and nERR@l can be defined as

$$ERR = \sum_{r} Pr(r) dsat(r-1)/r \qquad (4)$$

$$nERR@l = \frac{\sum_{r=1}^{l} Pr(r) dsat(r-1)/r}{\sum_{r=1}^{l} Pr^*(r) dsat^*(r-1)/r} \ . \qquad (5)$$

Thus, ERR is based on the expected probability that the user is finally satisfied at rank r and stops examining the ranked list.

All of the metrics mentioned so far can be regarded as an instance of the *Normalised Cumulative Utility* (NCU) metrics family [22], whose generic form is:

$$NCU = \sum_{r} Pr^{stop}(r) NU(r) \qquad (6)$$

where $Pr^{stop}(r)$ is the probability that the search engine user stops examining the ranked list at r and $NU(r)$ is a *normalised utility* function that should reflect the cost and benefit of examining the documents down to rank r. `ntcir_eval` supports two special stopping probability distributions, as described below.

The first is the *rank-biased* (RB) distribution, given by $Pr^{stop}(r) = \lambda^{C(r)-1}/\sum_{k=1}^{R} \lambda^{k-1}$ for every rank r with a relevant document. λ is a parameter which defaults to 0.95. The assumptions are that users stop examining the ranked list at a relevant document, and that users tend to stop at early ranks. For example, suppose that there are $R = 3$ relevant documents, and that two of them are retrieved at ranks 1 and 5, respectively. Then, $Pr^{stop}(1) = 1/(1 + 0.95 + 0.95^2) = 0.35$, $Pr^{stop}(5) = 0.95/(1 + 0.95 + 0.95^2) = 0.33$.

The second is the *graded-uniform* (GU) distribution, given by $Pr^{stop}(r) = g(r)/\sum_r g^*(r)$. The assumptions are that users stop examining the ranked list at a relevant document, and that users are more likely to stop at a highly relevant document than at a partially relevant document. For example, suppose that we have two L2-relevant documents and one L1-relevant documents, and we assign gain values of 2 and 1 to them, respectively. Then, for every rank where there is an L2-relevant document, $Pr^{stop}(r) = 2/(2+2+1) = 0.4$. At the rank where there is the L1-relevant document, $Pr^{stop}(r) = 1/(2+2+1) = 0.2$.

As for the normalised utility function, `ntcir_eval` supports $NU(r) = P(r)$ (precision) and $NU(r) = BR(r)$ (blended ratio). Thus, in Figure 1, `NCUgu,P` is the NCU with the GU-distribution with $NU(r) = P(r)$, `NCUrb,BR` is the NCU with the RB-distribution with $NU(r) = BR(r)$, and so on.

Recall that all of the other metrics described previously can be regarded as an NCU metric. For example, that Q-measure is an NCU with a uniform distribution over all relevant documents: $Pr^{stop}(r) = 1/R$; P^+ is an NCU with a uniform distribution over relevant documents retrieved between ranks 1 and r_p: $Pr^{stop}(r) = 1/C(r_p)$; P-measure (O-measure) is an NCU with a 100% stopping probability at rank r_p (r_1).

Also, `ntcir_eval` computes two versions of *normalised discounted cumulative gain* (nDCG). The *original* nDCG [9] (`nDCG@l`) is known to be counterintuitive:

Copyright © 2011 by Information Processing Society of Japan and
The Instiute of Electronics, Information and Communication Engineers
All rights reserved.

note that `nDCG@1000` in Figure 2 is 1, even though the top ranked document is nonrelevant and the first relevant document is at rank 2. This is because the original nDCG treats a relevant document at rank 1 and one at rank 2 equally [21]. Thus, the recommended version of nDCG is the widely-used "Microsoft version" (`MSnDCG@l`), given by [5]:

$$nDCG@l = \frac{\sum_{r=1}^{l} g(r)/\log(r+1)}{\sum_{r=1}^{l} g^*(r)/\log(r+1)} . \qquad (7)$$

`ntcir_eval` also computes *Ranked-Biased Precision* (RBP) [11], which is a rank-sensitive version of precision:

$$RBP = \frac{1-p}{g_h} \sum_r g(r)p^{r-1} \qquad (8)$$

where $p(\leq 1)$ is a parameter reflecting the persistence of the user.

In addition, `ntcir_eval` can compute a version of *Graded Average Precision* (GAP) [14] if the `-gap` option is used with the `compute` subcommand. GAP is not computed by default as it is more computationally expensive than other metrics. The exact definition of the `ntcir_eval` version of GAP can be found elsewhere [26].

An early version of NTCIREVAL with the `label` and `compute` subcommands has been used at the NTCIR *ACLIA IR4QA* [23], *GeoTime* [8] and *Community QA* [24] tasks.

3.2 Condensed-List Measures

As was discussed in Section 2.3, `ntcir_eval` can compute evaluation metrics after removing all unjudged documents from the original ranked list, i.e. based on a Condensed List (CL) [19]. The resultant metrics are referred to as *CL-measures*. For example, in Figure 2 (in which `-j` is used with the `label` subcommand), `AP` represents not the standard AP but its condensed-list version, or "CL-AP". (CL-AP has also been referred to as *induced AP* [30] and AP′ [19, 21].) CL-measures may be useful if the relevance assessments of the test collection being used are incomplete.

Also, `ntcir_eval` has a "hidden" option for computing CL-measures. Note that Figure 2 uses a `-j` option with the `label` subcommand but not with the `compute` subcommand. If `-j` is used with `compute` as well, `ntcir_eval` also outputs *bpref* [4], a binary-relevance metric specifically designed for evaluation with incomplete relevance assessments, as well as its variants [19]. (At least one L0 document is required in the `rel` file to compute bpref.) However, it has been shown that CL-measures such as CL-AP are more reliable and intuitive than bpref: for example, if `-j` is used with `compute` in Figure 2, then bpref would equal zero even though the ranked list has a relevant document at rank 2. Details can be found elsewhere [19, 21].

3.3 Equivalence-Class Measures

As was also discussed in Section 2.3, `ntcir_eval` can compute evaluation metrics based on Equivalence Classes (ECs). The resultant metrics are referred to as *EC-measures*. EC-measures are useful if some relevant items are interchangeable, e.g. answer strings in factoid QA evaluation [15, 20] and duplicate documents in IR evaluation.

By comparing Figure 1 and Figure 3, it can be observed that EC-based evaluation with `ntcir_eval` is basically the same as the traditional ranked retrieval evaluation. The only differences are:

- The gold standard (`erel`) file has a third field which specifies the ID of an EC.
- As the `-ec` option is used with `label`, redundant items from the same EC are ignored.
- As the `-ec` option is used with `compute`, the number of *ECs* in the `erel` files are treated as the number of relevant items. (Note that `jrel=1` in Figure 3 as there is only one EC, even though there are two relevant items.) Thus, an ideal list is contructed by picking only one of the most relevant items from each EC and then sorting them by the relevance levels. (For Figure 3, the ideal list contains `b` at rank 1 and nothing else.)

Note that a white space is used as the field separator in Figure 3. However, if the IA task to be evaluated involves ranking of strings which may contain white spaces (e.g. answer strings for factoid QA), an alternative field separator should be used in the `erel` files. For example, if the `erel` files use a semicolon as the separator, add `-sep ";"` to the `label` and `compute` subcommands.

4. Diversified Search

This section discusses the metrics for diversified search that `ntcir_eval` supports. Diversified search aims to accomodate different user needs by means of a single "entry-point" result page, when the query is *ambiguous* or *underspecified* [25, 26].

In diversity evaluation, we assume that, for each topic q, one or more *intents* i are available for evaluation in advance, as well as their *likelihoods* $Pr(i|q)$. For example, if the query "apple" has two possible intents, $i_1 =$"Apple the company" and $i_2 =$"apple the fruit," suppose that $Pr(i_1|q) = 0.8$ and $Pr(i_2|q) = 0.2$. (A few methods exit for estimating these probabilities [1, 27].) Moreover, we assume that *per-intent* graded relevance assessments are available: for example, a document about Steve Jobs may be L2-relevant to i_1, but nonrelevant (L0) to i_2; a Wikipedia disambiguation page for the word "apple" may be L1-relevant to both i_1 and i_2.

Given the above premises, a family of metrics called *D-measures* can be computed as follows [26]. Let $g_i(r)$ denote the gain value with respect to intent i for the document at rank r, assigned based on the aforementioned per-intent graded relevance assessments. Then, let the *global gain* of the document at rank r be $GG(r) = \sum_i Pr(i|q)g_i(r)$. Define an ideal ranked list, by sorting all relevant documents by the global gain. Let $GG^*(r)$ denote the global gain at rank r in this ideal list. By replacing $g(r)$ and $g^*(r)$ mentioned in Section 3.1 with $GG(r)$ and $GG^*(r)$, respectively, we can define "D-versions" of Q-measure, nDCG and so on. The assumptions behind D-measures are that intents are mutually exclusive, and that the gain value $g_i(r)$ is proportional to the probability that the document at rank r is relevant to intent i [26]. The intuitive interpretation of D-measures is that we want a system that rank documents that are highly relevant to major intents above those that are marginally relevant to minor intents.

`ntcir_eval` computes D-measures by means of two subcommands called `glabel` and `gcompute`. Unlike the aforementioned `label` subcommand, `glabel` reads a

Copyright © 2011 by Information Processing Society of Japan and
The Instiute of Electronics, Information and Communication Engineers
All rights reserved.

Grelv file, which is a list of documents in descending order of global gain values (i.e. the ideal list). Hence, glabel adds a global gain value (a real number) to each relevant document. For example, for the aforementioned "apple" query, suppose that document a is L2-relevant to intent i_1 and L1-relevant to intent i_2, and that we assign 2 and 1 to L2- and L1-relevant documents for each intent. Then the global gain for this document is $0.8 * 2 + 0.2 * 1 = 1.8$. Using the same system output example.res from Figure 1, we can compute D-measures as shown in Figure 4.

```
% cat example.Grelv
a 1.8
%  cat example.res | ntcir_eval glabel -I example.Grelv
c
b
a 1.8000
% cat example.res | ntcir_eval glabel -I example.Grelv |
ntcir_eval gcompute -I example.Grelv
 # syslen=3 jrel=1 jnonrel=0
 # r1=3 rp=3
 RR=                   0.3333
 O-measure=            0.5833
 P-measure=            0.5833
 P-plus=               0.5833
 AP=                   0.3333
 Q-measure=            0.5833
 NCUrb,P=              0.3333
 NCUrb,BR=             0.5833
 RBP=                  0.0451
 ERR=                  0.2143
 AP@1000=              0.3333
 Q@1000=               0.5833
 nDCG@1000=            0.6309
 MSnDCG@1000=          0.5000
 P@1000=               0.0010
 nERR@1000=            0.3333
 Hit@1000=             1.0000
```

Figure 4: Using ntcir_eval glabel and gcompute.

For example, the D-version of Q-measure ("D-Q") is 0.5833 because only the third document is relevant and its global gain value is 1.8; since the ideal list (example.Grelv) has this document at rank 1, $D\text{-}Q = BR(3) = (1 + 1.8)/(3 + 1.8) = 0.5833$ (See Eq. 3).

ntcit_eval can also compute *intent recall* (a.k.a. *subtopic recall* [31]). This is the number of intents covered by a ranked list divided by the total number of intents. The subcommand irec is used to compute intent recall, as shown in Figure 5.

```
% cat example.Irelv1
a 2
% cat example.Irelv2
a 1
% cat example.res
c
b
a
% ntcir_eval irec example.res example.Irelv1 example.Irelv2
 #intent_num=2
 I-rec@n=              0.0000
 I-rec@1000=           1.0000
```

Figure 5: Using ntcir_eval irec.

Here, the two Irelv files indicate that the *local* gain values for a with respect to the two intents are 2 and 1, respectively. As example.res has a at rank 3, and as this document covers both intents, intent recall (I-rec) at cutoff $l = 1000$ is 1. Whereas, I-rec@n is the intent recall at rank n, where n is the number of intents. In this example, since $n = 2$ and the top two documents are nonrelevant, $I\text{-}rec@n = 0$.

It is recommended that D-measure values be plotted against I-rec values in order to visualise the trade-off between diversity and relevance [25, 26]. However, NTCIREVAL can also combine a D-measure with I-rec to produce a single-value summary metric, called the D♯-measure. This is defined as follows:

$$D\sharp\text{-}measure = \gamma I\text{-}rec + (1 - \gamma) D\text{-}measure\ . \qquad (9)$$

D♯-measures are computed outside ntcir_eval, and the parameter γ can be changed within a shell script included in NTCIREVAL. It is also recommended that the document cutoff l for I-rec and D-measures is chosen so that $l \geq n$. This is to ensure that the maximum possible I-rec value (and therefore D♯-measure value) is 1. The ongoing NTCIR-9 INTENT task‖ is using NTCIREVAL for computing D(♯)-measures.

NTCIREVAL can also compute another set of diversity metrics called *Intent-Aware* (IA) metrics [1]. However, IA metrics have weaknesses in terms of intuitiveness, discriminative power, and in that they do not range fully between 0 and 1 [25, 26].

NTCIREVAL does not compute *α-nDCG*, a relatively widely-used diversity metric [7]. A precise computation of this metric involves an NP-hard problem. α-nDCG can be regarded as an extension of EC-measures in that it penalises retrieval of redundant items (but not as severely as EC-measures do).

5. Summary

This paper introduced NTCIREVAL, a general toolkit for IA evaluation. A shared toolkit like this provides a common ground for IA researchers on which systems can easily be compared and improved. It is hoped that IA researchers carefully examine and choose appropriate evaluation metrics for their purposes, and consider improving the metrics if necessary. Also, it is recommended that researchers use multiple evaluation metrics to examine systems from several *different* angles. "Different" is emphasised here, as some metrics may be redundant when used along with similar and more informative ones [28]. Table 2 provides some recommendations for typical search tasks [17, 21, 22, 26].

Table 2: Recommended evaluation metrics.

search task	recommended metrics
Traditional IR (navigational)	P^+, nERR
Traditional IR (informational)	nDCG (Microsoft version), Q-measure
Diversified IR	D(♯)-nDCG, D(♯)-Q, intent recall

There are many limitations to current approaches to IA evaluation, namely the use of "offline" tools such as NTCIREVAL. Such tools require pre-defined, static gold-standard data, as well as system outputs that are oversimplified compared to what are presented to real IA system users. Thus, for example, these approaches do not capture the real Web search user experiences, whose information needs change dynamically through rich interaction. Note also that the Web itself is dynamically evolving unlike a static test collection with relevance assessments. However, it is hoped that offline evaluation will remain useful for optimising basic system components such as those for ranking items, and will complement more complex and holistic evaluations (e.g. [2]) that tend to be unrepeatable.

‖http://www.thuir.org/intent/ntcir9/

Copyright © 2011 by Information Processing Society of Japan and
The Instiute of Electronics, Information and Communication Engineers
All rights reserved.

References

[1] Agrawal, R. *et al.*: Diversifying Search Results, *ACM WSDM 2009 Proceedings*, pp. 5-14, 2009.

[2] Bailey, P. *et al.*: Evaluating Search Systems Using Result page Context, *IIiX 2010 Proceedings*, 2010.

[3] Broder, A.: A Taxonomy of Web Search, *SIGIR Forum* 36(2), 2002.

[4] Buckley, C. and Voorhees, E. M.: Retrieval Evaluation with Incomplete Information, *ACM SIGIR 2004 Proceedings*, pp. 25-32, 2004.

[5] Burges, C. *et al.*: Learning to Rank Using Gradient Descent, *ICML 2005 Proceedings*, 2005.

[6] Chapelle, O. *et al.*: Expected Reciprocal Rank for Graded Relevance, *ACM CIKM 2009 Proceedings*, pp. 621-630, 2009.

[7] Clarke, C. L.A. *et al.*: Novelty and Diversity in Information Retrieval Evaluation, *ACM SIGIR 2008 Proceedings*, pp. 659-666, 2008.

[8] Gey, F. *et al.*: NTCIR-GeoTime Overview: Evaluating Geographic and Temporal Search, *NTCIR-8 Proceedings*, pp. 147-153, 2010.

[9] Järvelin, K. and Kekäläinen, J.: Cumulated Gain-Based Evaluation of IR Techniques, *ACM Transactions on Information Systems*, Vol. 20, No. 4, pp. 422-446, 2002.

[10] Mitamura, T. *et al.*: Overview of the NTCIR-8 ACLIA Tasks: Advanced Cross-Lingual Information Access, *NTCIR-8 Proceedings*, pp. 15-24, 2010.

[11] Moffat, A. and Jobel, J.: Rank-Biased Precision for Measurement of Retrieval Effectiveness, *ACM TOIS* 27(1), Article 2, 2008.

[12] Robertson, S.: On GMAP: and Other Transformations, *ACM CIKM 2006 Proceedings*, pp. 78-83, 2006.

[13] Robertson, S.: On the History of Evaluation in IR, *Journal of Information Sciences*, 34(4), pp. 439-456, 2008.

[14] Robertson, S., Kanoulas, E. and Yilmaz, E.: Extending Average Precision to Graded Relevance Judgments, *ACM SIGIR 2010 Proceedings*, pp. 603-610, 2010.

[15] Sakai, T.: New Performance Metrics based on Multigrade Relevance: Their Application to Question Answering, *NTCIR-4 Proceedings Open Submission Session*, 2004.

[16] Sakai, T.: Evaluating Evaluation Metrics based on the Bootstrap, *ACM SIGIR 2006 Proceedings*, pp. 525-532, 2006.

[17] Sakai, T.: Bootstrap-Based Comparisons of IR Metrics for Finding One Relevant Document, *AIRS 2006 Proceedings*, LNCS 4182, pp. 374-389, Springer, 2006.

[18] Sakai, T.: For Building Better Retrieval Systems: Trends in Information Retrieval Evaluation based on Graded Relevance (in Japanese), *IPSJ Magazine*, 47(2), pp. 147-158, 2006.

[19] Sakai, T.: Alternatives to Bpref, *ACM SIGIR 2007 Proceedings*, pp. 71-78, 2007.

[20] Sakai, T.: On the Reliability of Factoid Question Answering Evaluation, *ACM TALIP*, 6(1), Article 3, 2007.

[21] Sakai, T. and Kando, N.: On Information Retrieval Metrics Designed for Evaluation with Incomplete Relevance Assessments, *Information Retrieval*, 11(5), pp. 447-470, Springer, 2008.

[22] Sakai, T. and Robertson, S.: Modelling A User Population for Designing Information Retrieval Metrics, *EVIA 2008 Proceedings*, pp. 30-41, 2008.

[23] Sakai, T. *et al.*: Overview of NTCIR-8 ACLIA IR4QA, *NTCIR-8 Proceedings*, pp. 63-93, 2010.

[24] Sakai, T. *et al.*: Using Graded-Relevance Metrics for Evaluating Community QA Answer Selection, *ACM WSDM 2011 Proceedings*, 2011.

[25] Sakai, T.: Challenges in Diversity Evaluation, *ECIR 2011 Workshop on Diversity in Document Retrieval*, 2011.

[26] Sakai, T. and Song, R.: Evaluating Diversified Search Results Using Per-intent Graded Relevance, *ACM SIGIR 2011 Proceedings*, to appear, 2011.

[27] Song, R. *et al.*: Constructing a Test Collection with Multi-Intent Queries, *EVIA 2010 proceedings*, pp. 51-59, 2010.

[28] Webber, W. *et al.* Precision-At-Ten Considered Redundant, *ACM SIGIR 2008 Proceedings*, pp. 695-696, 2008.

[29] White, R. W. *et al.*: Supporting Exploratory Search, *Communications of the ACM* 49(4), 2006.

[30] Yilmaz, E. and Aslam, J. A.: Estimating Average Precision with Incomplete and Imperfect Judgments, *CIKM 2006 Proceedings*, 2006.

[31] Zhai, C., Cohen, W. W. and Lafferty, J.: Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval, *ACM SIGIR 2003 Proceedings*, pp. 10-17, 2003.

Copyright © 2011 by Information Processing Society of Japan and
The Instiute of Electronics, Information and Communication Engineers
All rights reserved.