EPRUM Metrics and INEX 2005

Benjamin Piwowarski

Centre for Web Research, Universidad de Chile
bpiwowar@dcc.uchile.cl

Abstract. Standard Information Retrieval (IR) metrics are not well suited for new paradigms like XML IR in which retrievable information units are document elements. These units are neither predefined nor independent, and the elements returned by IR systems may overlap and contain near misses. Part of the problem stems from the classical hypotheses on the user behaviour that do not take into account the structural or logical context of document elements or the possibility of navigation between retrievable units. The Expected Precision Recall with User Model (EPRUM) metric is based on a more realistic user model which encompasses a large variety of user behaviours. In this paper, we present the EPRUM metric used for evaluating the official submissions of INEX 2005 and detail the settings we used. We do not present the full derivation of the EPRUM metric but we give a thorough example of its computation along with the complete set of formulas needed to compute precision at different recall values. We also discuss the implication of such a metric on several key problems of XML Information Retrieval as the notion of the ideal list and the problem of the overlap.

1 Introduction

This document describes the EPRUM metric in the context of XML Retrieval. EPRUM is a metric that aims at providing a unique and comprehensive framework for the evaluation of XML Retrieval systems[1], by defining a user model which is a two fold extension of the classical user model. First, the evaluated list can be composed of more complex objects than simple links to documents or elements – thus allowing a natural evaluation of the Fetch&Browse task. Second, the user is "allowed" to browse into the structural context of a retrieved element: For example, if a subsection is returned, the user can browse to one of its paragraph or to to its enclosing section. Overlap and near misses are naturally handled, as we define an ideal list of non overlapping elements that can only be seen once by the user (a system is rewarded when an "ideal" element is seen by the user).

EPRUM is an extension of the of precision-recall. As Generalised Recall [3] and Precision-Recall with User Modelling [4], EPRUM is based on a user and relevance probabilistic model that are the basis of the precision and recall derivations. This user model has parameters that can be tuned so that they mimic the "average" user behaviour.

The plan of this paper is as follows. In Section 2, we show how we define precision at a given recall level. This definition is an alternative to the classical definition of precision-recall (PR) and yields the standard result when the user model parameters are set to mimic the usual assumed user behaviour (the list is composed of elements

[1] But not limited to: the relevance model could be used in standard information retrieval and its user model could be reused in passage retrieval, web retrieval, video retrieval, etc.

and the user does not browse from a retrieved element). In Section 3, we describe the different parameters of EPRUM and present the peculiar instantiation of the user model we chose for evaluating INEX 2005 runs. In Section 4, we present the main EPRUM formulas that can be used to evaluate any system run and illustrate them with examples of evaluation for both the Focussed and the Fetch&Browse tasks. Eventually, we discuss the implication of the EPRUM user model for the notion of ideal lists and of overlap in Section 5.

A note about relevance: We distinguish between the relevance of an element (the element contains some relevant material) and the ideality of an element (the fact that the element is the unit the user wants to see, i.e. that it belongs to the "ideal recall base").

2 The EPRUM Metric

EPRUM is an extension of precision-recall: It is based on a definition whose special case is the standard precision-recall as defined in TREC [6]. Precision is defined *as the ratio of the minimum number of ranks that a user has to consult in the list returned by an ideal system and by the evaluated system,* given that the user wants to see a given amount of ideal units. At a given recall level l ($0 < l \leq 1$), precision is defined formally as:

$$\text{Precision}(l) = \mathbb{E}\left[\text{Achievement indicator for a recall } l \times \frac{\text{Minimum number of consulted list items for achieving a recall } l \text{ (over all lists)}}{\text{Minimum number of consulted list items for achieving a recall } l \text{ (system list)}}\right] \quad (1)$$

It is easy to see that this is just an alternative definition of the precision at a given recall level. In classical IR, if a system retrieves $A+B$ documents, where A is the number of relevant documents and B the number of not relevant documents, then an ideal system would achieve the same recall with a list reduced to A documents. The above definition would result in a precision $\frac{A}{A+B}$ which is the exact definition of precision – the ratio of the number of relevant documents to the number of retrieved documents. The achievement indicator is used to set the precision to 0 if the recall level cannot be achieved; this is also the classical definition of precision-recall. This definition relates to the expected search length [1] and to the precision-recall as defined in [5].

The following example illustrates the definition of the EPRUM metric; let the list returned by a system be the following:

where gray nodes are ideal units while white nodes are not ideal. The standard definition of precision would assign a precision of respectively 1, 0.25 and 0 for recalls of 1, 2 and 3 (or more). With the definition we chose, we get the same values – the classical user model is deterministic in standard IR, so we can omit the mathematical expectation for now:

Recall 1. The minimum number of elements the user has to consult, over all possible lists, is 1. The value is the same for the system list and the user was able to see one element. Precision is 1.

Recall 2. The minimum number of elements the user has to consult, over all possible lists, is 2. For the evaluated system, the user will have to consult the list until d - that is, the minimum number of items that he has to consult is 4. Precision is 2/4=0.5.

Recall 3. In this case, the same process would give us a precision of 3/5 (because the user has to consult the whole list), but has the recall cannot be attained by the user, the achievement indicator is 0 and hence precision is also 0.

As shown in this example, this definition of precision-recall gives the same results as the standard definition. The interest of this formulation is that we can define and use more complex user and relevance models, and starting from the same definition, derive a generalisation of precision-recall. It is possible to prove that, using the final formula of EPRUM and setting its parameters so as to mimic the standard user behaviour in "flat" IR, we get exactly the same result as trec_eval [6].

3 What is Needed to Compute EPRUM?

EPRUM can be computed given three different sets of parameters:

1. The probability that a user *considers* an element of the corpus. This probability reflects the fact that the user clicks on a result in the list returned by a the IR system and will eventually have look at the element that is associated with the list item. In the context of XML Retrieval, we have to distinguish two cases: the Fetch&Browse task and the others. In the case of the Focussed task, we suppose that a user will *always* consider an element by clicking on its link in the returned list (this is the classical user model). In the case of the Fetch&Browse task, the user model is more complex and is described latter.
2. The probability that a user *browses* from a considered element to any neighbour element. That is, a user, when considering an element, will most probably navigate around it to its close context (i.e., in an XML document this would be the previous siblings, next siblings, ancestors, etc.). This behaviour is stochastic, that is defined by a probability, since we don't expect *all* the users to behave the same way. The probability of browsing from a considered element x to an element y could be measured, in a user experiment, by the proportion of users that would browse to y after having considered x. An element is *seen* if and only if the user browsed to it from a considered element.
3. The probability that a user finds a unit ideal. This probability is closely related to the concept of quantisation but has a well defined meaning in EPRUM: In a user experiment, its value would be the proportion of users that would find the given unit ideal if they had exactly the same information need.

Unfortunately, we still don't have enough user data to compute even an approximated user model. Nevertheless, it is possible to define simple yet realistic behaviours. In

INEX 2005, we chose user models close to the ones implied by xCG (where only elements overlapping with an ideal unit can be rewarded) which was the official INEX 2005 metric. The underlying motivation was to compare faithfully with EPRUM systems that were optimised for xCG. We defined the following user models:

1. For the consideration,

   Focussed. In the focussed task, we made the hypothesis, like in standard IR, that a user always considers an element pointed by a list item. That is, if the third list item is element x then the user will consult the element x (he will see the content of this element). At a given rank i, the probability of considering an element for the focussed task is either 0 (the element is not in the i first ranks) or 1 (the element is in the first i list elements):

$$\mathrm{P}(x \in C_i) = \begin{cases} 1 & \text{if } x \text{ is within the first } i \text{ elements of the list} \\ 0 & \text{otherwise} \end{cases}$$

   Fetch&Browse. An item in the list is not anymore only one element but it is rather a set of elements from the same document. We view this task as follows: The returned list, as displayed to the user, is a list of documents. When the user clicks on a document link, he is presented a document where system selected elements are highlighted and ordered – imagine that there is button that can focus on each selected element in turn. The user begins by seeing the first ranked element, then the second, etc. for a given article.

   We make the hypothesis that the probability that the user keeps on consulting the list of elements depends on the amount of irrelevant material contained in the previous consulted elements – this is somehow similar to the T2I (tolerance to irrelevance) user model [7]. That is, the probability that the user keeps on going after having consulted an element in the document directly depends on the element overlap with the ideal elements. For an element ranked j within the i^{th} document group, the probability that the user considers it is defined as:

$$\mathrm{P}(x_j \in C_i) = \mathrm{P}(x_{j-1} \in C_i) \times \left(k + (1-k) \times \frac{\text{size of intersection of } x_{j-1} \text{ with ideal elements}}{\text{size of the element}} \right)$$

   where $\mathrm{P}(x_1 \in C_i) = 1$ by definition (the user always consult the first highlighted element).

   The coefficient k is the minimum probability for a user to consider the next element in the list. For INEX 2005, we empirically set k to 0.8. For example, if the three first elements, say of size 10 characters, returned for an article have no intersection with an ideal element, the probability that the user considers the second one is $0.8 + 0.2 \times 0 = 0.8$, that she considers the third one is $0.8 \times (0.8 + 0.2 \times 0) = 0.64$, etc. Note that a run that returns only elements within (or equal to) ideal targets have their probability of considering an element always equal to 1. In this case (and only in this case), the order among elements within the article doesn't change the performance of the system with respect to this instantiation of EPRUM parameters.

2. For the browsing or navigational behaviour, we chose a simple user model – the user, from a considered element, can go up or down. We call this behaviour "hierarchic": The proportion of users that navigate from an element to one of its descendant, or from an element of its ancestor, is equal to the ratio of the sizes of the elements:

$$\mathrm{P}(x \to y) = \begin{cases} \frac{\text{size of } y}{\text{size of } x} & \text{if } y \text{ is an ancestor of } x \\ \frac{\text{size of } y}{\text{size of } x} & \text{if } x \text{ is an ancestor of } y \\ 0 & \text{otherwise} \end{cases}$$

   For instance, 30 % of the users would go from a section of size 10 its enclosing paragraph of size 3. Note that more realistic user models, like the T2I one, could be used. We chose this simple model because submitted runs were optimised for the inex_eval or the XCG metrics which have an implicit user definition which is close to this hierarchic behaviour. Note also that a user always browse from a considered element x to x (it implies that the fact that x is consulted is equivalent to the fact that x is seen by the user) as $\mathrm{P}(x \to x) = 1$.

3. The set of ideal elements was computed using the Kazai's algorithm [2] using the Exh quantisation:

$$q(x) = \begin{cases} 0 & \text{if "too small"} \\ 0 & \text{if exhaustivity is 0} \\ 0.5 & \text{if exhaustivity is 1} \\ 1 & \text{if exhaustivity is 2} \end{cases}$$

   This algorithm was chosen for its simplicity, and because it produces intuitively correct sets of ideal elements. Starting from the original assessments, the process is as follows. For each relevant path (*i.e.* a path from the deepest element with a non zero quantisation to the root of the document), the element with the higher quantisation within that path is added to the ideal set. In order to remove any overlap from the ideal set, elements that contain an ancestor in this set are all removed. The resulting set does not contain overlapping elements and is considered as the set of elements a user would want to see. The probability that a user – among all the user issuing the same query need – is satisfied by an element of this set is set to 0.5 (resp. 1) when its exhaustivity is 1 (resp. 2): Said otherwise, 100% of the users would be satisfied with an highly exhaustive element while only 50% of them would be satisfied with a fairly exhaustive element.

   As there are two possible quantisations (0.5 and 1), we have to "average" (an example is given in the next section) the precision-recall curves with two ideal sets: The first one is composed of all the ideal elements, while the second one is composed of the highly exhaustive ideal elements only.

   Note that when an element is not in this ideal set, it does not mean that a system returning this element will not be rewarded because a user can still browse to the ideal element.

4 EPRUM Formulas and Examples

We present in this section the evaluation of four lists for the Focussed (and SVCAS, VVCAS) and Fetch&Browse tasks. We used a small database where only two (or three) elements are ideal, as illustrated in Fig. 1. We first give the different formulas needed to compute EPRUM given an ideal set of elements $\mathfrak{I}$ and a particular user model instantiation.

Starting from formula (1), the precision at a given recall ℓ (ℓ is the number of ideal units the user wants to see) can be rewritten:

$$\text{Precision}(\ell) = \mathbb{E}\left[\begin{array}{c}\text{Minimum number of consulted list items}\\ \text{for achieving a recall } \ell \text{ (over all lists)}\end{array}\right]\Bigg\}(E1)$$

$$\times\ \mathbb{E}\left[\frac{\begin{array}{c}\text{Achievement indicator}\\ \text{for a recall } \ell\end{array}}{\begin{array}{c}\text{Minimum number of consulted list items}\\ \text{for achieving a recall } \ell \text{ (system list)}\end{array}}\right]\Bigg\}(E2)$$

It can be shown that:

$$(E1) = \sum_{\text{rank } i} i\left(\mathrm{P}(F_i^* \geq r) - \mathrm{P}(F_{i-1}^* \geq r)\right)$$

$$(E2) = \sum_{\text{rank } i} \frac{1}{i}\left(\mathrm{P}(F_i \geq r) - \mathrm{P}(F_{i-1} \geq r)\right)$$

where r is the smallest integer superior or equal to $\ell \times$ (number of ideal elements) and F_i (resp. F_i^*) is the number of ideal elements found by the user after he consulted the i first ranks of the system list (resp. the ideal list). If we consider the classical case, where an ideal element is retrieved or not at each rank, then $\mathrm{P}(F_i \geq r)$ is either 0 or 1. In this case, it is easy to see that the expected value E1 (resp. E2) is the actual value (or inverse value) of the rank where the r^{th} ideal element has been retrieved.

In order to compute the probability $\mathrm{P}(F_i = r)$ needed by formulas E1 and E2, we first need to know the value of the probability $\mathrm{P}(x \in \mathcal{S}_i)$ that an ideal element x is seen after the user has considered ranks 1 to i. As we use the same user model as in [3], we can use a nearly identical formula:

$$\mathrm{P}(x \in \mathcal{S}_i) = 1 - \prod_y (1 - \mathrm{P}(y \in C_i)\mathrm{P}(y \rightarrow x)) \tag{2}$$

where $\mathrm{P}(y \in C_i)$ and $\mathrm{P}(y \rightarrow x)$ are given by the chosen user model instantiation. In INEX 2005, we chose two different instantiations (one for the focussed/VVCAS/SVCAS tasks and the other for the Fetch&Browse task) as described in the preceding section.

Given $\mathrm{P}(x \in \mathcal{S}_i)$ for any ideal element x and any rank i, it is possible [3] to compute $\mathrm{P}(F_i = r)$ and hence $\mathrm{P}(F_i \geq r)$:

$$\mathrm{P}(F_i = r) = \sum_{\substack{A \subseteq \mathfrak{I} \\ |A| = r}} \prod_{x \in A} \mathrm{P}(x \in \mathcal{S}_i) \prod_{x \in \mathfrak{I} \setminus A} \mathrm{P}(x \notin \mathcal{S}_i) \tag{3}$$

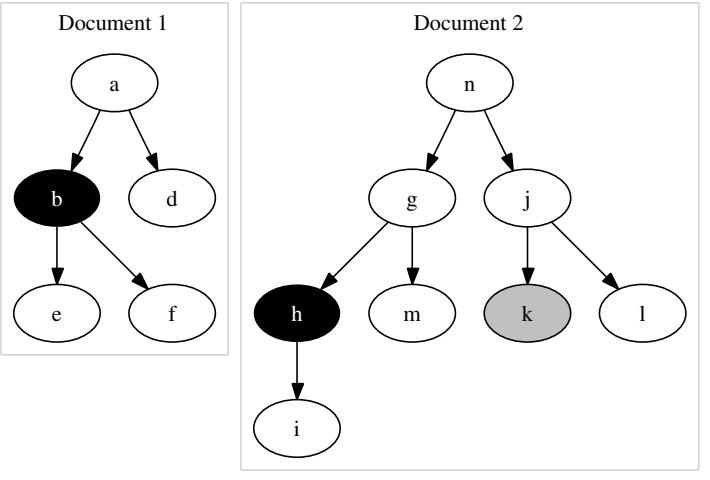

Fig. 1. The example database, composed of two documents and thirteen elements. There are three ideal elements. Two of them are highly exhaustive (b and h, with a black background) for the query while one of them is fairly exhaustive (k). The size of each element is 1 + the size of its children (in an imaginary unit: this could be for instance the number of words divided by 100): In this database, each element directly contains some text and possibly some descendants. The size of e (f, i, m, k or l) is 1, the size of b is 3, the size of a is 5, etc. The probability of navigating from an element to the other being the ratio of sizes, the probability to navigate from f to b is for instance $\frac{1}{3}$.

where $\mathfrak{I}$ is the set of ideal elements and the summation is taken over all the subsets A of cardinality r of the ideal set of element $\mathfrak{I}$. The above formula simply enumerates all the cases where *exactly r* ideal elements are seen by the user. It is possible to compute it in quadratic time with respect to the cardinality of $\mathfrak{I}$, or to approximate it using the normal law.

The above formulas can be used to compute the precision at any recall level. It is also possible to compute precision at a given rank but this is not described in this paper. The last important input, which was until now only evoked, is the ideal list. In the general case, it is not easy to derive an ideal list from an arbitrary user model. It can also be shown that there might not exist a unique ideal list for some user model instances, but rather a set of ideal lists – one for each recall value.

However, in the case of the user models we used in INEX 2005, the ideal lists are quite easy to generate: For the Focussed/VVCAS/SVCAS tasks, it is simply the ideal set elements ordered by exhaustivity value. For the Fetch&Browse task, the ideal list is composed of the documents ordered by decreasing sum of exhaustivity values of the ideal elements they contain. For each document, the ideal list to return is simply the set of ideal elements that that document contains.

In order to illustrate EPRUM behaviour, we now apply the different formulas with the two different user models we chose for INEX 2005.

4.1 Focussed and VVCAS, SVCAS

In order to illustrate the EPRUM metric, we use the following lists for the focussed, VVCAS and SVCAS tasks (all these tasks do not define precisely the target element, so the hierarchic behaviour makes sense):

A List b,h,k: This is the ideal list, composed of the ideal elements - with the "most" ideal first.
B List k, h, b: This is the ideal list, but ordered by increasing order of ideality.
C List f, h, k: The list **A** but with b (first element) replaced by one of its child.
D List h, f, k: The list **C**, swapping the two first elements.

We assume that the probability that the user has seen more than one ideal element *before* beginning to consult the list is 0; that is, $P(F_0 \geq r) = 0$ for $r > 0$. We then distinguish two cases:

1. If the user is satisfied with an element of exhaustivity at least 2 (75 % of the users – there is a justification that for simplicity we don't present in the paper):
   Recall 1 (level 1/2). $(E1)$ is 1; $(E2)$ is resp. 1, $\frac{1}{2}$, $1 \times (\frac{1}{3} - 0) + \frac{1}{2} \times (1 - \frac{1}{3}) = \frac{2}{3}$, and 1 for lists **A**, **B**, **C** and **D**. Precision is 1, $\frac{1}{2}$, $\frac{2}{3}$, and 1.
   Recall 2 (level 1). $(E1)$ is 2; $(E2)$ is resp. $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{2} \times (\frac{1}{3} - 0) = \frac{1}{6}$, and $\frac{1}{6}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, $\frac{2}{3}$, $\frac{1}{3}$ and $\frac{1}{3}$.
2. If the user is satisfied with an element of exhaustivity at least 1 (25 % of the users):
   Recall 1 (level 1/3). $(E1)$ is 1; $(E2)$ is resp. 1, 1, $1 \times (\frac{1}{3} - 0) + \frac{1}{2} \times (1 - \frac{1}{3}) = \frac{2}{3}$, and 1 for lists **A**, **B**, **C** and **D**. Precision is 1, 1, $\frac{2}{3}$, and 1.
   Recall 2 (level 2/3). $(E1)$ is 2; $(E2)$ is resp. $\frac{1}{2}$, $\frac{1}{2}$, $\frac{1}{2} \times (\frac{1}{3} - 0) + \frac{1}{3} \times (1 - \frac{1}{3}) = \frac{7}{18}$, and $\frac{7}{18}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, 1, $\frac{7}{9}$ and $\frac{7}{9}$.
   Recall 3 (level 1). $(E1)$ is 3; $(E2)$ is resp. $\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{3} \times (\frac{1}{3} - 0) = \frac{1}{9}$, and $\frac{1}{9}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, 1, $\frac{1}{3}$ and $\frac{1}{3}$.

There is a way to combine the two sets of precisions that we do not describe here but give an example instead. If a user wants to see more than two third of the ideal elements for list **B**, then for 75 % of the users this means a precision of $\frac{2}{3}$ (only highly exhaustive elements satisfy the user) and for 25 % of them this means a precision of 1: Hence the precision of $.\dot{7}5 \times \frac{5}{6} + .25 \times 1 = .875$. For the same list, if a user wants to see between $\frac{1}{2}$ (excluded) and $\frac{2}{3}$ (included) of the ideal elements, then for 75 % of the users that means seeing 2 ideal elements with a precision $\frac{2}{3}$, and for 25 % of the users that means seeing 2 ideal elements with a precision 1. Hence, a precision of $.75 \times \frac{2}{3} + .25 \times 1 = .75$.

The evaluations order the runs in an order which is appropriate: **A** has the maximum score and **C** is worse than **D** (**D** and **C** have their two first list item swapped, and **D** has a fully ideal element as its top ranked element). List **B**, containing only ideal elements, is overall better than **C** and **D**.

Table 1. Evolution of the different probabilities, with respect to the different lists (A, B, C, D) for the Focussed/VVCAS/SVCAS tasks. The three columns below probability $P(x \in \mathcal{S}_i)$ correspond respectively to the probability that element a, b, or c is seen by the user after rank i. The probability P_2 (resp. P_1) is the probability that the user found at least ... ideal elements after rank i, given that he is only satisfied with elements at least highly (resp. fairly) exhaustive.

	List (b,h,k): **A**								List (k,h,b): **B**							
	$P(x \in \mathcal{S}_i)$			$P_2(F_i \geq)$		$P_1(F_i \geq)$			$P(x \in \mathcal{S}_i)$			$P_2(F_i \geq)$		$P_2(F_i \geq)$		
rank	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**
1	1	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0
2	1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	0
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	List (f,h,k) **C**								List (b,i,k): **D**							
	$P(x \in \mathcal{S}_i)$			$P_2(F_i \geq)$		$P_1(F_i \geq)$			$P(x \in \mathcal{S}_i)$			$P_2(F_i \geq)$		$P_1(F_i \geq)$		
rank	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**
1	$\frac{1}{3}$	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	1	0	0	1	0	1	0	0
2	$\frac{1}{3}$	1	0	1	$\frac{1}{3}$	1	$\frac{1}{3}$	0	1	$\frac{1}{3}$	0	1	$\frac{1}{3}$	1	$\frac{1}{3}$	0
3	$\frac{1}{3}$	1	1	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$	1	1	$\frac{1}{3}$

Table 2. Precision-recall for the Focussed, VVCAS, and SVCAS tasks. The precision for the four lists and four recall intervals are shown in the four last lines. The line "correspondence" shows what is the number of ideal elements the user wants to see if, among the elements in the ideal set, (1) he considers that only elements with an exhaustivity 2 are ideal (2) he considers that elements with an exhaustivity at least 1 are ideal.

recall level	$]0, \frac{1}{3}]$	$]\frac{1}{3}, \frac{1}{2}]$	$]\frac{1}{2}, \frac{2}{3}]$	$]\frac{2}{3}, 1]$
correspondence	1,1	1,2	2,2	2,3
A	1	1	1	1
B	0.63	0.63	.75	0.88
C	0.67	0.69	0.44	0.33
D	1	0.94	0.44	0.33

4.2 Fetch and Browse

For the fetch&browse example, we illustrate EPRUM behaviour using the following lists:

A List D2[h,k] D1[b]: this is the ideal list composed of elements from two documents, D2 and D1. D2 is to be ranked before D1 because it contains more ideal elements.
B List D1[b] D2[h,k]: the ideal list in reverse order.
C List D2[i,k] D1[b]: the first document returned contains a near miss (i); as i is fully specific (contained in an ideal element), the probability that the user consults the next element (k) is 1.
D List D2[g,k] D1[b]: in this list, the first element of the first returned document is an element that overlaps partially with an ideal element; hence, the user will *consider*

the element k of D2 with a probability inferior to 1. Said otherwise, some users only will continue to consult the second highlighted highlighted elements within D2. The probability that the user consults the element k in D2 is $0.8+0.2\times\frac{1}{2}=0.9$ as one half of g overlaps with the ideal element h: At the first rank (document D2) the user sees h with a probability .5, and k with a probability .9. Another thing to note, is that if h and k satisfy the user, then he sees at first rank at least one ideal element with a probability $\frac{1}{2}\times.9+\frac{1}{2}\times.1+\frac{1}{2}\times.9=\frac{1.9}{2}$, the three terms of the sum being the case where (1) the user sees h and k, (2) the user sees k but not h, and (3) the user sees h but not k.

Note also that the only real difference between lists C and D is that the first element is not fully specific (because the same proportion of users will browse from element g to h and from element i to h).

Table 3. Evolution of the different probabilities, with respect to the different lists (A, B, C, D) for the Fetch&Browse task. The three columns below probability $P(x \in \mathcal{S}_i)$ correspond respectively to the probability that element a, b, or c is seen by the user after rank i. The probability P_2 (resp. P_1) is the probability that the user found at least 2 (resp. 1) ideal elements after rank i, given that he is only satisfied with elements at least highly (resp. fairly) exhaustive.

	List D2[h,k] D1[b]: A									List D1[b] D2[h,k]: B								
	$P(x \in \mathcal{S}_i)$			$P_2(F_i \geq)$		$P_1(F_i \geq)$				$P(x \in \mathcal{S}_i)$			$P_2(F_i \geq)$		$P_2(F_i \geq)$		
rank	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**		**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**	
1	0	1	1	1	0	1	1	0		1	0	0	1	0	1	0	0	
2	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	

	List D2[i,k] D1[b]: C									List D2[g,k] D1[f]: D								
	$P(x \in \mathcal{S}_i)$			$P_2(F_i \geq)$		$P_1(F_i \geq)$				$P(x \in \mathcal{S}_i)$			$P_2(F_i \geq)$		$P_1(F_i \geq)$			
rank	**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**		**b**	**h**	**k**	**1**	**2**	**1**	**2**	**3**	
1	0	$\frac{1}{2}$	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$	0		0	$\frac{1}{2}$	.9	$\frac{1}{2}$	0	$\frac{1.9}{2}$	$\frac{.9}{2}$	0	
2	1	$\frac{1}{2}$	1	1	$\frac{1}{2}$	1	1	$\frac{1}{2}$		1	$\frac{1}{2}$	.9	1	$\frac{1}{2}$	1	$\frac{1.9}{2}$	$\frac{.9}{2}$	

Like in the previous Section, we distinguish two cases:

1. If the user is satisfied with an element of exhaustivity at least 2 (75% of the users – there is a justification that we don't present here):
   Recall 1 (level 1/2). $(E1)$ is 1; $(E2)$ is resp. 1, 1, $1\times(\frac{1}{2}-0)+\frac{1}{2}\times(1-\frac{1}{2})=\frac{3}{4}$, and $\frac{3}{4}$ for lists **A**, **B**, **C** and **D**. Precision is 1, 1, $\frac{3}{4}$, and $\frac{3}{4}$.
   Recall 2 (level 1). $(E1)$ is 2; $(E2)$ is resp. $\frac{1}{2}$, $\frac{1}{2}$, $\frac{1}{2}\times(\frac{1}{2}-0)=\frac{1}{4}$, and $\frac{1}{4}$ for lists **A**, **B**, **C** and **D**. Precisions is 1, 1, $\frac{1}{2}$ and $\frac{1}{2}$.
2. If the user is satisfied with an element of exhaustivity at least 1 (25 % of the users):
   Recall 1 (level 1/3). $(E1)$ is 1; $(E2)$ is resp. 1, 1, 1, and $1\times(\frac{1.9}{2}-0)+\frac{1}{2}\times(1-\frac{1.9}{2})=\frac{3.9}{4}$ for lists **A**, **B**, **C** and **D**. Precision is 1, 1, 1, and $\frac{3.9}{4}$.
   Recall 2 (level 2/3). $(E1)$ is 1; $(E2)$ is resp. 1, $\frac{1}{2}$, $1\times(\frac{1}{2}-0)+\frac{1}{2}\times(1-\frac{1}{2})=\frac{3}{4}$, and $1\times(\frac{.9}{2}-0)+\frac{1}{2}\times(\frac{1.9}{2}-\frac{.9}{2})=\frac{2.8}{4}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, $\frac{1}{2}$, $\frac{3}{4}$ and $\frac{2.8}{4}$.

Recall 3 (level 1). $(E1)$ is 2; $(E2)$ is resp. $\frac{1}{2}$, $\frac{1}{2}$, $\frac{1}{2} \times (\frac{1}{2} - 0) = \frac{1}{4}$, and $\frac{1}{2} \times (\frac{.9}{2} - 0) = \frac{.9}{4}$ for lists **A**, **B**, **C** and **D**. Precisions are 1, 1, $\frac{1}{2}$ and $\frac{.9}{2}$.

Using the same technique that in the previous section, we now combine the precisions for these two sets of users.

Table 4. Precision-recall for the list A-D (Fetch&Browse). The precision for the four lists and four recall intervals are shown. The line "correspondence" shows what is the number of ideal elements the user wants to see if, among the elements in the ideal set, (1) he considers that only elements with an exhaustivity 2 are ideal (2) he considers that elements with an exhaustivity at least 1 are ideal.

recall level	$]0, \frac{1}{3}]$	$]\frac{1}{3}, \frac{1}{2}]$	$]\frac{1}{2}, \frac{2}{3}]$	$]\frac{2}{3}, 1]$
correspondence	1,1	1,2	2,2	2,3
A	1	1	1	1
B	1	.88	.88	1
C	.81	.75	.56	.50
D	.81	.74	.55	.49

The evaluations are as expected; list **A** has the maximum score, followed by list **B** (where the two documents were swapped). Then list **C** is superior to **D**, although very close: the difference lies in the fact that a part of the users did not continue to explore the document as the first element (for list **D**) was not fully specific.

5 Implications

The EPRUM metric makes explicit the user behaviour by defining a probabilistic user model. The probabilistic model is important since in XML IR we cannot expect all the users to behave the same way in a given document. This model has two interesting implications in XML IR:

The notion of ideal list. The ideal list is used both by EPRUM and xCG metrics to compute respectively the ideal precision/recall and effort/gain curves. While in xCG the notion of ideal list and ideal set of elements are the same, this is not the case in EPRUM: The ideal set of elements is, as in xCG, the set of elements that would satisfy users with the same query need. Contrarily to xCG, the ideal list of EPRUM *is not always* the set of ideal elements and depends on the specific user model. To illustrate this point, let us take a very peculiar (and not so realistic) user model: A user *always* browses to the first and the last paragraph of a consulted section. If the set ideal elements is reduced to two paragraphs – the first and the last of a section S, then the ideal set is composed of the two paragraphs while the ideal list is reduced to the section S.

A consequence of that separation is that there might exist more than one ideal list, one for any given recall r. Let's take an example to illustrate this point: Let **a**, **b**, and **c** be three elements; **b** and **c** are ideal. The list is composed of elements in the corpus. The probability of navigating from **a** to **b** (or **c**) is 0.9. For a recall 1, an ideal list would

be a simple list restricted to one of the ideal elements, **b** or **c**, with an expected length of 1. For a recall 2, an ideal list would be (**a**,**b**,**c**) because 81 % of the users would see two ideal elements after the first rank, 9 % after the second and 10 % after the third – thus implying an expected search length of 1.29 (instead of 2 with a list composed of **b** and **c**).

Overlap. The preceding example also illustrate another consequence of the EPRUM user model: Overlap might be a good property of the returned list. Let **a** be an ancestor of **b** and **c** in the previous example. As 19% of the users did not see the two elements after having consulted **a** in the list, it is necessary to return the two overlapping elements **b** and **c** so that 100% of the users eventually see all the ideal elements. This situation might arise whenever it is more interesting (in terms of expected search length) to first return one ancestor rather than each of its ideal descendants.

In INEX 2005, the chosen user model implies that there is only one ideal list (or a set of ideal lists but with the same expected search lengths for each recall value) and that the ideal list does not contain overlapping elements. However, it is interesting to underline the fact that choosing more complex user model might imply multiple ideal lists for different recall values and overlap in the ideal list. This would also be the case for the xCG metrics family if they reward near misses that are neither ancestors nor descendants of an ideal element.

6 Discussion

Most metrics used to compare the performance of semi-structured document search engines rely – sometimes implicitly – on a simplistic user behaviour: The user is supposed to consult exclusively the elements of the list returned by the engine. This user model is no more adapted to recent IR tasks like XML. In particular it does not allow considering user ability to navigate between elements, using the list as entry points to the information he seeks.

In this article, we briefly presented the EPRUM metric and showed how it was used in INEX 2005 to evaluate participant submissions. EPRUM is a generalisation of precision-recall that reduces to standard recall-precision if browsing between elements is not allowed and if element relevance is binary. It is also worth noting that EPRUM is somehow a natural extension of xCG (in particular of the new EP/GR metric) where the user and relevance model are explicit and the metric formulas derived from a theoretic user model, thus providing a consistent framework for XML IR evaluation.

Specifying explicitly the user model allows EPRUM to be set so that it measures the "average satisfaction" of users with the same query need. It is possible to set the metric's parameters so that the user model is closer to the real user behaviour in the context of XML IR when user experiments are available. We hope that such data will be available if, as discussed during the INEX 2005 workshop, topics in INEX 2006 are created using an interface that will log all the actions undertaken by the topic creator.

Further work are planned to investigate an instantiation of EPRUM parameters that would use experimental data from the INEX interactive track and to compare more thoroughly xCG and EPRUM. With respect to the latter, we expect to reach a quite high correlation when the EPRUM instantiation is as close as possible to the implied xCG

models: Preliminary experiments have shown that the correlations between average precision of each topic and each system run in INEX 2005 are respectively 0.75, 0.62 and 0.78 for the Pearson, Kendall and Spearman coefficients. In order to compare the two metrics, it would be necessary to analyse the differences in detail.

EvalJ

EPRUM is implemented in the EvalJ software, along with all the other metrics of INEX. It can be downloaded from this URL:http://evalj.sourceforge.net

References

[1] W. S. Cooper. Some inconsistencies and misidentified modelling assumptions in probabilistic information retrieval. In N. J. Belkin, P. Ingwersen, and A. M. Pej, editors, *Proceedings of the 14th ACM SIGIR*, Copenhagen, Danemark, 1992. ACM Press.

[2] G. Kazai and M. Lalmas. Notes on what to measure in inex. In A. Trotman, M. Lalmas, and N. Fuhr, editors, *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*. University of Otago, Univerisity of Glasgow, Information Retrieval Festival, 2005.

[3] B. Piwowarski and P. Gallinari. Expected ratio of relevant units: A measure for structured information retrieval. In N. Fuhr, M. Lalmas, and S. Malik, editors, *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, Dagstuhl, France, Dec. 2003.

[4] B. Piwowarski, P. Gallinari, and G. Dupret. An extension of precision-recall with user modelling (PRUM): Application to XML retrieval. submitted for publication, 2005.

[5] V. V. Raghavan, G. S. Jung, and P. Bollmann. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7(3):205–229, 1989.

[6] E. M. Voorhees. Common evaluation measures. In *The Twelfth Text Retrieval Conference (TREC 2003)*, number SP 500-255, pages 1–13. NIST, 2003.

[7] A. Vries, G. Kazai, and M. Lalmas. Tolerance to irrelevance: A user-effort oriented evaluation of retrieval systems without predefined retrieval unit. In *Proceedings of RIAO (Recherche d'Information Assistée par Ordinateur (Computer Assisted Information Retrieval)), Avignon, France*, Apr. 2004.