Ranking document fragments in XML Retrieval: an empirical study

Huyen-Trang Vu
LIP6, Université Paris 6
8, rue du capitaine Scott
75015 Paris, France
vu@poleia.lip6.fr

Benjamin Piwowarski
LIP6, Université Paris 6
8, rue du capitaine Scott
75015 Paris, France
bpiwowar@poleia.lip6.fr

Patrick Gallinari
LIP6, Université Paris 6
8, rue du capitaine Scott
75015 Paris, France
gallinar@poleia.lip6.fr

ABSTRACT

In the XML retrieval paradigm, document fragments may be returned as answers to a user query. This information being more specific than whole documents may therefore reduce the user effort for finding relevant information. However, since XML documents are composed of nested elements, many of which being possibly relevant to the user information need, retrieval systems must take care of the overlap between returned elements. In this paper, we consider this filtering problem. We first review related work from the IR literature. We then propose a framework for XML retrieval filtering and describe some ideas for this filtering process.

1. INTRODUCTION

With the widespread use of the eXtensible Markup Language (XML) structured document representations formats like e.g. DocBook have become popular or have been adopted as standards. New tools are being developed in order to manage, store and retrieve information from XML documents. In the information retrieval (IR) community, XML Retrieval has been considered as an important extension to standard IR: two SIGIR workshops were held on this subject [4, 2] and INEX (INitiative for the Evaluation of XML Retrieval) is now in its third year.

Within the new paradigm of XML IR, retrieval systems should retrieve the most specific elements for a query, and not the whole document anymore. The underlying idea is to reduce the ratio of irrelevant information the user sees, as in passage retrieval. User satisfaction not only depends on the ratio of irrelevant information but also on the amount of time (s)he has to spend consulting redundant information. In flat document IR, this is related to the problem of novelty: a document can contain information which was already seen by the user. In the XML framework, there is an additional form of redundancy which is related to the logical organization of the document itself: nested elements (e.g. a paragraph and its enclosing section) can be returned when both have been scored as relevant elements. The user will then be presented twice with the same information (the content of the paragraph).

An analysis of INEX 2003 participants' submissions showed that in average 28 % of elements in the ranked list returned by the participants search engines are duplicate, i.e. are either an ancestor or a descendant of an element already in the list (see fig. 1). This form of redundancy is then particularly damageable and the problem of overlap has to be addressed. We consider the subject of this paper. In this paper, we examine different possibilities for eliminating this type of redundant information. We consider this as a filtering task: a ranked list of document element has been retrieved from an XML collection and redundant information has to be reduced. Filtering is not performed by the retrieval engine itself but operates on the retrieved list.

The article is organized as follows. In section 2, we describe related works, some of which being recent works on XML filtering. We give a formal description of the problem in section 3, and present our propositions in section 4. In section 5, we describe some experiments which could allow comparing different propositions.

2. RELATED WORK

Let us first examine how redundancy has been considered in different IR settings like passage retrieval and hypertext IR.

In "flat document" IR the aim is to retrieve documents that fulfill a user information need. It has been argued that it would be useful to present to the user compact lists in which documents with a similar content have been removed [3, 20, 18]. This idea of novelty/redundancy detection has been extended to the case of passage retrieval e.g. at the sentence level ([16]). Novelty detection is difficult to handle while computing the relevance score of a document. All approaches to novelty detection consider as input a ranked list of documents output as relevant by an IR system. From that point, two different approaches have been used in the literature:

Adaptive filtering documents are considered one after the

other, in decreasing order and a binary decision (keep or remove) is taken for each document [20, 16].

Re-ordering new scores are computed for all documents, by considering all documents already seen in the list [3, 18]

In order to detect novelty/redundancy, different techniques have been proposed, ranging from simple ones (based on word occurrences) to elaborate ones (modeling content by language models or by graphsXX??).

XXreecrire Redundancy detection has also been used in the clarification phase of the TREC HARD Track ([1]). Due to limit of time that the user spends to give their relevance judgment on answers, it is crucial for system to select some *representative* snippets in order to get maximal feedback information from user. These selection techniques are called active feedback.

For hypertext IR, it is interesting to retrieve documents which are *entry points* to relevant ones [8, 11]. Simple techniques have been proposed for finding BEP (Best Entry Point)documents. They are similar to those used for adaptive filtering task. A simple one consists in removing a document whenever there is an incoming link from another document ranked higher in the list [8]. For hierarchically (or tree) structured web sites, [11] propose to compare the score of a document to its children (linked documents) scores and a binary decision is taken.

In the context of a structured retrieval task, the seminal work of Chiaramella et al. [5] made use of the notions of specificity and exhaustivity of an element with respect to a given query need. An element is specific if the only topics it discusses are query topics. It is exhaustive if it answers all aspects of the query need. They consider that for being consistent, answers must be completely distinct one from the other. If an element is exhaustive but not specific, then one of its child can be both exhaustive *and* specific. Searching for the most specific and most exhaustive elements is thus a recursive process. This is an interesting idea, however, their approach is purely logical and cannot be used in operational systems.

With XML retrieval systems, only simple strategies have been used for now. Some of them depend on the score value of the document element, while others do not.

Cui et al. [7] proposed to merge overlapped results: if an element in a candidate set formed by the set of elements containing at least one query term, contains another element, then the former is removed. The deepest element is favored with this selection method. Extensions of hypertext filtering were proposed by Lalmas et al. [12, 10]. They proposed to identify Best Entry Points (BEP) in documents with different rules such as the ratio (relevant children)/children of an element, the number of relevant consecutive siblings or the average relevance value of each layer in the document tree.

Crestani et al. [6] propose to rescore ranked elements by computing the expected utility (EU) of an element with respect to the relevance of its parent. This EU value is computed as follows. First, a utility value is assigned to each of the eight following possibilities: the element is (is not) relevant, the parent is (is not) relevant, the element is expected to be shown to the user or not. Then, an expected utility value is computed for the following two cases: the element is presented to the user (EU^+) or not (EU^-). The new score for an element can then be either $\mathtt{EU}^+$, or $\mathtt{EU}^+ - \mathtt{EU}^-$ or $\frac{\mathtt{EU}^+}{\mathtt{EU}^-}$. Preliminary experiments on a small collection shown an improvement in term of mean average precision. However, the values used for the computation of EU are empirical and the metrics used are not fully adapted to XML retrieval evaluation.

At INEX 2003 workshop, participants proposed different methods aiming at overcoming this overlapping problem. [13] suggested rules for merging local retrieved results while [14, 15] used simple filtering of the ranked lists. [17] proposed to select elements using a coverage criterion which is computed with respect to the number of query terms in each element. Evaluating the impact of these different strategies is difficult for now. There are two main reasons for that. First, the metrics used to evaluate XML retrieval systems only reflect very indirectly this overlapping problem. Second, it is difficult to make the distinction between the impact of the retrieval system itself and that of the filtering module.

Experiments with XML IR have already shown that filtering was an important problem. It has not been systematically studied yet and only very basic heuristic attempts have been tested for now. In the remainder of the article, we present a series of ideas which could be used in order to study more precisely the XML filtering problem. XXremove ??In general, there were two kind of approaches: one tries to avoid overlap by filtering out the list before retrieving elements, the other tries to take into account global effects.

3. PROBLEM FORMULATION

We will now describe more formally the XML filtering problem.

3.1 Problem Description

We consider filtering a ranked list of document elements. Each element has an associated score (RSV). The RSV of an element should measure the quality of the element for XML retrieval. In the following, we consider this RSV to be an estimate of the probability that the element is both *specific* and *exhaustive*.

The output of the filtering step will be a ranked filtered list. This list *should* be *consistent*, i.e. should contain as few redundant elements as possible. It should also be *optimal*, i.e. highly relevant elements should have the higher scores.

3.2 Problem Analysis

In standard IR, retrieved documents are ranked by decreasing order of relevance score for the given query (the RSV). The implicit assumption is that the documents are *independent* one from the other, this ranking principle does not hold in cases where there is a large number of potentially highly redundant relevant answers [3]. For XML retrieval, there is the additional problem of nested elements. In the remainder of this section, we describe the properties an XML

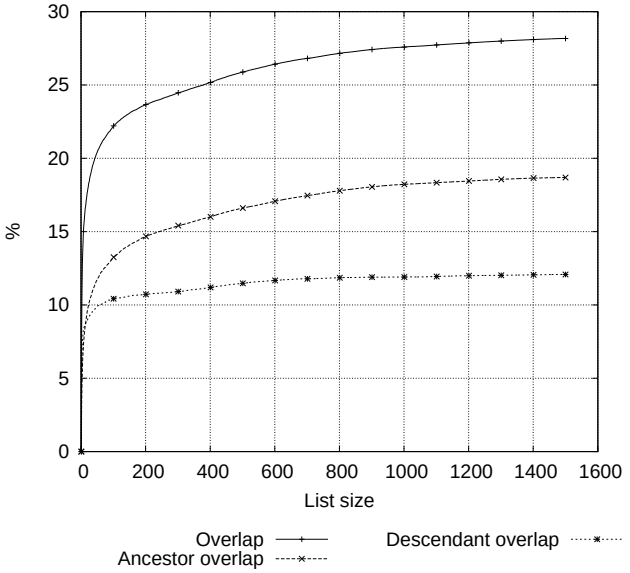

Figure 1: Overlap (INEX'03 submissions)

document element should fulfill in order to be truly relevant to an user. We also make some assumptions which are necessary for practical implementations.

Assumptions

We suppose that the answers are ranked linearly since we believe that despite the development of advanced visualization tools, users will still make a preeminent use of simple ranked lists.

We will suppose that each retrieved element is a meaningful information unit which is somehow self-contained. This implies that we will not consider the context of the element but focus on the structural relationships of an element.

We will also suppose that the user fully reads each answer - which is reasonable because of the above assumption. A consequence of this assumption is that any descendant of an element should be removed from the list since the user has already seen it.

The user can stop after consulting a certain number of answers. Highly relevant elements should therefore be ranked first in the list: this is related to the *optimality* criterion described later on.

Desiderata for answers

Answers are required to be *consistent*: users may be discouraged from viewing further elements if they are presented redundant information. They could also feel disoriented if they are directed to nested parts of the same document.

We consider that the filtering process must be *simple*: despite its importance with regard to the user effort, an XML retrieval system is a complex system and filtering must not take too much time.

The filtered list should be *optimal*: highly relevant elements should be ranked first. This point is important since filtering may remove some nested elements with a higher RSV. For example, consider the case of a section with a RSV of 0.8 and its paragraph with a RSV of 0.6. If for some reason the section is not present in the retrieved list, the paragraph could be lost for the user as it has a smaller score. Since this paragraph "represents" its section, it should be ranked higher. While it is easy to check if an element is enclosed in another, it is not simple to determine an optimal policy. It should be a compromise between the RSV, the structural relationships with other elements and the user preferences.

3.3 Processing Procedure

A filtering process can be decomposed into three distinct optional phases: (1) initial rescoring, (2) filtering and (3) final rescoring. Phase (1) will allow to include some information about the relations between the document elements. Phase (2) is a purely logical process: a binary decision (keep or remove) is taken for each element. Phase 2 ensures the list is *consistent*. The purpose of the last phase (3) is to ensure the optimality of the list. Existing attempts to filtering either performed phase (1) or (2). None of these approaches did both. The optimality problem -phase (3)- has not been considered for now.

4. PROPOSITIONS

In this section, we make concrete propositions that we plan to experiment in a near future. We follow the different phases which have been described before.

4.1 Re-scoring

If the RSV is to be considered as the probability that an element is both exhaustive and specific, it is necessary to transform the score. The motivation here is to make some preprocessing before we filter out the list. This phase is thus tightly coupled with the next one, so one should care about the filtering process before one selects the rescoring method. In some cases, it is also possible to skip this step.

4.1.1 Utility theory

The idea is to use the EU as defined in [6] whenever the underlying model outputs RSV that are probabilities. We plan to do further experiments in order to fully investigate this approach: in particular, we plan to try machine learning techniques to learn the utility values for the fact of showing an element (or not), knowing that the element is relevant (or not) and that its parent is relevant (or not). We denote this approach **RS1**.

One of our model is based on Bayesian Networks [15]. In this model, an element can be in three different and distinct states: not relevant, too big (exhaustive but not really specific) or exact (exhaustive and specific). For each element, we thus have two values (the third one is linked to the two previous) and not one. The extra information brought by the "too big" state could be used to transform our scores with more expressiveness than with a single score. We plan to use the EU methodology to transform the probabilities outputed by the Bayesian Network model (**RS2**).

4.1.2 Redundancy rescoring

It is also possible to consider the list order while computing the new scores. The general idea is to move elements one by one from a given set (the list of retrieved elements) to another (the set of rescored elements). Each selected element maximise a criterion which is a balance between relevance and redundance: the value of the criterion for a selected element is its new score.

Let us consider the sets E_t of n elements to rescore and the set E'_t of rescored elements at a given time t. Note that at the begining of the process ($t = 0$) $E'_0 = \emptyset$ and that at the end of the process ($t = n$) $E_n = \emptyset$.

We have to suppose that the *relevance* of an element to a given query is independent from the *redundance* of this element with respect to a given set of elements. At given time t, the RSV value of an element e in E_t is:

$$RSV_t^{(1)}(e) = \lambda RSV(e,q) - (1-\lambda) \max_{e' \in E'_t} Red(e,e')$$

where $\lambda \in [0,1]$ is a parameter, $RSV(e,q)$ the score of the retrieval system for element e with respect to the query q and $Red(e,e')$ is the redundancy of elements e with respect to e'. The higher is the latter value, the more redundant the element e is.

The element e^* with the highest value $RSV_t^{(1)}$ is selected: $E'_{t+1} = E'_t \cup \{e^*\}$ and $E_{t+1} = E_t \setminus \{e^*\}$.

The process is then reiterated until the set E_t is empty (**RS3**).

Zhai et al. pointed out that the linear combination is only meaningful when similarity measures $RSV()$ and $Red()$ are in the same scale [19]. It is unfortunately not the case here why?. We therefore use another criterion as defined in [19] which is more intuitive (**RS4**)

$$RSV_t^{(2)}(e) = p(Rel|e,q)(p(Red|e,E'_t) - \rho)$$

where ρ is a parameter that represents the cost ratio of user seeing an irrelevant element and of user seeing a relevant but redundant element, $\rho \geq 1$.

4.2 Filtering

In this section, we present filtering methods. We distinguish two different kind of filtering. The first one is related to adaptive filtering, while the other one is based on a recursive processing of the document tree. In the remainder of this section, we are interested in both strict (no overlap) and tolerant (some overlap) filtering.

4.2.1 List filtering

In this section, we present filtering methods which consider elements in the list one by one, begining with the first one which have the highest RSV. Let us denote $E = (e_1, ..., e_n)$ the ordered list of n elements which is the output of the retrieval system and FL_i the filtered list after i iterations of this process. The latter list can thus contain up to i elements and $RSV(e_i) \geq RSV(e_{i+1})$. The filtering process ends when FL_n is known.

We will also denote by $anc(e)$ (resp. $desc(e)$) the set of ancestors (resp. descendants) of an element e in the document tree.

The rules we plan to experiment are as follows:

Rule F1 (no descendant). With this rule, an element is removed whenever its ancestor is already in the filtered list. This is related to the assumption we made about the user behaviour: when (s)he consults the list, (s)he reads elements entirely. (S)he thus have already been reading the content of the element if its ancestor is in the list. Formally, if $anc(e_i) \in FL_{i-1}$ then the element is not added to the list.

Rule F2 (no ancestor). This rule is symmetric to rule R1: an element will not presented to the user if it one of its descendant is already in the list. For example, if a section contains only one paragraph, then if we present the paragraph to the user, we don't want to present the section. Formally, if $desc(e_i) \in FL_{i-1}$ the element is not added to the list. While this rule can be useful to remove redundant information, we believe that this rule is too harsh when the element already in the list is really small compared to the element which is filtered: consider for example the case of a document and a small passage in italics.

Rule F3 (some ancestors) We want to relax the rule R2, in order to allow some overlap in some cases. We have first to define the redundancy of e_i with respect to elements of FL_{i-1}. If the redundancy value is sufficiently high, then we filter out the element. The redundancy value of e_i wrt FL_{i-1} is estimated as:

$$Red(e_i, FL_{i-1}) = \max_{e \in FL_{i-1}} Red(e_i, e) \qquad (1)$$

where redundance of an element e_i with respect to e is proportional to the effort need to read the former given the latter:

$$Red(e_i, e) = f\left(\frac{length(e_i)}{length(e)}\right) \qquad (2)$$

where $length()$ is the length measured in an appropriate unit (number of words, number of caracters, etc.) If $Red(e_i, FL_{i-1}) < Red_{Threshold}$ then the element filtered out. Here, $Red_{Threshold}$ is a parameter which can be fixed empirically or learnt from data.

4.2.2 Filtering on the document tree

In the previous section, we have shown how it is possible to avoid some overlap, if not all, using an approach which is related to adaptive filtering. It is also possible to process a document after another, and to select a set of elements within that document for which there is no overlap.

For each node, starting with the root of the document tree, we take a binary decision: either we keep this element (and we do not consider its children) or we return at least one of its descendants by applying the same rule to each of its children. This algorithm guarantees the non-overlapping of selected elements as the recursive process is stopped whenever we return an element.

The criterion used to take the binary decision can be simple. For each element e in the tree, e is not added in the list if:

Rule F4 if the ratio of relevant elements among its children is superior to a given threshold, or

Rule F5 if the average score of relevant elements is superior to a given threshold.

These criterion are simple enough to be experimented. However, it is possible to do better. Our idea is to use an existing metric and try to optimise the list of returned elements according to this measure. We choose the Expected Ratio of Relevant Units which is a generalisation of standard recall in the context of XML retrieval. We thus denote GR this measure. We then suppose that the probability that an element is consulted by the user is inversely proportional to the score of this element:

$$P(e \text{ is seen}) \propto RSV(e, q)$$

This probability is used when computing the measure GR. We can then compute the expected value of GR if only the parent is in the list and compare it to the expected value of GR if only the children are in the list. If the latter is superior to the former, the element is discarded and the recursive process is iterated over its children (**F6**).

4.3 Final scoring

For the final score, we can apply two different strategies. The first one is to keep the score (**FS1**), the other one is to get the maximum RSV of elements which were removed (**FS2**).

5. PROPOSED EXPERIMENTS

Some of these methods were already used in experiments [14, 15]. As stated before, it is difficult to know if a given filtering is performing well as the results are not independent of the retrieval system performance. In order to examine the respective capabilities of filtering methods, we need to know how the retrieval system is working.

However, this is a rather difficult task. We thus propose to simplify the preliminary experiments: as we want more control over experiments, we simply do not use any retrieval system. We rather define typical systems by computing scores based upon the assessed relevance of these elements. The INEX scale [9] used to assess elements with respect to a given query has two dimensions: the exhaustivity and the specificity. We thus need to transform an assessment into a score. For the sake of clarity, we will not expose here the planned transformation.

In order to simulate the behaviour of a search engine, we will try to add some noise. As this noise should be related to real noise, we will compute for a sample of retrieval systems the distribution of scores with respect to a given assessment. The deterministic transformation of an assessment a into a RSV will then be followed by a random sampling with respect to the probability distribution computed for this assessment.

As we want extensive experiments, we will try every compatible combination of rules; parameters will be either fixed empirically or learnt from data.

6. CONCLUSION

In this article, we presented our planned work on XML filtering. We first motivated the XML filtering: it reduces the user effort to access relevant information as redundant elements are removed from the list presented to the user. Experiments on the INEX'03 dataset have shown that a high percentage of elements are redundant.

We made several assumptions on the user behaviour and described what are our requirements for XML filtering. We have shown that the filtering is a processing procedure which is composed of three phases: re-scoring, filtering and final scoring. For each of these phases, we proposed different techniques which we plan to experiment. The experiments will first investigate the effect of the different filtering components in an "ideal" environment, to clearly distinguish what the filtering from the scoring process.

7. REFERENCES

[1] J. Allan. HARD Track Overview in TREC 2003 High Accuracy Retrieval from Documents. In *Notebook Proceedings of TREC 2003*, 2003.

[2] R. Baeza-Yates, N. Fuhr, and Y. S. Maarek, editors. *ACM SIGIR 2002 Workshop on XML*, Aug. 2002.

[3] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual Int. ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM Press, 1998.

[4] D. Carmel, Y. Maarek, and A. Soffer, editors. *ACM SIGIR 2000 Workshop on XML*, July 2000.

[5] Y. Chiaramella, P. Mulhem, and F. Fourel. A Model for Multimedia Information Retrieval. Technical report, IMAG, Grenoble, France, July 1996.

[6] F. Crestani, L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. Ranking Structured Documents Using Utility Theory in the Bayesian Network Retrieval Model. In M. A. Nascimento, E. S. de Moura, and A. L. Oliveira, editors, *SPIRE 2003*, volume 2857 of *Lecture Notes in Computer Science*, pages 168 – 182, Brazil, Oct. 2003. Springer-Verlag Heidelberg.

[7] H. Cui, J.-R. Wen, and J.-R. Chua. Hierarchical Indexing and Flexible Element Retrieval for Structured Document. In *Advances in Information Retrieval, 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14-16, 2003, Proceedings*, volume 2633 of *Lecture Notes in Computer Science*, pages 73–87, Pisa, Italy, Apr. 2003. Springer.

[8] M. E. Frisse. Searching for information in a hypertext medical handbook. *Communications of the ACM*, 31(7):880–886, 1988.

[9] N. Fuhr, S. Malik, and M. Lalmas. Overview of the initiative for the evaluation of xml retrieval (inex) 2003. In *Proceedings of the Second INEX Workshop*, March 2004.

[10] G. Kazai, M. Lalmas, and T. Rölleke. Focussed Structured Document Retrieval. In *9th Int. Symposium on String Processing and Information Retrieval SPIRE'02*, Lisbon, Portugal, Sept. 2002.

[11] M. Lalmas and E. Moutogianni. A dempster-shafer indexing for the focussed retrieval of a hierarchically structured document space: Implementation and experiments on a web museum collection. In *6th RIAO Conference, Content-Based Multimedia Information Access*, Paris, France, Apr. 2000.

[12] M. Lalmas and J. Reid. Automatic Identification of Best Entry Points for Focussed Structured Document Retrieval. In *CIKM Conference on Information and Knowledge Management*, New Orleans, Louisiana, USA, Nov. 2003. Poster.

[13] Y. Mass and M. Mandelbrod. Retrieving the most relevant XML Components. In N. Fuhr, M. Lalmas, and S. Malik, editors, *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, Schloss Dagstuhl, Germany, Dec. 2003.

[14] P. Ogilvie and J. Callan. Using Language Models for flat text queries in XML 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, Germany, Dec. 2003.

[15] B. Piwowarski, H.-T. Vu, and P. Gallinari. Bayesian Networks and INEX'03. In N. Fuhr, M. Lalmas, and S. Malik, editors, *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, Dagstuhl, Germany, Dec. 2003.

[16] I. Soboroff and D. Harman. Overview of the TREC 2003 Novelty Track. In *Notebook Proceedings of TREC 2003*, 2003.

[17] A. Trotman and R. A. O'Keefe. Identifying and Ranking Relevant Document Elements. In N. Fuhr, M. Lalmas, and S. Malik, editors, *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, Dagstuhl, Germany, Dec. 2003.

[18] C. Zhai. *Risk Minimization and Language Modeling in Text Retrieval*. PhD thesis, Carnegie Mellon University, July 2002.

[19] C. Zhai, W. W. Cohen, and J. Lafferty. Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, ACM Press, New York, NY, USA, 2003.

[20] Y. Zhang, J. P. Callan, and T. Minka. Novelty and redundancy detection in Adaptive Filtering. In *Proceedings of the 25th annual Int. ACM SIGIR conference on Research and development in information retrieval*, pages 81–88. ACM Press, 2002.