Bringing Undergraduate Students Closer to a Real-World Information Retrieval Setting: Methodology and Resources

Julián Urbano, Mónica Marrero, Diego Martín and Jorge Morato
University Carlos III of Madrid
Department of Computer Science
Leganés, Madrid, Spain
{ jurbano, mmarrero, dmandres, jmorato }@inf.uc3m.es

ABSTRACT
We describe a pilot experiment to update the program of an Information Retrieval course for Computer Science undergraduates. We have engaged the students in the development of a search engine from scratch, and they have been involved in the elaboration, also from scratch, of a complete test collection to evaluate their systems. With this methodology they get a whole vision of the Information Retrieval process as they would find it in a real-world setting, and their direct involvement in the evaluation makes them realize the importance of these laboratory experiments in Computer Science. We show that this methodology is indeed reliable and feasible, and so we plan on improving and keep using it in the next years, leading to a public repository of resources for Information Retrieval courses.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *information systems education*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *performance evaluation (efficiency and effectiveness).*

General Terms
Management, Performance, Experimentation.

Keywords
Information Retrieval, Education, Evaluation, Experiment.

1. INTRODUCTION
Information Retrieval (IR) is the discipline that studies the automatic search for documents and the information they contain in an effective and efficient manner. According to a recent market study, between 2009 and 2020 the amount of digital information in the world will grow by a factor of 44, and yet the staffing and investment to manage it will grow by a factor of just 1.4 [1]. Dealing with this rate difference is a real challenge, and the need for Computer Science (CS) experts with adequate IR knowledge is a clear necessity to alleviate the problem of managing the ever-growing information surrounding us. However, IR courses are not always part of the core program in Computer Science majors [2].

Many decisions in the Computer Science industry are taken on the basis of mere experience and bias towards or against certain technologies. Thus, it is imperative for CS professionals to have sufficient background on basic experimental methods and training on critical analysis of experimental studies [3]. There are very few courses that can really show CS students how to run computer-related laboratory experiments and analyze their validity. Information Retrieval courses are very suitable for this purpose because it is a highly experimental discipline with a major focus on the evaluation of search engine effectiveness [4]. However, the experimental aspect of IR is not that common in IR courses [2].

In the University Carlos III of Madrid, the course on Information Retrieval is an elective course for senior CS undergraduates, where we discuss several aspects of IR with a special focus on the Web. In the Spring 2010 edition we incorporated many changes to assess the feasibility of a much more experimental-oriented program to resemble, as closely as possible, a real-world scenario where IR techniques are needed to manage relatively large amounts of digital information. This last year we have had 32 students and 5 faculty members available for the experiment, 2 of which are the course instructors. We developed with the students a brand new test collection of web documents, and a meta-analysis with well-established techniques shows that using such collections is indeed trustworthy for an IR course. In this paper we show how we adapted the methodology followed in TREC (Text Retrieval Conference) [5] and the results obtained with it.

The rest of the paper is organized as follows. Section 2 briefly revises other IR courses taught and analyzes critical factors missed. Next, we describe how we adapted our course to cover those factors, and in Section 4 we show that the collection created by students is indeed reliable for an IR course. Section 5 finishes with conclusions and lines for future work.

2. CURRENT IR COURSES
The development of IR systems from scratch or based on existing technology is a widely used method to teach IR in Computer Science. Some promote the use of frameworks or tools that can be customized and extended by students. The trouble in using these tools is variable, and some authors consider that these frameworks should be simple enough to use, for the students to focus on the course concepts rather than struggle with their intricacies [6]. For instance, IR Base [7] allows the integration of components, documentation and services so that prototypes can be quickly developed for research and educational purposes. Other free tools to build search engines are Alkaline, Greenstone and SpidersRUs, recently compared by students [8]. It is also frequent to find open source tools such as Lucene and Lemur for educational purposes.

But it is also possible to guide students through the development of an IR system from scratch. This approach is only possible in clearly technology-oriented majors, as it is necessary to have a good deal of experience on software development, algorithms, data structures, database management, etc. This need, obviously, has to be taken into account to evaluate the difficulty of the course assignments. In return, students get a much wider view of the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ITiCSE'11, June 27–29, 2011, Darmstadt, Germany.
Copyright 2011 ACM 978-1-4503-0697-3/11/06...$10.00.

© ACM, 2011. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution.
The definitive version was published in the Proceedings of the 16th ACM ITiCSE Conference, 2011. ISBN 978-1-4503-0697-3.
http://doi.acm.org/10.1145/1999747.1999829

techniques and processes explained in the lectures. Examples of this approach can be found in the "Build your search engine in 90 days" method [9] and others applied to Digital Libraries [10].

As to the evaluation of IR systems, the IEEE/ACM Computer Curricula establishes the importance of the Scientific Method in CS, and the necessity for students to learn it [3]. Some tools such as ONTAP [11] have been used to evaluate query performance in terms of classical precision and recall measures, but knowledge concerning evaluation methods is hardly found in IR curricula, let alone the design and analysis of experiments [2]. Students usually learn how to build an IR system, but they do not always learn how to evaluate it. Evaluation laboratory experiments are very useful in Computer Science to verify and tune tools being developed, and even in less technical majors like Library Science (LS) it is essential to choose the adequate tool on the basis of usability, effectiveness and efficiency. It is also interesting from the point of view of research, as these experiments tackle some issues widely studied by the IR research community and yet to be solved.

Furthermore, the solution of problems in CS demands capabilities with data analysis and experimentation, but for them to be meaningful they must be carried out under real-world conditions [12]. In the case of IR it is not always possible to freely access representative document collections and distribute them among the students. Some tools incorporate document collections and evaluation components. However, these resources are very limited and it is difficult to find diversity in terms of domain and documental types. It is therefore normal in IR courses to stick to the same test collection for several years.

Therefore, students should be able to use existing tools as long as they do not lose sight of the main purpose of their assignment; or develop their own as long as they are guided not to focus just on the implementation. Also, they should be involved in the evaluation of their systems to appreciate and understand the problems it bears. Finally, teaching resources are scarce in IR, and so it would be desirable that all resources elaborated by the students be somehow reused in the following years.

3. NEW TEACHING METHODOLOGY

3.1 Search Engine

The main lab assignment in our course consists in developing, from scratch, a complete IR system for web pages using C#. These systems are developed in 3 modules to hand in separately: a naïve retrieval system implementing some basic retrieval model; an extension to include query expansion; and the adaptation to identify named entities such as persons and localizations. To facilitate the development of their systems, we provided the students with the skeleton of a web crawler [13] and a framework to model the IR process their systems undergo, as well as to evaluate their effectiveness with the test collection. The development of the actual IR techniques has been guided by the course instructors, but no source code was provided in this case.

3.2 Test Collection

A test collection for Information Retrieval evaluation is made up of three basic components: a document collection, a set of information needs (usually called topics), and the relevance judgments (usually assessed by humans) telling what documents are relevant to the topics [4]. The systems to evaluate are run for each topic and return documents in the collection deemed relevant to the topic. Then, some effectiveness measures are used to assess, according to the relevance judgments, how well the systems actually answered the information needs.

The TREC conferences, organized by NIST, have traditionally followed a well-studied methodology to evaluate IR systems. In a typical TREC ad-hoc setting, the document collection is selected first, depending on the specific purpose and characteristics of the task [5]. Then, each relevance assessor available comes up with a set of candidate topics and estimates the difficulty and number of relevant documents with the help of a search engine. Out of all candidate topics, usually 50 of them are selected for the test collection. Both the document collection and the topic set are made public for the participants, who run their systems and submit a ranked list with the first 1,000 documents per topic. Then, NIST forms pools of documents for each topic, taking the first 100 documents (the pool depth) from each participating run. Depending on the particular topic complexity, the pool sizes vary significantly, as for some topics the systems tend to return the same results (leading to small pool sizes) and for others they differ greatly (leading to larger pools). The documents in these pools are the ones judged for relevance, and documents not in the pools are considered to be not relevant.

In our case, the methodology to create the test collection has to be different. First of all, the document collection cannot be as big as TREC's because undergraduate students do not have the appropriate level to handle that much information efficiently. Indexing and storing such collections requires a level of knowledge and expertise that these students do not have yet, restricting the size of the collection they can handle without much trouble. Because of this constraint, the election of topics has to be careful. We decided that all topics should have a common theme so that the document collection would not have to be too heterogeneous and hence be too large. We opted for all topics to be related to computing, as it sure is a theme attractive to Computer Science students. Thus, the test collection depends on the topics and not the other way around as usual.

The problem at this point is how to create the document collection assuring that at least some relevant material is included for every topic. We decided to issue queries to Google just as if we were trying to answer the information needs and find as much relevant material as possible (manually considering term proximity, query expansion, etc.). Doing so, we could discard topics apparently too difficult or for which there appeared to be no clearly relevant web pages. We narrowed down the topic set to 20 computing-related topics (plus 2 noisy topics, see below). Using a focused crawler [13], similar to the one developed by the students, we downloaded all results that Google returned for each topic. The union of these web pages was our complete document collection (see Table 1).

Now that we had documents and topics, we needed relevance judgments. There is another glaring difference here for us: it would not be safe to have the students judge documents once they have submitted their results, as some might try cheating and judge relevant all documents retrieved by their system. Therefore, we had to come up with a solution to create a reliable pool of documents and have the students judge them *before* they even start the development of their systems. We decided to create pools from the results of well-known and freely available IR tools such as Lemur and Lucene. These systems implement most of the techniques we regularly teach in our course, so it is fair to expect the student systems to retrieve similar documents. Thus, we configured 8 Lemur and 4 Lucene instances with different IR techniques (call these the pooling systems) and ran them with the documents and topics we developed. The pools formed with these systems are the ones the students judged and the ones they actually used for their systems (call this the biased collection).

Table 1. Summary of topics: documents contributed to the complete and biased collections, pool size and depth, and assessor agreement, when possible. * for topics judged by one faculty member, ** judged by two faculty members. † for the noisy topics.

Topic	Downloaded	Pool size	Pool depth	Kappa	Precision	Recall
001	328	100	28	0.362	0.811	0.545
002	417	100	26	0.243	0.233	1.000
003	616	100	30	0.517	0.906	0.829
004	220	102	25	0.470	0.769	0.638
005*	417	101	17	-	-	-
006**	768	102	19	0.468	0.622	0.821
007	547	100	25	0.456	0.889	0.429
008	729	100	23	0.096	0.818	0.164
009	374	100	37	0.625	1.000	0.550
010	609	101	26	0.217	1.000	0.111
011	218	100	56	0.192	0.250	0.500
012*	338	100	19	-	-	-
013	384	100	21	0.333	0.269	1.000
014	247	100	58	0.342	0.595	0.556
015	435	102	34	0.624	0.810	0.791
016	417	103	28	0.433	0.526	0.741
017	516	101	23	0.574	0.500	0.571
018	474	101	20	0.735	0.895	0.773
019*	488	100	15	-	-	-
020	459	105	20	0.395	0.278	0.833
021†	79	-	-	-	-	-
022†	689	-	-	-	-	-
Average	444	100.90	27.5	0.417	0.657	0.638
Total	9,769	1,967	-	-	-	-

But there is another difference with TREC here: if we took the first *k* documents (the pool depth) from the 12 pooling systems, the pool size could be very big for certain topics. Consider for instance topic 019, where each pooling system contributed only 15 documents to the pool. If we had taken around 27 documents, as the average resulted, the pool for topic 019 would have probably had around 200 documents to judge. If some students were assigned a pool this large, they could just stop judging or judge carelessly once they think they have done enough work compared to their classmates. To prevent this situation, we decided to pool documents until the pool had a minimum size of 100, making each topic's pool to have a different depth.

To measure the quality of the judgments made by students, we decided to include in those (at least) 100 documents 10 noisy documents not related to any topic, so that we could check afterwards whether students actually judged them non-relevant or not (we obtained these documents by issuing two queries to Google explicitly excluding the terms in the other 20 topics). Also, to be sure that the pools had some relevant documents, we added the first 10 results given by Google for each topic, as we checked before, when selecting topics, that some relevant material was found by Google. Therefore, all pools have 10 noisy documents, the first 10 documents retrieved by Google, and documents retrieved by the pooling systems up to a minimum of 100 documents altogether.

Documents for topic 006 were judged by two faculty members, topic 005 was judged by one faculty member, and topics 012 and 019 were judged by the two course instructors (one topic each). The other 16 topics were judged by the 32 students, two students per topic. Note that the two noisy topics needed not be judged. All documents were judged on a 3-point relevance scale (not relevant, somewhat relevant and highly relevant). Of all the 326 judgments on noisy documents, only once did a student judge the document as relevant. Therefore, we decided to trust the relevance judgments to some degree.

As can be seen in Table 1, the complete collection has 9,769 documents (735 MB) and 20 topics (plus the two noisy topics: 021 and 022). The biased collection (made up just by the pooled documents) has a total of 1,967 documents (161 MB). The pool sizes ranged from 100 to 105, so all students judged the same amount of documents (it took them about 2 hours to complete the task). However, the pool depths varied considerably, ranging from 15 (topic 019) to 58 (topic 014). This shows that the pooling systems tended to agree much more for some topics than others, which is to be expected considering the diversity in the topics purposes and difficulty, which agrees with current IR practice.

3.3 Search Engine Evaluation

While the students had a previous, much smaller collection to test their systems before submission, the evaluation of each of their three modules has been performed by the course instructors using the very test collection elaborated with the students. This evaluation followed well accepted criteria and effectiveness measures traditionally used in TREC [5]. In particular, we evaluated their effectiveness with NDCG (Normalized Discounted Cumulative Gain) [14]. This measure allows us to easily assess not only if the systems retrieve relevant material, but whether it is ranked properly, that is, with the most relevant documents before the less relevant ones. The measure ranges from 0 (no relevant information is retrieved at all) to 1 (all relevant information in the collection is retrieved at the best possible ranks).

We also evaluate the continuous learning process, as the development of the search engine is incremental and the students can further improve previous modules when submitting new ones, which they actually did. We also consider the difficulty of the IR techniques they decide to implement, their understanding and evaluation, which should be reflected in the reports they have to submit and present with each module. Also, the lab sessions are useful to assess the teamwork capabilities of the students.

4. RELIABILITY OF THE COLLECTION

The critical point of our methodology is the reliability of the test collection developed with the students. This type of IR evaluations has two main drawbacks: the inconsistency and incompleteness of relevance judgments [4][15][16]. Topical relevance is highly subjective, and some people might judge the same document differently, sometimes even the same person judges the same document differently over time. Therefore, the use of different judges to assess the relevance of documents could make the evaluation results inconsistent and little reliable, as some systems could benefit more from some judges than others. On the other hand, it is important to note that only the documents in the pools are judged for relevance, and every document outside the pool is considered non-relevant. As such, the judgments are incomplete, and if the student systems returned documents not included in the pool they would be penalized, even if some of them were actually relevant.

TREC test collections are generally considered reliable because they are built with many documents, topics and systems, and they have enough man-power to judge tens of thousands of documents. Inconsistency of judgments could be accentuated in our case because they are made by undergraduate students rather than trained IR experts, and the very small scale of our experiments could make the pools too incomplete to be reliable. The effect of these problems has been studied in TREC for over a decade with well accepted meta-evaluation techniques [15][16]. Next, we show the results of this meta-analysis in our collection to assess its reliability and hence the feasibility of the whole methodology.

4.1 Inconsistency of Judgments

4.1.1 Assessor Agreement

For 17 of the 20 topics we collected judgments from two different students (faculty members in the case of topic 006). We measured the agreement between each topic's two assessors using Cohen's Kappa (equal weights), resulting in an average score of 0.417 (see Table 1), which can be considered fairly high. We also measured the precision and recall of one student's judgments according to the other's, and the averages were 0.657 and 0.638. These results agree exceptionally well with Ellen Voorhees' finding that a practical upper bound on performance is 65% precision at 65% recall, as that is the level at which TREC's experts tended to agree with one another [15]. As such, the judgments from our students seem reliable compared to TREC's.

4.1.2 Effect on System Performance

Despite the students apparently tended to agree on the relevance judgments, the differences found could still have an impact on the evaluation of their systems. For 17 topics we have judgments from two assessors, and we can measure the effect of having used one or the other by randomly choosing one assessor per topic and then combining them all across topics.

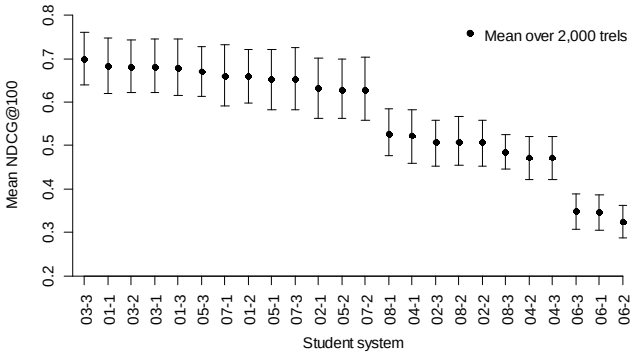

Figure 1. Mean, minimum and maximum NDCG@100 score for the student systems over a sample of 2,000 trels.

We made 2,000 such random combinations (call each of these a *trel* or topic relevance list) and used them to re-evaluate the student systems (see Figure 1). The differences between minimum and maximum NDCG@100 scores ranged from 0.075 and 0.146 across systems, which are about 1.5 times larger than those found for TREC data [15]. This is to be expected though, as our students have far less experience than TREC assessors, our judgments are inherently more variable because they are on a 3-point scale rather than the 2-point scale used in TREC, and the absolute effectiveness of the student systems was itself larger than those evaluated in TREC (about twice as much).

4.1.3 Effect on System Ranking

The interesting result of an IR evaluation experiment is not really how well the systems perform in absolute terms, as this highly depends on the particular test collection used, but rather what systems perform better; that is, their relative ranking. For example, in Figure 1 it can be seen that system 03-2 (module 2 by student group 03) performed much better than system 08-3. Because the actual effectiveness of the systems depend on the relevance judgments used, it is interesting to see how the final ranking of systems varies with one set of judgments or another. We measured this variation with Kendall's tau correlation coefficient between 2,000 random pairs of system rankings generated by random trels. The average correlation was 0.926, ranging from 0.811 to 1. This indicates that the final ranking of systems is quite stable to variations in judgments, compared to Voorhees' findings with TREC data (an average coefficient of 0.938 [15]). Indeed, none of the ranking swaps between two systems were significant (Wilconxon, α=0.05).

4.2 Incompleteness of Judgments

4.2.1 Effect of Pool Size

Small pool sizes with incomplete relevance judgments are not reliable because many relevant documents could be left out of the pool. However, large pools are more expensive to evaluate, and because our assessors are the students themselves we need to find a reasonable midpoint for reliable pools at the minimum cost. We can assess the effect of progressively incrementing the pool size and re-evaluating the student systems to measure the differences in effectiveness that the pool size increment causes. We started with pools of just 20 documents (the 10 top results in Google and the 10 noisy documents), and re-evaluated the systems for average NDCG@100 score. Next, we incremented the pool sizes to 25 documents, and re-evaluated again. The increment in NDCG@100 score ranged between 14.55% and 37.13%. We kept doing this with increments of 5 documents up until the last increment from pool sizes 95 to 100 documents (see Figure 2). The increment in NDCG@100 performance here ranged from 0.4% to 1.63%, with an average of just 1%.

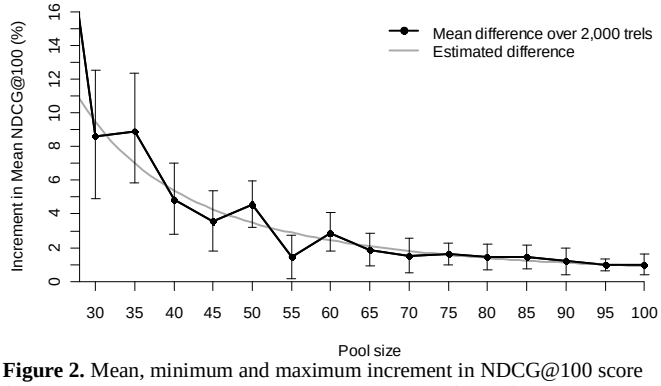

Figure 2. Mean, minimum and maximum increment in NDCG@100 score for the student systems evaluated with 2,000 trels, for different pool size increments (20 and 25 omitted for clarity). Estimation is plotted in grey.

As the figure shows, differences in performance decrease as pool size increases, because as the pools get larger there are fewer relevant documents yet to be discovered. Average performance differences were found between 0.5% and 3.5% in TREC data, with some observations of up to 19% [16]. In our case, differences in this range can be found for pool sizes of about 60-65 documents and more, so our collection is fairly reliable despite the small pool sizes and the fact that none of the student systems contributed directly to the pools.

4.2.2 Extrapolation from Pool Size

Using the points in Figure 2, we can extrapolate the average increment in NDCG@100 scores for pools over 100 documents. If the increments decreased considerably for small additions to the pools, we could consider having a few more documents to judge for the sake of completeness and stability of results. We fitted a non-linear regression model (see gray curve in Figure 2), according to which we would need to have about 135 documents per pool to have average increments in NDCG@100 below 0.5%. Therefore, it does not seem reasonable to increment the pool sizes that much, considering that the average increment is just 1% for pools with 100 documents and 35 more are just too many for such a small improvement.

5. CONCLUSIONS AND FUTURE WORK

Information Retrieval courses do not always pay much attention to the IR process as a whole. We believe, though, that it is very important for students to be involved in the process from the very beginning, not using any data or tool as given. In particular, it is of major importance for undergraduate students to focus on the actual techniques learned in class rather than on particular technologies, as well as getting used to laboratory experimental settings. In IR courses, these evaluations usually take place with small test collections that are maintained from year to year, with the students facing always the same documents, the same topics and the same relevance judgments beforehand. Therefore, we updated the program of our IR course to give more importance to evaluation experiments, involving the students in the development and use of a brand new test collection.

We have described how to adapt TREC's ad-hoc methodology to build such collections for an IR course. The first main difference is that the documents in the collection are gathered after selecting the topics, and not the other way around as usual. The second main difference is related to the pools of documents to judge. The systems developed by the students cannot contribute directly to the pools to prevent cheating, and the judging effort is limited because the students cannot be asked to judge as many documents as we would want. Due to this limitation, the pools are formed differently, with the help of freely available IR tools. With typical meta-analysis techniques we measured the reliability of this methodology in terms of judgments inconsistency and incompleteness. We observed high agreement scores between students, and very high correlations between their systems when using different sets of relevance judgments. In terms of pool reliability, we estimated that pools of size 100 and different depths are quite reliable and do not seem to affect the evaluation significantly. We conclude that the judgments made by students can be trusted, and that the pooling method proposed seems to work reasonably well for these small-scale evaluations.

Having students participate in the whole process of building a test collection helps them realize the complexities and limitations of IR evaluations, especially regarding the concept of relevance and the scale of the evaluation itself, and gives them a good sense of empirical experiments for computer-related tasks. The results in our pilot experiment have been satisfactory both in terms of reliability of the collection and response by the students. Their reviews by the end of the semester show that apparently they had more technical problems this year (the score dropped from 3.27 to 2.82, over 5), unveiling possible gaps in the CS program, especially in terms of database technology and big data management. Nonetheless, they showed the same satisfaction levels as previous years, which is very appealing for us given that this year they had to work significantly more than previous students, and carrying out a pilot study like this always bears daily, intangible logistics problems both for them and for us.

We plan on improving the methodology and build one new collection each year. A clear benefit of this is that each year the students will have more and more small test collections to train and tune their systems, which helps them in the development and improvement phases. Moreover, beginning the next year we will have considerably more students, and so more collaborative methodologies can be explored to build larger collections, or maybe two different ones with different purposes. We are also planning the coordination of this course with another CS course where students have to develop a large application based on already available components, and a Library Science course where students are more focused on the user side of the IR process. The idea is that the students in the development course could develop better tools and frameworks for the IR students to build their search engines, and the LS students to contribute to the test collections for them to be more realistic and reliable.

The collection and frameworks described in this paper are publicly available for research and educational purposes at http://ir.kr.inf.uc3m.es. All forthcoming collections will be freely available as well, with reports commenting on their development.

ACKNOWLEDGEMENTS

We would like to thank the students that participated in this pilot experiment for their effort and understanding, as well as Sonia Sánchez-Cuadrado. We would also like to thank Ellen Voorhees for her comments on the inconsistency analysis.

REFERENCES

[1] J. Gantz and D. Reinsel. *The Digital Universe Decade, Are You Ready?* IDC iView, 2010.

[2] J.M. Fernández-Luna, J.F. Huete, A. MacFarlane, and E.N. Efthimiadis. Teaching and Learning in Information Retrieval. *Journal of Information Retrieval*. 12(2): 201-226, 2009.

[3] IEEE/ACM. . Computing Curricula 2001. http://www.acm.org/education/curricula-recommendations.

[4] E.M. Voorhees. The Philosophy of Information Retrieval Evaluation. In *CLEF Workshop*, pages 355-370, 2002.

[5] E.M. Voorhees and D.K. Harman. . *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[6] N. Madnani and B.J. Dorr. Combining Open-Source with Research to Re-Engineer a Hands-On Introductory NLP Course. In *Workshop on Issues in Teaching Computational Linguistics*, pages 71-79, 2008.

[7] P. Calado, A. Cardoso-Cachopo, and A.L. Oliveira. IR-BASE: An Integrated Framework for the Research and Teaching of Information Retrieval Technologies. In *Int. Workshop on Teaching and Learning of Inf. Retrieval*, 2007.

[8] M. Chau, C.H. Wong, Y. Zhou, J. Qin, and H. Chen. Evaluating the Use of Search Engine Development Tools in IT Education. *JASIST*. 61(2): 288-299, 2010.

[9] M. Chau, Z. Huang, and H. Chen. Teaching Key Topics in Computer Science and Information Systems Through a Web Search Engine Project. *ACM Journal on Educational Resources in Computing*. 3(3): , 2003.

[10] D.G. Hendry. History Places: A Case Study for Relational Database and Information Retrieval System Design. *ACM Journal on Educational Resources in Computing*. 7(1), 2007.

[11] K. Markey and P. Atherton. *ONTAP: Online Training and Practice Manual for ERIC Data Base Searchers*, 1978.

[12] G. Braught, C.S. Miller, and D. Reed. Core Empirical Concepts and Skills for Computer Science. *ACM SIGCSE Bulletin*. 36(1): 245-249, 2004.

[13] J. Urbano, J. Lloréns, Y. Andreadakis, and M. Marrero. Crawling the Web for Structured Documents. In *ACM International CIKM Conference*, pages 1939-1940, 2010.

[14] K. Järvelin and J. Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems*. 20(4): 422-446, 2002.

[15] E.M. Voorhees. Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness. *Information Processing and Management*. 36(5): 697-716, 2000.

[16] J. Zobel. How Reliable are the Results of Large-Scale Information Retrieval Experiments? In *International ACM SIGIR Conference*, pages 307-314, 1998