Good Benchmarks are Hard To Find: Toward the Benchmark for Information Retrieval Applications in Software Engineering

Alex Dekhtyar and Jane Huffman Hayes

ABSTRACT

Seven to eight years ago, the number of applications of Information Retrieval (IR) methods in Software Engineering was close to zero. These days, IR and text mining methods are accepted approaches to analysis of textual artifacts generated during the software lifcycle. The incentive to try IR methods in such analysis is strong: the field comes with a reputation for proven industrial and academic success, and some important Software Engineering problems related to textual artifacts, can be translated into an instance of a standard IR problem in a reasonably straightforward manner.

In this position paper, we observe that part of the success of IR as a field came from the use of established, well-maintained, and almost universally accepted benchmarks for testing the work of IR methods. We elaborate on the question "Is the field mature enough to talk about benchmarking?" asked by the working session organizers. Our position is that *without robust, well-designed time-tested, and, eventually well-established and accepted benchmarks, research on application of IR methods to problems in Software Engineering will not reach its full potential.*

1. WHY?

Information Retrieval has become a textbook example of a field that found its "killer application." Web search did not just put IR at the fingertips of millions. Over the past 12 years, it has defined the shape and form, size and key properties of the core IR tasks. IR methods had to be built to work with collections of billions of text documents which came in a variety of sizes and languages. They had to work on-line — i.e., the most relevant answers to the user queries had to be returned in real time. And they had to have high accuracy: the top documents retrieved and shown to the users had better be relevant.

But the academic success of Information Retrieval as the field of Computer Science had one other important reason: since 1992, IR research has been supported by a series of well-designed, robust benchmarks, distributed via the infrastructure of TREC (Text REtrieval Conference) series, co-sponsored by the National Institute of Standards and Technology (NIST) and the U.S. Department of Defense.

TREC benchmarks evolved from year to year. Having started with simple retrieval tasks from document collections, they have expanded in recent years to include (i) retrieval of special kinds of information, such as filtering spam, (ii) retrieval from legal texts, (iii) retrieval from genomics data, (iv) retrieval from very large datasets, (v) cross-language retrieval, (vi) retrieval of video data, and more.

The importance of TREC to the development of IR in the last 15 years can hardly be overstated. TREC datasets and the framework of TREC, which included research groups submitting the answers derived by their tools and further publication of side-by-side tool comparisons served as the measuring stick and the measuring process for the majority of IR researchers. The diversity of TREC tracks ensured that TREC benchmarks covered an ever-expanding number of IR tasks and applications.

Software Engineering is somewhat unique in terms of how the IR methods are used in it. In a typical Software Engineering setting, IR methods are used on a document collection that consists

of artifacts produced within a single project or a group of related projects. The size of such document collections is much smaller than the typical sizes of document collections featured in "standard" IR applications. This opens up the possibility to use a wider range of IR methods, including techniques that scale poorly, but are known to provide good results (Latent Semantic Indexing is an example of such a technique). In addition, individual elements in Software Engineering (SE) data collections come at a higher level of granularity: individual requirements form a collection of elements in a requirements document, in source code often "elements" are class descriptions, or even method/function code. Such elements are often rather short and are written in a technical lingo. These two features often mean that some traditional assumptions behind "standard" IR document collections, such as Zipf law of keyword frequencies do not work in Software Engineering collections.

Do we need benchmarks for testing IR methods on Software Engineering datasets? In short, our view is that the answer is "yes." At present, it is clear that IR methods are applicable to a reasonably wide range of Software Engineering problems, such as (borrowing from the list provided by the working section organizers): traceability, reverse engineering, concept location, reuse, impact analysis, and more. Missing from the current state-of-affairs in the field are:

- *Robust comparisons of different techniques for solving the same problem.* While there is an occasional apples-to-apples comparison of IR methods [1, 5] that uses the same problem and the same dataset, such work is just beginning to emerge.
- *View of the behavior of the same methods on different problems.* Standard IR techniques are applied with different success to various problems in Software Engineering. These problems, however, come from very different areas of Software Engineering, sprouting different researcher communities. There is very little communication across the problem boundaries, and very little analysis of the behavior of different approaches across the entire problem set in Software Engineering to which IR methods are applicable.

We believe that a well-designed, robust, feature-rich benchmark for testing application of IR methods to Software Engineering problems will go a long way toward addressing both issues above. It will establish the sorely needed "measuring stick" for testing uses of different IR techniques within the same problem set. Additionally, the process of creating, maintaining and the framework of using the benchmark have the ability to bring together the Software Engineering researchers applying IR methodology to different problems.

In the following two sections we give a brief overview of our vision of what such a benchmark should be and how it can potentially be built.

2. WHAT?

In a nutshell, an IR benchmark can be described by the following formula:

Benchmark = DataSet +Tasks + Answers +Measures + So ftwarelDataFormats.

We give a brief overview of each component below.

Data Set. We identify five important features of the dataset: *origin*, *size*, *diversity*,*flexibility*, *nature*. Given our need to test application of IR methods to Software Engineering problems, the overall *origin* of the dataset is straightforward: it should be a collection of artifacts from a

software project. The exact origin may vary from a software project specially manufactured to be the benchmark, to the use of artifact collections from existing projects. The issue of *data availability* looms large here.

Because *scalability* (if only for the sizes of software lifcycle artifact collections) is an important aspect of the methodologies applied, it would be good to have benchmark artifacts of varying *sizes*, and, in particular, ensure that the overall size of the data set (both in terms of number/type of artifacts and in terms of the sizes of individual artifacts) is consistent with industry standards.

Project artifacts come in *diverse* forms, from requirements and design documents, to code, to test cases, to bug reports. Additional artifacts, such as glossaries, dictionaries, charts and diagrams, etc., may also be included. Related to the *diversity* is the *nature* of the artifacts. Should the benchmark contain only textual artifacts, or should it include non-textual artifacts of different kinds?

While the purpose of IR benchmarks is to compare different IR methods to each other, in Software Engineering, it is often the case that different *modus operandi* need to be compared. For example, in our work on traceability, we have argued that it is important to measure the accuracy of the requirements tracing matrix built by a human analyst[3]. In such cases, the notion of an "IR method"

responsible for the answers is extended to include some "operating procedures." Will the data set in the benchmark be*flexible* enough to support such different procedures?

Tasks. Diversity in the types of artifacts in the benchmarks shall lead to diversity in the tasks, that call for application of IR methods. Even when limited to traceability only, a number of different tasks can be envisioned, including: (a) traceability between different textual artifacts, (b) traceability between a textual and a non-textual artifact, (c) traceability between non-textual artifacts, (d) detection of different types of links, (e) tracing of non-functional requirements, and more. Other subdisciplines can contribute tasks of their own.

Answers. In our practice, testing of methods on "real" artifacts is significantly hampered by the lack of available answers, the *ground truth*, against which the work of the methods will be compared. It is very hard to build the answer sets from scratch when the artifacts are large, doubly so in an academic environment. One of the drawbacks of diversity of tasks in the benchmark is the need to have answer sets for all of them.

Measures. We have standard measures for the accuracy of IR algorithms: *precision*, *recall* (both macro- and micro- variants), *average expected precision*, *f-measure*, etc. Additional measures, related to various aspects of the tasks important from the Software Engineering point of view can be developed and used. For example, in [4] we have described a measure called *selectivity*, which quantifies the difference between the size of the result of the IR method, and the total number of candidate links an analyst would have to check manually.

Software/Data Formats. The only optional part of the benchmark, development of convenient data formats and inclusion of software that understands these formats in the benchmark can go a long way in both making the benchmark popular and establishing standard means of encoding "real" artifacts for the purpose of running IR methods on them.

3. HOW?

In summer of 2006, the NASA Independent Verification and Validation (IV&V) Facility has sponsored, as part of its annual research funding activities, the creation of the Center of Excellence for Traceability[2]. The Center is viewed as a partnership of government agencies, businesses and research institutions to facilitate, sponsor and infuse traceability research focusing on development of practical solutions. As the first step, the Center sponsored a working meeting of leading researchers in the field of traceability devoted to establishing the Grand Challenges in Traceability. At the meeting, the participants outlined 18 subareas (such as traceability knowledge, traceability education, traceability methodology, scalability, human factors, etc.) in which advancement of state-of-the-art in traceability research and practice was desired. This, in turn gave rise to 40 Grand Challenges in traceability [2].

Among the areas identified, *Measurement and Benchmarks* was considered one of the key. In particular, the following Grand Challenges related to this area, have been proposed in [2]:

L-GC1 Define standard processes and related procedures for performing empirical studies during traceability research.
L-GC2 Build benchmarks for evaluating traceability methods and techniques.
L-GC3 Define measures for assessing the quality of individual and sets of traceability links.
L-GC4 Develop techniques to assess traceability methods and processes.

In the next year, the Center of Excellence for Traceability is planning to establish the process for addressing this set of challenges. Because availability of benchmarks is crucial, in our view, to the successful development of the area, the intent is to proceed expediently with putting together a working group tasked with building a robust benchmark for testing automated traceability techniques.

At the same time we note that while the challenges above specifically address the problems related to traceability, they are common with other applications of IR to Software Engineering. Different applications of IR to Software Engineering can share the dataset, measures and formats/parsers: the difference can be built into the tasks that would be separate for each application.

We hope that our efforts in building the traceability benchmark can be extended to create a more general benchmark for testing applications of IR methods to a wide range of Software Engineering problems. We invite all interested parties to join and/or support these efforts.

Acknowledgments
Our work is funded by NASA under grants NNG05GQ58G8G and NNX06AD02G and by NSF under grant CCF-0647443. We thank Marcus Fisher, Stephanie Ferguson, Jane Cleland-Huang and all the participants of the workshop on Grand Challenges in Traceability, which took place on August 4-5 in Fairmont, WV.

4. REFERENCES

[1] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., and Merlo, E. Recovering Traceability Links between Code and Documentation. *IEEE Transactions on Software Engineering*, Volume 28, No. 10, October 2002, 970-983.

[2] J. Cleland-Huang, A. Dekhtyar, J. Huffman Hayes (Eds.). Grand Challenges in Traceability, *draft*, *Center of Excellence for Traceability tech. report* COET-GCT-06-01-

0.9, http://www.traceabilitycenter.org/downloads/ documents/GrandChallenges/, September 10, 2006.

[3] J. Huffman Hayes and A. Dekhtyar. Humans in the Traceability Loop: Can't Live with 'Em, Can't Live Without 'Em, (2005), in *Proceedings, 3d International Workshop on Traceability in Emerging Forms of Software Engineering*, pp. 20-23, Long Beach, CA, November 7, 2005.

[4] J. Huffman Hayes, A. Dekhtyar, S. Sundaram. Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods. *IEEE Trans. Software Eng*. 32(1): 4-19 (2006)

[5] Marcus, A., Maletic, J. Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing, in *Proceedings of the Twenty-Fifth International Conference on Software Engineering*, 2003, Portland, Oregon, 3 - 10 May 2003, pp. 125-135.