An Architecture for Privacy-Preserving and Replicable High-Recall Retrieval Experiments

Adam Roegiest
University of Waterloo

Gordon V. Cormack
University of Waterloo

ABSTRACT

We demonstrate the infrastructure used in the TREC 2015 Total Recall track to facilitate controlled simulation of "assessor in the loop" high-recall retrieval experimentation. The implementation and corresponding design decisions are presented for this platform. This includes the necessary considerations to ensure that experiments are privacy-preserving when using test collections that cannot be distributed. Furthermore, we describe the use of virtual machines as a means of system submission in order to to promote replicable experiments while also ensuring the security of system developers and data providers.

1. INTRODUCTION

The Total Recall track at TREC 2015 [15, 14] sought to investigate methods for achieving high-recall, with an assessor in the loop, through controlled simulation. The Total Recall track offered an online evaluation platform that allowed participants to produce systems that could request document assessments in a document-at-a-time manner, which is similar to the Microblog track's "evaluation as a service" methodology [11, 12]. This evaluation platform was primarily contained in a Web server (Section 3) that facilitated run creation, corpora distribution, the aforementioned document assessment process, and some basic online evaluation for training collections. Included in this platform was a baseline model implementation (the "BMI", Section 4.1), distributed as a VirtualBox virtual machine (VM), which participants could freely modify as they saw fit.

In developing this platform we sought to mitigate issues that have arisen in previous information retrieval research, and, in particular, high-recall retrieval experimentation. The primary issues we sought to address are as follows:

- Assessor-participant interaction and the effect on performance [9]
- Data anonymization and preserving the privacy of individuals in a corpus [1, 2]
- Re-usability of participant systems as on-going baselines for subsequent iterations of the track
- Lowering the barrier of participation in the track

We feel that we have successfully met these goals but we acknowledge that there are still improvements to be made. By controlling interaction through a simulated assessor (e.g., a system receives a binary judgement on a document), participant interaction does not rely on the ability of participants to get information out of the assessor but rather on the documents themselves. The BMI provides a working system to participants that they can use as a baseline for their experiments. Accordingly, there is less burden on participants to have an in-depth understanding of the Web server and can focus on their particular algorithms.

Furthermore, the track offered two modes of participation: an `At-Home` mode where interaction occurred over the Web; and a `Sandbox` mode where interaction occurred locally on a single machine. Participation in the `Sandbox` mode required participants to submit a working VM that would be ran, without Internet access, on test collections that were unknown to participants. Figure 1 provides a high-level conceptual depiction of how the live and sandbox modes differ.

We note that regardless of configuration, the data collections (the types and formats described in Section 2) are largely "plug-and-play", meaning that new data collections can, with little effort, be added to the platform. The result is that new experiments can be run on participant systems without necessitating participant action and without requiring participant effort — other than that required to run the system. Furthermore, the `Sandbox` mode of participation allows collections to be used that would otherwise be "too hot" to distribute or would require onerous and time consuming anonymization that may not work.

In the remainder of this work, we describe the major components of the Total Recall track's platform: the test collections and their format (Section 2); the Web server and its sandboxing (Section 3); the baseline system, participant systems, and their sandboxing (Section 4). We conclude with a discussion of limitations and future improvements to the platform (Section 5). Throughout these sections, we will provide the underlying reasoning as to why a particular approach was taken when another might have sufficed.

SIGIR '16, July 17 - 21, 2016, Pisa, Italy

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4069-4/16/07...$15.00

DOI: http://dx.doi.org/10.1145/2911451.2911456

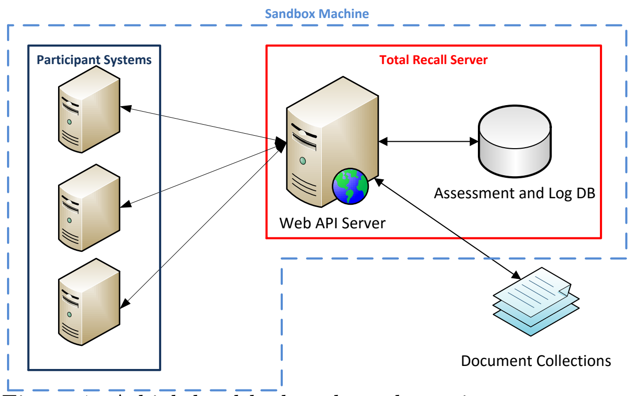

Figure 1: A high-level look at how the various components interact in live and sandbox environments. Note that the dashed blue line denotes the Total Recall server and participant VMs running on the same machine.

2. TEST COLLECTIONS

For our purposes, we consider a test collection to be a set of documents along with a set of qrels, which are tuples mapping documents and topics (information needs) to relevance assessments. Data collections are transmitted to participant clients through the Total Recall service first by transmitting the set of documents and then by having systems requests relevance assessments in a document-at-a-time manner. This simulates the interaction between participants and a gold standard assessor. that was present in previous TREC Legal tracks [10, 7], TREC Filtering tracks [13], and the TREC Spam tracks [4, 3, 8], while also controlling that interaction to ensure that it is equitable amongst all participants. Furthermore, to ease system development we curated each test collection so that a single file contained a single document. This is contrary to many past TREC collections where a file might contain multiple documents and was performed to simplify document parsing. We additionally translated each document into a plain text rendering from whatever native format it was originally stored in (e.g., PST, WARC, etc) to ease the burden of system design as participants would not have to worry about processing multiple filetypes.

By taking this controlled approach to high-recall retrieval experimentation, we hope to mitigate any bias that might be perceived by spending "too much time" with a topic authority (gold standard assessor) or the ability to ask the right questions. The focus is then on the algorithms used and less so on the human interaction between topic authorities and participants. This should not be construed to suggest that such interaction is bad but that we merely sought to eliminate the confound that different approaches to such interaction may have on system evaluation. Furthermore, this approach facilitates experimentation with more topics over more test collections since there is no requirement to provide access to a high-quality topic authority (that is willing to volunteer valuable time), which has previously limited how many topics and how much interaction could be performed [10, 7].

For the purposes of system development, we used three publicly available test collections: 20-Newsgroups[1]; Reuters-21758[2]; and, a variant of the Enron v2 e-mail collection[3]. The inclusion of the first two datasets is to facilitate rapid development and testing of participant systems as they both contain approximately 20,000 documents. Neither was meant to be representative of a valid test collection but were merely there to help participant's gain confidence that their system was working. Moreover, we offered samples of these first two corpora as a mechanism to ensure participants understood and could correctly interact with the Web server, but without requiring the extensive processing time of a larger corpus. The Enron collection was our attempt to provide a representative collection, as it had previously been used in the TREC 2009 Legal track [10], that would more accurately indicate the effectiveness of participant systems. For all of these collections, we provided an online mechanism to provide recall, effort, precision, and F1 scores for checking system performance.

At the beginning of July, the `At-Home` phase of participation began and three additional corpora were released to participants (after signing the appropriate usage agreements). These collections were the official test collections and so participants did not receive any explicit system performance feedback as a means of preventing any potential meta-learning by systems. Although, we acknowledge that some participants may have had internal quality assurance processes that would have helped determine their own performance.

Not all corpora can be distributed by the service due to usage agreements or the risk of divulging sensitive data to the public, and so, generally, such restricted corpora will be distributed only in the form of `Sandbox` participation (discussed in Section 3.1). This helps to ameliorate issues with imperfect anonymization and the inability to give all interested parties access to the data. While no system that emits any information (e.g., even summary evaluation metrics) can ever be entirely secure, we believe that structuring the release of collections in this way aids in preserving the privacy of all parties when private collections are used. For all test collections, we refer interested parties to the track overview [15] for in-depth descriptions.

3. THE SERVER

The Total Recall server operates as a Web service that participant systems interact with over HTTP requests (i.e., a REST(ful) API). There are three main types of interactions between systems and the service:

- Request for information on a corpus and a link to download the corpus. Such details include the type of corpus (e.g., e-mail, newswire) and the language of the corpus (English only for 2015).
- Request for information on a topic to process, which includes the corresponding corpus and a description of the information need.
- Request for relevance assessment of documents with respect to a particular topic.

Secondary interactions can occur, including: starting and finalizing a run; result generation for developmental test collections; error log checking; and, other track specific actions beyond the scope of this paper.

[1]http://qwone.com/~jason/20Newsgroups/
[2]http://www.daviddlewis.com/resources/testcollections/reuters21578/
[3]http://trec-legal.umiacs.umd.edu/corpora/trec/legal10/

Figure 2 provides the general workflow that a client system might be expected to perform. For brevity we omit the actual API that was used to interact with the service and instead direct interested parties to the API documentation[4]. Note that all requests and responses to and from the server are encoded in JSON format, which has become a standard format for passing data between a Web server and client software.

In terms of implementation the entirety of the Total Recall web server is written in Node.js[5], which is a Javascript runtime that uses an event-driven model for developing sever-side Web applications (e.g., REST(ful) APIs). Node.js is designed to be efficient and lightweight and is extremely popular in web development. While other architectures would have sufficed, Node.js was chosen due to its popularity and the opportunity to explore current state-of-the-art technologies in Web development. Where necessary (e.g., in storing judgments and a log of participant requests), the server relies on an installation of MySQL for persistant storage. As will be discussed in Section 5, using a default installation of MySQL was perhaps a poor choice and resulted in a small bottleneck near the end of the experimental period but has been corrected for following iterations. All code for the Total Recall track is available for public download in a git repository[6].

3.1 Sandboxing the Server

To facilitate the use of test collections that require complicated transmission protocols (e.g., transmitted by a third party, such as NIST) or are "too hot to handle" (e.g., raw versions of confidential email), we provided the `Sandbox` mode of participation. In this mode, participants submit their systems as a VM (more details in Section 4) which we run in a restricted environment with access to the data.

Test collections do not necessarily have to be stored directly on the sandbox machine; instead data providers will be able to distribute test collections via USB flashdrive or external hard drive. Once the test collection is loaded into the service (using some relatively simple shell scripts), the participant systems are can be ran with these collections much in the same way that they would in the live scenario. However, the participant's VM is prohibited from accessing the Internet or Web in any way. This is done so that we can prevent any blatant data leakage from the sandbox environment. This restriction is enforced by limiting access of the clients machines only to the service and (optionally) by air-walling the sandbox machine (i.e., never connecting it to the Internet once the test collections are present). Air-walling occurred for one of the `Sandbox` collections due to the nature of the of the collection.

By enforcing these restrictions on the sandbox machine, the data is protected from unintended transmission as well as unintended dissemination of personal or private data. Furthermore, we can limit the only output of the sandbox server to be only summary evaluation measures and statistics (discussed in the track overview [15, 14]) once all participant systems have been ran. In doing so, the goal would be to prevent accidental distribution of private data that may be contained in qrels and document identifiers.

[4]quaid.uwaterloo.ca:33333/#/api
[5]See https://nodejs.org/en/ for more information.
[6]http://repo.trec-total-recall.com

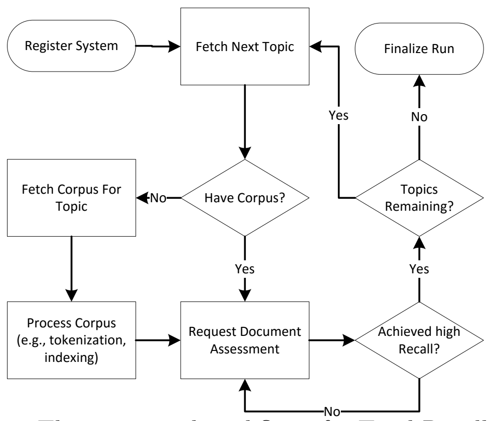

Figure 2: The envisioned workflow of a Total Recall participant system.

4. PARTICIPANT SYSTEMS

The envisioned workflow for a client system is depicted in Figure 2, and is implemented by the the Baseline Model Implementation (Section 4.1). Participants can construct several different systems to interact with the Total Recall service. Clients for `At-Home` experiments can be developed in several different ways: a purely automatic program (or set of programs) that performs the Total Recall task; a customized virtual machine that runs the above; as a "sidecar" (i.e., a directory containing additional scripts) supplied to the BMI's Virtual Machine. A "sidecar" submission can be run without any additional dependencies being added to the default BMI VM.

The Total Recall task is one that can be performed manually or through some combination of manual and automatic approaches. Accordingly, we allowed participants to submit a single manual run, which was to be performed before developing any automatic systems. Participants could submit manual runs using either the API, used by automatic systems, or a Web-based interface that we provided. The Web-based interface was quite simple and was largely just a pretty interface for the API. Out of the 3 manual teams, 2 used the Web interface and 1 directly communicated with the Web server via the API.

For sandbox submission, participants are required to submit systems that fall under the full VM or sidecar categories. This is done with the hope that systems would work out-of-the-box, which was mostly true for TREC 2015. Previous experience with running participant systems in a sandbox for the TREC Spam tracks [8, 3, 4] led us to believe that allowing the submission of arbitrary code would lead to wasted effort in getting code to run correctly. As coordinators would then have to debug and fix-up any code that was not compatible with the sandbox environment (e.g., installing libraries, having necessary compiler versions). By using a VM, we hoped to limit the necessity of such tasks and were successful in this by having only one system require major coordinator intervention (out of 11 different systems).

A final benefit to requiring submissions of VMs or sidecars is that it allows commercial vendors of high-recall retrieval software to submit their products without requiring source code submission (as was required in previous tracks that used sandboxing). Vendors can submit a fully compiled version of their software that runs on the VM and so do not expose their intellectual property to track coordinators, data providers, or any other parties that might come into contact with the software. Extra steps can also be taken, such as encrypting the VM and/or software so that outside parties cannot access the software at all (outside of the normal interaction between client and server).

4.1 Baseline Model Implementation (BMI)

Our primary baseline is an augmented version of the Continuous Active Learning (CAL) method originally presented by Cormack and Grossman [5], which is called AutoTAR [6]. Unlike the original version of CAL, AutoTAR uses exponentially increasing batches sizes, and unlike the reported results of the AutoTAR paper, we use a tuned version of `sofia-ml`[7] (based upon suggestions from the package's author).

This baseline is implemented as part of the virtual machine with sidecar approach, where the AutoTAR algorithm is refactored as a sidecar for a simple Debian VM. This was done to provide a model implementation of a sidecar and a meaningful working implementation that participants would be free to augment in the course of experimentation. The BMI is implemented using a combination of C++ and bash scripts (along with associated command line tools). The canonical implementation is available for download under GPL[8].

Cormack and Grossman have shown that AutoTar generally outperforms a simple CAL implementation. Accordingly, our intent was to provide a reasonable baseline that could "fast-track" participants to a working system, without requiring them to worry about API programming. We hoped that the BMI would provide ample opportunity for participants to improve upon its results or be inspired by the technique and devise original algorithms of their own. Several of the participants made use of the BMI in their own submissions.

5. DISCUSSION

As with the development and use of any long running software, issues were likely to arise throughout the course of the track and did. Primarily, the biggest issue was a bottleneck in document assessment requests near the submission deadline. This was primarily due to the vanilla installation of MySQL which became bogged down when there were many queries being ran. In hindsight, this issue is relatively easy to solve by configuring MySQL to handle queries more effectively and by adding additional connections to the Web server so that multiple requests are not held up by a single connection to the database.

Surprisingly, the use of a REST(ful) API did not appear to cause participants too much trouble, in spite of our fears that it might during development. These troubles were likely ameliorated by providing tools like the BMI and the manual participation interface.

Another design decision to consider for future iterations is the use of full virtual machines. There are many merits to the use of virtual machines, primarily the fact that the entire system is self-contained and so security issues are more manageable due to more complete isolation from other software on the machine. This isolation comes with performance setbacks. For example, we could only have one participant system running at a time due to the high overhead of both participant systems and the cost of virtualization on our sandbox machines. An alternative may be to use a more lightweight system, like Docker, which makes use of software containers. In short, software containers package an application together with its dependencies for running on an arbitrary (Linux) server. The easiest way to envision software containers is somewhere between virtual machines and auto-building tools (e.g., make, Maven). Accordingly, there is less emulation overhead that occurs with containers but at the expense greater exposure to the underlying system. A more thorough analysis of the costs and benefits is on-going and the inclusion of Docker containers may occur for TREC 2016.

[7]https://code.google.com/archive/p/sofia-ml/

[8]http://plg.uwaterloo.ca/˜gvcormac/trecvm/

6. CONCLUSIONS

We have presented a brief overview of the Total Recall track's online evaluation platform and the design decisions that went into its development. The use of this service is an attempt to develop a reusable system for evaluating information retrieval systems on both public and private data. In addition, we attempted to learn from issues that have arisen in previous TREC tracks dealing with high-recall retrieval and private collections. Interested parties are encouraged to read the track overview [15], visit the track website (trec-total-recall.org), join the mailing list[9] to follow the progress of the track, and contribute to the development of the platform.

7. REFERENCES

[1] http://www.securityfocus.com/news/11497.

[2] http://www.securitypronews.com/new-scam-tricking-people-by-offering-new-facebook-profile-look-2012-11.

[3] G. V. Cormack. TREC 2006 Spam Track Overview. In *Proc. TREC-2006*.

[4] G. V. Cormack. TREC 2007 Spam Track Overview. In *Proc. TREC-2007*.

[5] G. V. Cormack and M. R. Grossman. Evaluation of Machine-learning Protocols for Technology-assisted Review in Electronic Discovery. In *Proc. SIGIR 2014*.

[6] G. V. Cormack and M. R. Grossman. Autonomy and reliability of continuous active learning for technology-assisted review. *arXiv Preprint*, 2015.

[7] G. V. Cormack, M. R. Grossman, B. Hedin, and D. W. Oard. Overview of the TREC 2010 Legal Track. In *Proc. TREC-2010*.

[8] G. V. Cormack and T. R. Lynam. TREC 2005 Spam Track Overview. In *Proc. TREC-2005*.

[9] B. Hedin and D. W. Oard. Replication and automation of expert judgments: Information engineering in legal e-discovery. In *Proc. Systems, Man and Cybernetics*, 2009.

[10] B. Hedin, S. Tomlinson, J. R. Baron, and D. W. Oard. Overview of the TREC 2009 Legal Track. In *Proc. TREC-2009*.

[11] J. Lin and M. Efron. Evaluation as a service for information retrieval. *SIGIR Forum*.

[12] J. Lin and M. Efron. Overview of the trec-2013 microblog track. In *Proc. TREC-2013*.

[13] S. Robertson and I. Soboroff. The TREC 2002 Filtering Track Report. In *Proc. TREC 2002*.

[14] A. Roegiest and G. V. Cormack. Total recall track tools architecture overview. In *Proc. TREC-2015*.

[15] A. Roegiest, G. V. Cormack, C. L. Clake, and M. R. Grossman. TREC 2015 Total Recall Track overview. In *Proc. TREC-2015*.

[9]Link available from track website.