Evaluation as a Service for Information Retrieval

Jimmy Lin
The iSchool
University of Maryland, College Park
jimmylin@umd.edu

Miles Efron
Graduate School of Library and Information Science
University of Illinois, Urbana-Champaign
mefron@illinois.edu

Abstract

How can we run large-scale, community-wide evaluations of information retrieval systems if we lack the ability to distribute the document collection on which the task is based? This was the challenge we faced in the TREC Microblog tracks over the past few years. In this paper, we present a novel evaluation methodology we dub "evaluation as a service", which was implemented at TREC 2013 to address restrictions on data redistribution. The basic idea is that instead of distributing the document collection, we (the track organizers) provided a service API "in the cloud" with which participants could accomplish the evaluation task. We outline advantages as well as disadvantages of this evaluation methodology, and discuss how the approach might be extended to other evaluation scenarios.

1 Introduction

The Cranfield Paradigm [5], especially operationalized in the Text Retrieval Conferences (TRECs) [12], provides the cornerstone of batch evaluation in information retrieval today. A fundamental assumption of this methodology is that researchers can acquire the document collection under study—whether via physical CD-ROMs or DVDs (in the early days), hard drives (today), or directly downloadable "from the cloud". What if this is not possible? One example is a collection of messages posted to Twitter ("tweets"): the company's terms of service forbid redistribution of tweets, and thus it would not be permissible for an organization to host a collection of tweets for download by researchers.[1] Other examples of data that make wide distribution difficult include electronic medical records, a subject of substantial interest by researchers today—for obvious privacy concerns. Similar issues exist for email search and desktop search as well.

In this paper, we discuss one workable solution to this challenge that has been operationalized in the TREC 2013 Microblog track—a concept we've called "evaluation as a service", a play on current cloud computing technologies such as "infrastructure as a service" (IaaS), "platform as a service" (PaaS), and "software as a service" (SaaS). The basic idea is that, instead of distributing the collection, the evaluation organizers provide an API through which

[1] Although there are third-party resellers of Twitter data, the costs are too high to be a practical mechanism for distributing research collections.

the collection can be accessed for completing the evaluation task. This approach makes it not only possible to conduct large-scale evaluations on non-distributable collections, but offers other advantages as well. To be fair, however, there are disadvantages that must also be considered. We attempt to provide a balanced exposition of this evaluation methodology, highlighting new opportunities as well as issues of concern. Although to date this evaluation methodology has only been applied to a single TREC task, we conclude by discussing how the approach might be adapted to other evaluation scenarios as well.

2 Evolution of Evaluation as a Service

The "evaluation as a service" approach implemented in the TREC 2013 Microblog track evolved out of an attempt to address deficiencies in the data distribution approach implemented in the previous two iterations of the track (which began in 2011). Here, we provide a quick recap, but refer the reader to previous track overview papers for more details [8, 10]. In TREC 2011 and 2012, the Microblog track used the Tweets2011 collection, specifically created for those evaluations. Since Twitter's terms of service prohibit redistribution of tweets, it was necessary to develop an alternative mechanism for researchers to obtain the collection. The track organizers devised a process whereby NIST distributed the *ids* of the tweets (rather than the tweets themselves). Given these ids and a downloading program we developed, a participant could "recreate" the corpus. Since the downloading program accessed the `twitter.com` site directly, the tweets were delivered in accordance with Twitter's terms of service.

The "download-it-yourself" approach adequately addressed the no-redistribution issue but exhibited scalability limits. In particular, the speed of the downloading program, which had built-in rate limiting for "robotic politeness", set a practical upper bound on the size of the collection. The Tweets2011 collection originally contained 16 million tweets, which is small by modern standards. For 2013, we hoped to increase the collection size by at least an order of magnitude, which required a completely new approach.

Our solution was the "evaluation as a service" model. The basic idea is that instead of distributing the tweets themselves, we provide a service API through which participants could access the tweets. As the track organizers, we gathered the "official" collection by crawling the public sample tweet stream from Twitter during a data collection period lasting two months, from February 1 to March 31, 2013 (inclusive). This data collection period was publicized on the track mailing list, which gave researchers an opportunity to "follow along" and gather contemporaneous tweets (although they were unlikely to be the same exact tweets due to the sampling process the underlies Twitter's streaming API).[2] In total, we gathered 259 million tweets, although at the time of the evaluation the collection was reduced to 243 million tweets after the removal of deleted tweets.

We built a search API (implemented using Lucene[3] and Thrift[4]) that provides standardized access to the collection. The functionality of the API was determined based on needs solicited from the community via the track mailing list, which were balanced against available resources necessary to implement the requested features. All code associated with the

[2]The participants could of course do whatever they wished with their local tweet collection, but they were under the same no-distribution restrictions.

[3]`http://lucene.apache.org/`

[4]`http://thrift.apache.org/`

retrieval infrastructure is open source,[5] including crawler, so that the community can inspect the code and understand exactly what the software is doing.

The service API provides top k retrieval capabilities using Lucene's query language and returns tweets with associated metadata (author, timestamp, tweet text, etc.). The service API was made available on Amazon's EC2 service to all registered TREC participants starting in June until the TREC evaluation deadline in August. It was the only method by which the collection could be accessed, which meant that *all* participants were required to use the API to obtain an initial ranking of tweets for a given query (which their custom retrieval algorithms could then further process).

We are presently analyzing results from the evaluation, and our detailed findings will be disseminated in the TREC proceedings. However, the evaluation methodology appears to have been successful. The track received submissions from twenty groups in total, which represents a healthy community (based on experiences with previous tracks). Since participation at TREC is entirely voluntary, this shows that the evaluation-as-a-service model is tenable: the service API appears to be simple enough for teams to adopt and flexible enough to enable teams to answer their research questions of interest.

2.1 Advantages

There are several advantages to the evaluation-as-a-service model, which we detail below:

Evaluation on collections that cannot be distributed. This is an obvious point, but worth emphasizing—the evaluation-as-a-service concept makes large-scale evaluation of tweet search possible.

More meaningful system comparisons. Modern information retrieval systems have become complex collections of components for document ingestion, inverted indexing, query evaluation, document ranking, and machine learning. As a result, it can be difficult to isolate and attribute differences in effectiveness to specific components, algorithms, or techniques. Consider a baseline retrieval model such as BM25 or query-likelihood within the language modeling framework—alternative implementations may produce substantially different retrieval results due to small but consequential decisions such as the tokenization strategy, stemming algorithm, method for pruning the term space (e.g., discarding long or rare terms), and other engineering issues. Although the prevalence of open-source retrieval engines makes it possible for a researcher to see exactly what a system is doing, in practice few cross-system comparisons are performed with "calibration" on baseline models, making it sometimes difficult to compare advanced techniques based on different system implementations. In some cases, the effects that we are hoping to study are masked by differences we are not interested in.

One concrete example of this phenomenon is observed in biomedical retrieval. Jiang and Zhai [6] showed that different tokenization strategies yielded statistically significant differences in effectiveness due to the prevalence of complex jargon that, for example, frequently combines letters and numbers (e.g., gene names). It is likely that tokenization will also have significant effects in tweets to due hashtags, abbreviations, shortened URLs, and other Twitter idiosyncrasies.

The evaluation-as-a-service model addresses many of these issues by deploying a common

[5] `http://twittertools.cc/`

API that is used by all participants. This means that everything "below" the API (e.g., indexing, tokenization, etc.) is *exactly* the same for everyone. Thus, we can be confident that differences in effectiveness can be attributed to retrieval techniques on top of the API, rather than "uninteresting" issues such as tokenization and stemming.

Support of open-source community development. One decision we made early on in the development of the evaluation infrastructure for the TREC 2013 Microblog track was that all associated software would be open source. It is desirable that all participants know exactly how the API works and have access to all the minor but potentially important decisions that were made in its implementation (see above).

We hope that this decision has the additional effect of more effectively fostering an open-source community of pluggable system components. There is growing recognition within the IR community that open source software helps advance the state of the art; a common API increases the likelihood that code components inter-operate, thus increasing the likelihood of adoption. Although there is already widespread availability of open-source retrieval engines, nearly all systems are monolithic in the sense that they were not designed for service decomposition along functional boundaries. This means that a particular algorithm developed for one system cannot be easily used by researchers who have written their code on another system due to interface incompatibilities. A common API begins to address this issue.

Solution for systems engineering issues. To reflect today's retrieval environment, modern document collections for IR evaluations have grown quite large—sizes in the tens of terabytes are common. Manipulating these large collections represent non-trivial engineering challenges. Despite the field's growing familiarity with large-scale distributed frameworks such as Hadoop, the available open-source solutions are not quite yet "turn key". Although the size of the tweet collection used in the TREC 2013 Microblog track remains manageable (∼100 GB compressed), the large sizes of other document collections (e.g., ClueWeb12 or the TREC KBA corpus) pose a barrier to entry for many research groups. Even for well-resourced research teams with access to large compute clusters, the effort devoted to systems engineering challenges might be better spent on the development of retrieval techniques (unless, of course, the focus of the research is on scalability).

The evaluation-as-a-service model provides a solution to these systems engineering issues. Scalability challenges only need to be solved once, by the developers of the API. Participants need not be concerned about systems issues that are hidden behind the abstraction.

2.2 Disadvantages

To be fair, the evaluation-as-a-service model suffers from several challenges:

Limited diversity in retrieval techniques. The biggest drawback of a common service API is that it prescribes a particular architecture for accomplishing the evaluation task. In the case of the TREC Microblog track, the API essentially forces the participants to design their retrieval algorithms around reranking initial results. Despite the fact that such an architecture is commonplace in both industry [3] and academia [7, 1], any abstraction necessarily restricts the flexibility of researchers to tackle the problem in creative ways.

We see this as an issue in at least two ways. First, TREC has developed a community-driven effort to define the retrieval challenges of the day, but (purposely) remains agnostic to how researchers might go about tackling those challenges. It would be a philosophical shift

to define *both* the problem and the methodology (as instantiated in the API). Second, TREC collections are routinely used in ways for which they were not originally intended: although this requires some care, creative uses of TREC resources amplify the usefulness of the data and broadens the impact of the evaluation efforts. The imposition of a common service API makes creative reuse more difficult.

One possible way to mitigate this concern is to institute community-driven processes for refining the API so that researchers' needs are met. Participants could propose new features to be included in subsequent iterations of an evaluation (for example, at the TREC workshop); they could even implement the feature themselves (since the API is open source) and then offer the code contribution for review and integration by the track organizers.

Unknown evaluation characteristics. The information retrieval literature has a tradition of studies that enhance our understanding of the limits of test collections, e.g., their reusability, stability, topic effects, and other related issues (e.g., [11, 13, 2, 9], just to name a few). These studies collectively give us confidence that our evaluation tools can be "trusted". The evaluation-as-a-service model violates some assumptions about how test collections are created, and these issues need to be examined in more detail so that we may gain confidence in the results. For example, previous work (e.g., [2]) has suggested that the reusability of test collections is impacted by the diversity of the original judgment pool. It is likely that pools created using the service API are less diverse because all participants are working from similar result sets. However, since *all* participants, both those who participated in test collection creation and future users of the evaluation resources, must access the collection through the API, it is less clear how we might encourage diversity.

Note that these concerns represent *unknowns* rather than outright problems with the evaluation-as-a-service model. In fact, this model might exhibit all the desirable characteristics (reusability, stability, etc.) of well-crafted, traditional test collections—we just don't know yet. There is a need for studies to examine these issues.

Long-term permanence of service APIs. One essential property of well-built test collections is reusability by researchers who did not participate in their original creation, often long after the initial evaluation. A traditional test collection, once created, requires relatively modest resources to maintain. Relevance judgments are small in size and can be easily hosted on a website (for example, the NIST website, as is customary today); the community has developed sustainable mechanisms for distributing (even large) document collections based on a cost-recovery model (e.g., physically mailing hard drives).

Service APIs, on the other hand, are far more expensive to operate: first, hardware resources must be procured (whether physical servers or infrastructure "in the cloud"); second, humans must be in the loop for both administrative functions (e.g., granting access to new users) and to ensure availability (e.g., restarting the service when it crashes). Longevity is especially a concern—for example, TREC collections from the 1990s are still used today. In the evaluation-as-a-service model, we have not developed a process for sustaining the service API in the long term. Currently, the TREC Microblog organizers have been responsible for maintaining the service, but it would be unrealistic to assume availability in perpetuity. Should NIST take ownership of the service? Or a third-party organization? And if so, when? Ultimately, who pays for machine and people time? These are issues yet to be worked out and reflect the changing roles of NIST and researchers in organizing and running large-scale community evaluations.

3 Extensions and Conclusions

On balance, the evaluation-as-a-service model presents intriguing possibilities for assessing the effectiveness of information retrieval systems. We believe that the disadvantages noted above do not point to fundamental limitations of the evaluation methodology, but rather highlight its immaturity and our inexperience with the approach—none of the issues appear to be insurmountable.

One design decision we made in building the service API was to provide the participants with direct access to the tweet text and associated metadata. This appears to be a sensible choice given that Twitter's native search API delivers comparable data, but for other tasks this may not be tenable.[6] For example, it may be problematic to deliver electronic medical records to evaluation participants. In this case, we see two possible solutions: The first and more restrictive is to return a surrogate of the original document—for example, extracted features that can be subsequently analyzed. Of course, it may be difficult to come up with a useful set of features that doesn't reveal the original document content.

Alternatively, we can imagine pushing the general idea of evaluation as a service even further, by shipping code to the evaluation infrastructure. In this approach, participants would submit their systems, and evaluation would be conducted (either at NIST, at a third-party site, or "in the cloud") without the participants ever touching the sensitive evaluation data.[7] Although this approach is not new – previous examples include the TREC 2005 Spam track [4] and the Music Information Retrieval Evaluation eXchange (MIREX)[8] – the maturation of technologies such as virtualization and standardized RPC mechanisms such as Thrift make this approach easier to implement.

The "evaluation as a service" concept began originally as an attempt to overcome a particular usage restriction on a specific type of data, but has evolved into an evaluation methodology that may have much broader applicability. The model challenges the way our field thinks about evaluation, and hopefully will stimulate advances in our understanding of retrieval systems.

References

[1] N. Asadi and J. Lin. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013)*, pages 997–1000, Dublin, Ireland, 2013.

[2] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004)*, pages 25–32, Sheffield, United Kingdom, 2004.

[6] As a side note: What prevents teams from using the API to crawl the collection? Two answers: First, such behavior is prohibited by the usage agreement of the API. Second, since all accesses are logged, we can monitor the logs for suspicious activity.

[7] How participants would develop systems and algorithms that work on data they don't have access to is another important issue to address, but beyond the scope of this short paper.

[8] `http://www.music-ir.org/mirex/`

[3] B. B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, and J. Degenhardt. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*, pages 411–420, New York, New York, 2010.

[4] G. Cormack and T. Lynam. TREC 2005 spam track overview. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, Maryland, 2005.

[5] D. Harman. *Information Retrieval Evaluation*. Morgan & Claypool Publishers, 2011.

[6] J. Jiang and C. Zhai. An empirical study of tokenization strategies for biomedical information retrieval. *Information Retrieval*, 10(4–5):341–363, 2007.

[7] C. Macdonald, R. L. Santos, and I. Ounis. The whens and hows of learning to rank for web search. *Information Retrieval*, 16(5):584–628, 2013.

[8] I. Ounis, C. Macdonald, J. Lin, and I. Soboroff. Overview of the TREC-2011 Microblog Track. In *Proceedings of the Twentieth Text REtrieval Conference (TREC 2011)*, Gaithersburg, Maryland, 2011.

[9] M. Sanderson and J. Zobel. Information retrieval system evaluation: Effort, sensitivity, and reliability. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005)*, pages 162–169, Salvador, Brazil, 2005.

[10] I. Soboroff, I. Ounis, C. Macdonald, and J. Lin. Overview of the TREC-2012 Microblog Track. In *Proceedings of the Twenty-First Text REtrieval Conference (TREC 2012)*, Gaithersburg, Maryland, 2012.

[11] E. M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998)*, pages 315–323, Melbourne, Australia, 1998.

[12] E. M. Voorhees. The philosophy of information retrieval evaluation. In *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum, Lecture Notes in Computer Science Volume 2406*, pages 355–370, 2002.

[13] E. M. Voorhees and C. Buckley. The effect of topic set size on retrieval experiment error. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 316–323, Tampere, Finland, 2002.