The Pareto Frontier of Utility Models as a Framework for Evaluating Push Notification Systems

Gaurav Baruah and Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada
{gbaruah,jimmylin}@uwaterloo.ca

ABSTRACT

We propose a utility-based framework for the evaluation of push notification systems that monitor document streams for users' topics of interest. Our starting point is that users derive either positive utility (i.e., "gain") or negative utility (i.e., "pain") from consuming system updates. By separately keeping track of these quantities, we can measure system effectiveness in a gain vs. pain tradeoff space. The Pareto Frontier of evaluated systems represents the state of the art: for each system on the frontier, no other system can offer more gain without more pain. Our framework has several advantages: it unifies three previous TREC evaluations, subsumes existing metrics, and provides more insightful analyses. Furthermore, our approach can easily accommodate more refined user models and is extensible to different information-seeking modalities.

1 INTRODUCTION

There is growing interest in push notification systems that prospectively monitor continuous document streams to identify relevant, novel, and timely content, delivered to users as push notifications. The Temporal Summarization [2, 8], Microblog [11], and Real-Time Summarization [12] Tracks at recent Text Retrieval Conferences (TRECs) all explore this basic idea.

We propose a novel framework for evaluating push notification systems that focuses on the Pareto Frontier of a utility model in which users derive positive utility (i.e., gain) from consuming relevant content and negative utility (which we refer to as "pain") from consuming non-relevant content. A system occupies a point in this pain vs. gain tradeoff space, and instead of trying to collapse both quantities into a single-point metric (as previous evaluations attempt to do), we examine the Pareto Frontier of systems in this utility space. Systems that lie on the frontier are all "optimal" in the sense that for a particular level of gain, no other system offers a user experience with less pain.

Our framework makes the following contributions to the evaluation of push notification systems:

- It provides a unified evaluation methodology that encompasses multiple recent TREC evaluations.
- It subsumes previous metrics used in those evaluations.
- It provides more insightful comparisons of system effectiveness that capture different operating points.
- It is extensible to different information-seeking modalities and more refined user models.

This paper details our evaluation framework, applies it to previous TREC evaluations, and elaborates on each of the points above.

2 BACKGROUND AND RELATED WORK

Work on prospective information needs against document streams dates back at least a few decades. Early examples include the TREC Filtering Tracks from 1995 [10] to 2002 [15] and research under the umbrella of topic detection and tracking (TDT) [1]. Over the past few years, we have seen renewed interest in this area, due to the growing popularity of social media and the tighter response cycles demanded by various end users (e.g., journalists, politicians, traders, etc.) to act on high-velocity information.

The TREC Temporal Summarization (TS) Tracks [3, 8], which took place from 2013 to 2015, imagined a stream of newswire articles that a system processed in real time to incrementally select relevant non-redundant sentences pertinent to events of interest such as the 2012 Pakistan garment factory fires or the 2012 Buenos Aires train crash. The putative user of such a system might be a disaster relief coordinator or a journalist, and although somewhat underspecified in the evaluation itself, system output (i.e., "updates") might be proactively delivered to the user via push notifications. Alternatively, based on the model of Baruah et al. [6], users may periodically poll the system for updates.

The Temporal Summarization Tracks evaluated system output in terms of two metrics, expected gain, $(1/|S|) \sum_{s \in S} G(s)$, and comprehensiveness, $(1/|Z|) \sum_{s \in S} G(s)$, where S is the set of updates returned by the system, Z is the number of relevant updates, and G computes the gain for an update. Typically, G factors in redundancy (i.e., systems are not rewarded for returning two updates that "say the same thing"), verbosity (i.e., shorter updates are preferred over longer updates), and also timeliness (i.e., incorporating some type of latency penalty). The exact details varied across track iterations, but the overall form remained constant. Expected gain is analogous to precision and comprehensiveness is analogous to recall, so these two metrics are quite familiar to any IR researcher.

At around the same time as the TS evaluations, the TREC Microblog (MB) Tracks were exploring information needs against social media streams (specifically Twitter). For the evaluation in 2015, systems were given "interest profiles" and the task was to monitor the live Twitter stream to identify relevant and novel tweets in a timely fashion, which were putatively delivered to users as

978-1-4503-4490-6/17/10…$15.00
DOI: http://dx.doi.org/10.1145/3121050.3121089

push notifications. For example, a user might be interested in anecdotes of people injured while playing Pokémon Go and wished to be notified whenever there are such reports. The convergence of the TS and MB Tracks led to their merger in TREC 2016 to form the Real-Time Summarization (RTS) Track, following the basic setup of content filtering on tweets. The novel aspect of the evaluation in 2016 was live user participation, in which system updates were actually delivered as push notifications to users' mobile devices, who provided judgments *in situ* [12, 16].

The MB and RTS evaluations used metrics that were heavily borrowed from the TS evaluations, although the exact formulation differed. Nevertheless, both evaluations measured expected gain as well as cumulative gain (what TS called comprehensiveness). Once again, these are basically variants of precision and recall. The MB and RTS evaluations additionally re-introduced a linear utility metric that dates back to the TREC Filtering Track in 1990s [10, 15]: gain minus pain (GMP), which is a linear interpolation between positive and negative utility based on an arbitrary weight. Our utility-based framework is derived from this formulation, but with a critical difference: instead of interpolating gain and pain to produce a single-point metric, we separately present the two quantities in an effectiveness tradeoff space.

3 EVALUATION MODEL

We assume a task model in which users specify prospective information needs against a stream of discrete text units (sentences, tweets, etc.). The system monitors this stream, and whenever it identifies a relevant piece of text, the user is proactively notified—in operational terms, this might be a push notification to the user's mobile device, a notification pop-up on the user's desktop, or some other mechanism of grabbing the user's attention. Generically, we call these updates. This task setup broadly encompasses the TREC evaluations described in Section 2.

To evaluate such systems, we assume a simple user model: that the user will read each update with some probability p, which we call the persistence parameter. A high persistence value models an "eager" user, and a low persistence value models a user who is likely to ignore the updates. In addition, we assume that unread updates are accumulated in a "holding area" somewhere on the user's device, e.g., a mobile app, a desktop notification tray, etc. Whenever the user examines an update, we assume that the user will also read each subsequent item in the holding area with probability p. The decision is made independently with respect to each item and does not take into account the item's relevance (similar to RBP [14]).

Given this user model, we can construct a utility model for evaluating systems. Whenever the user reads an update, she derives some utility: we refer to positive utility as "gain" and negative utility as "pain". For a particular topic, over a specified evaluation period, a particular system or algorithm delivers to the user a certain amount of pain and gain. We average across multiple topics to arrive at a system's overall effectiveness. To enable meaningful averaging, gain is normalized to the maximum gain based on the evaluation ground truth (e.g., all nuggets in the evaluation pools).[1] Thus, each system can be represented as a point in the gain vs. pain tradeoff space. Below, we explain precisely how gain and pain are operationalized in the three evaluations we examined.

[1]Note that this maximum gain may or may not be achievable based on a particular user model. For example, a low persistence user may never have the opportunity to accumulate much gain, even if every update contains relevant information. This design represents one of many possible choices, each with advantages and disadvantages.

Temporal Summarization. We applied our evaluation framework to data from the TS Track at TREC 2014 (TS14). All the TS evaluations employed a nugget-based evaluation methodology [3], and thus expected gain (i.e., the precision-like metric) and comprehensiveness (i.e., the recall-like metric) were both measured in terms of nuggets. For a given run, TS14 evaluated updates using normalized Expected Gain (nEG) and Comprehensiveness (C).

In our analysis, positive utility (i.e., gain) is accumulated when the user encounters a nugget in an update and negative utility (i.e., pain) is accumulated when the user encounters a non-relevant update. Reading behavior is dictated by the user persistence model described above. We do not penalize gain for latency or verbosity [3, 6] in order to keep our implementation straightforward and intuitive, and pain in this case is simply the number of non-relevant updates consumed. As will be described later, our basic framework can be enriched with more refined user models.

There is one additional detail to note: A few systems in TS14 returned an enormous number of updates (on the order of 10^4 updates per topic). It is inconceivable that so many updates would actually be delivered to a user, and therefore we preprocessed these runs prior to analysis. Fortunately, systems returned a confidence score with each update, and therefore in our experiments only updates above a normalized confidence score of 0.7 are pushed to the simulated user. We set this threshold to balance the volume of system output with the number of unjudged updates; because of the pooled evaluation procedure, most of the updates from the high volume systems were unjudged. This preprocessing was necessary to bring the TS14 runs closer in line to a "reasonable" push notification scenario, although changing the threshold does not impact our findings (since it has the effect of throwing away mostly unjudged updates and affects only the high-volume runs). Updates with a normalized score below 0.7 are assumed to still be available in the "holding area", but in practice since the user consumes these updates in score order, they are rarely read based on our user model.

Microblog and Real-Time Summarization. We analyzed data from the TREC 2015 Microblog Track (MB15) and the TREC 2016 Real-Time Summarization Track (RTS16). For both, positive utility accumulates when the user reads a relevant tweet (defined in the same way as in the official evaluation), and negative utility accumulates when the user reads a non-relevant tweet (as in the official evaluation, just the count of non-relevant tweets).

Since in both MB15 and RTS16, each system was limited to pushing ten tweets per topic per day, we assumed that all system updates were delivered to users. Unread notifications are assumed to be stored in a time-ordered queue on the user's mobile app, to be consumed or ignored based on our persistence model.

4 RESULTS AND ANALYSIS

Evaluation results for TS14, MB15, and RTS16 are shown in Figure 1 based on the pain and gain definitions described in the previous section, modeled with a persistence of $p = 0.5$, i.e., half the push notifications are ignored. Each point represents an individual system in the evaluation.

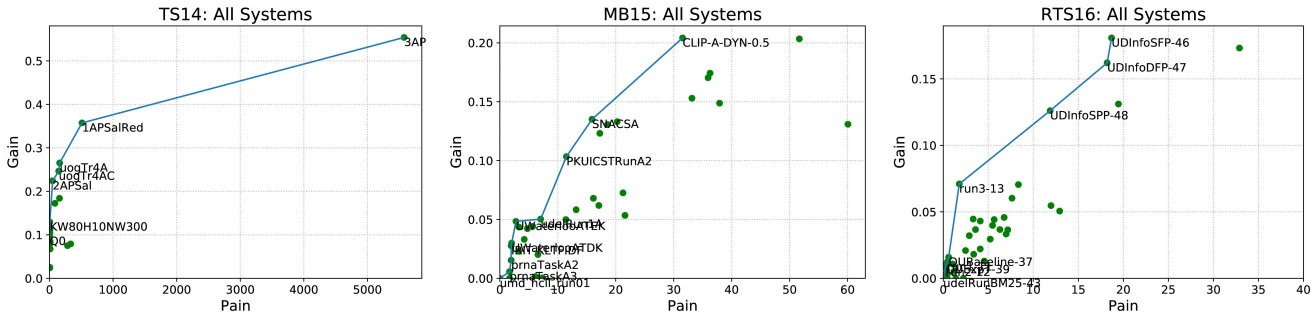

Figure 1: Application of our evaluation framework to the TS14, MB15, and RTS16 evaluations, with persistence $p = 0.5$. The Pareto Frontier represents the best achievable gain for a given level of pain.

The Pareto Frontier (outlined in each figure) contains the set of systems such that, for each system, it is not possible to achieve greater gain without incurring greater pain. Each system along the frontier can claim to be "optimal", in the sense that for the particular level of gain achieved, the system has the minimum pain. Put differently, all systems in the interior are dominated by systems along the frontier, in that for a particular level of gain, there is an alternate system (on the frontier) which provides at least that level of gain with smaller pain—and therefore a rational user would always prefer the system on the frontier. Systems on the frontier, however, are not comparable since they represent different gain vs. pain tradeoffs—deciding between them would require knowing actual user preferences (more below). Naturally, the frontier is not continuous, and hence we interpolate between systems.

In what follows, we elaborate on the four key advantages of our evaluation framework that were outlined in the introduction:

Unified evaluation framework. We provide a unified view across three different TREC evaluations that share underlying similarities, but which up until now have employed different evaluation methodologies. Across these evaluations, we see the broad contours of system design and the tradeoffs in building push notifications: in all cases, the Pareto Frontier rises steeply in the beginning and then falls off. This suggests that there are a number of "obviously good" updates that are fairly easy to identify, but the marginal cost of discovering additional useful updates increases, such that the highest levels of recall can't be achieved (with present systems) without an inordinate amount of pain. This is most apparent in the TS14 evaluation, where the frontier really levels off at higher gain.

Subsumption of previous metrics. In our framework, each system is represented by a point in the gain vs. pain tradeoff space. This position encodes representations that are comparable to all existing metrics for TS, MB, and RTS. Expected gain relates to the slope of the line connecting each point to the origin. Recall metrics (e.g., comprehensiveness) can be visualized as the position on the y axis of each system. Linear utility corresponds to a weighted linear combination of the x and y positions. Furthermore, it is easy to construct iso-metric lines (i.e., lines along which systems have the same metric value): radial lines of different slopes emanating from the origin, parallel horizontal lines, and parallel diagonal lines for the three metrics above, respectively. Thus, our evaluation framework not only unifies but also subsumes previous evaluations—no information about previous metrics is lost.

More insightful analyses. Beyond unifying previous evaluations and subsuming existing metrics, our framework addresses specific issues with previous metrics and supports more insightful analyses. In previous evaluations, precision and linear utility are hard to interpret because they don't factor in update volume: for example, a system that returns 100 "good" and 100 "bad" updates receives the same score as a system that returns 1000 "good" and 1000 "bad" updates. In both cases, the precision is 0.5 and the linear utility is zero (assume equal weight on gain and pain). However, it is obvious that these two systems provide a completely different user experience. If we sort systems in terms of effectiveness by either metric, we are hiding important differences in how systems actually appear to real users.

In fact, any single-point metric necessarily encodes a particular user model representing a specific operating point, which is not ideal because we currently have little evidence of what users actually want in a push notification system (e.g., guidance from user studies). Specifically, the Pareto Frontier recognizes that there are different operating points for push notification systems in the space of pain vs. gain. Some operating points are more plausible than others (for example, in TS14 it is unlikely that users would be willing to endure over 5000 units of pain), but ultimately we need guidance from real-world users. Until then, we argue that it is dangerous to optimize on any single-point metric.

With guidance from our user model, we can explore different usage scenarios in our evaluation framework to conduct more insightful analyses. For example, the current user model contains a persistence parameter, which characterizes the extent to which a user will pay attention to or ignore a notification: we can vary the persistence parameter to model different types of users and note how the Pareto Frontier changes in each case. Figure 2 shows some of these results: we plot the Pareto Frontiers for three different values of persistence $p = \{0.1, 0.5, 0.9\}$. In all three evaluations, we only focus on low-pain systems (note the range of the x axis compared to Figure 1). From the plots, we see that although the Pareto Frontier changes, its overall shape and the systems that lie on the frontier remain similar. Put differently, "good" systems seem to be invariant with respect to different user models, which suggests that our evaluation framework is robust with respect to different parameter settings.

Extensibility. Beyond merely exploring parameters in the existing model, our framework can be extended with richer user and utility

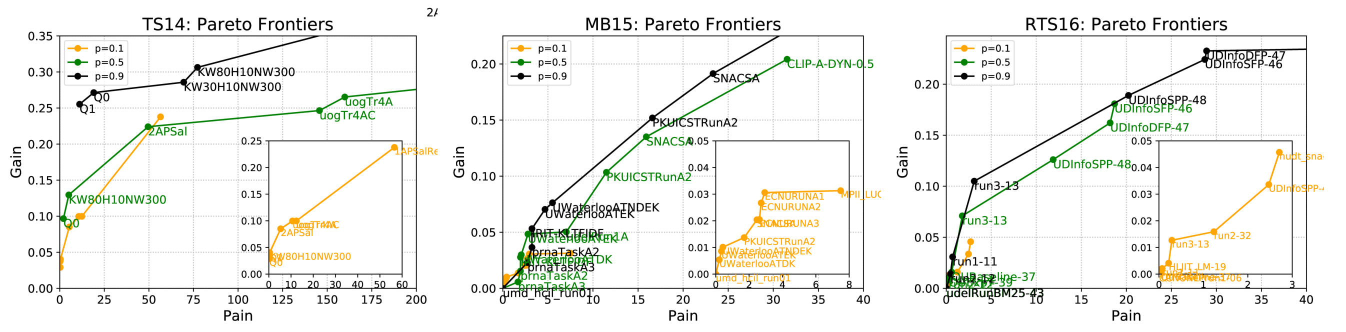

Figure 2: Pareto frontiers for the TS14, MB15, and RTS16 evaluations, modeling users with different persistence, $p = 0.1, 0.5, 0.9$

models based on advances in our understanding of user behavior, operationalized in user simulations [4, 5]. Concrete examples include accurate models of user reading speed [7] in the calculation of pain and gain, time-based calibration of effectiveness measures for more accurately capturing utility [17], incorporating negative higher-order relevance [9], and modeling users' propensity to browse deep down a ranked list of items [13]. These improvements could all be applied to replace our simple persistence model.

In fact, our framework can even accommodate completely different modalities of information seeking with respect to prospective information needs. In this work we assume that system updates are delivered as push notifications, potentially interrupting the user. Alternatively, Baruah et al. [6] considered the scenario where updates accumulate without explicit notification, and a simulated user periodically returns to examine system output (unprompted by the system itself). Pull-based and push-based consumption of updates are merely the ends of a spectrum, and real-world users are likely to employ a combination of both strategies. It is certainly possible to capture these different modalities in our framework using a more refined user model, which would translate into different gain and pain measurements. However, our notion of a gain vs. pain tradeoff space remains relevant, and the Pareto Frontier can still serve as an important construct for understanding the state of the art.

5 FUTURE WORK AND CONCLUSION

Our evaluation framework immediately suggests a gap in our understanding of push notifications systems: What is the pain tolerance of users in operational systems, and how does that tolerance vary with the volume of updates? Based on the latest evaluation results from the TREC 2016 Real-Time Summarization Track, the best systems can achieve a gain vs. pain ratio of roughly one for low volume pushes [12], which corresponds to a precision of 0.5. In other words, roughly half of the content pushed is not relevant. Furthermore, beyond a certain volume, current systems are not able to maintain even a one-to-one gain vs. pain ratio. Is this "good enough"? It seems unlikely to us, but empirically, what is the threshold for a system to be usable in practice?

An important related question concerns update volume [16]: What is the right number of notifications to push within some time interval? One imagines that even for a fast moving topic, users will get annoyed if they are constantly bombarded by notifications, even if all the updates are relevant. On the other hand, a user might also express annoyance at a system for failing to deliver an important update. All of the questions above point to the need for user studies to enrich our understanding of the push notification problem. Our framework can help to contextualize these studies.

Extensions to the user and utility models mentioned in the previous section represent another promising area of future work, as well as modeling hybrid push/pull interactions. We believe that our framework can provide a unified view to evaluate systems that address prospective information needs. Specifically, there is great potential in mixed-initiative interfaces that are both reactive and proactive as appropriate.

REFERENCES

[1] James Allan. 2002. *Topic Detection and Tracking: Event-Based Information Organization.* Kluwer Academic Publishers, Dordrecht, The Netherlands.
[2] Javed Aslam, Matthew Ekstrand-Abueg, Virgil Pavlu, Richard McCreadie, Fernando Diaz, and Tetsuya Sakai. 2014. TREC 2014 Temporal Summarization Track Overview. In *TREC.*
[3] Javed Aslam, Matthew Ekstrand-Abueg, Virgil Pavlu, Fernando Diaz, and Tetsuya Sakai. 2013. TREC 2013 Temporal Summarization. In *TREC.*
[4] Leif Azzopardi. 2016. Simulation of Interaction: A Tutorial on Modelling and Simulating User Interaction and Search Behaviour. In *SIGIR.* 1227–1230.
[5] Leif Azzopardi, Kalervo Järvelin, Jaap Kamps, and Mark D. Smucker. 2011. Report on the SIGIR 2010 Workshop on the Simulation of Interaction. *SIGIR Forum* 44, 2 (2011), 35–47.
[6] Gaurav Baruah, Mark D. Smucker, and Charles L. A. Clarke. 2015. Evaluating Streams of Evolving News Events. In *SIGIR.* 675–684.
[7] Charles L. A. Clarke and Mark D. Smucker. 2014. Time Well Spent. In *IIiX '14.* 205–214.
[8] Qi Guo, Fernando Diaz, and Elad Yom-Tov. 2013. Updating Users about Time Critical Events. In *ECIR.* 483–494.
[9] Heikki Keskustalo, Kalervo Järvelin, Ari Pirkola, and Jaana Kekäläinen. 2008. Intuition-Supporting Visualization of Userfis Performance Based on Explicit Negative Higher-Order Relevance. In *SIGIR.* 675–682.
[10] David D. Lewis. 1995. The TREC-4 Filtering Track. In *TREC.* 165–180.
[11] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. 2015. Overview of the TREC-2015 Microblog Track. In *TREC.*
[12] Jimmy Lin, Adam Roegiest, Luchen Tan, Richard McCreadie, Ellen Voorhees, and Fernando Diaz. 2016. Overview of the TREC 2016 Real-Time Summarization Track. In *TREC.*
[13] David Maxwell, Leif Azzopardi, Kalervo Järvelin, and Heikki Keskustalo. 2015. Searching and Stopping: An Analysis of Stopping Rules and Strategies. In *CIKM.* 313–322.
[14] Alistair Moffat and Justin Zobel. 2008. Rank-Biased Precision for Measurement of Retrieval Effectiveness. *ACM TOIS* 27, 1, Article 2 (Dec. 2008), 27 pages.
[15] Stephen Robertson and Ian Soboroff. 2002. The TREC 2002 Filtering Track Report. In *TREC.*
[16] Adam Roegiest, Luchen Tan, and Jimmy Lin. 2017. Online In-Situ Interleaved Evaluation of Real-Time Push Notification Systems. In *SIGIR.*
[17] Mark D. Smucker and Charles L. A. Clarke. 2012. Time-based Calibration of Effectiveness Measures. In *SIGIR.* 95–104.

Acknowledgments. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, with additional contributions from the U.S. National Science Foundation under CNS-1405688.