Update Delivery Mechanisms for Prospective Information Needs: An Analysis of Attention in Mobile Users

Jimmy Lin, Salman Mohammed, Royal Sequiera, and Luchen Tan
David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada

ABSTRACT

Real-time summarization systems that monitor document streams to identify relevant content have a few options for delivering system updates to users. In a mobile context, systems could send push notifications to users' mobile devices, hoping to grab their attention immediately. Alternatively, systems could silently deposit updates into "inboxes" that users can access at their leisure. We refer to these mechanisms as push-based vs. pull-based, and present a two-year contrastive study that attempts to understand the effects of the delivery mechanism on mobile user behavior, in the context of the TREC Real-Time Summarization Tracks. Through a cluster analysis, we are able to identify three distinct and coherent patterns of behavior. As expected, we find that users are likely to ignore push notifications, but for those updates that users *do* pay attention to, content is consumed within a short amount of time. Interestingly, users bombarded with push notifications are less likely to consume updates on their own initiative and less likely to engage in long reading sessions—which is a common pattern for users who pull content from their inboxes. We characterize users as exhibiting "eager" or "apathetic" information consumption behavior as an explanation of these observations, and attempt to operationalize our findings into design recommendations.

ACM Reference Format:
Jimmy Lin, Salman Mohammed, Royal Sequiera, and Luchen Tan. 2018. Update Delivery Mechanisms for Prospective Information Needs: An Analysis of Attention in Mobile Users. In *SIGIR '18: 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8-12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210018

1 INTRODUCTION

Prospective information needs are typically posed against document streams such as posts on Twitter, RSS feeds, newswire articles, etc. For example, a user might be concerned about tensions on the Korean Peninsula and wishes to receive updates whenever there is a new development. Real-time summarization systems monitor these streams and identify documents that are relevant, typically operationalized as on-topic, non-redundant, and timely. The nature of prospective information needs means that relevant documents may appear at any time, and thus an open question is how system output should be delivered to users. We explore this question in the context of mobile devices that are ubiquitous today.

There are two basic mechanisms for delivering updates from real-time summarization systems: In one approach, an update can be delivered to a user's mobile device as a push notification, typically accompanied by a visual and/or auditory signal that purposely interrupts the user to attract attention. Alternatively, updates can be silently deposited into an "inbox", waiting for the user to examine on the user's own initiative (much like email). These two mechanisms can be characterized as either "push-based" or "pull-based"—terminology we will use throughout this paper. Although in reality a system can adopt both mechanisms, for simplicity we examine either approach in isolation.

We present a two-year study drawn from the TREC Real-Time Summarization (RTS) Tracks involving over 50 users who evaluated live system updates on their mobile devices *in situ*, i.e., as they were going about their daily lives. These users interacted with a mobile app that employed either the push- or pull-based mechanism described above. We are specifically interested in three research questions focused on users' attention to system updates:

(RQ1) How do users in the push vs. pull treatments differ in terms of quantifiable behavioral characteristics?
(RQ2) Can we generalize patterns of user behavior within and across push- and pull-based update delivery?
(RQ3) Can we operationalize our findings into design recommendations for developers of real-time summarization systems?

Contributions. In answering the above research questions, we view our work as having the following contributions:

- We present, to our knowledge, the first contrastive study of push- vs. pull-based delivery mechanisms for addressing prospective information needs for mobile users. Since the main experimental manipulation was the delivery mechanism, we can attribute differences in user behavior to the system treatment.
- We present a novel methodology for data analysis that exploits different visualizations to empirically characterize *how much* and *when* users pay attention to updates from systems with different delivery mechanisms.
- We identify, via a clustering analysis, distinct and coherent patterns of user behaviors and the impact of delivery mechanisms. We hypothesize that users can be characterized as *eager* or *apathetic* with respect to information consumption, which provides an explanation of observed behaviors.
- We operationalize our findings into concrete design recommendations for system developers, identifying categories of users for which push notifications might or might not be valuable.

As an initial foray into information delivery mechanisms on mobile devices for real-time summarization systems, our work raises as many interesting questions as it answers. We are forthright in discussing the limitations of our study, most of which point directly to future avenues of exploration.

2 BACKGROUND AND RELATED WORK

The literature on prospective information needs dates back to the 1960s, with work on "selective dissemination of information" to help scientists stay up to date on relevant literature [8, 9]. The relationship between prospective and retrospective (i.e., *ad hoc*) information needs has long been noted [4], although the former has received far less attention over the years. However, two related research threads beginning in the 1990s substantially advanced our state of knowledge. The TREC Filtering Tracks, which ran from 1995 [11] to 2002 [17], brought modern evaluation methodologies and shared evaluations to bear on filtering document streams. The task can be best understood as binary classification on *every* document in the collection with respect to standing queries. The research program known as Topic Detection and Tracking (TDT) encompassed a number of related tasks around streams of news stories [1], including event detection, tracking, and linking. The Temporal Summarization Tracks at TREC, which ran from 2013 to 2015 [2], represented the evolution of TREC Filtering and TDT; the goal was to generate concise update summaries from news sources about unexpected events as they developed [7].

The Real-Time Summarization (RTS) Track at TREC, which provides the context for this work, is a direct descent of Temporal Summarization and captures a number of important differences compared to TREC Filtering and TDT. Both were formulated with an intelligence analyst in mind, with an emphasis on recall—e.g., in filtering the goal is to identify *all* relevant documents. In contrast, for delivery to mobile devices, we are concerned with selecting only a small set of the most relevant updates. Furthermore, in TREC Filtering and TDT, systems were forced to make immediate decisions about incoming documents; instead, real-time summarization systems are given the option of delaying decisions, which allows tradeoffs between output quality and timeliness.

Another major difference is that previous evaluations, including TDT, TREC Filtering, and Temporal Summarization, merely *simulated* the streaming nature of the document collection, whereas in RTS the participants were required to build working systems that operated on tweets posted in real time. This enabled an evaluation setup whereby system updates are delivered directly to users, matching real-world deployment scenarios. The RTS Track is an example of a "living labs" approach [19, 20] that attempts to evaluate systems in more realistic settings.

There is a rich body of work on interruptions in general and mobile push notifications in particular from the perspective of human–computer interaction [14]. However, we are not aware of any previous work specifically on prospective information needs, and thus our study represents a novel contribution to the literature.

Despite the term "summarization" in RTS, the systems we consider are very different from the multi-document summarization tasks conceived by the NLP community (e.g., DUC evaluations [5]). Those summarization tasks are retrospective in nature, and the goal is for a system to construct summaries of varying lengths "on demand". In contrast, RTS is designed to deliver relevant, timely, and novel updates incrementally over time.

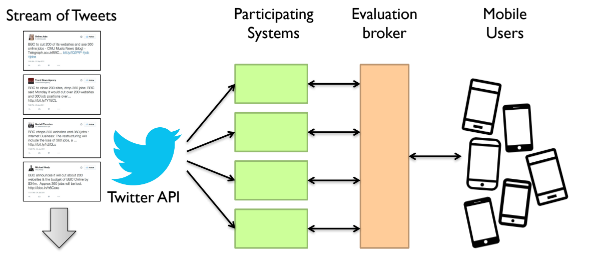

Figure 1: The RTS evaluation setup: systems "listen" to the live Twitter sample stream and send results to the evaluation broker, which then delivers results to users, either via push notifications or silent deposit into users' inboxes.

3 EVALUATION METHODOLOGY

The Real-Time Summarization (RTS) Tracks at TREC 2016 and 2017, for which we were co-organizers, deployed a "living labs" evaluation setup whereby real users were recruited to provide relevance judgments *in situ* on their mobile devices. In this paper, we focus only on the "Scenario A" task, whose overall design is shown in Figure 1. Although the point of the RTS Track was to evaluate systems using judgments from the mobile users, in this study we analyze the behavior of the mobile users, treating the systems as black boxes. Here, we present the salient aspects of the evaluation methodology, referring readers to the track overviews for details [12, 13].

The evaluations occurred at pre-specified intervals during the summer of 2016 (10 days) and the summer of 2017 (8 days). The prospective information needs (called interest profiles) were distributed to participating groups well in advance. Due to their time-dependent nature, different sets of interest profiles were used in each evaluation. A few weeks prior to the beginning of the evaluation period, we recruited mobile users: In 2016, the recruitment pool consisted of undergraduate and graduate students at the University of Waterloo. In 2017, the subject pool was additionally expanded to include developers of the participating systems. In total, 13 users participated in 2016 and 42 in 2017. All users were compensated $1 per 20 judgments (the same in both years). As part of the recruitment process, users were asked to "subscribe" to interest profiles, i.e., to indicate which ones they desired system updates for. In 2016, there were 203 profiles to choose from, all created by the organizers. In 2017, there were 188 profiles to choose from, a combination of organizer-created and user-suggested profiles. A large and diverse set of profiles meant that users were likely able to find profiles they were genuinely interested in.

In the 2016 evaluation, 18 participating groups submitted 41 system runs; in 2017, 15 groups submitted 41 runs. During the evaluation period, whenever a system identified a relevant tweet with respect to an interest profile, the system submitted the tweet id to the evaluation broker (see Figure 1). Each system was allowed to submit at most ten tweets per interest profile per day as not to overwhelm the user. The evaluation broker then "delivered" the update (tweet) to the mobile users who had subscribed to the interest profile, following the temporal interleaving strategy proposed by

Qian et al. [16] and further examined by Roegiest et al. [18]. Note that each update was delivered only once, even if it was submitted by multiple systems at different times. The main difference between the two evaluations was the update delivery mechanism:

- In 2016 (RTS16), the mobile users installed a custom assessment app, and each delivered update was *immediately* rendered as a push notification containing both the tweet and the corresponding interest profile. Modulo device-specific settings, each notification was accompanied by an audible and/or tactile signal specifically designed to attract the user's attention. The user may choose to attend to the update immediately or to ignore it. Either way, the update is added to an assessment queue in the app on the user's mobile device. The user can access the app at any time to examine the queue of accumulated updates in reverse chronological order (i.e., most recent first). We call this the push condition, since system updates are directly pushed to users.
- In 2017 (RTS17), delivered updates were deposited in LIFO (Last-In, First-Out) queues we call the users' inboxes. This delivery was not accompanied by any attempt to interrupt the user. They could, anytime they desired, visit a web app (different from the app used in 2016) on their mobile devices to examine the queue of accumulated updates. The users were shown updates in reverse chronological order, and so the presentation order was the same in both years. We call this the pull condition, since the users are pulling updates from their inboxes at their leisure.

Irrespective of the delivery mechanism for each update (tweet), the user can make one of three judgments with respect to the associated interest profile: *relevant*, if the update contains relevant and novel information; *redundant*, if the update is relevant but contains redundant information (with respect to previously-seen updates); *not relevant*, if the update does not contain relevant information. However, as we explain below, our study does not specifically make use of these judgments.

One salient characteristic of the evaluation methodology is its *in situ* design, which realistically mirrors real-world deployment scenarios since users are going about their daily lives throughout the multi-day evaluation period. The updates are delivered in real time, but judgments are provided at the users' leisure under the two system treatments. Of course, some updates are never seen. How many updates a user consumes and the delay between update delivery and user action are key aspects of user attention that we examine, detailed next.

4 POPULATION-LEVEL ANALYSES

Given the experimental setup described above, this paper examines user attention to system updates under different delivery mechanisms (push vs. pull). To be precise, we recorded a user as "paying attention" or "consuming" an update as the moment the user submitted a relevance judgment in the mobile app. In actuality, though, these are distinct events, since the user first notices the update, reads the content, and finally provides a judgment. However, since tweets are short, we believe that reading time can be ignored without affecting our overall findings. Thus, in our analyses, we refer to "consuming an update" and "providing a relevance judgment" interchangeably as the point in time when a user "paid attention" to a system update.

To simplify analyses, we do not consider the actual relevance judgment, since consuming an update incurs effort and cost whether or not the update is relevant. For push notifications, the user has already been interrupted. For the pull condition, we note that effort models for batch relevance assessment in the literature (e.g., [21]) do not distinguish between judgment grades, so this is a reasonable simplification. Nevertheless, we return to discuss system quality as a confounding factor in Section 7. In this section, we focus on population-level analyses, and then use these findings to guide closer examination of individual users in Section 5.

4.1 Volume and Response Rate

We begin by defining volume and response rate, two straightforward ways of characterizing user behavior:

- **Volume** refers to the number of updates that a user has potentially been exposed to, which is affected by the number of interest profiles the user subscribed to as well as the nature of those information needs. In the push condition (RTS16), volume forms the upper bound on the number of push notifications the user *could* have received during the evaluation period. In the pull condition (RTS17), volume refers to the total number of messages that was deposited in the user's inbox during the evaluation period.
- **Response rate** refers to the proportion of updates that a user actually paid attention to (i.e., provided a relevance judgment).

Figure 2 shows the volume and response rates for 13 users in the push condition (RTS16, left) and 42 users in the pull condition (RTS17, right). The total height of each blue bar indicates volume and each orange bar indicates the number of updates judged, so the response rate is the ratio between the two. The bars are also annotated with the number of interest profiles that each user subscribed to and sorted by the total number of judgments provided. Each user is identified by a three letter alphanumeric code, prefixed with either RTS16 or RTS17—these identifiers are used to refer to specific users anonymously throughout the paper. We see that in both cases, a few users contributed many judgments, while many users contributed relatively few. This comes as no surprise, as the distribution of judgments is consistent with crowd-sourced relevance judgments, e.g., via Amazon Mechanical Turk [6].

It is quite obvious that the response rate for the push condition is much lower than for the pull condition. For the push condition, the maximum response rate was 46% for a volume of 7141 updates; only four (out of 13) users had response rates greater than 20%. For the pull condition, there were 9 users (out of 42) who consumed over 90% of the updates; of these, the maximum volume was 14477 updates. This finding makes sense as push notifications are disruptive. As a point of reference for the push condition, 5000 updates over an evaluation period of ten days translates into an average of ~21 messages per hour (assuming a constant arrival rate). Factoring in sleeping time, we would already consider a response rate of nearly 50% (the most "diligent" user in RTS16) quite impressive.

The dramatic differences in response rates can be seen in Figure 3, which shows histograms of the response rates for both the push (green) and pull (purple) conditions, bucketed in increments of 10%. There is insufficient data to draw inferences about the shapes of the distributions, except to confirm that response rates are much lower for the push condition.

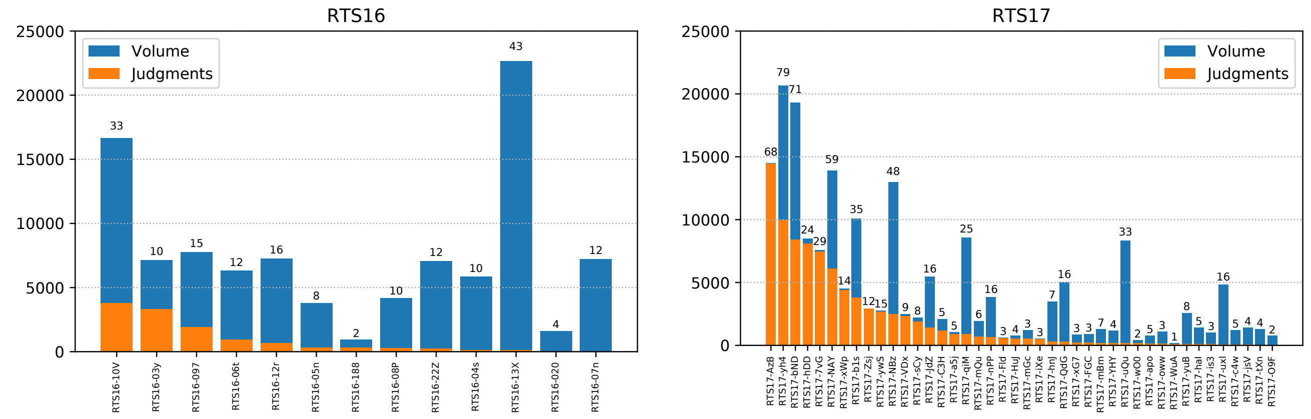

Figure 2: The volume and number of judgments received for the push condition (left) and the pull condition (right). Bars are sorted by the number of judgments provided and annotated with the number of profiles the user subscribed to.

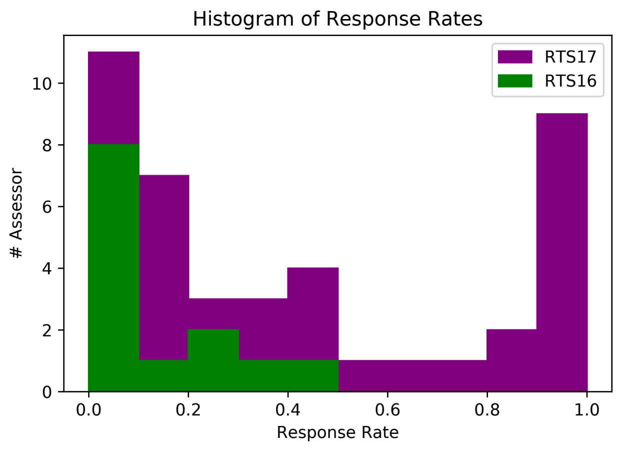

Figure 3: Histogram of response rates for RTS16 (push) and RTS17 (pull), bucketed in 10% increments.

Interestingly, it appears that volume and response rate vary independently in both the push and pull conditions: For example, in RTS16, many users subscribed to a similar number of profiles and hence received similar numbers of updates; yet their response rates varied greatly. In RTS17, we see users who received a high volume of updates (by subscribing to many profiles). In some cases, they attended to nearly all updates, and in other cases they ignored most of them. On the flip side, we see users who received a low volume of updates but attended to nearly all of them, and users who still ignored most updates.

4.2 Response Profiles

We define *response delay*, or simply *delay*, as the difference in time from when an update was delivered to when the user paid attention to it. In other words, how long did it take the user to notice and consume the update? To better understand user behavior, we can empirically characterize the temporal distribution of these delays. To simplify analyses, we bucket delays into one hour blocks—that is, how many updates did the user attend to within the first hour, the second hour, etc. under either the push or pull conditions? We can normalize the absolute counts to yield a probability distribution. We refer to these as *response profiles*, which characterize empirically *when* each user pays attention to system updates. Note that compensation for participants in our studies was not tied to how quickly they provided judgments, and therefore these delays are not the result of experimental manipulation.

In Figure 4, we visualize these probability distributions as a heatmap. Each column represents a specific user: the push condition (RTS16) on the left and the pull condition (RTS17) on the right. Each row represents a particular hour; there are 13 rows in total: the first 12 represent individual hours, while the last row captures the remaining probability mass (longer than 12 hours). Our rationale for lumping all delays longer than 12 hours together is that beyond a certain point, we are operating outside the intended design space of real-time summarization systems. In the heatmap, color is used to encode the amount of probability mass in each condition—darker reds indicate more mass. Within each condition (push vs. pull), the columns (users) are sorted in ascending order of the amount of probability mass in the first row (i.e., the proportion of messages that the user paid attention to in the first hour).

From the heatmap, big differences between the push and pull conditions are readily apparent. In the push condition, for most users, the probability mass is concentrated in the first row, which means that users pay attention to updates within the first hour. This is expected: if systems interrupt users with push notifications, they will indeed pay attention—and therefore the delays are generally short. For a system designed to address prospective information needs, short delays seem like a desirable outcome. However, this must be balanced against a much lower response rate in the push condition (Figures 2 and 3). In fact, most updates are never seen at all. For users in the pull condition, we see a broad range in terms of the amount of probability mass in the first row. Interestingly, we also observe significant mass in the 12+ row for many users—we'll come back and expand on this observation later.

More formally, we can visualize the amount of mass in the first row of the heatmap in an empirical distribution function (EDF), shown in Figure 5 (separately for the push and pull conditions), where the y axis represents the cumulative probability and the x

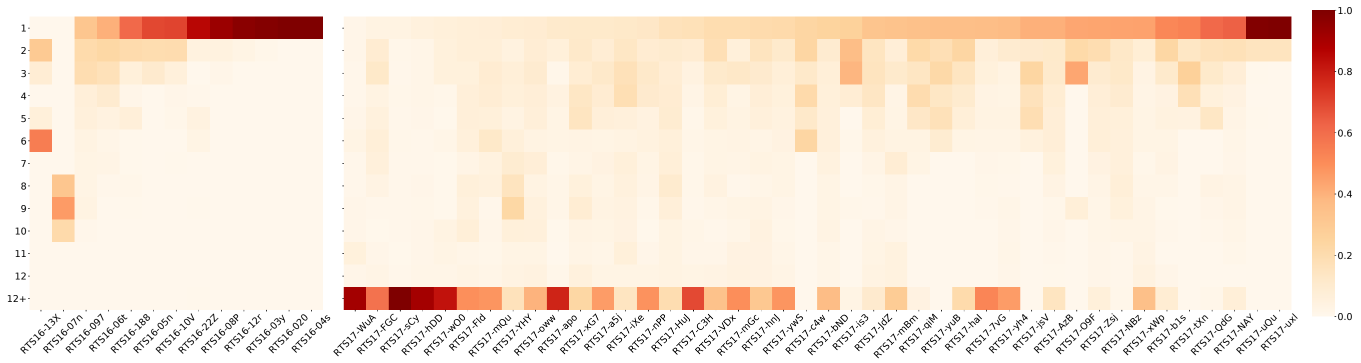

Figure 4: Heatmap visualizing user response profiles: for each user (column), each cell shows the fraction of updates consumed in the *n*-th hour (row) after the system update was delivered. The last row represents delays more than 12 hours. The push condition (RTS16) is shown on the left, the pull condition (RTS17) on the right.

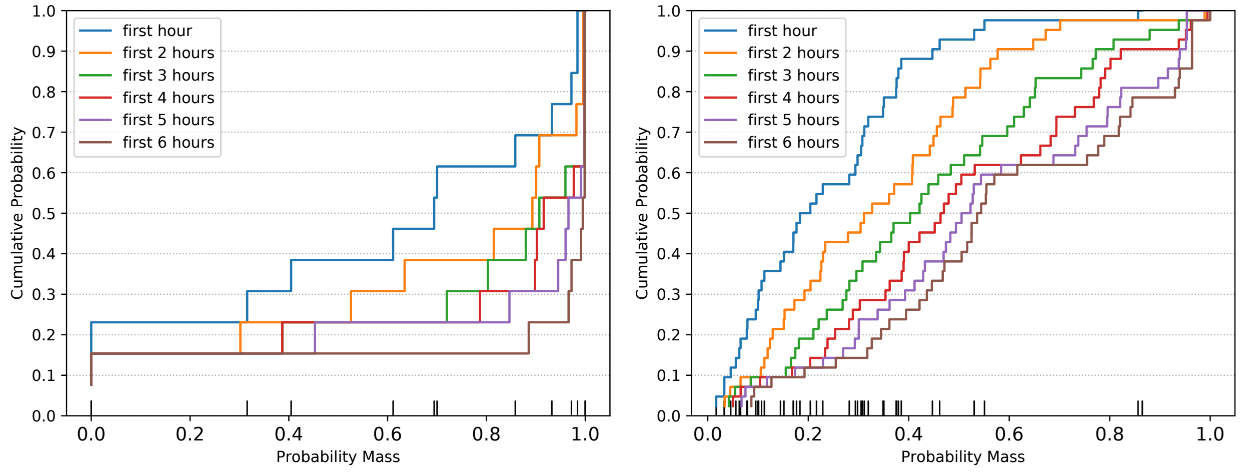

Figure 5: Empirical distribution functions (EDFs) of the fraction of updates consumed within *n* hours under the push (left) and pull (right) conditions. Data for the blue lines (first hour) correspond to the first row of the heatmap in Figure 4; ticks on the bottom represent the probability mass in the first hour of those individual users.

axis represents the random variable—in this case, the proportion of updates consumed in the first hour (the top blue line). Each black tick at the bottom of each plot represents the probability mass in the first hour for an individual user. If we assume that our users are iid sampled from a particular population, then the EDF coarsely approximates the true CDF of the population (and the EDF converges to the CDF as the number of samples increases). If we further assume that the evaluations are drawing from a homogeneous population (more discussion in Section 7), then the differences between the EDFs can be attributed to differences in system treatment (i.e., delivery mechanism).

Focusing for now only on the top blue line (the first hour): How can we interpret these EDFs? It is more intuitive to consider one minus the cumulative distribution: these distributions show the fraction of users ($1 - y$ axis value) that have paid attention to at least some fraction of the updates in the first hour. For example, under the push condition, 20% of users or the top quintile (corresponding to 0.8 on the y axis) attended to at least ~90% of the updates. In contrast, under the pull condition, the top quintile attended to at least ~35% of the updates (a much lower bound). The median user ($y = 0.5$) in the push condition paid attention to at least ~70% of updates in the first hour; the median user in the pull condition paid attention to at least ~20% of updates during the same period. It is worth emphasizing that these EDFs show delays for updates that the user paid attention to; the plots say nothing about updates that were never seen. Although users in the push condition have shorter delays, their response rates are also much lower.

Our analysis can be extended beyond the first hour to the *n*-th hour: that is, we can plot the EDF of users paying attention to updates within *n* hours. These curves for the push and pull conditions are shown in Figure 5, for each hour up to six hours (in different colors). The "rightward shift" of the curves indicate that for every passing hour of delay, more and more users pay attention to system updates. Within six hours, under the push condition, the median user ($y = 0.5$) would have consumed nearly all updates (that the user is ever going to read), while under the pull condition, the user would have only consumed around half of the updates (albeit out of a larger fraction of the total volume).

The combination of this analysis and the previous section forms one contribution of this paper—a novel methodology (using heatmaps and EDFs) to empirically characterize *how much* and *when* users pay attention to updates from real-time summarization systems under different delivery mechanisms. These findings provide an answer to (RQ1) "How do users in the push vs. pull treatments differ in terms of quantifiable behavioral characteristics?" There are quite obvious and easily quantifiable differences in user behavior: Overall, users have much lower response rates when bombarded with push notifications, but for the updates they do pay attention to, information is consumed quickly (typically within an hour).

4.3 Clustering Response Profiles

It is no surprise that push notifications lead to shorter response delays. What *is* surprising, though, is that even in the pull condition, there are users whose response profiles have probability mass concentrated in the early hours. Examining the heatmap in Figure 4, there are a few users from the pull condition whose behaviors are quite similar to those from the push condition, except that these users *are not receiving push notifications*!

We can formalize this observation by identifying clusters of users who have similar response profiles, i.e., by clustering the columns in the heatmap in Figure 4. This is accomplished by treating each profile (i.e., the bucketed distributions) as a feature vector and applying hierarchical clustering using cosine similarity as the distance metric (with average link merging). With this analysis we answer (RQ2): Can we generalize patterns of user behavior within and across push- and pull-based update delivery?

Results are presented in Figure 6 as a dendrogram showing the sequence of hierarchical merges that comprise the clustering. Each row is exactly the same as the corresponding column in Figure 4, except rotated; the leftmost column is a color code for presentational convenience. From the hierarchical structure of the merges we can identify three distinct clusters of profiles:

- *Early-heavy distributions.* In these response profiles (coded orange), probability mass is concentrated in the early hours, with short response delays. Within this cluster, however, we still observe gradations—some profiles have more mass in the early hours than others. These users are "eager" in consuming updates, typically with more frequent but shorter sessions.
- *Late-heavy distributions.* In these profiles (coded purple), the probability mass is concentrated in the 12+ bucket, meaning that these users do not consume updates until long after they have been delivered. However, we do observe degrees of late "heaviness", i.e., some profiles have more mass in the 12+ bucket than others. There is inevitably probability mass in the early hours because a user encounters older updates in the assessment interface only after consuming more recent updates (by design). These users are "apathetic" in consuming updates, with much fewer but longer sessions.
- *Bimodal distributions.* Interestingly, we observe a distinct third cluster of response profiles (coded blue), where the distribution of delays is bimodal. There is a non-negligible amount of probability mass in both the early bucket (first hour) and the late bucket (i.e., 12+ hours). In other words, these users consume some updates soon after delivery, but they also consume a significant number of old updates. These users appear to be *both* eager and apathetic, with a combination of long and short sessions.

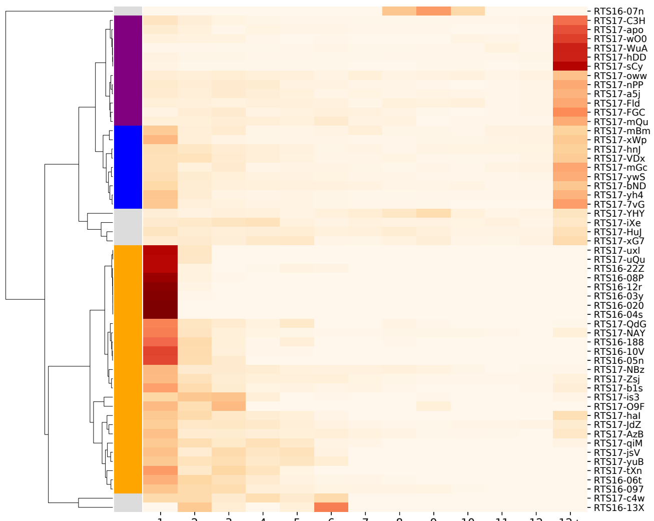

Figure 6: Hierarchical clustering of response profiles for the push and pull conditions, showing early-heavy (orange), late-heavy (purple), and bimodal (blue) distributions.

It is worth emphasizing that these clusters (our color codes) are based on the induced hierarchical structures generated by the clustering algorithm. This lends credence to their empirical validity, as opposed to subjective "inkblot reading" by the authors. There are seven outliers that appear to defy our characterizations, which we have color-coded gray; these users, however, for the most part, provided few judgments. Two provided less than 100 judgments during the entire evaluation period, and all except for one provided less than 500 judgments.

Note that in this analysis we applied clustering over users from both the push (RTS16) and pull (RTS17) conditions, which means that the observed user behaviors represent a complex mix of users' intrinsic characteristics as well as the system treatment (i.e., push vs. pull). Nevertheless, two interesting observations jump out:

Observation #1: *Some users in the pull condition behave as if they were in the push condition.* The cluster of early-heavy distributions (orange) contains a mix of users from both the push and pull conditions. For RTS17 (pull condition) users, this means that they paid attention to most system updates within a short amount of time *without* the interruption of push notifications. As we examine more closely in Section 5, these users appear to be frequently returning to their inboxes without an external prompt.

Observation #2: *No users in the push condition exhibit late-heavy and bimodal delay distributions.* These findings are related, in that the second follows from the first. In other words, when a system pushes a notification, the user either pays attention to it within a short amount of time or never looks at it. As a result, we also do not see late-heavy or bimodal distributions.

Let us add some statistical rigor to this observation: Users can be classified into four categories (the three profiles above and a fourth "other" category corresponding to the rows color-coded gray), and we wish to know if users from RTS16 and RTS17 are sampled from the same underlying distribution. Typically, Pearson's χ^2 test of homogeneity is the appropriate statistical test to apply, but our sample size (particularly RTS16) is smaller than the threshold where the χ^2 test is generally considered reliable. Instead, we applied Fisher's exact test, which indicates that we can reject the null hypothesis ($p < 0.01$) that the RTS16 and RTS17 observations are drawn from the same distribution. The biggest difference between these evaluations is the delivery mechanism (push vs. pull), although in Section 7 we discuss and rule out some possible confounds.

5 INDIVIDUAL-LEVEL ANALYSES

The previous section focused on population-level analyses of user behavior, where we are able to broadly characterize the effects of push vs. pull delivery. Clustering the delay distributions allowed us to identify three distinct types of user response profiles. However, such a coarse-grained analysis does not give us insight into what users are actually doing. In this section, we examine in detail the behavior of a few users, guided by the previous analyses.

5.1 Early-Heavy Distributions

In Figure 7 we show details of two users from the early-heavy distribution cluster: RTS16-03y (top) and RTS17-NAY (bottom). We refer the reader to the cluster dendrogram in Figure 6 to see how these two users relate to each other and the other users.[1]

The timeline scatterplots visualize all updates that each user consumed, i.e., each dot represents a judgment. Time (in UTC) runs along the y axis, from the top and flowing downward, measured in days from the beginning of the evaluation period. The x position indicates the response delay. The dots are superimposed on top of the delay distribution. As with all previous analyses, the distribution is divided into 13 buckets, one for each hour up to 12, and the final one for 12+ hours. In other words, the histogram shows exactly the same information as each row in Figure 6 (the cluster dendrogram) and each column in Figure 4 (the heatmap).

In these timelines, each horizontal "run" of blue dots represents a session with our mobile assessment app, where the width of each run indicates the number of judgments provided in that session. It is not immediately obvious why the sessions have time gaps, since users always consume updates in reverse chronological order and cannot encounter older updates unless they have assessed all earlier ones (by design). Such gaps arise when users consume many updates in a session and "overrun" their previous sessions temporally. This is illustrated in Figure 8, where the small squares represent delivered updates arranged on a timeline. At time A (the red block arrow), the user examined five updates, marked in yellow. The user then returned at time B (the green block arrow) and examined all the updates colored blue. After that, the interface would show older updates (shown in tan) and if the user continued to assess those updates, this "jump" in the age of the updates would show up as a gap in the response delays.

For user RTS16-03y (Figure 7, top), we see many frequent but short sessions. This user had a response rate of 46% (the highest),

[1]Our clustering analysis identified RTS17-xul and RTS17-uQu as more similar to the RTS16 users. However, these users supplied relatively few judgments, and thus we opted to present a user who was more engaged.

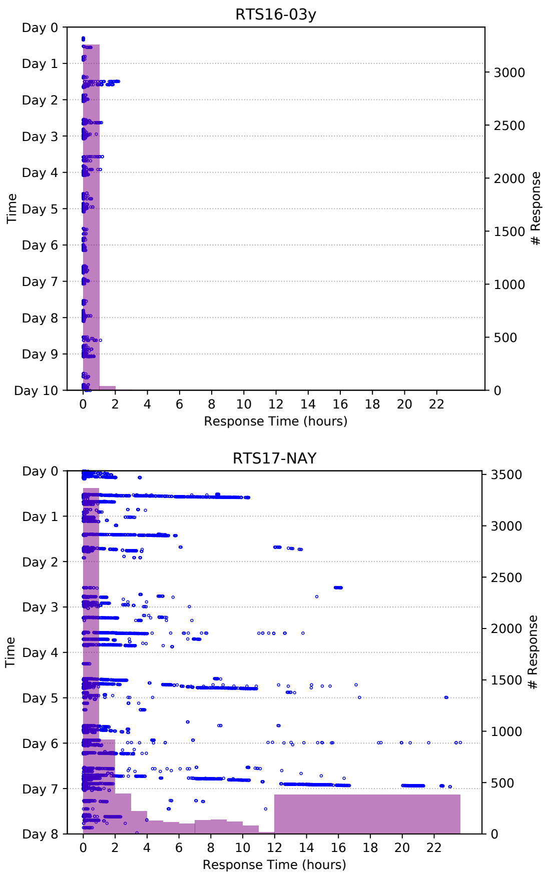

Figure 7: Two users with early-heavy delay distributions, RTS16-03y (top) and RTS17-NAY (bottom): frequent but (relatively) short sessions.

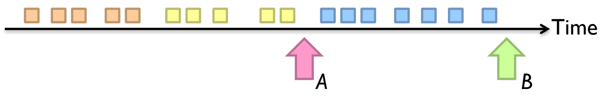

Figure 8: Hypothetical example showing how "gaps" may arise in the timeline scatterplots shown in Figure 7.

subscribed to 10 profiles, and had a volume of 7141 updates. Nearly all updates were consumed within an hour after system delivery (in most cases, *well* within an hour); the user did not consume any updates older than two hours. It appears that the user was reacting quickly to the push notifications. However, because of the frequent interruptions, the sessions were never very long.

User RTS17-NAY (Figure 7, bottom) shares many similarities with RTS16-03y in having many short sessions (note that the x axis has the same scale). This user subscribed to 59 profiles and had a response rate of 44% on a volume of 13.9k updates. The vast majority of updates were consumed within two hours, which yields an early-heavy delay distribution. For this reason, our clustering analysis identified these two profiles as similar. While there are a number of longer sessions, their visual prominence in the timeline makes them seem more frequent than they actually are. From the

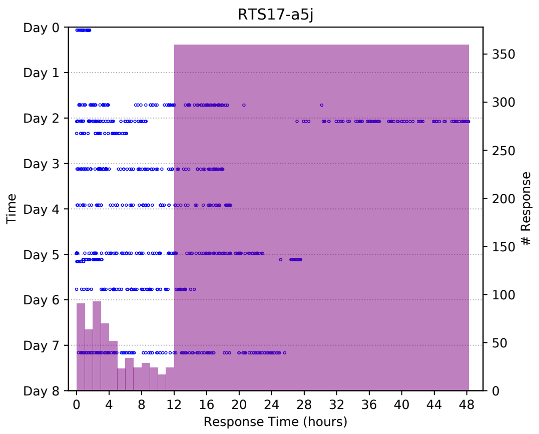

Figure 9: A user with a late-heavy delay distribution from RTS17: few but long sessions.

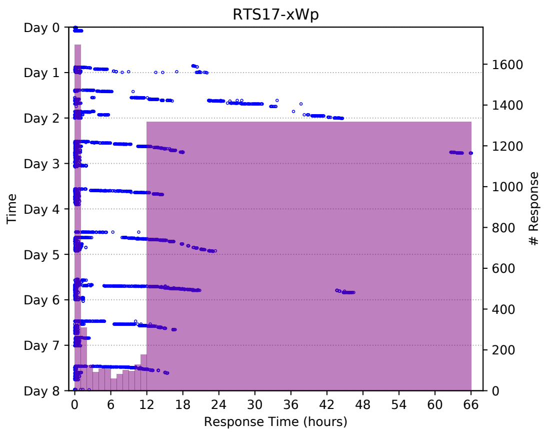

Figure 10: A user with a bimodal delay distribution from RTS17: a combination of long and short sessions.

histogram, we see that there is relatively little probability mass in the 12+ bucket, so it is accurate to say that this user engaged in predominantly short sessions.

It is worth emphasizing that RTS17-NAY behaved as described *without push notifications*! The user visited the mobile app frequently to consume system updates without any prompting. Because the sessions were not triggered by interruptions (i.e., push notifications), the delays were longer, but the user was also more likely to read "deeply" for longer periods of time.

5.2 Late-Heavy Distributions

At the other end of the delay distribution spectrum, we see late-heavy distributions as a coherent and distinct cluster of users (the purple-coded users in Figure 6). One example selected from this group, user RTS17-a5j, is shown in Figure 9. The timeline visualization takes exactly the same form as above, but note the different time scale on the x axis. This user subscribed to five profiles and had a response rate of 85% on a volume of 1068 updates.

Here, we see far fewer sessions—in this case, averaging about once a day. However, these sessions were all quite long. That is, the user was engaging with our mobile assessment app and "deeply" reading system updates over relatively long spans of time. The result was that all the delays were quite long, and in fact, there were only updates with short delays because the user must examine newer updates before encountering older updates.

Many other users in this cluster exhibit similar behavior patterns. In fact, we see several examples of users engaging in very few (e.g., three) sessions with long observed delays. Note that if a user is encountering updates several days old, then we are operating outside the intended design space of RTS systems—and the assessment behavior is not very different from batch-style evaluation by TREC assessors. We observe that *all* late-heavy distributions are from pull users (RTS17). Although we had fewer users in the push condition (RTS16), we find this absence striking (more discussion below).

5.3 Bimodal Distributions

In addition to early-heavy and late-heavy distributions, which are in some ways not surprising, our clustering analysis also revealed bimodal distributions—this can be clearly seen in the blue-coded block in Figure 6. From this group we show the timeline for user RTS17-xWp in Figure 10. This user subscribed to 14 profiles and had a response rate of 98% on a volume of 4515 updates.

Remarkably, from the timeline we see both short frequent sessions and periodic long sessions. Once again, we emphasize that this user did not receive push notifications, so the user was returning to the mobile app on the user's own initiative. This was happening frequently in short sessions where only a few updates were consumed at a time. However, the user also engaged in long, "deep reading" sessions that led to consumption of old updates, much like in the late-heavy distributions. For this reason, we see a clear bimodal distribution in delays.

6 SYNTHESIS AND RECOMMENDATIONS

Thus far, we have answered (RQ2): Yes, there are indeed distinct and coherent patterns of user behavior. At a high level, we see early-heavy, late-heavy, and bimodal distributions of response delays. Users under the pull condition can exhibit any of these patterns, but push condition users are all early-heavy. Next, we synthesize our findings, provide an explanation of the observed behavior, and attempt to operationalize design recommendations for developers of real-time summarization systems (RQ3).

We hypothesize that users' information behavior with respect to real-time updates can be characterized as *eager* or *apathetic*. Eager users desire the most up-to-date information—they want updates *now*. We imagine an eager user as someone who might also compulsively check email. In contrast, apathetic users desire relevant information, but they don't really care about the recency of the updates delivered to them. We believe that eagerness or apathy is an intrinsic user characteristic, and such notions are not foreign in the literature. For example, Järvelin and Kekäläinen talk about *patient* and *impatient* users in the formulation of nDCG [10] and these constructs are captured in subsequent work on evaluation metrics (e.g., RBP [15]) and user models (e.g., [3]).

Within our framework, observed user behavior is the product of intrinsic user characteristics and system treatment. At a high level, we draw the following conclusions:

Pull-based delivery allows users to manifest their intrinsic behavior. The critical point here is that the users engage the system for updates on their own initiative: whenever they want and for how ever long they please. Eager users still "check in" frequently (e.g., RTS17-NAY, shown on the bottom of Figure 7), even without system interruptions. On the other hand, apathetic users "check in" less frequently, but in each session they are prepared to engage with system updates and read "deeply". Finally, bimodal users exhibit a mixture of eager and apathetic behavior—they check in frequently and read a bit, but then come back periodically to catch up on updates they may have missed.

Push-based delivery suppresses user initiative. Why don't we observe any late-heavy or bimodal distributions in the push condition? Admittedly, our sample size is small, but this asymmetry is striking and statistically significant. We believe that this is because constantly bombarding the user with push notifications causes fatigue and suppresses any additional motivation to open up our app and engage with content on the user's own initiative. As a result, we observe only short (but frequent) sessions—none of the users in the push condition exhibit "deep" reading behavior, where the user engages with system updates for long periods of time.

From these two findings we can come up with design recommendations for system developers. From the system's perspective, the designer has two "knobs" to control: the volume of updates, which is directly related to algorithms and users' needs, and the delivery mechanism (push vs. pull). We return to discuss volume in Section 7, but for now let us hold this factor constant. The system developer typically wishes to shape what industry calls "engagement", which is operationalized in this context as response rate and delay (the two user response variables). At a high level, we already know (from Section 4), that push notifications trade off a lower response rate for shorter delays (i.e., getting information to users faster). With respect to engagement, we offer the following design recommendations, thus answering (RQ3):

For apathetic users, push notifications are unlikely to affect user behavior in terms of increasing engagement. In other words, if users fundamentally don't care about receiving up-to-date information, then no amount of A/B testing of delivery algorithms is going to make a difference. Developers should just leave such users alone. In fact, trying to engage apathetic users with push notifications does double damage in that (1) users will simply ignore or turn off notifications and (2) users will become less likely to engage on their own initiative. Such users are easy to identify, and system developers should simply exclude them altogether from attempts to optimize for push notification engagement.

For apathetic users, consider other modalities of update delivery. Since most updates encountered by late-heavy users are more than 12 hours old, daily email digests might be a more appropriate mechanism for delivering a batch of updates for deep reading. This is also explored in the RTS Track in "Scenario B" [12, 13].

For eager users, push notifications can decrease response delays, but developers must be aware of the costs. Despite similarities between early-heavy users from RTS16 and RTS17, users receiving push notifications still exhibit shorter delays overall. It is believable that eager users are less annoyed by push notifications and more likely to pay attention to interruptions, and so push notifications can be an effective system intervention if shorter delays are desired.

However, the utility (from the user's perspective) of reducing delays for eager users is questionable, other than perhaps emergency scenarios (i.e., incoming ballistic missile) or a few specialized needs (e.g., information about a stock). For users who frequently check our mobile app, they will already encounter the updates the next time they "check in". Furthermore, as we argue above, there are costs to constant interruptions in terms of suppressing user initiative. For the designer of a production system, it is possible to operationalize this design principle by weighing the urgency (i.e., temporal utility of the update) against the mean time between user-initiated sessions.

We note that these design recommendations run somewhat counter to how modern social media organizations optimize engagement, which can be characterized as "more, more, more". For users with already high engagement, there is a tendency to want to push that engagement even higher. The thinking goes, "if the user is logging in every two hours, can we find additional mechanisms (e.g., push notifications) to get the user to log in every hour?" Rarely is consideration given to different types of users who may have different information consumption behaviors. This relates to how user buckets are selected in an A/B test: a random sample, for example, conflates users who may be fundamentally different. Increasing stimuli and treating users like a homogeneous population of lab rats provide little value to the user, and may be self-defeating from the system's perspective. For push notifications, we argue that interruptions take away users' motivation to engage with the system on their own initiative.

7 LIMITATIONS AND FUTURE WORK

Due to their complexity, all user studies exhibit methodological imperfections, and ours is no different. Nevertheless, TREC still remains the gold standard in multi-participant IR evaluations today, and we as co-organizers have adopted all evaluation best practices. The biggest difference between the 2016 and 2017 evaluations was the delivery mechanism (push vs. pull), but other differences are worth discussing:

- The evaluations used different interest profiles and examined different systems (although there were a number of teams that participated both years). In both evaluations, there were sufficient interest profiles to choose from such that users were likely interacting with topics they were interested in. Looking through TREC system papers, we did not notice any significant differences in algorithmic approaches across both years. Furthermore, due to the broker interleaving in delivering updates, users were seeing results from all systems in aggregate, and it is unlikely that there were significant differences in systems as a whole.
- The user studies recruited different participants each year. However, since our recruitment procedure did not substantively change and we were drawing from the same general student population, we are skeptical that this would be a source of major differences. Overall, we believe that our setup is comparable to a between-subjects user study. Of course, within-subjects designs can lead to stronger conclusions, but the nature of the delivery mechanism makes such a design difficult.

- The two evaluations deployed different assessment interfaces; both were similar in terms of the overall flow, but graphical elements and other details differed. This difference was the direct result of needing push notifications in RTS16; without the requirement, we had greater flexibility for the RTS17 interface to simplify deployment with a web app. No doubt that interface design would have some effect on user behavior, but push vs. pull delivery remains the salient experimental manipulation.

While we cannot rule out that the above differences (or others) affect our findings, nothing jumps out as obviously suspect in our explanation of the large behavioral differences observed.

One obvious limitation of this work, as with most user studies, is the size of our user population. However, in comparison to typical academic user studies, our sample size is already quite large, considering the significant involvement of test subjects. Furthermore, this study was not possible without substantial investment from NIST and the framework of multi-participant evaluations over two years. We believe that our study already pushes the limits of scale for user studies that are realistically feasible in academia.

One direct implication of the user sample size is the strength of evidence supporting our findings, the most striking of which is that nearly all users in the push condition exhibit early-heavy behavior, whereas pull condition users exhibit three distinct profiles. Of course, it is difficult to prove the absence of a behavior, but the results of our significant testing (Section 4.3) do allow us to reject the null hypothesis that there are no significant differences between the push and pull conditions. Coupled with our explanation in terms of eager and apathetic information consumption behavior, we believe that our conclusions at minimum have face validity.

In answering (RQ1) and (RQ2), beyond our actual findings, we believe that our data analysis methodology (use of heatmaps, EDFs, clustering, etc.) is itself a contribution. This approach can be directly applied to analyze log data from production services (e.g., Twitter, news reading apps, etc.). With a much larger population of users, the next obvious step would be to fit a parametric family of models in order to characterize user populations (for example, Figure 5). We leave this for future work.

At a high level, our eager vs. apathetic categorization of users' information consumption behavior is of course too simplistic. While it serves as a good starting point, there are many factors to tease apart. System update volume and quality are two obvious issues (although note that volume and response rate vary independently, per analysis in Section 4.1): in our case, with a multi-participant evaluation, the mobile users receive interleaved results from all systems, which cover the entire spectrum of quality. It would be interesting future work to examine how users respond to relevant vs. non-relevant updates. This also factors into characteristics of user sessions, whose lengths would likely be influenced by how good system updates are.

Another interesting direction is to examine user motivation in more detail, which we could externally manipulate by the compensation for participation. Our studies paid subjects by the number of updates assessed, which incentivizes a large volume of judgments but provides no incentive for shorter delays. It might be interesting to modify compensation based on some function of delay or even a fixed compensation (regardless of users' actions).

Finally, we would like primarily observational user studies to inform the construction of well-specified user models that can then be applied to help developers iterate rapidly on different algorithms within requiring actual users. There is work on such models for *ad hoc* retrieval (e.g., [21]) and even formulations for streaming tasks [3], but we need richer models for prospective information seeking, as well as calibration and validation efforts.

8 CONCLUSIONS

This paper makes an initial foray into understanding how mobile users consume the output of real-time summarization systems. We have examined push vs. pull mechanisms, but ultimately, the best approach is likely to be a hybrid. Before we get there, however, there remain many unanswered questions for even simple setups. Our hope is that this work increases the community's interest in prospective information needs and spurs additional future work.

9 ACKNOWLEDGMENTS

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. We'd like to thank our RTS co-organizers M. Abualsaud, F. Diaz, N. Ghelani, R. McCreadie, D. Milajevs, A. Roegiest, and E. Voorhees for help running the evaluations, and all our mobile users for their participation.

REFERENCES

[1] J. Allan. 2002. *Topic Detection and Tracking: Event-Based Information Organization.* Kluwer Academic Publishers, Dordrecht, The Netherlands.
[2] J. Aslam, F. Diaz, M. Ekstrand-Abueg, R. McCreadie, V. Pavlu, and T. Sakai. 2015. TREC 2015 Temporal Summarization Track Overview. *TREC.*
[3] G. Baruah, C. Clarke, and M. Smucker. 2015. Evaluating Streams of Evolving News Events. *SIGIR.* 675–684.
[4] N. Belkin and W. Croft. 1992. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *CACM* 35, 12 (1992), 29–38.
[5] H. Dang. 2005. Overview of DUC 2005. *DUC.*
[6] C. Grady and M. Lease. 2010. Crowdsourcing Document Relevance Assessment with Mechanical Turk. *NAACL HLT 2010 Workshop on AMT.* 172–179.
[7] Q. Guo, F. Diaz, and E. Yom-Tov. 2013. Updating Users about Time Critical Events. *ECIR.* 483–494.
[8] C. Hensley. 1963. Selective Dissemination of Information (SDI): State of the Art in May, 1963. *AFIPS Spring Joint Computer Conference.*
[9] E. Housman and E. Kaskela. 1970. State of the Art in Selective Dissemination of Information. *IEEE Transactions on Engineering Writing and Speech* 13, 2 (1970), 78–83.
[10] K. Järvelin and J. Kekäläinen. 2002. Cumulative Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.
[11] D. Lewis. 1995. The TREC-4 Filtering Track. *TREC.* 165–180.
[12] J. Lin, S. Mohammed, R. Sequiera, L. Tan, N. Ghelani, M. Abualsaud, R. McCreadie, D. Milajevs, and E. Voorhees. 2017. Overview of the TREC 2017 Real-Time Summarization Track. *TREC.*
[13] J. Lin, A. Roegiest, L. Tan, R. McCreadie, E. Voorhees, and F. Diaz. 2016. Overview of the TREC 2016 Real-Time Summarization Track. *TREC.*
[14] A. Mehrotra and M. Musolesi. 2018. Intelligent Notification Systems: A Survey of the State of the Art and Research Challenges. *arXiv:1711.10171v2.*
[15] Alistair Moffat and Justin Zobel. 2008. Rank-Biased Precision for Measurement of Retrieval Effectiveness. *TOIS* 27, 1 (2008), Article 2.
[16] X. Qian, J. Lin, and A. Roegiest. 2016. Interleaved Evaluation for Retrospective Summarization and Prospective Notification on Document Streams. *SIGIR.* 175–184.
[17] S. Robertson and I. Soboroff. 2002. The TREC 2002 Filtering Track Report. *TREC.*
[18] A. Roegiest, L. Tan, and J. Lin. 2017. Online In-Situ Interleaved Evaluation of Real-Time Push Notification Systems. *SIGIR.* 415–424.
[19] A. Said, J. Lin, A. Bellogín, and A. de Vries. 2013. A Month in the Life of a Production News Recommender System. *CIKM Workshop on Living Labs for Information Retrieval Evaluation.* 7–10.
[20] A. Schuth, K. Balog, and L. Kelly. 2015. Overview of the Living Labs for IR Evaluation (LL4IR) CLEF Lab 2015. *CLEF.*
[21] M. Smucker and C. Clarke. 2012. Time-Based Calibration of Effectiveness Measures. *SIGIR.* 95–104.