Overview of the TREC 2016 Real-Time Summarization Track

Jimmy Lin,[1] Adam Roegiest,[1] Luchen Tan,[1]
Richard McCreadie,[2] Ellen Voorhees,[3] and Fernando Diaz[4]

[1] David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada
[2] School of Computing Science, University of Glasgow, Scotland, the United Kingdom
[3] National Institute for Standards and Technology, Maryland, USA
[4] Microsoft Research, New York, USA

{jimmylin, aroegies, luchen.tan}@uwaterloo.ca
richard.mccreadie@glasgow.ac.uk, ellen.voorhees@nist.gov, fdiaz@microsoft.com

1. INTRODUCTION

The TREC 2016 Real-Time Summarization (RTS) Track aims to explore techniques and systems that automatically monitor streams of social media posts such as Twitter to keep users up to date on topics of interest. We might think of these topics as "interest profiles", specifying the user's prospective information needs. In real-time summarization, the goal is for a system to "push" (i.e., recommend or suggest) interesting and novel content to users in a timely fashion. For example, the user might be interested in poll results for the 2016 U.S. presidential elections and wishes to be notified whenever new results are published. We can imagine two methods for disseminating updates:

- **Scenario A: Push notifications.** As soon as the system identifies a relevant post, it is immediately sent to the user's mobile device via a push notification. At a high level, push notifications should be relevant (on topic), novel (users should not be pushed multiple notifications that say the same thing), and timely (provide updates as soon after the actual event occurrence as possible).
- **Scenario B: Email digests.** Alternatively, a user might wish to receive a daily email digest that summarizes "what happened" that day with respect to the interest profiles. One might think of these emails as supplying "personalized headlines". At a high level, these results should be relevant and novel; timeliness is not particularly important, provided that the tweets were all posted on the previous day.

For expository convenience and to adopt standard information retrieval parlance, we write of users desiring *relevant* content, even though "relevant" in our context might be better operationalized as interesting, novel, and timely.

Real-Time Summarization is a new track at TREC 2016 and represents a merger of the Microblog (MB) Track, which ran from 2010 to 2015, and the Temporal Summarization (TS) Track, which ran from 2013 to 2015 [2]. The creation of RTS was designed to leverage synergies between the two tracks in exploring prospective information needs over document streams containing novel and evolving information. The task this year directly evolved from the real-time filtering task in the TREC 2015 Microblog Track [6].

Despite superficial similarities, our task is very different from document filtering in the context of earlier TREC Filtering Tracks, which ran from 1995 [4] to 2002 [9], and the general research area of topic detection and tracking (TDT) [1]. The TREC Filtering Tracks are best understood as binary classification on *every* document in the collection with respect to standing queries, and TDT is similarly concerned with identifying *all* documents related to a particular event—with an intelligence analyst in mind. In contrast, we are focused on identifying a small set of the most relevant updates to deliver to users. Furthermore, in both TREC Filtering and TDT, systems must make online decisions as soon as documents arrive. In our case, for scenario A, systems can choose to push older content (latency is one aspect of the evaluation), thus giving rise to the possibility of algorithms operating on bounded buffers. Finally, previous evaluations, including TDT, TREC Filtering, and Temporal Summarization, merely *simulated* the streaming nature of the document collection, whereas participants in our evaluation actually operated on tweets posted in real time.

2. EVALUATION DESIGN

2.1 General Setup

The design of the TREC 2016 Real-Time Summarization Track largely follows the design of the real-time filtering task in the TREC 2015 Microblog Track [6]. Although we are interested in exploring filtering techniques over streams of social media posts in general, this year's track restricted the content under consideration to tweets due to their widespread availability. In particular, Twitter provides a streaming API through which clients can obtain a sample (approximately 1%) of public tweets, colloquially known as the "spritzer". This level of access is available to anyone who signs up for an account.

During the official evaluation period, which began Tuesday, August 2, 2016 00:00:00 UTC and lasted until Thursday, August 11, 2016 23:59:59 UTC, participants' systems "listened" to Twitter's live tweet sample stream to identify relevant tweets with respect to users' interest profiles.

System behavior during the evaluation period varied according to the evaluation scenario:

Scenario A: Push notifications. As soon as the system identifies a relevant tweet with respect to an interest profile, it pushes (i.e., submits) the tweet to the RTS evaluation broker (via a REST API). The evaluation broker records

the system submission and then immediately delivers the tweet to the mobile devices of a group of human assessors as a push notification *in real time* (more details in Section 2.4).

Each system was allowed to push at most ten tweets per interest profile per day. This per-day tweet delivery limit represents a crude attempt to model user fatigue in mobile push notifications. Note, however, that in this design we are not modeling real-world constraints such as "don't send users notifications in the middle of the night". This simplification was intentional.

Scenario B: Email digests. The system is tasked with identifying up to 100 tweets per day per interest profile. These posts are putatively delivered to the user daily. For simplicity, all tweets from 00:00:00 to 23:59:59 UTC are valid candidates for that particular day. It is expected that systems will compute the results in a relatively short amount of time after the day ends (e.g., at most a few hours), but this constraint was not enforced. Each system recorded the results (i.e., ranked lists) for each day, which were then uploaded to NIST servers in batch shortly after the evaluation period ended.

The per-day limit of 100 tweets was arbitrarily set, but at a value that is larger than what one might expect from a daily email digest, primarily to enrich the judgment pool (more details in Section 2.5). As with scenario A, we neglected to model real-world constraints in favor of simplicity, since defining a "day" in terms of UTC does not take into account the reading habits of users in different time zones around the world.

For scenario A, the RTS evaluation broker records system outputs as they are received and thus we can be sure that the participating systems are actually operating in real time. For scenario B, systems were expected to conform to the temporal constraints imposed by the task scenario (for example, to not use "future knowledge" when ranking the tweets), but there was no enforcement mechanism due to the post-hoc batch submission setup.

An important consequence of the evaluation design is that, unlike in most previous TREC evaluations, no collection or corpus was distributed ahead of time. Since each participant "listened" to tweets from Twitter's streaming API, the collection was generated in real time and delivered to each participant independently. In a 2015 pilot study [7], we verified that multiple listeners to the public Twitter sample stream receive effectively the same tweets (Jaccard overlap of 0.999 across six independent crawls over a three day sample in March 2015). This evaluation setup was adopted in the TREC 2015 Microblog Track without any issue, thus providing large-scale validation of the design. For evaluation purposes (i.e., pool formation for judgments), the organizers also collected the live Twitter stream: this was accomplished by two independent crawlers in two geographically-distributed datacenters on Amazon's EC2 service. Note that independent crawls do not increase coverage of the tweets received; the sole purpose of the setup was to increase redundancy, particularly robustness with respect to transient network glitches that sometimes affect tweet delivery. The union of these two crawls was designated as the "official" collection.

Another substantial departure from most previous TREC evaluations is the requirement that participants maintain a running system that continuously monitors the tweet sample stream during the evaluation period. The track organizers provided boilerplate code and reference implementations, but it was the responsibility of each individual team to run its own system, connect with the RTS evaluation broker to submit results, and cope with crashes, network glitches, power disruptions, etc. The TREC 2015 Microblog Track, as well as other recent tracks at TREC that required participants to maintain "live" systems, showed that this requirement does not present an onerous barrier to entry for participating teams.

2.2 Run Submission

In both scenarios, systems were asked to only consider tweets in English. Each team was allowed to submit up to three runs for scenario A and three runs for scenario B. Runs for scenario A involved registering with the RTS evaluation broker to request a unique token, which was used to associate all submitted tweets to a particular run submission (see Section 2.4).

Runs were categorized into three different types based on the amount of human involvement:

- **Automatic Runs:** In this condition, system development (including all training, system tuning, etc.) must conclude prior to downloading the interest profiles from the track homepage (which were made available before the evaluation period). The system must operate without human input before and during the evaluation period. Note that it is acceptable for a system to perform processing on the profiles (for example, query expansion) before the evaluation period, but such processing cannot involve human input.
- **Manual Preparation:** In this condition, the system must operate without human input during the evaluation period, but human involvement is acceptable before the evaluation period (i.e., after downloading the interest profile). Examples of manual preparation include human examination of the interest profiles to add query expansion terms or manual relevance assessment on a related collection to train a classifier. However, once the evaluation period begins, no further human involvement is permissible.
- **Manual Intervention:** In this condition, there are no limitations on human involvement before or during the evaluation period. Crowd-sourcing judgments, human-in-the-loop search, etc. are all acceptable.

Participants were asked to designate the run type at submission time for the scenario B runs. For scenario A runs, we asked each team about the type of each of their runs over email after the evaluation period.

All types of systems were welcomed; in particular, manual preparation and manual intervention runs are helpful in understanding human performance and enriching the judgment pool.

2.3 Interest Profiles

Interest profiles for real-time summarization are difficult to develop because of their prospective nature—this was one of the lessons learned from the real-time filtering task in the TREC 2015 Microblog Track [6]. For retrospective *ad hoc* topics over a static collection, it is possible for topic developers to explore the document collection to get a sense of the amount of relevant material, range of topical facets, etc.

for a particular information need. Typically, topic developers prefer topics that have neither too many nor too few relevant documents. This is not possible for RTS interest profiles, which essentially requires "predicting the future". The track overview paper from TREC 2015 [6] provides more discussion of these issues.

Just as in the TREC 2015 Microblog Track, we adopted the "standard" TREC *ad hoc* topic format of "title", "description", and "narrative" for the interest profiles. The so-called title consists of two to three keywords that provide the gist of the information need, akin to something a user might type into the query box of a search engine. The description is a one-sentence statement of the information need, and the narrative is a paragraph-length chunk of prose that sets the context of the need and expands on what makes a tweet relevant. By necessity, these interest profiles are more generic than the needs expressed in typical retrospective topics because the topic developer does not know what future events will occur. Thus, despite superficial similarities in format, we believe that interest profiles are qualitatively different from *ad hoc* topics.

Given the prospective nature of interest profiles, we employed the strategy of "overgenerate and cull". That is, we created many more interest profiles than there were resources available for assessment, with the understanding that we could cull a set of profiles after the fact to assess, guided by actual assessor interest. For 2016, the interest profiles were drawn from three sources:

1. 51 interest profiles that were assessed from the TREC 2015 Microblog Track, so that participants have access to training data.
2. 107 additional interest profiles culled from the TREC 2015 Microblog Track—the old profiles were manually filtered to retain those that were still applicable (e.g., throwing away profiles about events that have happened already) and profiles for which there would hopefully be a reasonable volume of relevant tweets.
3. 45 new interest profiles that were specifically developed from scratch for this year's track.

All interest profiles were made available to the participants before the beginning of the evaluation period.

2.4 Online Judgments and Metrics

On key feature introduced in this year's track is an online evaluation component for scenario A whereby system outputs are assessed in an online manner. Our general approach builds on growing interest in so-called "Living Labs" [11] and related Evaluation-as-a-Service (EaaS) [3] approaches that attempt to better align evaluation methodologies with user task models and real-world constraints to increase the fidelity of research experiments.

Our evaluation architecture is shown in Figure 1 and was previously described in Roegiest et al. [10]; the entire evaluation infrastructure is open source and available on GitHub.[1] As the participating systems identify relevant tweets, they are immediately pushed to the RTS evaluation broker, which then immediately routes the tweets to assessors who have installed a custom app on their mobile devices. The tweets are

[1] `https://github.com/trecrts/trecrts-eval/`

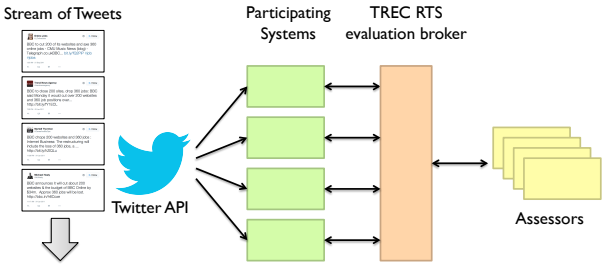

Figure 1: Evaluation setup for scenario A showing the use of mobile assessors who judge tweets in real time, mediated by the RTS evaluation broker.

rendered as push notifications on the assessors' mobile devices and are added to an assessment queue in their app for consideration.

This setup has a number of distinct advantages over traditional post-hoc batch evaluations:

- Gathering relevance judgments in an online fashion has the potential to yield more situationally-accurate assessments, particularly for rapidly developing events. With post-hoc batch evaluations, there is always a bit of disconnect as the assessor needs to "imagine" herself at the time the update was pushed. With our evaluation framework, we remove this disconnect.
- An online evaluation platform allows for the possibility of user-submitted information needs, thus giving assessors the ability to judge tweets for interest profiles they are genuinely interested in.
- An online evaluation platform opens the door to providing realistic, online feedback to participants, thus potentially facilitating active learning approaches.

In this first year of the evaluation, we did not provide a mechanism for user-submitted interest profiles or an API for participants to receive feedback. However, we hope to introduce these features in the future, and the existing infrastructure provides a solid foundation to build on.

In more detail, the evaluation proceeded as follows:

1. Prior to the beginning of the evaluation period, each participant's system "registers" with the RTS evaluation broker (via a REST API call) to request a unique token, which is used in future requests to associate all submitted tweets to a particular system. For the purposes of this discussion, each participant "run" is considered a separate system.
2. Whenever a system identifies a relevant tweet with respect to an interest profile, the system submits the result to the RTS evaluation broker via a REST API, which records the submission time.
3. The RTS evaluation broker immediately routes the tweet to the mobile device of an assessor, where it is rendered as a push notification containing both the text of the tweet and the corresponding interest profile.
4. The assessor may choose to judge the tweet immediately, or if it arrived at an inopportune time, to ignore it. Either way, the tweet is added to the queue in a custom app on the assessor's mobile device, which she can access

Figure 2: Screenshot of the mobile assessment app.

at any time to judge the queue of accumulated tweets. Users have the option of logging out of the app completely, at which point they will cease to receive notifications.

5. As the assessor examines tweets and provides judgments, the results are relayed back to the RTS evaluation broker and recorded.

Our setup largely follows the interleaved evaluation methodology for prospective notifications proposed by Qian et al. [8]. For each tweet, the user makes one of three judgments:

- *relevant*, if the tweet contains relevant and novel information;
- *redundant*, if the tweet contains relevant information, but is substantively similar to another tweet that the assessor had already seen;
- *not relevant*, if the tweet does not contain relevant information.

A screenshot of the mobile assessment app is shown in Figure 2. The icons below each tweet represent the *relevant*, *not relevant*, and *redundant* judgments, respectively.

The entire evaluation is framed as a user study (with appropriate ethics review and approval). A few weeks prior to the beginning of the evaluation period, we recruited assessors from the undergraduate and graduate student population at the University of Waterloo, via posts on various email lists as well as paper flyers on bulletin boards. The assessors were compensated $5 CAD to install the mobile assessment app and $1 CAD per 20 judgments.

As part of the assessor training process, they subscribed to receive notifications for profiles they were interested in, selecting from the complete list given to all participants via an online web interface. To encourage diversity, we did not allow more than three assessors to select the same profile (on a first come, first served basis).

The RTS evaluation broker followed the temporal interleaving strategy proposed by Qian et al. [8], which meant that tweets were pushed to the assessors as soon as the broker received the submitted tweets. Although Qian et al. only discussed interleaving the output of two systems, it is straightforward to extend the strategy to multiple systems. The broker made sure that each tweet was only pushed once (per profile), in the case where the same tweet is submitted by multiple systems at different times. Although one can imagine a variety of different "routing" algorithms for pushing tweets to different assessors that have subscribed to a topic, this year we implemented the simplest possible algorithm where the tweet was pushed to *all* assessors (that had subscribed to the profile). This meant that the broker might receive more than one judgment per tweet.

Another implication of this interleaved evaluation setup is that an assessor will likely encounter tweets from different systems, which makes proper interpretation of *redundant* judgments more complex. A tweet might only be redundant because the same information was contained in a tweet pushed earlier by another system (and thus it is not the "fault" of the particular system that pushed the tweet). That is, the interleaving of outputs from different systems was directly responsible for introducing the redundancy. Although Qian et al. [8] proposed a heuristic for more accurate credit assignment to cope with interleaving, in this evaluation we simply counted the absolute number of judgments of each type. From these counts, we computed "strict" precision, defined as:

$$\frac{\text{relevant}}{\text{relevant} + \text{redundant} + \text{not relevant}} \tag{1}$$

as well "lenient" precision, defined as:

$$\frac{\text{relevant} + \text{redundant}}{\text{relevant} + \text{redundant} + \text{not relevant}} \tag{2}$$

Precision seemed like an appropriate metric given the cost of push notifications in terms of interrupting the user. Note that these precision computations represent a micro-average (and *not* an average across per-topic scores). This choice was made because of the sparisty of judgments, which would magnify the effects of interest profiles with few judgments.

Finally, we made the (arbitrary) decision of using "strict" precision as the primary metric for assessing scenario A runs using mobile assessors.

2.5 Batch Judgments and Metrics

In addition to the online evaluation by mobile assessors, the track also employed a standard post-hoc batch evaluation methodology that has been refined and validated over many iterations in previous TREC evaluations. For scenario A, the dual evaluation approach helps us validate the reliability of our online mobile assessment methodology.

We adopted the Tweet Timeline Generation (TTG) evaluation methodology that was originally developed for the TREC 2014 Microblog Track [5] and also used in the TREC

2015 Microblog Track [6]. The methodology has been externally validated [15], and similar methodologies have been deployed in evaluations dating back at least a decade; thus, we can consider this approach mature and reliable. The assessment workflow proceeded in two major stages: relevance assessment and semantic clustering. Both were accomplished by NIST assessors.

Relevance assessments were performed using pooling with a single pool across both scenario A and scenario B runs. The pools were constructed from all submitted runs, taking all tweets from Scenario A runs and up to 90 tweets (per profile) from Scenario B runs. For scenario B runs, tweets were added to the judgment pool in a round-robin fashion across days. That is, the top-ranked tweet from each day was first added to the pool, then the second-ranked tweet from each day, and so on. If the process exhausted tweets from a particular day before the 90 tweet limit had been reached, tweets were selected from the remaining days until the limit.

After pool formation, the next decision was the selection of interest profiles to manually assess. In this case, the selections of the mobile assessors provided an obvious guide. Profiles to assess were selected by first taking those interest profiles that had at least 50 distinct tweets judged by the mobile assessors (there were 67 of these), and then eliminating profiles whose pools were enormous or those about events from 2015. NIST assessors ended up judging 56 profiles. The mean size of the pools was 1206 tweets, with minimum 917 and maximum 1651.

These pools were then examined by NIST assessors. To facilitate consistent judgments, tweets were first clustered by lexical similarity. Each tweet was independently assessed on a three-way scale of "not relevant", "relevant", and "highly relevant". Non-English tweets were marked as not relevant by fiat. If a tweet contained a mixture of English and non-English content, discretion was left to the assessor. As with previous TREC Microblog evaluations, assessors examined links embedded in tweets, but did not explore any additional external content beyond those. Retweets did not receive any special treatment and were assessed just like any other tweet.

All 56 profiles judged by NIST assessors have at least one relevant judgment from the mobile assessors. However, based on the NIST assessors, one interest profile has no relevant tweets, three other interest profiles have exactly one relevant tweet, and a total of 14 interest profiles have fewer than 10 relevant tweets. At the other end of the scale, three interest profiles have more than 200 relevant tweets, the maximum being RTS10 (Hiroshima bomb reactions), with 364 relevant tweets.

After the relevance assessment process, the NIST assessors proceeded to perform semantic clustering on the relevant tweets using the tweet timeline generation (TTG) protocol, originally developed for the TREC 2014 Microblog Track [5, 15]. Unlike in previous years, where the clustering was performed outside NIST, this year the same assessor performed both the relevance judgments and the clustering.

The TTG protocol was designed to reward novelty (or equivalently, to penalize redundancy) in system output. In both scenario A and scenario B, we assume that users would not want to see multiple tweets that "say the same thing", and thus the evaluation methodology should reward systems that eliminate redundant output. Following the TREC 2014 Microblog Track, we operationalized redundancy as follows: for every pair of tweets, if the chronologically later tweet contains substantive information that is not present in the earlier tweet, the later tweet is considered novel; otherwise, the later tweet is redundant with respect to the earlier one. In our definition, redundancy and novelty are antonyms, so we use them interchangeably but in opposite contexts.

Due to the temporal constraint, redundancy is *not* symmetric. If tweet A precedes tweet B and tweet B contains substantively similar information found in tweet A, then B is redundant with respect to A, but not the other way around. We also assume transitivity. Suppose A precedes B and B precedes C: if B is redundant with respect to A and C is redundant with respect to B, then by definition C is redundant with respect to A.

In the instructions given to the NIST assessors, they were not provided a particular target regarding the number of clusters to form. Instead, they were asked to use their best judgment, considering both the interest profile and the actual tweets.

For the semantic clustering, the assessors were shown all the relevant tweets (from the judgment pool) for a single interest profile within a custom assessment interface. The tweets were shown in the left pane in chronological order, while the list of current clusters were shown in a pane on the right side. For each tweet in the left pane, the assessor could either use that tweet as the basis for a new cluster, or add it to one of the existing clusters. In this way, clusters representing important pieces of information (comprised of semantically similar tweets) are constructed incrementally. To aid the clustering process, assessors could enter a short textual description for each cluster and then sort the tweets by similarity to a selected cluster, as a way to speed up the process of finding additional relevant tweets for that cluster. Users could also retroactively move a tweet from a cluster back into the left pane, such that it could then be assigned to a different cluster. The output of the assessment process (for each interest profile) is a list of clusters, where tweets in each cluster represent a particular "facet" of the overall information need.

2.5.1 Scenario A Metrics

For Scenario A, we computed a number of metrics from the relevance judgments and clusters provided by the NIST assessors, detailed below. As previously discussed, push notifications should be relevant (on topic), novel (users should not be pushed multiple notifications that say the same thing), and timely (provide updates as soon after the actual event occurrence as possible). Unlike the TREC 2015 Microblog Track as well as previous Temporal Summarization Tracks (cf. [2]), which devised single-point metrics that attempted to incorporate both relevance, novelty, and timeliness, we decided this year to separately compute metrics of output quality (relevance and novelty) and latency (timeliness).

We envision that systems might trade off latency with output quality: For example, a system might wait to accumulate evidence before pushing tweets, thus producing high-quality output at the cost of high latency. Alternatively, a low-latency system might aggressively push results that it might "regret" later. Computing metrics of output quality separately from latency allows us to understand the potential tradeoffs. Additionally, we believe this approach is appropriate because we have no empirical evidence as to what the "human response curve" to latency looks like—

that is, how much should we discount a quality metric based on tardiness? Attempting to formulate a single-point metric collapses meaningful distinctions in what users may be looking for in systems.

Expected Gain (EG) for an interest profile on a particular day is defined as follows:

$$\frac{1}{N}\sum \mathrm{G}(t) \tag{3}$$

where N is the number of tweets returned and $\mathrm{G}(t)$ is the gain of each tweet:

- Not relevant tweets receive a gain of 0.
- Relevant tweets receive a gain of 0.5.
- Highly-relevant tweets receive a gain of 1.0.

Once a tweet from a cluster is retrieved, all other tweets from the same cluster automatically become not relevant. This penalizes systems for returning redundant information.

Normalized Cumulative Gain (nCG) for an interest profile on a particular day is defined as follows:

$$\frac{1}{\mathcal{Z}}\sum \mathrm{G}(t) \tag{4}$$

where $\mathcal{Z}$ is the maximum possible gain (given the ten tweet per day limit). The gain of each individual tweet is computed as above. Note that gain is not discounted (as in nDCG) because the notion of document ranks is not meaningful in this context.

The score for a run is the mean of scores for each day over all the profiles. Since each profile contains the same number of days, there is no distinction between micro- vs. macro-averages. An interesting question is how scores should be computed for days in which there are no relevant tweets: for rhetorical convenience, we call days in which there are no relevant tweets for a particular interest profile (in the pool) "silent days", in contrast to "eventful days" (where there are relevant tweets). In the EG-1 and nCG-1 variants of the metrics, on a "silent day", the system receives a score of one (i.e., a perfect score) if it does not push any tweets, or zero otherwise. In the EG-0 and nCG-0 variants of the metrics, for a silent day, all systems receive a gain of zero no matter what they do. For more details about this distinction, see Tan et al. [14].

Therefore, under EG-1 and nCG-1, systems are rewarded for recognizing that there are no relevant tweets for an interest profile on a particular day and remaining silent (i.e., does not push any tweets). The EG-0 and nCG-0 variants of the metrics do not reward recognizing silent days: that is, it never hurts to push tweets.

Gain Minus Pain (GMP) is defined as follows:

$$\alpha \cdot \sum G - (1-\alpha)\cdot P \tag{5}$$

The G (gain) is computed in the same manner as above. Pain P is the number of non-relevant tweets that the system pushed, and α controls the balance between the two. We investigated three α settings: 0.33, 0.50, and 0.66. Note that this metric is the same as the linear utility metric used in the TREC Filtering Tracks [4, 9], although our formulation is slightly different. Thus, our metric is not novel, which we see as an advantage since it builds on previous work.

In summary, for scenario A, we report EG-1, EG-0, nCG-1, nCG-0, and GMP (with $\alpha = \{0.33, 0.50, 0.66\}$). EG-1 was considered the primary metric.

Latency. In addition to the quality metrics above, we report, only for tweets that contribute to gain, the mean and median difference between the time the tweet was pushed and the first tweet in the semantic cluster that the tweet belongs to (based on the NIST assessors).

For example, suppose tweets A, B, and C are in the same semantic cluster, and were posted 09:00, 10:00, and 11:30, respectively. No matter which of the three tweets is pushed, the latency is computed with respect to the creation time of A (09:00). Therefore, pushing tweet C at 11:30 and pushing tweet A at 11:30 gives the same latency.

2.5.2 *Scenario B Metrics*

Scenario B runs were evaluated in terms of nDCG as follows: for each interest profile, the list of tweets returned per day is treated as a ranked list and from this nDCG@10 is computed. Note that in this scenario, the evaluation metric *does* include gain discounting because the email digests can be interpreted as ranked lists of tweets. Gain is computed in the same way as in scenario A with respect to the semantic clusters. Systems only receive credit for the first relevant tweet they report from a cluster.

The score of an interest profile is the mean of the nDCG scores across all days in the evaluation period, and the score of the run is the mean of scores for each profile. Once again, the micro- vs. macro-average distinction is not applicable here. As with scenario A, we computed two variants of the metric: with nDCG-1, on a "silent day", the system receives a score of one (i.e., a perfect score) if it does not push any tweets, or zero otherwise. In nDCG-0, for a silent day, all systems receive a gain of zero no matter what they do.

3. RESULTS

To provide a track-wide baseline and also a point of comparison for this year's participants, we deployed the "YoGosling" system [13], which is a simplified reimplementation of the best-performing automatic system from the TREC 2015 Microblog Track [12]. The system was originally designed for scenario A, but we adapted it for scenario B by simply running the system on all tweets collected at the end of the day, keeping the same exact scoring model and scoring thresholds as implemented for scenario A.

3.1 Scenario A

For Scenario A, we received 41 runs from 18 groups. These runs pushed a total of 161,726 tweets, or 95,113 unique tweets after de-duplicating within interest profiles (but not de-duplicating across profiles).

For the online evaluation of scenario A systems, we recruited a total of 18 assessors, 13 of whom ultimately provided judgments. Of these, 11 were either graduates or undergraduate students at the University of Waterloo. In total, we received 12,115 judgments over the assessment period, with a minimum of 28 and a maximum of 3,791 by an individual assessor. We found that 10,605 tweets received a single judgment, 743 tweets received two judgments, and 8 tweets received three judgments. Overall, 122 interest pro-

Assessor	Judgments	Profiles	Messages	Response
1	53	4	1619	3.27%
2	3305	10	7141	46.28%
3	136	10	5860	2.32%
4	327	8	3795	8.62%
5	949	12	6330	14.99%
6	28	12	7211	0.39%
7	281	10	4162	6.75%
8	1908	15	7754	24.61%
9	3791	33	16654	22.76%
10	680	16	7257	9.37%
11	107	43	22676	0.47%
12	324	2	938	34.54%
13	226	12	7058	3.20%

Table 1: Assessor statistics. For each assessor, columns show the number of judgments provided, the number of interest profiles subscribed to, the maximum number of push notifications received, and the response rate.

files received at least one judgment; 93 received at least 10 judgments; 67 received at least 50 judgments; 44 received at least 100 judgments.

The distribution of judgments by assessors is shown in Table 1. The columns list: assessor id, the number of judgments provided, the number of profiles subscribed to. The fourth column shows the sum of all push notifications for the profiles that each assessor subscribed to: this captures the maximum number of push notifications that the assessor *could have received* during the evaluation period. Note that we do not have the *actual* number of notifications each assessor received because the assessor could have logged out during some periods of time or otherwise adjusted the local device settings (e.g., to disable notifications). The final column shows the response rate, computed as the fraction between the second and fourth columns, which is a lower-bound estimate. From this table, we see that some assessors are quite diligent in providing judgments, while others are more sporadic.

It was originally our intention to build mobile assessment apps for both Android and iOS, but due to technical issues with the app development framework we were using, we were unable to deploy a stable iOS app in time. As a result, all assessors used the Android app. Some assessors encountered display issues with tweets during the evaluation period, due to the wide range of devices owned by the assessors. Since this was not anticipated during testing, we did our best to support these assessors and to provide workarounds on the fly. While the overall assessment experience could have been more refined, the entire setup worked as expected.

After the evaluation, while compiling results, we discovered that from the RTS evaluation broker's perspective, some tweets were pushed before they were actually posted on Twitter. Since it is unlikely that participants had created time traveling devices, we attributed the issue to clock skew on the broker. Note that since the broker was an EC2 instance in the cloud that was shut down soon after the evaluation ended, there was no way to debug this issue to obtain confirmation. The only reasonable solution we could come up with was to add a temporal offset to all pushed tweets. We set this offset to 139 seconds, the maximum gap between a system push time and the posted time of the tweet (on Twitter itself).

Results of the evaluation by the mobile assessors are shown in Table 2. For each run, the columns show the number of tweets that were judged relevant (R), redundant (D), and not relevant (N); the number of unjudged tweets (U); the length of each run (L), defined as the total number of tweets pushed by the system for the interest profiles that have at least one judgment. The next column shows the fraction of pushed tweets that were judged (C), defined as $(R+D+N)/L$. The table also reports the mean ($\bar{t}$) and median ($\tilde{t}$) latency of pushed tweets in seconds, measured with respect to the time the original tweet was posted. Next, the table shows "strict" and "lenient" precision (as defined in Section 2.4), with 95% binomial confidence intervals. The final column shows the run type: 'A' denotes automatic and 'P' manual preparation.

The rows in Table 2 are sorted by "strict" precision, but sorting by "lenient" precision doesn't greatly affect the rankings of the systems. The YoGosling baseline (WaterlooClarke, WaterlooBaseline-50) is noted in the results table. The placement of the YoGosling baseline suggests that the community has made quite a bit of progress on this task, since the best performing run from last year now falls in the middle of the pack.

Results of the evaluation by NIST assessors are shown in Table 3. The columns list the various metrics discussed in Section 2.5 and also the mean and median latency in seconds. Note that latency is computed with respect to the first tweet in each cluster, and thus a system may have a high latency even if it pushes a tweet immediately. The second to last column shows the length of each run, defined as the number of notifications pushed for the interest profiles that were assessed. The final column shows the run type: 'A' denotes automatic and 'P' manual preparation. The rows are sorted by EG-1, the primary metric. The YoGosling baseline is also marked in the results table; we see that it also places in the middle of the pack.

For reference, an empty run (i.e., doing nothing) would receive a score of 0.2339 for EG-1 and nCG-1 (with all other scores being zero). This is also shown in Table 3. As with the TREC 2015 Microblog Track, the baseline of doing nothing is surprisingly competitive given the current battery of metrics. The same observation has been noted in previous TREC Filtering Tracks. In a precision-focused task such as this, it is very important for systems to "keep quiet", which translates into the task of recognizing when there are no relevant documents.

Figure 3 shows a heatmap of the distribution of relevant and highly-relevant tweets by the NIST assessors: each column corresponds to an interest profile and each row corresponds to a day in the evaluation period. Figure 4 is organized in the same manner, but we only show the first tweet in each cluster.

Figure 5 shows scatterplots for "strict" precision (left) and "lenient" precision (right) vs. median latency. Each solid square represents a run. We do not see, overall, a trade-off between output quality and latency. That is, systems with higher latencies, which have more time to accumulate evidence on relevance and novelty, do not tend to perform better in terms of the various quality metrics.

The same scatterplots for batch evaluation metrics are

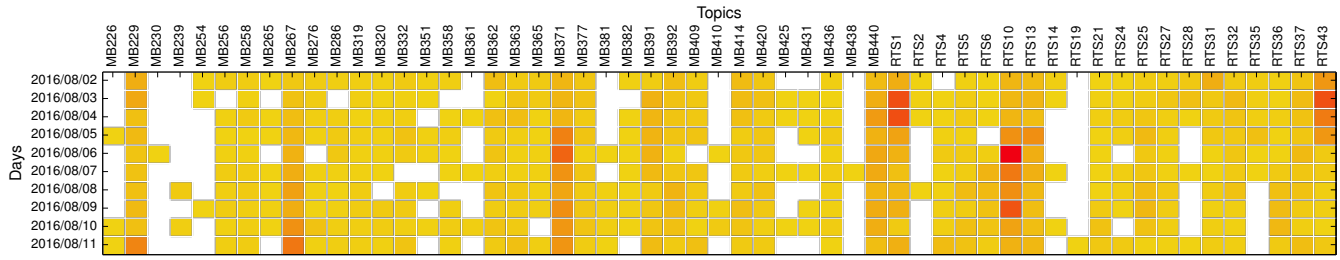

Figure 3: Heatmap of the distribution of all relevant and highly-relevant tweets: interest profiles in columns, days of the evaluation in rows.

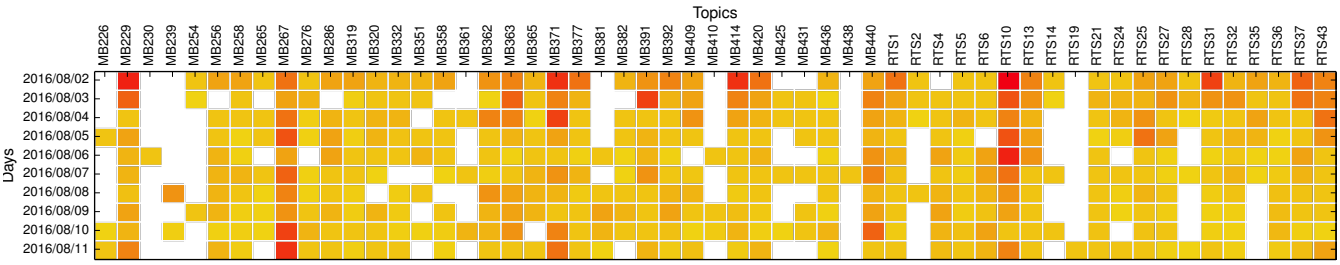

Figure 4: Heatmap of the distribution of the first tweet in each semantic cluster: interest profiles in columns, days of the evaluation in rows.

shown in Figure 6 (EG-1, EG-0, nCG-1, and nCG-0) and Figure 7 (GMP with $\alpha = \{0.33, 0.50, 0.66\}$). The scenario A runs are shown as solid squares. Once again, we do not observe any strong relationship between system output quality (as measured by the various metrics) and latency.

3.2 Scenario B

For scenario B, we received a total of 40 runs from 15 groups. Evaluation results based on NIST assessors are shown in Table 4. Runs are sorted by nDCG-1, with the YoGosling baseline (YoGoslingBSL) marked. For reference, the empty run would have received an nDCG-1 score of 0.2339, also shown in the results table.

The separation of quality metrics from latency allows us to unify the evaluation of scenario A and scenario B runs—we can simply convert scenario B runs into scenario A runs by pretending that all tweets were emitted at 23:59:59, and then running the evaluation scripts for scenario A exactly as before. The results of this conversion are shown in Figure 6 and Figure 7, where the scenario B runs are shown as empty squares. We would have expected that scenario B runs, on the whole, outperform scenario A runs (on quality metrics), since they had the advantage of accumulating evidence throughout the entire day. This, however, does not appear to be the case. Nevertheless, we believe this way of visualizing the results frames mobile push notifications and email digests as variants of the same underlying task, just differing in the amount of latency that is tolerated.

4. CONCLUSIONS

The TREC 2016 Real-Time Summarization Track had several innovative elements. Building on previous Microblog evaluations, we emphasized working systems that operate on the live Twitter stream, in an attempt to narrow the gap between research and practice. We continued to refine evaluation metrics as we better understand the nuances of push notifications. Most notably, this track represents, to our knowledge, the first deployment of an interleaved evaluation framework for prospective information needs, providing an opportunity to examine user behavior in a realistic setting. Our efforts will continue with another instance of the track in TREC 2017.

5. ACKNOWLEDGMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Additional support came from the U.S. National Science Foundation under IIS-1218043 and CNS-1405688. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsors.

6. REFERENCES

[1] J. Allan. *Topic Detection and Tracking: Event-Based Information Organization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[2] J. Aslam, F. Diaz, M. Ekstrand-Abueg, R. McCreadie, V. Pavlu, and T. Sakai. TREC 2015 Temporal Summarization Track overview. In *Proceedings of the Twenty-Fourth Text REtrieval Conference (TREC 2015)*, Gaithersburg, Maryland, 2015.

[3] A. Hanbury, H. Müller, K. Balog, T. Brodt, G. V. Cormack, I. Eggel, T. Gollub, F. Hopfgartner, J. Kalpathy-Cramer, N. Kando, A. Krithara, J. Lin, S. Mercer, and M. Potthast. Evaluation-as-a-Service: Overview and outlook. In *arXiv:1512.07454*, 2015.

[4] D. D. Lewis. The TREC-4 Filtering Track. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 165–180, Gaithersburg, Maryland, 1995.

[5] J. Lin, M. Efron, Y. Wang, and G. Sherman. Overview of the TREC-2014 Microblog Track. In *Proceedings of the Twenty-Third Text REtrieval Conference (TREC 2014)*, Gaithersburg, Maryland, 2014.

[6] J. Lin, M. Efron, Y. Wang, G. Sherman, and E. Voorhees. Overview of the TREC-2015 Microblog Track. In *Proceedings of the Twenty-Fourth Text REtrieval Conference (TREC 2015)*, Gaithersburg, Maryland, 2015.

[7] J. H. Paik and J. Lin. Do multiple listeners to the public Twitter sample stream receive the same tweets? In *Proceedings of the SIGIR 2015 Workshop on Temporal, Social and Spatially-Aware Information Access*, Santiago, Chile, 2015.

[8] X. Qian, J. Lin, and A. Roegiest. Interleaved evaluation for retrospective summarization and prospective notification on document streams. In *Proceedings of the 39th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016)*, pages 175–184, Pisa, Italy, 2016.

[9] S. Robertson and I. Soboroff. The TREC 2002 Filtering Track report. In *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg, Maryland, 2002.

[10] A. Roegiest, L. Tan, J. Lin, and C. L. A. Clarke. A platform for streaming push notifications to mobile assessors. In *Proceedings of the 39th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016)*, pages 1077–1080, Pisa, Italy, 2016.

[11] A. Schuth, K. Balog, and L. Kelly. Overview of the Living Labs for Information Retrieval Evaluation (LL4IR) CLEF Lab 2015. In *Proceedings of the 6th International Conference of the CLEF Association (CLEF'15)*, 2015.

[12] L. Tan, A. Roegiest, and C. L. Clarke. University of Waterloo at TREC 2015 Microblog Track. In *Proceedings of the Twenty-Fourth Text REtrieval Conference (TREC 2015)*, Gaithersburg, Maryland, 2015.

[13] L. Tan, A. Roegiest, C. L. A. Clarke, and J. Lin. Simple dynamic emission strategies for microblog filtering. In *Proceedings of the 39th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016)*, pages 1009–1012, Pisa, Italy, 2016.

[14] L. Tan, A. Roegiest, J. Lin, and C. L. A. Clarke. An exploration of evaluation metrics for mobile push notifications. In *Proceedings of the 39th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016)*, pages 741–744, Pisa, Italy, 2016.

[15] Y. Wang, G. Sherman, J. Lin, and M. Efron. Assessor differences and user preferences in tweet timeline generation. In *Proceedings of the 38th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015)*, pages 615–624, Santiago, Chile, 2015.

team	run	R	D	N	U	L	C	$\bar{\tau}$	$\tilde{\tau}$	P (strict)	P (lenient)	type
COMP2016	run3-13	193	4	141	1243	1573	0.215	14	14	0.5710 (0.5177, 0.6227)	0.5828 (0.5296, 0.6342)	P
COMP2016	run2-12	47	1	38	424	508	0.169	13	13	0.5465 (0.4416, 0.6475)	0.5581 (0.4529, 0.6584)	P
COMP2016	run1-11	54	1	50	498	600	0.175	13	13	0.5143 (0.4199, 0.6077)	0.5238 (0.4291, 0.6168)	P
CLIP	CLIP-A-1-08	91	1	89	507	679	0.267	493	40	0.5028 (0.4306, 0.5748)	0.5083 (0.4360, 0.5802)	A
umd_hcil	UmdHcilBaseline-49	20	0	22	176	218	0.193	3863	1323	0.4762 (0.3336, 0.6228)	0.4762 (0.3336, 0.6228)	A
CLIP	CLIP-A-2-09	158	7	171	911	1227	0.274	485	25	0.4702 (0.4175, 0.5236)	0.4911 (0.4380, 0.5443)	A
CLIP	CLIP-A-3-10	170	7	189	1071	1418	0.258	472	25	0.4645 (0.4140, 0.5157)	0.4836 (0.4328, 0.5347)	A
prna	PRNATaskA3-36	80	10	116	936	1134	0.182	380	39	0.3883 (0.3244, 0.4564)	0.4369 (0.3709, 0.5052)	A
IRIT	iritRunBiAm-21	201	6	323	1674	2177	0.243	13	13	0.3792 (0.3389, 0.4213)	0.3906 (0.3500, 0.4327)	A
PKUICST	run2-32	245	16	389	2197	2813	0.231	39	37	0.3769 (0.3405, 0.4148)	0.4015 (0.3645, 0.4397)	A
prna	PRNABaseline-34	26	0	44	228	293	0.239	502	30	0.3714 (0.2677, 0.4885)	0.3714 (0.2677, 0.4885)	A
prna	PRNATaskA2-35	117	7	191	1337	1640	0.192	1074	77	0.3714 (0.3199, 0.4260)	0.3937 (0.3413, 0.4486)	A
QU	QUExpP-38	45	1	76	348	463	0.263	155	15	0.3689 (0.2885, 0.4573)	0.3770 (0.2960, 0.4656)	A
PKUICST	run1-31	220	18	360	1997	2566	0.233	38	38	0.3679 (0.3302, 0.4073)	0.3980 (0.3595, 0.4378)	A
QU	QUExpT-39	33	1	56	280	365	0.247	108	17	0.3667 (0.2745, 0.4698)	0.3778 (0.2846, 0.4810)	A
PKUICST	run3-33	200	15	333	1830	2347	0.233	38	37	0.3650 (0.3257, 0.4061)	0.3923 (0.3523, 0.4338)	A
QU	QUBaseline-37	56	3	108	477	635	0.263	219	17	0.3353 (0.2681, 0.4099)	0.3533 (0.2848, 0.4283)	A
WaterlooClarke	WaterlooBaseline-50	148	12	286	1461	1888	0.236	44	42	0.3318 (0.2897, 0.3768)	0.3587 (0.3156, 0.4043)	A
ISIKol	MyBaseline-24	184	18	375	2610	3169	0.182	13	14	0.3189 (0.2822, 0.3580)	0.3501 (0.3123, 0.3899)	A
WaterlooLin	WaterlooBaseline-51	145	8	303	1367	1804	0.253	46	46	0.3180 (0.2769, 0.3621)	0.3355 (0.2937, 0.3801)	A
NUDTSNA	nudt_sna-29	262	34	546	3187	4011	0.210	47	46	0.3112 (0.2808, 0.3432)	0.3515 (0.3200, 0.3844)	A
NUDTSNA	nudt_sna-30	49	19	94	776	937	0.173	35	34	0.3025 (0.2370, 0.3771)	0.4198 (0.3465, 0.4967)	A
udel	udelRunTFIDF-44	119	8	292	1101	1504	0.279	30	28	0.2840 (0.2429, 0.3290)	0.3031 (0.2610, 0.3487)	A
udel_fang	UDInfoDFP-47	632	91	1526	6945	9133	0.246	37954	33732	0.2810 (0.2628, 0.3000)	0.3215 (0.3025, 0.3411)	A
IRLAB_DA-IICT	runA_daiict_irlab-23	105	10	259	1721	2083	0.180	1314	1224	0.2807 (0.2376, 0.3283)	0.3075 (0.2629, 0.3560)	P
HLJIT	MyBaseline-17	86	3	220	692	993	0.311	34	23	0.2783 (0.2313, 0.3308)	0.2880 (0.2404, 0.3409)	A
udel_fang	UDInfoSPP-48	467	66	1180	5171	6841	0.250	37900	33537	0.2726 (0.2521, 0.2942)	0.3112 (0.2897, 0.3335)	A
udel_fang	UDInfoSFP-46	591	89	1537	7014	9179	0.242	37522	33162	0.2666 (0.2486, 0.2854)	0.3067 (0.2879, 0.3262)	A
HLJIT	MyBaseline-18	63	7	168	833	1067	0.223	16	22	0.2647 (0.2127, 0.3242)	0.2941 (0.2399, 0.3549)	A
udel	udelRunTFIDFQ-45	113	6	327	1138	1568	0.284	29	28	0.2534 (0.2152, 0.2957)	0.2668 (0.2279, 0.3097)	P
HLJIT	HLJIT_LM-19	47	4	141	560	744	0.258	23	20	0.2448 (0.1894, 0.3102)	0.2656 (0.2082, 0.3323)	A
IRIT	Hamid-20	354	35	1067	5470	6887	0.211	80	56	0.2431 (0.2218, 0.2658)	0.2672 (0.2451, 0.2905)	A
IRIT	IritIrisSDA-22	136	17	448	1467	2060	0.292	14	14	0.2263 (0.1946, 0.2614)	0.2546 (0.2214, 0.2909)	A
NUDTSNA	nudt_sna-28	2	0	7	61	69	0.130	32	32	0.2222 (0.0632, 0.5474)	0.2222 (0.0632, 0.5474)	A
udel	udelRunBM25-43	2	1	6	38	47	0.191	27	27	0.2222 (0.0632, 0.5474)	0.3333 (0.1206, 0.6458)	P
CCNU2016NLP	CCNUNLPrun1-06	19	0	95	728	842	0.135	72	18	0.1667 (0.1094, 0.2457)	0.1667 (0.1094, 0.2457)	P
CCNU2016NLP	CCNUNLPrun2-07	17	3	89	763	870	0.125	50	26	0.1560 (0.0997, 0.2356)	0.1835 (0.1220, 0.2665)	P
BJUT	BJUTmyrf-03	136	53	1411	9656	11240	0.142	10059	549	0.0850 (0.0723, 0.0997)	0.1181 (0.1032, 0.1349)	A
BJUT	BJUTmydt-04	97	57	1141	9390	10677	0.121	13912	540	0.0749 (0.0618, 0.0905)	0.1189 (0.1024, 0.1377)	A
BJUT	BJUTmydt-05	54	10	899	8145	9102	0.106	6912	542	0.0561 (0.0432, 0.0724)	0.0665 (0.0524, 0.0840)	A
QUT_RTS	QUT_RTS-40	0	0	11	89	100	0.110	88103	94647	0.0000 (0.0000, 0.2588)	0.0000 (0.0000, 0.2588)	A

Table 2: Evaluation of scenario A runs by the mobile assessors. The first two columns show the participating team and run. The next columns show the number of tweets that were judged relevant (R), redundant (D), and not relevant (N); the number of unjudged tweets (U); the length of each run (L), defined as the total number of tweets pushed by the system for the interest profiles that have at least one judgment. The next columns show the fraction of pushed tweets that were judged (C), defined as $(R+D+N)/L$; the mean ($\bar{t}$) and median ($\tilde{t}$) latency of pushed tweets in seconds, measured with respect to the time the original tweet was posted; "strict" and "lenient" precision, with 95% binomial confidence intervals. The final column shows the run type: 'A' denotes automatic and 'P' manual preparation. Rows are sorted by "strict" precision, and the YoGosling baseline (WaterlooClarke, WaterlooBaseline-50) is noted.

team	run	EG-1	EG-0	nCG-1	nCG-0	$GMP_{.33}$	$GMP_{.50}$	$GMP_{.66}$	mean	median	length	type
COMP2016	run3-13	0.2698	0.0483	0.2909	0.0695	-0.3262	-0.2054	-0.0916	91549	24	443	P
QU	QUBaseline-37	0.2643	0.0321	0.2479	0.0157	-0.1357	-0.0888	-0.0447	173843	62478	169	A
COMP2016	run1-11	0.2565	0.0244	0.2515	0.0194	-0.0804	-0.0464	-0.0144	120947	7545	128	P
COMP2016	run2-12	0.2559	0.0220	0.2483	0.0143	-0.0585	-0.0326	-0.0082	154939	10055	101	P
QU	QUExpT-39	0.2552	0.0230	0.2455	0.0133	-0.0986	-0.0647	-0.0329	141163	46025	124	A
QU	QUExpP-38	0.2519	0.0233	0.2413	0.0127	-0.1641	-0.1134	-0.0657	161403	56863	178	A
IRIT	iritRunBiAm-21	0.2493	0.0332	0.2541	0.0380	-0.5464	-0.3817	-0.2267	102630	23	572	A
prna	PRNABaseline-34	0.2423	0.0119	0.2402	0.0098	-0.0770	-0.0522	-0.0289	81480	317	88	A
CLIP	CLIP-A-2-09	0.2407	0.0354	0.2382	0.0328	-0.2556	-0.1656	-0.0809	121940	12090	323	A
CLIP	CLIP-A-3-10	0.2397	0.0361	0.2415	0.0380	-0.3149	-0.2085	-0.1083	122959	3346	378	A
NUDTSNA	nudt_sna-30	0.2392	0.0214	0.2417	0.0238	-0.4295	-0.3067	-0.1911	370851	468940	422	A
CLIP	CLIP-A-1-08	0.2366	0.0206	0.2254	0.0093	-0.0950	-0.0629	-0.0328	227092	178997	113	A
PKUICST	run2-32	0.2347	0.0400	0.2433	0.0487	-0.7343	-0.5183	-0.3150	145028	22229	751	A
NUDTSNA	nudt_sna-28	0.2344	0.0004	0.2352	0.0013	-0.0416	-0.0299	-0.0189	436939	583469	39	A
PKUICST	run1-31	0.2342	0.0342	0.2447	0.0447	-0.6382	-0.4500	-0.2729	135444	22242	655	A
prna	PRNATaskA2-35	0.2342	0.0253	0.2302	0.0213	-0.4666	-0.3317	-0.2047	120735	210	463	A
Empty run		0.2339	0.0000	0.2339	0.0000	0.0000	0.0000	0.0000	-	-	-	-
PKUICST	run3-33	0.2329	0.0311	0.2343	0.0325	-0.5735	-0.4071	-0.2506	135116	42691	574	A
prna	PRNATaskA3-36	0.2329	0.0240	0.2290	0.0201	-0.3365	-0.2348	-0.1391	172796	3322	351	A
QUT_RTS	QUT_RTS-40	0.2315	0.0011	0.2306	0.0003	-0.0688	-0.0509	-0.0340	145162	145162	60	A
WaterlooLin	WaterlooBaseline-51	0.2298	0.0244	0.2315	0.0261	-0.5773	-0.4165	-0.2652	81515	74	549	A
WaterlooClarke	WaterlooBaseline-50	0.2289	0.0253	0.2330	0.0295	-0.6000	-0.4317	-0.2733	120909	8718	576	A
HLJIT	MyBaseline-18	0.2276	0.0383	0.2283	0.0390	-0.3698	-0.2576	-0.1520	185335	18755	391	A
udel	udelRunBM25-43	0.2205	0.0009	0.2202	0.0006	-0.0093	-0.0067	-0.0043	267113	267113	9	P
IRIT	IritIrisSDA-22	0.2181	0.0270	0.2317	0.0406	-1.1275	-0.8161	-0.5229	123013	13047	1059	A
umd_hcil	UmdHcilBaseline-49	0.2145	0.0038	0.2114	0.0007	-0.0664	-0.0482	-0.0311	186006	226867	60	A
HLJIT	MyBaseline-17	0.2085	0.0246	0.2018	0.0178	-0.4070	-0.2929	-0.1854	96672	10044	388	A
NUDTSNA	nudt_sna-29	0.1891	0.0320	0.2261	0.0689	-1.7681	-1.2835	-0.8273	135997	32730	1643	A
udel	udelRunTFIDF-44	0.1885	0.0475	0.1779	0.0368	-0.5584	-0.3897	-0.2310	118965	1783	589	A
udel	udelRunTFIDFQ-45	0.1879	0.0415	0.1781	0.0317	-0.6412	-0.4536	-0.2770	118760	5905	648	P
HLJIT	HLJIT_LM-19	0.1752	0.0109	0.1788	0.0145	-0.3256	-0.2357	-0.1511	174582	56865	305	A
ISIKol	MyBaseline-24	0.1748	0.0391	0.1766	0.0409	-1.0095	-0.7246	-0.4564	127914	13907	981	A
IRLAB_DA-IICT	runA_daiict_irlab-23	0.1708	0.0440	0.1546	0.0278	-0.7448	-0.5397	-0.3467	176709	36152	698	P
CCNU2016NLP	CCNUNLPrun1-06	0.1699	0.0003	0.1714	0.0018	-0.1732	-0.1290	-0.0874	355559	355559	146	P
CCNU2016NLP	CCNUNLPrun2-07	0.1643	0.0000	0.1643	0.0000	-0.2070	-0.1545	-0.1050	0	0	173	P
IRIT	Hamid-20	0.1224	0.0402	0.1916	0.1095	-2.5321	-1.8348	-1.1785	112089	75	2372	A
udel_fang	UDInfoSPP-48	0.0915	0.0594	0.1859	0.1538	-2.2833	-1.6344	-1.0237	137202	65971	2236	A
udel_fang	UDInfoDFP-47	0.0699	0.0574	0.2150	0.2025	-2.9761	-2.1313	-1.3361	130313	59284	2906	A
udel_fang	UDInfoSFP-46	0.0642	0.0517	0.1972	0.1847	-3.0633	-2.2018	-1.3909	124759	59085	2954	A
BJUT	BJUTmydt-05	0.0639	0.0014	0.0711	0.0086	-3.8657	-2.8813	-1.9547	149217	851	3250	A
BJUT	BJUTmydt-04	0.0339	0.0017	0.0408	0.0087	-4.5325	-3.3781	-2.2917	281845	283652	3809	A
BJUT	BJUTmyrf-03	0.0113	0.0077	0.0276	0.0240	-4.5495	-3.3763	-2.2722	232487	186427	3885	A

Table 3: Evaluation of scenario A runs by NIST assessors. The columns marked "mean" and "median" show the mean and median latency with respect to the first tweet in each cluster. The second to last column shows the length of each run, defined as the number of notifications pushed for the interest profiles that were assessed. The final column shows the run type: 'A' denotes automatic and 'P' manual preparation. Rows are sorted by EG-1. The YoGosling baseline (WaterlooClarke, WaterlooBaseline-50) and the empty run are noted.

team	run	nDCG-1	nDCG-0	type
COMP2016	PolyURunB3	0.2898	0.0684	manual preparation
NUDTSNA	nudt_sna	0.2708	0.0529	automatic
QU	QUJM16	0.2621	0.0300	automatic
QU	QUJMDR24	0.2558	0.0237	automatic
COMP2016	PolyURunB1	0.2536	0.0215	manual preparation
COMP2016	PolyURunB2	0.2523	0.0184	manual preparation
IRIT	RunBIch	0.2481	0.0321	automatic
WaterlooLin	YoGoslingBSL	0.2352	0.0299	automatic
PKUICST	PKUICSTRunB3	0.2348	0.0151	automatic
QU	QUDR8	0.2344	0.0094	automatic
Empty run		0.2339	0.0000	
prna	PRNATaskB1	0.2334	0.0352	automatic
NUDTSNA	nudt_bironT	0.2306	0.0681	automatic
WaterlooLin	YoGoslingLMGTFY	0.2273	0.0327	automatic
prna	PRNATaskB2	0.2244	0.0226	automatic
ISIKol	isikol_tag	0.2213	0.0196	automatic
IRIT	AmILPWSEBM	0.2208	0.1262	automatic
ISIKol	isikol_ti	0.2189	0.0171	automatic
udel	udelRunBM25B	0.2151	0.0008	manual preparation
udel	udelRunTFIDFQB	0.1991	0.0330	manual preparation
prna	PRNATaskB3	0.1987	0.0665	automatic
IRLAB_DA-IICT	IRLAB2	0.1972	0.0169	manual preparation
udel	udelRunTFIDFB	0.1970	0.0363	automatic
CCNU2016NLP	CCNUNLPrun1	0.1732	0.0018	manual preparation
PKUICST	PKUICSTRunB2	0.1569	0.1569	automatic
CCNU2016NLP	CCNUNLPrun2	0.1554	0.0000	manual preparation
IRLAB_DA-IICT	IRLAB	0.1532	0.0711	manual preparation
udel_fang	UDInfo_TlmN	0.1451	0.1416	automatic
udel_fang	UDInfo_TlmNlm	0.1445	0.1410	automatic
PKUICST	PKUICSTRunB1	0.1423	0.1423	automatic
udel_fang	UDInfo_TN	0.1315	0.1279	automatic
CLIP	CLIP-B-MAX	0.1244	0.0173	automatic
BJUT	bjutgbdt	0.1200	0.0914	automatic
HLJIT	HLJIT_LM	0.1155	0.1155	automatic
HLJIT	HLJIT_LM_TIME	0.1145	0.1145	automatic
IRIT	IritIrisSDB	0.1062	0.0651	automatic
BJUT	bjutdt	0.0978	0.0978	automatic
CLIP	CLIP-B-2015	0.0718	0.0718	automatic
HLJIT	HLJIT_LM_URL	0.0638	0.0638	automatic
BJUT	bjutrf	0.0582	0.0582	automatic
CLIP	CLIP-B-MIN	0.0312	0.0312	automatic

Table 4: Evaluation of scenario B runs by NIST assessors.

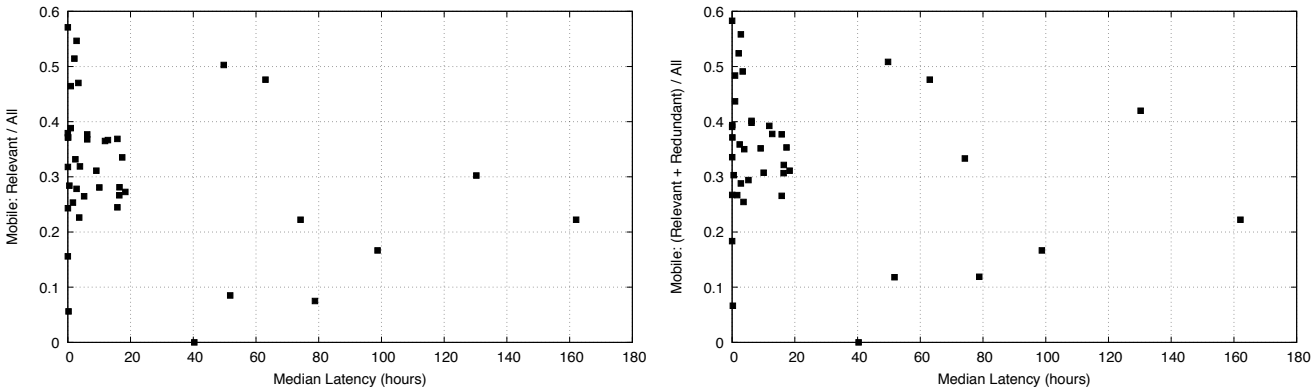

Figure 5: Scatterplots for "strict" precision (left) and "lenient" precision (right) vs. median latency. Each point represents a run.

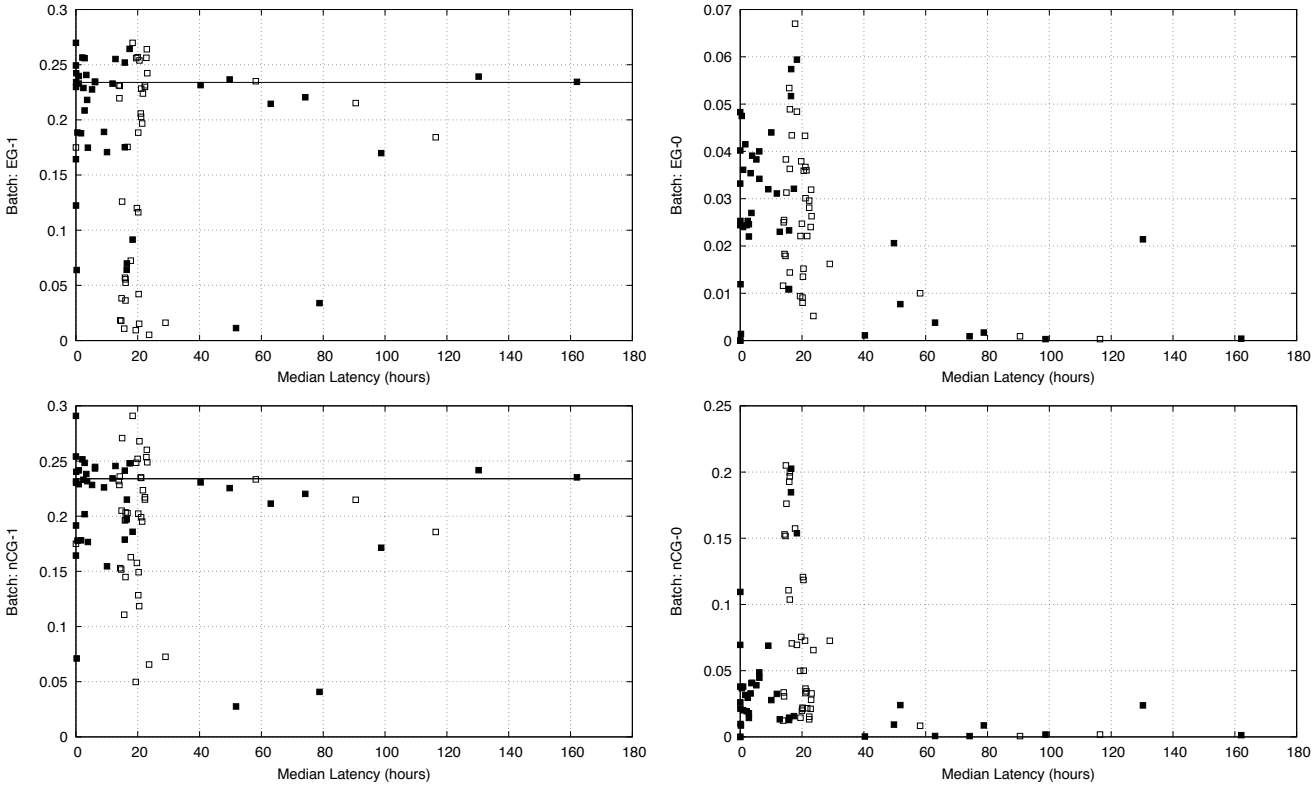

Figure 6: Scatterplots for various metrics vs. median latency. Each point represents a run: solid squares denote scenario A runs; empty squares denote scenario B runs treated as if they were scenario A runs. The solid horizontal line denotes the score of the empty run for EG-1 and nCG-1.

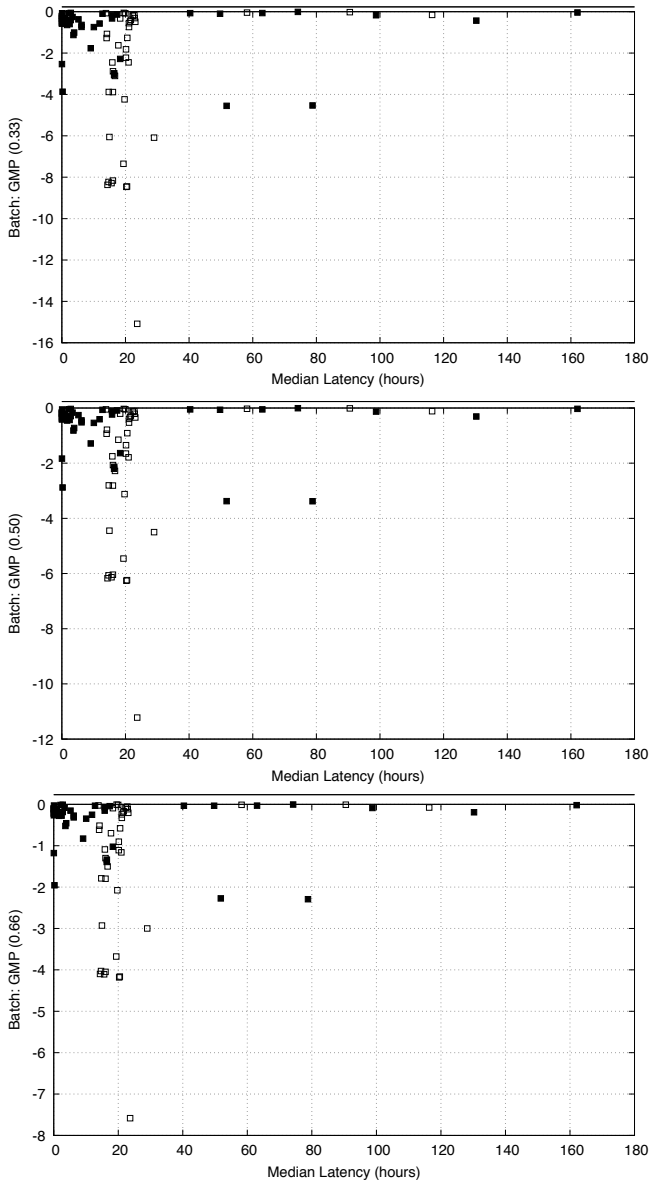

Figure 7: Scatterplots for GMP (different α) vs. median latency. Each point represents a run: solid squares denote scenario A runs; empty squares denote scenario B runs treated as if they were scenario A runs.