TREC 2014 Temporal Summarization Track Overview

Javed Aslam Fernando Diaz Matthew Ekstrand-Abueg
Richard McCreadie Virgil Pavlu
Tetsuya Sakai

Homepage: `http://www.trec-ts.org/`

February 17, 2015

1 Introduction

News events such as protests, accidents or natural disasters represent a unique information access problem where traditional approaches fail. For example, immediately after an event, the corpus may be sparsely populated with relevant content. Even when, after a few hours, relevant content becomes available, it is often inaccurate or highly redundant. At the same time, crisis events demonstrate a scenario where users urgently need information, especially if they are directly affected by the event.

The goal of this track is to develop systems for efficiently monitoring the information associated with an event over time. Specifically, we are interested in developing systems which can broadcast short, relevant, and reliable sentence-length updates about a developing event. The track has the following four main aims:

- To develop algorithms which detect sub-events with low latency,
- To model information reliability in the presence of a dynamic corpus,
- To understand and address the sensitivity of text summarization algorithms in an online, sequential setting, and
- To understand and address the sensitivity of information extraction algorithms in dynamic settings.

The remainder of this overview is structured as follows. Section 2 describes the temporal summarization task in detail. In Section 3, we discuss the corpus of documents from which the summaries are produced, while in Section 4, we discuss how temporal summarization systems are evaluated within the track. Section 5 details the process via which sentence updates were assessed. Finally,

```
<event>
  <id>1</id>
  <title>2012 Buenos Aires rail disaster</title>
  <description>...</description>
  <start>1329910380</start>
  <end>1330774380</end>
  <query>buenos aires train crash</query>
  <type>accident</type>
</event>
```

Figure 1: Example topic description for the topic '2012 Buenos Aires Rail Disaster'.

in Section 6, we summarize the performance of the participant systems to the 2014 track.

2 Task Description

The aim of this task is to emit a series of sentence updates over time about a named event given a high volume stream of input documents. In particular, the temporal summarization task focuses on large events with a high impact, such as protests, accidents or natural disasters. Each event is represented by a topic description, providing a textual query representing that event, along with start and end timestamps defining a period of time within which to track that event. An example topic description is illustrated in Figure 1.

For an event, participant systems process a stream of Web documents from the tracking period as defined in the topic in temporal order. The aim is to select sentences from those documents to emit as updates describing that event. The set of sentences emitted form a summary of that event over time. An optimal summary is one that covers all of the essential information about the event with no redundancy, where each new piece of information was added to the summary as soon as it became available.

For the 2014 task, participants produced temporal summaries for 15 different events, spanning accidents, natural disasters, storms, shootings and protests. Table 1 summarizes these 15 topics.

3 Corpus

The 2014 Temporal Summarization track used documents from the TREC KBA 2014 Stream Corpus. This corpus consists of a set of timestamped documents from a variety of news and social media sources covering the time period October 2011 through April 2013. Each document contains a set of sentences, each with a unique identifier.

Each event topic defines a subset of the time period covered by this corpus, representing the period to track that event. Participant systems had two options available when working with the corpus:

1. Extract the topic time periods from the TREC KBA 2014 Stream Corpus and process all documents from these time periods. This was the only option available to partatipants during the 2013 track.

2. Use a pre-filtered version of the TREC KBA 2014 Stream Corpus, denoted TREC-TS-2014F, which only contains documents from the 2014 event topic time periods. TREC-TS-2014F was also subject to pre-filtering such that it focuses on documents that are more likely to contain relevant sentences.

Each document within the TREC KBA 2014 Stream Corpus contains zero or more sentences (the sentence boundaries are pre-defined) and a timestamp representing when that document was crawled. Participants return a list of sentences extracted from the KBA corpus documents for each event. Each sentence is identified by the combination of a document identifier (which document the sentence came from) and a sentence identifier (the position of the sentence within the document). Additionally, when a sentence is emitted, the participant system also records the time with respect to the underlying document stream of that emission. If the participant system is making immediate binary emit/ignore decisions on a per sentence basis, then this timestamp will correspond to crawl-time of the document. However, some participant systems opted to delay the emission of sentences to collect more information before issuing updates - in these cases the timestamps recorded reflect the additional latency of these systems.

Participants were allowed to include runs that use information external to the KBA corpus. The use of external data had the following requirements:

- External data must have existed before the event start time, or
- External data must be time-aligned with the KBA corpus and no information after the simulation decision time can be used.

Similarly, supporting statistical models or auxiliary programs were subject to the same requirements. For example, participants were not to use a statistical model trained on data that existed after the event end time.

4 Evaluation

We evaluate runs according to their relevance, coverage, novelty, and latency of the updates.

- The relevance or precision of the summary with respect to the event topic, i.e. the degree to which the updates within the summary are on-topic and novel. This is measured by the (normalized) Expected Gain metric ($n\mathbf{EG}(\mathcal{S})$).

topic	type	#gold nuggets	#pooled updates
Costa Concordia disaster and recovery	accident	226	1008
Early 2012 European cold wave	natural disaster	73	654
2013 Eastern Australia Floods	storm	68	570
Boston Marathon bombings	shooting	76	984
Port Said Stadium riot	protest	47	813
2012 Afghanistan Quran burning protests	protest	75	759
In Amenas hostage crisis	hostage	48	768
2011-13 Russian protests	protest	89	916
2012 Romanian protests	protest	100	758
2012-13 Egyptian protests	protest	35	612
Chelyabinsk meteor	natural disaster	126	919
2013 Bulgarian protests against the Borisov	protest	117	608
2013 Shahbag protests	protest	138	723
February 2013 nor'easter	storm	105	951
Christopher Dorner shootings and manhunt	shooting	88	701

Table 1: TS2014 topics, with number of gold nuggets extracted by assessors, and number of participant updates pooled for matching.

- The coverage of the summary with respect to all of the essential information that could have been retrieved for the event. This is measured by the Comprehensiveness metric ($\mathbf{C}(\mathcal{S})$).
- The degree to which the information contained within the updates is outdated. This is measured by the Expected Latency metric (E[Latency]).

We also report the performance of all of the participant runs under a combined measure (that incorporates Expected Gain and Comprehensiveness with Latency included), i.e. the Harmonic Mean of normalized Expected Latency Gain ($n\mathbf{EG}_\tau(\mathcal{S})$) and Latency Comprehensiveness ($\mathbf{C}_\tau(\mathcal{S})$), denoted $\mathcal{H}$. This is the official target metric for the 2014 task. Detailed descriptions of metrics and how they are calculated can be found in Appendix A.

5 Judging

The evaluation process occurred in two phases:

(a) Gold Nugget Extraction, and

(b) Update-Nugget Matching

The first phase defined the space of relevant information for the queries. In particular, this involves the creation of a set of 'information nuggets' about each event that represent all of of the essential information that a good summary should contain. This phase also associates each information nugget with

a timestamp representing approximately when that information became public knowledge. The second phase generates a matching between updates provided by the participants to the information nuggets. It is this matching that forms the basis for evaluating a system's accuracy and coverage. A detailed description of these phases of judging can be found in Appendix B.

6 Results

We present an overview of the performance of the participant systems (runs) in Table 3. The last column in Table 3 reports the $\mathcal{H}$ of each participant run and the TREC average. From Table 3, we observe that the 2APSal run by cunlp was the best performing overall, closely followed by the BJUT and uogTr runs.

To examine why these runs are the best performing, we next report performance under both Expected Gain and Comprehensiveness separately. Note that we would expect there to be some degree of trade-off between these two metrics. The average performance of participating system runs under (normalized) Expected Gain and Comprehensiveness are reported in Table 3. From Table 3, we observe that in terms of (normalized) Expected Gain, the three BJUT runs and the 2APSal run by cunlp produced the most precise summaries, i.e. they returned the least content not matching one or more nuggets (that were not previously covered). Meanwhile, the systems that produced the most Comprehensive summaries (those that matched the most unique information nuggets) were the three uogTr systems, the cunlp 3AP run and the BUPT_PRIS Cluster1 approach. Figure 2 illustrates the distribution of all submitted runs in terms of (normalized) Expected Gain and Comprehensiveness. As we can see from Figure 2, the performance of the submitted runs varies greatly with respect to Expected Gain and Comprehensiveness, indicating that participants applied very different approaches to select sentences for inclusion into the summaries.

Next, we examine the performance of participating systems in terms of Latency. The fifth column of Table 3 reports the average Expected Latency weight observed for each system. Somewhat contrary to its name, a higher latency weight is better, above one means that on average, the system has found the matched information before it was added to Wikipedia. From Table 3, we observe that in general, there is not a large correlation between the latency of a system and its combined metric score ($\mathcal{H}$). In fact, the lowest performing run also has the best overall latency. On the other hand, the best performing runs have a good overall, but relatively average Expected Latency. We can compare any two runs to visualize the latency of their individual updates. To illustrate this, Figure 3 shows the times after the event began at which the first 75 updates were issued by the BJUT Q1 and uogTr2A runs, for event topic 13 (2013 Eastern Australia Floods). From Figure 3, we see that the uogTr2A runs began issuing updates very close to the beginning of the event (+3 hours from the event start), while the first update by the BJUT Q1 run was two days later (+86 hours from the event start). As expected, the overall latency of the uogTr2A run is higher. However, the BJUT Q1 run outperformed the uogTr2A run by

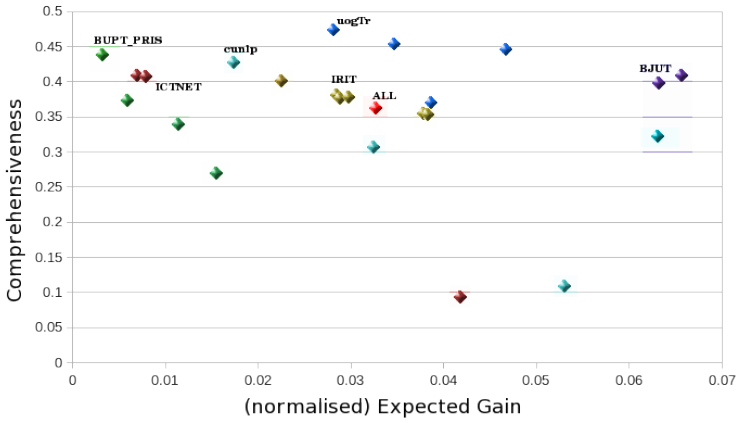


Figure 2: Participant run plot of (normalized) Expected Gain vs. Comprehensiveness.

a large margin under the combined measure (0.0992 vs 0.0526) for this topic. From this, we can conclude that the evaluation metrics are much less sensitive to latency considerations in comparison to Expected Gain or Comprehensiveness.

7 Conclusion

In general, the best performing runs submitted to the 2014 track balanced the need for high precision and novelty with topic coverage. Systems that focused on one or other components were less effective overall. From the scale of the results, it appears that attaining high precision is more difficult than achieving recall for this task, and hence it is here that further research is needed.

A Metrics

To evaluate the performance of the summaries produced by participant systems, we define the concept of explicit sub-events or 'nuggets', each with a precise timestamp and text describing the sub-event. An effective summary should cover as many of these nuggets as possible, while minimizing redundancy.

A sentence *update* is a timestamped short text string. We generally denote an update as the pair (string, time): $u = (u.string, u.t)$. For example $u =$ ("`The hurricane was upgraded to category 4`", 1330169580) represents an update describing the hurricane category, now 4, pushed out by system $\mathcal{S}$ at UNIX time 1330169580 (i.e. 1330169580 seconds after 0:00 UTC on January 1, 1970). In this year's evaluation, the update string is chosen from the set of segmented sentences in the corpus as defined in the guidelines.

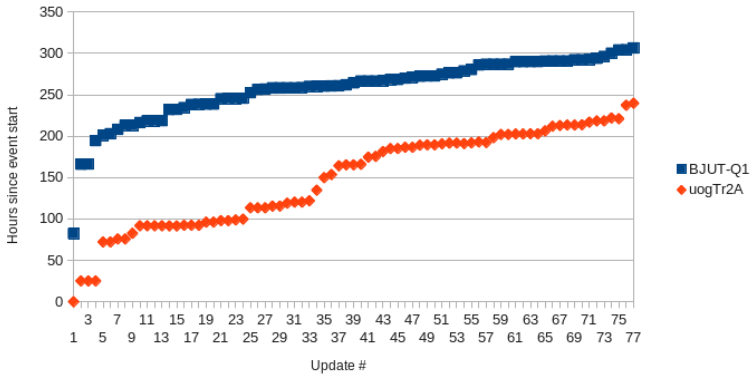


Figure 3: Comparison of update issue times for the BJUT Q1 run and the uogTr2A run for event topic 13 (2013 Eastern Australia Floods).

TeamID	RunID	$n\mathbf{EG}(\mathcal{S})$	$\mathbf{C}(\mathcal{S})$	E[Latency]	$\mathcal{H}$
cunlp	2APSal	0.0631	0.3220	1.2068	**0.1162**
BJUT	Q1	**0.0657**	0.4088	1.1491	0.1110
BJUT	Q2	0.0632	0.3979	1.1669	0.1091
BJUT	Q0	0.0632	0.3979	1.1669	0.1091
uogTr	uogTr2A	0.0467	0.4453	1.2322	0.0986
uogTr	uogTr4AC	0.0347	0.4539	1.2751	0.0793
uogTr	uogTr4ARas	0.0387	0.3691	1.2328	0.0772
IRIT	KW30H5NW3600	0.0383	0.3521	1.2221	0.0723
IRIT	KW30H5NW300	0.0378	0.3538	1.2208	0.0714
uogTr	uogTr4A	0.0281	**0.4733**	1.2522	0.0677
average		0.0327	0.3615	1.2943	0.0620
IRIT	KW80H5NW3600	0.0289	0.3764	1.2191	0.0604
IRIT	KW30H10NW300	0.0298	0.3780	1.2617	0.0602
cunlp	1APSalRed	0.0325	0.3058	1.1507	0.0602
IRIT	KW80H5NW300	0.0285	0.3806	1.2164	0.0596
ICTNET	run3	0.0531	0.1081	0.7004	0.0530
BUPT_PRIS	Cluster4	0.0155	0.2692	1.9140	0.0508
IRIT	KW80H10NW300	0.0225	0.4012	1.2621	0.0503
BUPT_PRIS	Cluster3	0.0115	0.3380	1.9165	0.0407
cunlp	3AP	0.0174	0.4265	1.3689	0.0403
ICTNET	run2	0.0418	0.0934	0.6266	0.0311
BUPT_PRIS	Cluster2	0.0059	0.3728	1.9170	0.0222
ICTNET	run4	0.0079	0.4070	1.2364	0.0178
ICTNET	run1	0.0070	0.4090	1.2314	0.0160
BUPT_PRIS	Cluster1	0.0033	0.4369	**1.9175**	0.0127

Table 2: Performance metrics per participant run, averaged over topics.

TeamID	RunID	#submitted updates	#pooled updates (std)
cunlp	2APSal	381.40	100.80 (33.58)
BJUT	Q1	116.86	89.40 (28.31)
BJUT	Q0	155.20	93.73 (22.70)
BJUT	Q2	155.20	93.73 (22.70)
uogTr	uogTr2A	1769.20	121.26 (46.23)
uogTr	uogTr4AC	2532.73	133.00 (45.15)
uogTr	uogTr4ARas	1735.73	104.06 (42.08)
IRIT	KW30H5NW3600	880.20	132.26 (29.11)
IRIT	KW30H5NW300	907.53	135.73 (30.69)
uogTr	uogTr4A	2860.46	159.40 (50.14)
cunlp	1APSalRed	1064.93	114.26 (15.30)
IRIT	KW30H10NW300	1401.66	178.40 (48.17)
IRIT	KW80H5NW3600	1388.33	148.93 (33.41)
IRIT	KW80H5NW300	1444.13	154.20 (34.46)
ICTNET	run3	1005.13	60.66 (38.24)
BUPT_PRIS	Cluster4	31444.20	387.53 (811.95)
IRIT	KW80H10NW300	2257.53	204.60 (52.18)
cunlp	3AP	5967.00	238.80 (50.64)
BUPT_PRIS	Cluster3	29937.93	351.00 (479.75)
ICTNET	run2	1101.73	68.40 (46.20)
BUPT_PRIS	Cluster2	39082.80	370.66 (359.43)
ICTNET	run4	5223.40	195.40 (74.50)
ICTNET	run1	5917.13	201.60 (73.28)
BUPT_PRIS	Cluster1	65954.86	568.66 (551.56)
ALL	-	8528.55	183.60 (266.38)

Table 3: For each run, the number of submitted updates and the number of pooled updates (with std over topics), averaged over topics. Pooling takes into account the number of updates per run, and the confidence stated for each update per run. The number of pooled updates per run includes the automatic matched updates in the average.

Two updates are semantically comparable using a text similarity measure or a manual annotation process applied to their *string* components; if two updates u and u' refer to the same information (semantically matching), then we write this as $u \approx u'$, irrespective of their timestamps. Because two systems might deliver the same update *string* at different times, it is generally not the case that $u.string = u'.string$ implies $u.t = u'.t$.

Given an event, our manual annotation process generates a set of gold standard updates called *nuggets*, extracted from wikipedia event pages and timestamped according to the revision history of the page. Editorial guidelines recommend that nuggets be a very short sentence, including only a single sub-event, fact, location, date, etc, associated with topic relevance. We refer to the canonical set of updates as $\mathcal{N}$. This manual annotation process is retrospective and subject to error in the precision of the timestamp. As a result we might encounter situations where the timestamp of the nugget is later than the earliest matching update.

In response to an event's news coverage, a system/run broadcasts a set of timestamped updates generated in the manner described in the Guidelines. We refer to a system's set of updates as $\mathcal{S}$. The set of updates received before time τ is,

$$\mathcal{S}_\tau = \{u \in \mathcal{S} : u.t < \tau\} \tag{1}$$

Our goal in this evaluation is to measure the precision, recall, timeliness, and novelty of updates provided by a system.

A.1 Preliminaries

Our evaluation metrics are based on the following auxiliary functions.

- **Nugget Relevance.** Each nugget $n \in \mathcal{N}$ has an associated relevance/importance grade,

$$\mathbf{R} : \mathcal{N} \to [0, 1] \tag{2}$$

  $\mathbf{R}(n)$ measures the importance of the content (information) in the nugget. Nugget importance was provided on a 0-3 scale by assessors (no importance to high importance). For graded relevance, we normalize on an exponential scale, since high importance nuggets are described as "of key importance to the query", whereas low importance nuggets are "of any importance to the query". When binary relevance is needed, everything of any relevance is relevant (0 is the only non-relevant grade). The actual relevance functions used are presented below; $n.i$ denotes the nugget importance as assigned by the assessor.

$$\mathbf{R}_{\text{graded}}(n) = \frac{e^{n.i}}{e^{\max_{n' \in \mathcal{N}} n'.i}} \qquad \text{Graded relevance} \tag{3}$$

$$\mathbf{R}_{\text{binary}}(n) = \begin{cases} 1 & \text{iff } n.i > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{Binary relevance} \tag{4}$$

Note that for graded relevance, returning exactly the nugget set as the system output updates and nothing more ("perfect system"), would usually *not* result in an expected gain of 1. However, using binary relevance, the perfect system would score an expected gain of 1.

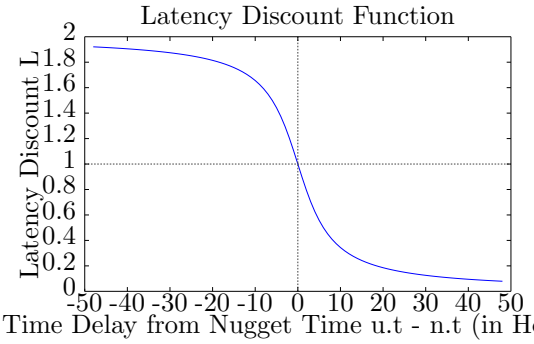
The relevance can be discounted in time or in size, hence the following discounting functions.

- **Latency Discount.** Given a reference timestamp of a matching nugget, t^*, a latency penalty function $\mathbf{L}(t^*, t)$ is a monotonically *decreasing* function of $t - t^*$. A system may return an update matching Wikipedia information before the Wikipedia information exists; thus we use a function that is smooth and decays on both the positive and negative sides.

  The actual function used is presented below with arguments the nugget Wikipedia time (wiki-edit timestamp) $n.t$, and the update time $u.t$ as indicated by the system.

$$\mathbf{L}(n.t, u.t) = 1 - \frac{2}{\pi} \arctan(\frac{u.t - n.t}{\alpha}) \qquad \text{latency-discount} \qquad (5)$$

$$\alpha = 3600 * 6 \qquad \text{latency-step (6 hours)} \qquad (6)$$



  Current parameters allow the latency discount factor to vary from 0 to 2 (1 means nugget time equal to update time), and flattens at around one day($\pm$ 24 hours). Note that as a result, gain and expected gain can be greater than 1.

- **Verbosity Normalization.** The task definition assumes that a user receives a stream of updates from the system. Consequently, we want to penalize systems for including unreasonably long updates, since these easily lead to significantly higher reading effort. The verbosity can be defined as a string length penalty function, monotonically increasing in the number of words of the update *string*. We will refer to this normalization function as $\mathbf{V}(u)$.

  For the actual verbosity implementation, we approximate the number of extra nuggets worth of information in a given update. This is done by

finding all text which did not match a nugget (as defined by the assessors), and dividing the number of words in the text by the average number of words in a nugget for that query.

$$\mathbf{V}(u) = 1 + \frac{|all\_words_u| - |nuggetmatching\_words_u|}{AVG_n|words_n|} \tag{7}$$

$$= 1 + \frac{|u| - |\cup_{n \in \mathbf{M}^{-1}(u,\mathcal{S})} \mathbf{M}(n,\mathcal{S})|}{\text{avg}_{n \in \mathcal{N}}|n|} \tag{8}$$

where $|u|$, $|n|$ are the length (in number of words) of the update u, and nugget n.

Note that if an update has all its words being part of some match to a nugget, the verbosity is $V(u)$=1; otherwise $V(u)-1$ is an approximation of the "extra non-matching words" in terms of equivalent number of nuggets.

- **Update-Nugget Matching.** We also define a key *earliest matching function* between a nugget and an update set,

$$\mathbf{M}(n,\mathcal{S}) = \text{argmin}_{\{u \in \mathcal{S}: n \approx u\}} u.t \tag{9}$$

or $\emptyset$ if there is no matching update for n. $\mathbf{M}(n,\mathcal{S})$ should be interpreted as "given n, the earliest matching update in $\mathcal{S}$."

We also define the set of nuggets for which u is the earliest matching update as,

$$\mathbf{M}^{-1}(u,\mathcal{S}) = \{n \in \mathcal{N} : \mathbf{M}(n,\mathcal{S}) = u\} \tag{10}$$

Note that an update can be the earliest matching update for more than one nugget.

A.2 Metrics

Using the previously defined notion of *relevance, latency, verbosity, and matching* we can define several measures of interest for Temporal Summarization.

Given an update u and a matching nugget n (i.e. $u \approx n$), we can define the **discounted gain** as,

$$\mathbf{g}(u,n) = \mathbf{R}(n) \times \text{discounting factor} \tag{11}$$

Given the previously defined discounts, we have the following family of discounted gains,

$$\mathbf{g_F}(u,n) = \mathbf{R}(n) \qquad \text{discount-free gain} \tag{12}$$

$$\mathbf{g_L}(u,n) = \mathbf{R}(n) \times \mathbf{L}(n.t, u.t) \qquad \text{latency-discounted gain} \tag{13}$$

Since an update can be the earliest to match several nuggets ($u \approx n$), we define the gain of an update with respect to a system (or participant run) $\mathcal{S}$

as the sum of [latency-discounted] relevance of the nuggets for which it is the earliest matching update:

$$\mathbf{G}(u, \mathcal{S}) = \sum_{n \in \mathbf{M}^{-1}(u,\mathcal{S})} \mathbf{g}(u, n) \tag{14}$$

where the gain can be either of the discounted gains described earlier. Note that for an appropriate discounting function, $\mathbf{G}(u, \mathcal{S}) \in [0, 1]$, although for the latency-discounted gain, given the imperfect nature of model timestamps, $\mathbf{G_L}(u, \mathcal{S}) \in [0, 2]$.

One way to evaluate a system is to measure the **expected gain** for a system update. This is similar to traditional notions of precision in information retrieval evaluation. Over a large population of system updates, we can estimate this measure reliably. The computation of the expected update gain for system $\mathcal{S}$ by time τ is the average of the gain per update:

$$\begin{aligned} n\mathbf{EG}(\mathcal{S}) &= \frac{1}{Z|\mathcal{S}|} \sum_{u \in \mathcal{S}} \mathbf{G}(u, \mathcal{S}) && (15) \\ &= \frac{1}{Z|\mathcal{S}|} \sum_{u \in \mathcal{S}} \sum_{n \in \mathbf{M}^{-1}(u,\mathcal{S})} \mathbf{g}(u, n) \\ &= \frac{1}{Z|\mathcal{S}|} \sum_{\{n \in \mathcal{N} : \mathbf{M}(n,\mathcal{S}) \neq \emptyset\}} \mathbf{g}(\mathbf{M}(n, \mathcal{S}), n) && (16) \end{aligned}$$

where Z is the maximum obtainable expected gain per topic (similar to DCG normalization). Additionally, we may penalize "verbosity" by normalizing not by the number of system updates, but by the overall **verbosity of the system**

$$n\mathbf{EG_V}(\mathcal{S}) = \frac{1}{\sum_{u \in \mathcal{S}} \mathbf{V}(u)} \frac{1}{Z} \sum_{\{n \in \mathcal{N} : \mathbf{M}(n,\mathcal{S}) \neq \emptyset\}} \mathbf{g}(\mathbf{M}(n, \mathcal{S}), n) \tag{17}$$

Our definition of $\mathbf{g}$ is such that it:

- does not penalize for large a update matching several nuggets, as opposed to a few small updates each matching a nugget, due to verbosity weighting,
- penalizes for late updates (against matched nugget reference timestamp), and
- penalizes "verbosity" of updates text not matching any nuggets.

Furthermore, we have that $\mathbf{G}(u, \mathcal{S}_\tau) \in [0, 1]$ if all update timestamps are at or after matching model timestamps. Over a set of events, the mean expected gain is defined as,

$$\mathbf{MEG} = \frac{1}{|\mathcal{E}|} \sum_{\epsilon \in \mathcal{E}} \mathbf{EG}(\mathcal{S}^\epsilon) \tag{18}$$

where $\mathcal{E}$ is the set of evaluation events and $\mathcal{S}^\epsilon$ is the system submission for event ϵ.

Because a user interest may be concentrated immediately after an event and because a system's performance (in terms of gain) may be dependent on the time after an event, we will also consider a **time-sensitive expected gain** for the first τ seconds,

$$\mathbf{EG}_\tau(\mathcal{S}) = \mathbf{EG}(\mathcal{S}_\tau) \tag{19}$$

with $\mathbf{MEG}_\tau$ defined similarly.

In addition to good expected gain, we are interested in a system providing a comprehensive set of updates. That is, we would like the system to cover as many nuggets as possible. This is similar to traditional notions of recall in information retrieval evaluation. Given a set of system updates, $\mathcal{S}$, we define the **comprehensiveness** (and **latency-comprehensiveness**) of the system as:

$$\mathbf{C}(\mathcal{S}) = \frac{1}{\sum_{n\in\mathcal{N}} \mathbf{R}(n)} \sum_{\{n\in\mathcal{N}:\mathbf{M}(n,\mathcal{S})\neq\emptyset\}} \mathbf{g}(\mathbf{M}(n,\mathcal{S}),n) \tag{20}$$

$$= \frac{1}{\sum_{n\in\mathcal{N}} \mathbf{R}(n)} \sum_{u\in\mathcal{S}} \sum_{n\in\mathbf{M}^{-1}(u,\mathcal{S})} \mathbf{g}(u,n)$$

$$= \frac{1}{\sum_{n\in\mathcal{N}} \mathbf{R}(n)} \sum_{u\in\mathcal{S}} \mathbf{G}(u,\mathcal{S}) \tag{21}$$

We also define a time-sensitive notion of comprehensiveness,

$$\mathbf{C}_\tau(\mathcal{S}) = \mathbf{C}(\mathcal{S}_\tau) \tag{22}$$

with an aggregated measure defined as,

$$\int_{t_s}^{t_e} \mathbf{C}_\tau(\mathcal{S}) d\tau \tag{23}$$

which measures how quickly a system captures nuggets.

B Judging

B.1 Gold Nugget Extraction

In this first phase, assessors were asked to read all edits of the Wikipedia article for each query, manually extracting text perceived as relevant and novel for that edit. Additionally, assessors assigned an importance grade to every text fragment, or nugget, as well as noted any dependencies in the information. An example portion of the extraction interface can be seen in Figure 4.

In order to simplify later matching, assessors were told to extract nuggets such that they were atomic pieces of information relevant to the query. However,

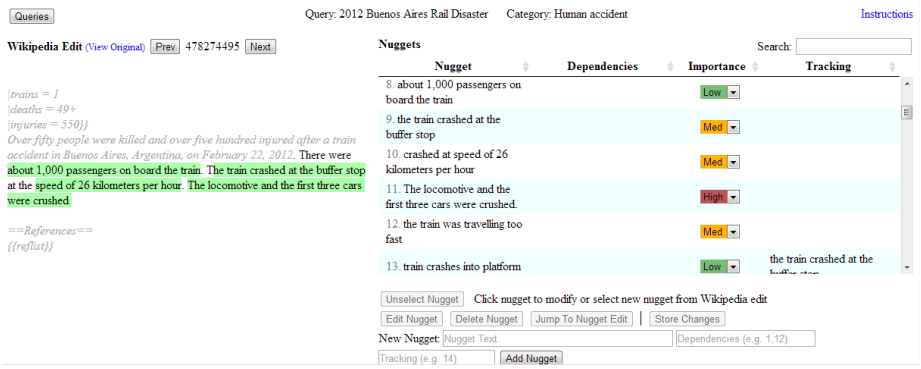


Figure 4: Extraction interface used by assessors to extract nuggets from Wikipedia edits.

knowing that information can be highly contextual, we allow for the notion of dependencies between nuggets: a nugget may be relevant to a query if and only if another nugget is also present. For evaluation purposes, a nugget was considered unmatched if it had unmatched nuggets on which it depended.

Additionally, we provided a method for assessors to track updates to pieces of information. This was primarily used to allow them to collate their work and reduce redundancy, but it was also used in the matching phase to help the assessor find the closest piece of information to a match.

B.2 Update-Nugget Matching

Once submissions were received, we performed a variant of depth-pooling in order to sample updates for evaluation. We sampled the top 60 updates per query and run as sorted by the provided confidence scores (highest first). Additionally, we performed near-duplicate detection among update text to increase the covered set. This resulted in sampled update counts as per Table 2. One note here is that not all runs contained 60 updates per query; for the run-query pairs with less than 60 updates, all updates were sampled.

The sampled updates were presented in an interface similar to the one for extraction. Assessors examined and matched updates to nuggets by selecting portions of updates which matched a given nugget, as nuggets are atomic but updates are not. An assessor was allowed to break a nugget into two or more new nuggets to improve atomicity if desired. Note that a nugget may match multiple updates, and an update may match multiple nuggets. An example view of the matching interface can be seen in Figure 5.

B.2.1 Automatic matches for unpooled updates

The participant updates that did not make it to the pool for manual matching form the set of "unpooled updates". We performed an automatic exact match

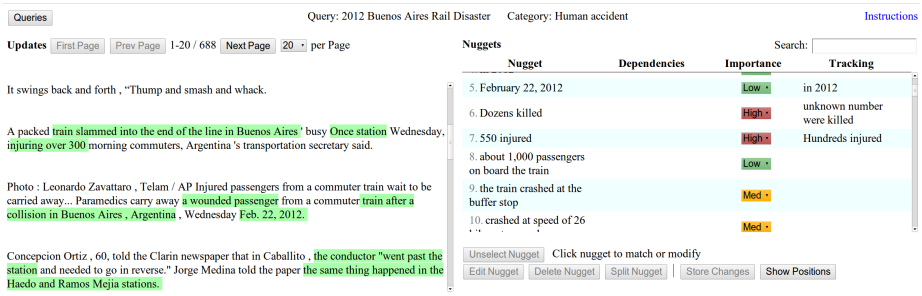


Figure 5: Matching interface used by assessors to match updates and nuggets.

between these unpooled updates and the known relevant pooled updates (manually matched); the updates that matched a known relevant pooled update are also considered relevant and included as matching nuggets for evaluation purposes. All updates, both pooled and unpooled, that do not match any nugget (manual) or other relevant update (automatic), are considered nonrelevant for evaluation metrics.