TREC 2015 Temporal Summarization Track Overview

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

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

1 Introduction

There are many summarization scenarios that require updates to be issued to users over time. For example, during unexpected news events such as natural disasters or mass protests new information rapidly emerges. The TREC Temporal Summarization track aims to investigate how to effectively summarize these types of event in real-time. In particular, the goal is to develop systems which can detect useful, new, and timely sentence-length updates about a developing event to return to the user. In contrast to classical summarization challenges (such as DUC or TAC), the summaries produced by the participant systems are evaluated against a ground truth list of information nuggets representing the space of information that a user might want to know about each event. An optimal summary will cover all of the information nuggets in the minimum number of sentences. Also in contrast to classic summarization and newer timeline generation tasks, the Temporal Summarization track focuses on performing this analysis *online* as documents are indexed.

For the third 2015 edition of the Temporal Summarization track, we had four main aims. First, to better address the issues with run incompleteness by producing larger run pools and by using pool expansion based on sentence similarity. Second, to lower the barrier to entry for new groups by providing multiple sub-tasks using corpora of varying sizes, allowing groups to pick the task(s) that their infrastructure can cope with. Third, to refine the metrics to better incorporate latency by considering timeliness against the corpus as well as against updates to the Wikipedia page. Finally, to continue to increase the number of events covered by the evaluation.

This is the final year of the Temporal Summarization track. For 2016, the track will merge with the Microblog track to become the new Real-Time Summarization (RTS) Track. This new RTS track will still tackle the same challenges as the Temporal Summarization track, but will incorporate microblog streams and will include a new Living-Lab style evaluation in addition to the classical

```
<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'.

dataset-based evaluation.

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, 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.

More precisely, 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. In contrast to previous years, there are three sub-tasks running in 2015:

Task 1: Full Filtering and Summarization

- Participants will be provided *very high-volume streams* of news articles and blog posts crawled from the Web for a set of events. Only a very small portion of the stream will be relevant to the event.

- Each participant will need to process those streams in time order, *filter out irrelevant content* and then select sentences from those documents to return to the user as updates describing each event over time.

Task 2: Partial Filtering and Summarization

- Participants will be provided *high-volume streams* of news articles and blog posts crawled from the Web for a set of events.
- Each participant will need to process those streams in time order, *filter out irrelevant content* and then select sentences from those documents to return to the user as updates describing each event over time.

Task 3: Summarization Only

- Participants will be provided low-volume streams of *on-topic documents* for a set of events.
- Each participant will need to process those streams in time order selecting sentences from the documents contained within each stream to return the user as updates over time.

In summary, the sub-task defines the corpus that the participant uses to find sentences to return to the user. Task 1 uses a generic crawl of the Web from the time period of the event, which will require a large amount of filtering. Task 2 uses an automatically filtered Web crawl, that removed documents that are very unlikely to be relevant, but this crawl will still need significant further filtering. Task 3 uses a low-volume set of manually selected documents. For the 2015 task, participants produced temporal summaries for 20 different events, spanning accidents, natural disasters, storms, shootings and protests. Table 1 summarizes these 20 topics.

3 Corpus

The 2015 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 three options available when working with the corpus, which defines which sub-task they were involved in:

1. Extract the topic time periods from the TREC KBA 2014 Stream Corpus and process all documents from these time periods. Using this approach results in a Task 1 run. This was the only option available to participants during the 2013 track.

EID	Event Title	# Nuggets	# Pooled Updates
26	Vauxhall helicopter crash	22	1,222
27	Cyclone Nilam	24	1,191
28	2013 Savar building collapse	60	1,471
29	2013 Hyderabad blasts	90	1,347
30	Brazzaville arms dump blasts	77	1,275
31	2012 India blackouts	33	991
32	Reactions to Innocence of Muslims	226	1,645
33	Battle of Konna	41	1,233
34	February 2013 Quetta bombing	26	1,137
35	15 April 2013 Iraq attacks	20	1,233
36	19 March 2013 Iraq attacks	48	1,373
37	2011-12 Los Angeles arson attacks	62	1,336
38	2013 Thane building collapse	29	1,390
39	2013 United States embassy bombing in Ankara	10	755
40	22 December 2011 Baghdad bombings	37	1,053
41	Aleppo University bombings	26	1,136
42	Carnival Triumph 2013 Engine Room Fire	46	1,271
43	USS Guardian (MCM-5) January 2013 Grounding	11	769
44	2012 Indian Ocean earthquakes	65	1,223
45	2012 Haida Gwaii earthquake	57	1,129
46	2012 Catalan independence demonstration	60	1,036

Table 1: TRECTS 2015 topics, with number of gold nuggets extracted by assessors, and number of participant updates pooled for matching.

2. Use a pre-filtered version of the TREC KBA 2014 Stream Corpus, denoted TREC-TS-2015F, which only contains documents from the 2015 event topic time periods. TREC-TS-2015F was also subject to pre-filtering such that it focuses on documents that are more likely to contain relevant sentences. Using this approach results in a Task 2 run. This option was also available to participants during the 2014 track.

3. Use a manually pre-filtered version of the TREC KBA 2014 Stream Corpus, denoted TREC-TS-2015RelOnly, which only contains only documents that were annotated as containing some relevant content from the 2015 event topic time periods. TREC-TS-2015RelOnly is a subset of TREC-TS-2015F. Using this approach results in a Task 3 run. This option was new for 2015.

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 ($n\mathbf{EG}(\mathcal{S})$).
- 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, Comprehensiveness and Expected Latency), i.e. the Harmonic Mean of normalized Expected Latency Gain ($\mathbf{EG}_{\tau}(\mathcal{S})$) and Latency Comprehensiveness ($\mathbf{C}_{\tau}(\mathcal{S})$), denoted $\mathcal{H}$. This is the official target metric for the 2015 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

Table 2: Performance on task 1. Performance for systems summarizing the entire document stream, without using any of the filtered set. Runs sorted by $\mathcal{H}$, the harmonic mean of latency gain and latency comprehensiveness.

TeamID	RunID	$n\mathbf{EG}(\mathcal{S})$	$\mathbf{C}(\mathcal{S})$	E[Latency]	$\mathcal{H}$
cunlp	2LtoSnofltr20	0.1224	0.4691	0.8086	0.1531
CWI	IGnPrecision	0.1894	0.4678	0.6273	0.1396
Mean		**0.1533**	**0.4575**	**0.6507**	**0.1279**
CWI	IGn	0.1620	0.5137	0.6538	0.1248
CWI	docs	0.1242	0.4680	0.6658	0.1222
CWI	titles	0.1915	0.3107	0.5171	0.1150

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 Tables 2 (Task 1), 3 (Task 2), and 4 (Task 3). The last column of the tables reports the $\mathcal{H}$ of each participant run and the TREC average. Due to per-task normalization, metric values across tasks are not comparable.

Only two teams participated in Task 1 due to the overhead involved in processing the full KBA corpus. Although the ranking of cunlp is consistent with its position in Task 2, we note its expected gain is below average so the performance of the run comes from its strong comprehensiveness and lower latency. The seven teams participating in Task 2 exhibited a range of performance, with even the three above-average teams representing high gain (WaterlooClarke) and high comprehensiveness (cunlp, IRIT). In fact, the comprehensiveness of the high gain runs was below average, emphasizing the impressive magnitude of gains from these runs. The tradeoff between gain and comprehensiveness can be visualized in Figure 2. Regardless of whether they focused on gain or comprehensiveness, top performing runs also consistently had better than average latency. Runs in the Task 3 exhibited a similar tradeoff between gain and comprehensiveness although, in this case, the performance in different regimes was more pronounced.

Although we normalized metric values per task, when we normalized the values across tasks, we observed a similar ordering of systems. This suggests that, although sampling the corpus removes the ability to match certain nuggets,

Table 3: Performance on task 2. Performance for systems summarizing TREC-TS-2015F. Runs sorted by $\mathcal{H}$, the harmonic mean of latency gain and latency comprehensiveness.

TeamID	RunID	$n\mathbf{EG}(\mathcal{S})$	$\mathbf{C}(\mathcal{S})$	E[Latency]	$\mathcal{H}$
WaterlooClarke	UWCTSRun1	0.2350	0.3520	0.6612	0.1762
WaterlooClarke	UWCTSRun3	0.2252	0.3421	0.6643	0.1718
WaterlooClarke	UWCTSRun2	0.2872	0.2584	0.6551	0.1710
cunlp	3LtoSfltr5	0.1371	0.4870	0.6392	0.1282
cunlp	1LtoSfltr20	0.1203	0.5372	0.6287	0.1100
IRIT	FS1A	0.0849	0.4959	0.6051	0.0719
cunlp	4APSAL	0.1011	0.4584	0.5108	0.0674
Mean		**0.0666**	**0.4342**	**0.4697**	**0.0499**
IRIT	FS2A	0.0518	0.5899	0.6285	0.0476
BJUT	DMSL1NMF2	0.0445	0.6123	0.4539	0.0354
BJUT	DMSL1AP1	0.0413	0.6155	0.4701	0.0338
l3sattrec15	l3sattrec15run1	0.0408	0.3612	0.3743	0.0268
l3sattrec15	l3sattrec15run3	0.0400	0.3669	0.3712	0.0262
IRIT	FS1B	0.0422	0.2939	0.3913	0.0259
IRIT	FS2B	0.0306	0.3391	0.4491	0.0239
USI	InL2DecrQE1ID1	0.0182	0.5713	0.5806	0.0196
USI	InL2DecrQE2ID2	0.0169	0.5758	0.5836	0.0184
udel_fang	WikiOnlyFS2	0.0206	0.5819	0.4600	0.0176
udel_fang	ProfOnlyFS3	0.0258	0.5294	0.4122	0.0174
USI	InL2StabQE2ID3	0.0171	0.6133	0.5238	0.0170
udel_fang	WikiProfMixFS1	0.0189	0.5965	0.4660	0.0166
l3sattrec15	l3sattrec15run2	0.0283	0.2276	0.2560	0.0164
USI	InL2IncrQE2ID4	0.0179	0.5837	0.2888	0.0108

Table 4: Performance on task 3. Performance for systems summarizing TREC-TS-2015RelOnly. Runs sorted by $\mathcal{H}$, the harmonic mean of latency gain and latency comprehensiveness.

TeamID	RunID	$n\mathbf{EG}(\mathcal{S})$	$\mathbf{C}(\mathcal{S})$	E[Latency]	$\mathcal{H}$
WaterlooClarke	UWCTSRun4	0.1840	0.1710	0.3983	0.0853
BJUT	DMSL2N2	0.0645	0.6557	0.5606	0.0649
uogTr	uogTrhEQR2	0.0667	0.5459	0.5335	0.0639
uogTr	uogTrhEEQR4	0.0714	0.5342	0.5249	0.0632
BJUT	DMSL2A1	0.0600	0.6777	0.5787	0.0622
uogTr	uogTrdEQR1	0.0402	0.6590	0.6741	0.0508
uogTr	uogTrdEEQR3	0.0418	0.6096	0.6401	0.0505
Mean		**0.0595**	**0.5627**	**0.5524**	**0.0472**
UvA.ILPS	COS	0.0428	0.5708	0.5951	0.0471
UvA.ILPS	COSSIM	0.0281	0.7325	0.6952	0.0372
udel_fang	WikiOnly2	0.0446	0.5522	0.5008	0.0353
UvA.ILPS	LexRank	0.0224	0.7490	0.6836	0.0299
ISCASIR	runvec2	0.0190	0.7881	0.7210	0.0250
UvA.ILPS	LDAv2	0.0202	0.7423	0.6338	0.0241
ISCASIR	runvec1	0.0174	0.7852	0.6458	0.0215

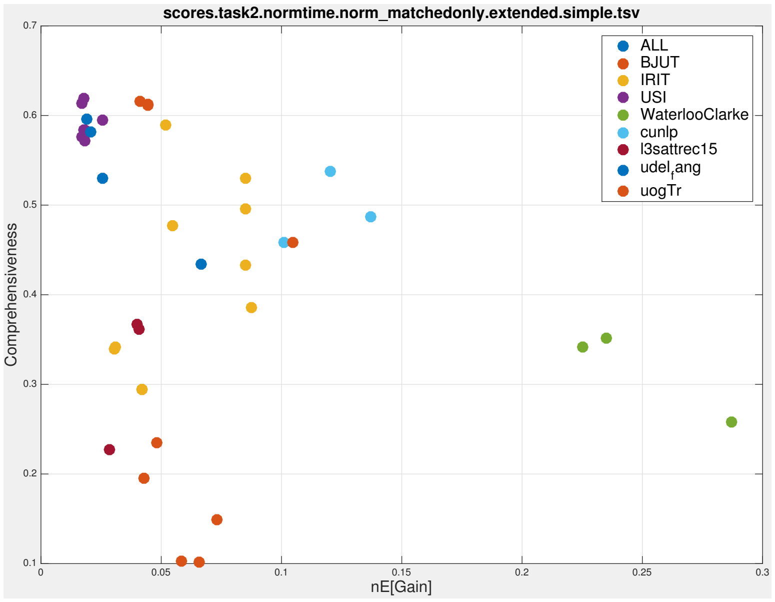


Figure 2: Participant run plot of (normalized) Expected Gain vs. Comprehensiveness.

on average, the effect was not substantial. We plan on developing additional robustness checks in further analysis.

7 Conclusion

In general, the runs submitted to the 2015 track either had a fairly high precision or novelty with topic coverage, but it appears that it was difficult for systems to do both. 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. Because of the similarity in experiment design, we recommend participants continue studies in the TREC 2016 Real-Time Summarization Track.

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.

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.

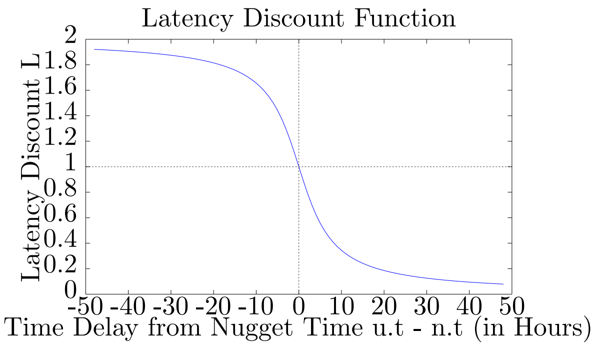
  The relevance can be discounted in time or in size, hence the following discounting functions.

- **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.

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

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



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(± 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.

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

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

  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:

$$n\mathbf{EG}(\mathcal{S}) = \frac{1}{Z|\mathcal{S}|}\sum_{u\in\mathcal{S}}\mathbf{G}(u,\mathcal{S}) \tag{15}$$

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

$$= \frac{1}{Z|\mathcal{S}|}\sum_{\{n\in\mathcal{N}:\mathbf{M}(n,\mathcal{S})\neq\emptyset\}}\mathbf{g}(\mathbf{M}(n,\mathcal{S}),n) \tag{16}$$

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**

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

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:

$$\begin{aligned}
\mathbf{C}(\mathcal{S}) &= \frac{1}{\sum_{n\in\mathcal{N}}\mathbf{R}(n)} \sum_{\{n\in\mathcal{N}:\mathbf{M}(n,\mathcal{S})\neq\emptyset\}} \mathbf{g}(\mathbf{M}(n,\mathcal{S}),n) && (20)\\
&= \frac{1}{\sum_{n\in\mathcal{N}}\mathbf{R}(n)} \sum_{u\in\mathcal{S}} \sum_{n\in\mathbf{M}^{-1}(u,\mathcal{S})} \mathbf{g}(u,n) \\
&= \frac{1}{\sum_{n\in\mathcal{N}}\mathbf{R}(n)} \sum_{u\in\mathcal{S}} \mathbf{G}(u,\mathcal{S}) && (21)
\end{aligned}$$

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.

In order to summarize expected gain and comprehensiveness, we use an F measure as our primary metric based on these values,

$$\mathbf{F}(\mathcal{S}) = \frac{\mathbf{EG}_\mathbf{V}(\mathcal{S}) \times \mathbf{C}(\mathcal{S})}{\mathbf{EG}_\mathbf{V}(\mathcal{S}) + \mathbf{C}(\mathcal{S})} \tag{24}$$

B Judging

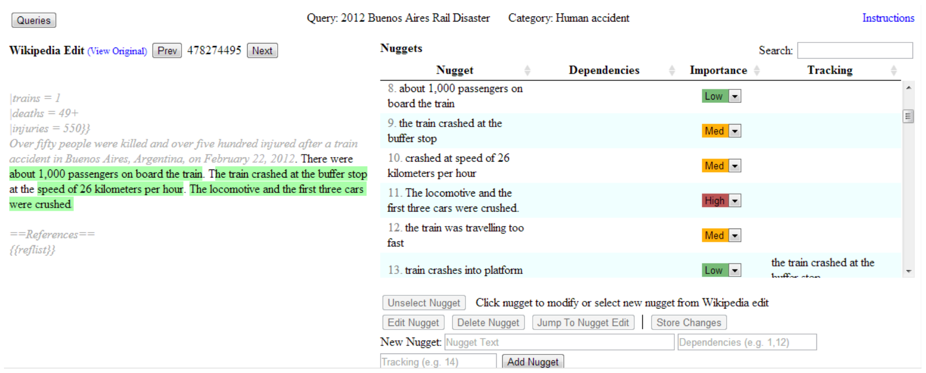
B.1 Gold Nugget Extraction

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. An example portion of the extraction interface can be seen in Figure 3.

In order to simplify later matching, assessors were told to extract nuggets such that they were atomic pieces of information relevant to the query. Unlike in previous years, no dependency extractions were performed, as we found it sufficient in previous years to simply allow splitting of nuggets during the matching phase and to remove the notion of dependencies.

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 approximately 60


Queries

Query: 2012 Buenos Aires Rail Disaster   Category: Human accident

Instructions

Wikipedia Edit (View Original) Prev 478274495 Next

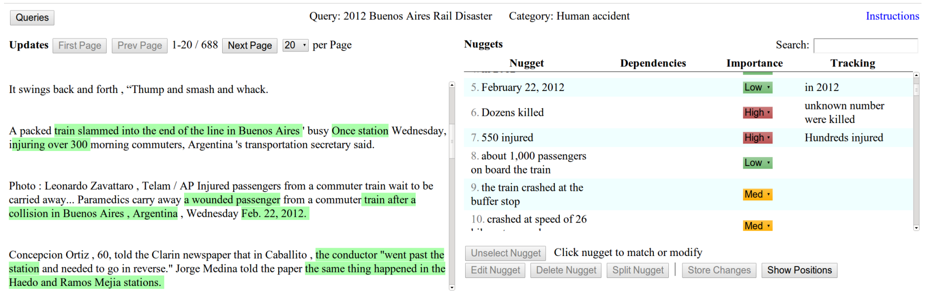
|trains = 1
|deaths = 49+
|injuries = 550}}
Over fifty people were killed and over five hundred injured after a train accident in Buenos Aires, Argentina, on February 22, 2012. There were about 1,000 passengers on board the train. The train crashed at the buffer stop at the speed of 26 kilometers per hour. The locomotive and the first three cars were crushed.

==References==
{{reflist}}

Nuggets   Search:

Nugget	Dependencies	Importance	Tracking
8. about 1,000 passengers on board the train		Low	
9. the train crashed at the buffer stop		Med	
10. crashed at speed of 26 kilometers per hour		Med	
11. The locomotive and the first three cars were crushed.		High	
12. the train was travelling too fast		Med	
13. train crashes into platform		Low	the train crashed at the

Figure 3: Extraction interface used by assessors to extract nuggets from Wikipedia edits.


Queries

Query: 2012 Buenos Aires Rail Disaster   Category: Human accident

Instructions

Updates First Page Prev Page 1-20 / 688 Next Page 20 per Page

It swings back and forth , “Thump and smash and whack.

A packed train slammed into the end of the line in Buenos Aires ' busy Once station Wednesday, injuring over 300 morning commuters, Argentina 's transportation secretary said.

Photo : Leonardo Zavattaro , Telam / AP Injured passengers from a commuter train wait to be carried away... Paramedics carry away a wounded passenger from a commuter train after a collision in Buenos Aires , Argentina , Wednesday Feb. 22, 2012.

Concepcion Ortiz , 60, told the Clarin newspaper that in Caballito , the conductor "went past the station and needed to go in reverse." Jorge Medina told the paper the same thing happened in the Haedo and Ramos Mejia stations.

Nuggets   Search:

Nugget	Dependencies	Importance	Tracking
5. February 22, 2012		Low	in 2012
6. Dozens killed		High	unknown number were killed
7. 550 injured		High	Hundreds injured
8. about 1,000 passengers on board the train		Low	
9. the train crashed at the buffer stop		Med	
10. crashed at speed of 26		Med	

Figure 4: Matching interface used by assessors to match updates and nuggets.

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 1. 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 4.

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 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.