TREC 2014 Temporal Summarization

Draft Guidelines

June 11, 2014 Aslam, Diaz, Ekstrand-Abueg, McCreadie, Pavlu, Sakai

1 Definitions

1.1 Corpus

We will be using 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. A document contains a set of sentences, each with a unique identifier. Participants should get access to the corpus by submitting the appropriate KBA User Agreements. You may use the KBA 2014 'english-and-unknown-language' streamcorpus.

Due to the size and computational cost of processing the full KBA 2014 corpus, we will also be releasing a smaller version that has been pre-filtered to focus on documents that are likely to contain relevant sentences. Participants may use either the full or filtered version of the corpus. We will evaluate the runs using the

filtered corpus separately from those using complete KBA 2014 corpus.

1.1.1 Retrievable Units

Participants should return a list of sentences from the KBA corpus for each event in the format detailed later in Section 3. Each sentence is identified by the combination of a document identifier and a sentence identifier. Documents in the KBA corpus are segmented into sentences in the 'ner' field. A sentence identifier is the index of the sentence in the document, *beginning at zero*. If there are three sentences in a document, the first sentence would be identified as '0', the second would be identified as '1', and the third would be identified as '2'. If there is only one sentence in a document, it would be identified as '0'. **It is your responsibility to make sure your output format is consistent with this indexing.**

1.1.2 Processing Order

Temporal Summarization track aims to investigate the summarisation of events over time in as realistic a setting as possible. As such, participant systems should treat the documents from within the time period of each event as a stream. Therefore, documents should be iterated over in temporal order. Documents should be sorted by `stream_time.zulu_timestamp` (equivalently `stream_time.epoch_ticks`). We encourage participants to use efficient data structures to perform many experiments (i.e. 're-running the simulation'). **However, any data structure should not expose the simulated system to information with a timestamp after the decision time.** This means that participants should be careful about precomputing temporally sensitive global statistics such as IDF, a value which will change as documents are processed.

When emitting a sentence into the summary, a timestamp should be provided indicating when (with respect to the underlying document stream) the sentence was emitted. In the case of a fully real-time system that makes include/exclude decisions on a per-document basis, this timestamp would correspond to timestamp of the document containing each sentence. Alternatively, a system that buffers documents for a period of time before emitting (e.g. emitting at the end of each hour for instance) should produce a timestamp indicating when each batch of sentences were emitted with respect to the underlying document stream.

One alternative is to treat timestamps within the corpus' hour directory as the end of that hour and process all documents in the hour directory together (i.e. without the need for sorting). If your algorithm uses data across hour directories, you still must iterate over the hour directories in time order.

1.1.3 External Resources

Participants are allowed to include runs that use information external to the KBA corpus. We stress 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 are subject to the same requirements. For example, participants should **not** use a statistical model trained on data that existed **after** the event end time.

When submitting, any external data used should be declared. Unlike for the 2013 task, it is not required to submit a 'basic' run without such external data.

1.2 Topics

An event refers to a temporally acute topic and is represented as,

- **Title**: A short retrospective description of the event (string).

```
<event>
  <id>1</id>
  <title>2012 Buenos Aires rail disaster</title>
  <description>http://en.wikipedia.org/wiki/2012_Buenos_Aires_rail_disaster</description>
  <start>1329910380</start>
  <end>1330774380</end>
  <query>buenos aires train crash</query>
  <type>accident</type>
</event>
```

Figure 1: Complete topic definition for '2012 Buenos Aires Rail Disaster'.

```
<event>
  <id>1</id>
  <start>1329910380</start>
  <end>1330774380</end>
  <query>buenos aires train crash</query>
  <type>accident</type>
</event>
```

Figure 2: Masked topic definition for '2012 Buenos Aires Rail Disaster'.

- **Description**: A retrospective free text event description (url).
- **Start**: Time when the system should start summarization (UNIX timestamp in GMT).
- **End**: Time when the system should end summarization (UNIX timestamp in GMT).
- **Query**: A keyword representation of the event description expressed by a user during the event (string).
- **Type**: The type of event (one of {accident, bombing, earthquake, hostage, impact event, protest, riot, shooting, storm}).

We present an example topic in Figure 1. We will provide participants with a set of masked event topics containing only the topic id, query, start, and end (Figure 2).

2 Task Definition

During the simulation, a system should emit relevant and novel sentences to an event (exact metrics will be released in a separate document). Conceptually, a simulator should be structured as in Figure 3. The arguments to the simulator are the participant summarization system, the time-ordered corpus, the keyword query, and the relevant time range. In line 1, we initialize the output summary to empty. In line 2, we initialize the sequential update summarization system with the event query. The system should store some representation of this query for later processing and filtering. We iterate over the corpus in temporal order (line 3), processing each document in sequence (line 5). If the document we are processing is in the event timeframe (line 7), then we check to see if adding the document resulted in the system deciding to output a set of summary sentence ids (line 9). We then add these sentence ids to the summary timestamped with the time of the decision (lines 10-12).

We have tried to present an abstract representation of sequential update summarization. There are several comments worth making. First, if a participant is interested in efficiency and does not anticipate needing documents outside of the event timeframe, then the call to **S**.PROCESS(d) can be moved inside of the condition in line 7. Participants should be clear about any filtering of $\mathcal{C}$. For example, a participant should note if they are just iterating over documents with a high BM25 score. However, if this is done, care must be taken to make sure that filtering out a document d does not exploit information from sources after d.TIME() (e.g. retrospective IDF values).

3 Result Format

We expect team result formats to be in the following tab-separated file format,

```
1 HelloWorldUniversity  Cluster1  1357052200-54f6f6e096a4cae27bee55dc2e0dc2b6 0 1330432283 0.90
1 HelloWorldUniversity  Cluster1  1357052200-54f6f6e096a4cae27bee55dc2e0dc2b6 1 1330472283 0.75
1 HelloWorldUniversity  Cluster1  1357052190-8f7a8b30a9a8f671dcc979677b04fab4 1 1330482283 0.99
```

where the columns are defined as,

1. query identifier
2. team identifier
3. run identifier
4. document identifier
5. sentence identifier: base 0 index of a segmented sentence in the KBA corpus
6. decision timestamp
7. confidence value: a strictly positive number (> 0) which encodes the system's confidence in this being a reasonable update; this value may be used to prioritize updates if we cannot judge all of them.

```
TemporalSummarization($\mathbf{S}, \mathcal{C}, q, t_s, t_e$)
    $\mathbf{S}$      ▷ Participant system.
    $\mathcal{C}$      ▷ Time-ordered corpus.
    $q$      ▷ Event keyword query.
    $t_s$     ▷ Event start time.
    $t_e$     ▷ Event end time.
    $\mathcal{U} \leftarrow \{\}$
    $\mathbf{S}$.Initialize($q$)
    for $d \in \mathcal{C}$
        do
           $\mathbf{S}$.Process($d$)
           $t \leftarrow d$.Time()
           if $t \in [t_s, t_e]$
             then
                 $\mathcal{U}_t \leftarrow \mathbf{S}$.Decide()
                 for $u \in \mathcal{U}_t$
                     do
                        $\mathcal{U}$.Append($u, t$)
    return $\mathcal{U}$
```

Figure 3: Sequential update summarization simulator.

4 Important Dates

June 13, 2014	train and test events released
June 13, 2014	metrics released
June 25, 2014	filtered corpus released
late July 2014	submission website open
September 3, 2014	submission website closed