Automatic Ground Truth Expansion for Timeline Evaluation

Richard McCreadie, Craig Macdonald and Iadh Ounis
University of Glasgow, Scotland, UK
(firstname.lastname)@glasgow.ac.uk

ABSTRACT

The development of automatic systems that can produce timeline summaries by filtering high-volume streams of text documents, retaining only those that are relevant to a particular information need (e.g. topic or event), remains a very challenging task. To advance the field of automatic timeline generation, robust and reproducible evaluation methodologies are needed. To this end, several evaluation metrics and labeling methodologies have recently been developed - focusing on information nugget or cluster-based ground truth representations, respectively. These methodologies rely on human assessors manually mapping timeline items (e.g. tweets) to an explicit representation of what information a 'good' summary should contain. However, while these evaluation methodologies produce reusable ground truth labels, prior works have reported cases where such labels fail to accurately estimate the performance of new timeline generation systems due to label incompleteness. In this paper, we first quantify the extent to which timeline summary ground truth labels fail to generalize to new summarization systems, then we propose and evaluate new automatic solutions to this issue. In particular, using a depooling methodology over 21 systems and across three high-volume datasets, we quantify the degree of system ranking error caused by excluding those systems when labeling. We show that when considering lower-effectiveness systems, the test collections are robust (the likelihood of systems being miss-ranked is low). However, we show that the risk of systems being miss-ranked increases as the effectiveness of systems held-out from the pool increases. To reduce the risk of miss-ranking systems, we also propose two different automatic ground truth label expansion techniques. Our results show that our proposed expansion techniques can be effective for increasing the robustness of the TREC-TS test collections, markedly reducing the number of miss-rankings by up to 50% on average among the scenarios tested.

ACM Reference Format:
Richard McCreadie, Craig Macdonald and Iadh Ounis. 2018. Automatic Ground Truth Expansion for Timeline Evaluation. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210034

1 INTRODUCTION

With the increasing usage of social media platforms and online reporting channels, information is produced and disseminated online faster and in larger volumes than ever before. As a result, users expect to have easy access to up-to-date information about topics of interest, resulting in a large number of new real-time information-seeking scenarios. These scenarios require solutions that can identify relevant (topical), non-redundant (avoids repeated information), and timely (up-to-date) content from noisy high-volume text streams. A common class of solutions that require these characteristics are event timeline/real-time summary generation systems. Such systems take as input a topic of interest and a large volume of textual items (e.g. news articles or tweets), most of which are non-relevant and/or redundant, and select a subset of those items to be emitted over time into a timeline or an updating summary [12, 19, 26]. An example extract from the output of such systems is shown in Table 1.

Table 1: Example timeline summary extract with nuggets.

Timestamp	Update Text	Information Units
01/14/2012, 5:02pm	Carrying 3,206 passengers and 1,023 crew members, the Costa Concordia was on its usual route across the Mediterranean Sea and departed Civitavecchia - three hours before disaster struck.	Crew and Passenger count, Ship route, Time of departure
01/14/2012, 9:38pm	As the Costa Concordia keeps shifting on its rocky ledge, many have raised the prospect of a possible environmental disaster if the 2,300 tons of fuel on the half-submerged cruise ship leaks into the sea.	Fuel oil environmental hazard
01/15/2012, 5:17pm	The Costa Concordia death toll has risen by two - as all British passengers and crew were confirmed to have survived. Two French nationals and a Peruvian died after the cruiser ran aground near the island of Giglio off the Tuscan coast on Friday night.	People killed increased by 2. Location of event

This work is concerned with how to effectively and efficiently evaluate the quality of timeline items produced by such systems. Over the last few years new methodologies to evaluate the quality of timelines have been proposed [4, 17]. These methodologies typically use human annotators to manually identify atomic units of information that form a ground truth representing the information a 'good quality' summary about a topic should contain (see Figure 1). Next, textual items (e.g. sentences or tweets) returned by a diverse set of timeline generation systems for the topic are pooled. Finally, the pooled text items are manually checked to see what atomic information units for the topic they cover (if any), forming <text item,information unit> pairs. Metrics such as Expected Latency Gain [12] use the resultant pairs to estimate the degree to which individual text items included in a timeline (and hence the timeline as a whole) contains relevant, non-redundant, and timely information.

The use of atomic information units as a ground truth for evaluating timelines/real-time summaries is generally accepted and has been successfully deployed within the Temporal Summarization and Real-time Summarization tracks at the Text Retrieval Conference (TREC) [4, 17]. However, while these tracks produced test collections that can in theory be used to evaluate any timeline generation system, prior works have reported cases where these test collections fail to accurately estimate the performance of *new* timeline generation systems [19, 20]. In particular, it was observed in these past works that the overlap between items included in the initial pools (i.e. the assessed set) and those returned by their new proposed systems was insufficient to facilitate a robust comparison

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

of systems. As a result, it is unclear to what extent the test collections produced during these tracks can be used to evaluate the quality of new systems that were not included in the initial pooling [5]. In this paper, we investigate to what extent current atomic information unit-based test collections are able to distinguish between timeline summarization systems with different effectiveness levels, as well as propose and evaluate automatic solutions to reduce the likelihood of errors occurring when evaluating such systems.

Contributions. The main contribution of our work is an in-depth analysis of the TREC 2013-2015 Temporal Summarization track test collections that quantifies how robust these collections are when evaluating unpooled systems, as well as an effectiveness evaluation of different automatic <text item,information unit> expansion techniques aimed at increasing the robustness of these test collections. Specifically, we tackle two main research questions:

- **RQ1:** To what extent can the TREC Temporal Summarization track test collections accurately rank unpooled systems?
- **RQ2:** If we use automatic methods to generate additional <text item,information unit> pairs can we reduce the likelihood of new systems being miss-ranked?

Our results show that the TREC 2013-2015 Temporal Summarization track test collections do not accurately estimate the effectiveness of unpooled systems. Moreover, the discrepancies observed between actual and estimated performances are sufficient to cause errors when ranking those systems. Furthermore, we found that the likelihood of encountering ranking errors is not uniform across system effectiveness levels – the better a system is, the more negatively it is impacted by not being pooled. For this reason, we conclude that it is potentially risky to use the TREC-TS test collections out-of-the-box. We then experiment with two types of automatic <text item,information unit> expansion techniques aimed at reducing these discrepancies, namely: item-item similarity expansion and item-item semantic expansion. Our experiments using these two expansion techniques show that automatically adding even a small number of <text item,information unit> pairs can markedly reduce the number of ranking errors observed when using the text collections. In particular, we found that item-item similarity expansion can reduce the number of ranking errors by up to 30% while item-item semantic expansion can reduce the number of ranking errors by up to 50%. We conclude that these expansion techniques improve the robustness of the TREC-TS test collections, reducing the risk of miss-ranking new systems that were not pooled.

2 BACKGROUND AND RELATED WORK

2.1 Classical Summarization Evaluation

In the summarization domain, a range of evaluation methodologies have previously been proposed and examined in the literature. Early works focused on estimating the quality of fixed-length textual summaries produced by either single-document or multi-document summarization systems [22]. This is a type of textual comparative evaluation, where a summary produced by an automatic system is compared against one or more gold standard summaries authored by humans. The idea underpinning this type of evaluation is that good summaries will be textually similar to the gold-standard summaries. To perform the similarity comparison, the ROUGE [16] suite of metrics have become the defacto standard and were used extensively as part of the Document Understanding Conference (DUC) [9] and Text Analysis Conference (TAC) [10] evaluations.

2.2 Timeline Summaries and Evaluation

Comparative evaluation approaches were used for many years to evaluate multi-document summarization systems [1], however the shift toward real-time information sharing and the associated development of timeline generation and real-time summarization solutions [19, 26, 30, 33] that push updates to users over an extended period of time required new evaluation methodologies. A timeline summary can be defined as a number of (approximately) sentence-length timestamped text items. These text items might be sentences extracted from news articles [4] or tweets [17]. A timeline summary is usually about a topic or event, and hence the text items it contains should be relevant to that topic or event. A timeline is normally visualized as a list of text items in chronological or reverse-chronological order. New text items may be added to the timeline over time, as new information emerges and is found by the summarization system. Classical comparative evaluation approaches that use metrics like ROUGE [16] and its temporal extensions [8, 12] make the assumption that both the summary to be evaluated and the gold-standard summaries are of (roughly) equal length, and that the gold-standard summaries do not change over time. As such, these classical comparative summary evaluation approaches are unsuitable to evaluate timelines.

To solve this issue, *atomic information units* were introduced as an alternative means to evaluate the quality of a timeline summary [12]. Atomic information units had been used in a wide range of domains prior to their application to timeline evaluation such as Web search diversification [25] and question answering [29], although the terminology used to describe them changes depending on the domain they are applied to. Indeed, atomic information units are equivalent to as sub-topics, aspects, facets, clusters or nuggets [12, 19, 23, 26, 32]. The core concept behind atomic information unit-based evaluations is that all of the units that contribute to the evaluation score for a system should be explicitly defined. In this way, evaluation can be reduced to counting the proportion of all units covered by a system. The more units covered (typically within some range constraint such as the top k documents), the better that system is. This concept maps naturally into a summarization context, where each 'unit' represents a piece of information that 'good' timeline summary for a topic should contain.

In practice, for evaluating timeline summaries, atomic information units have been implemented in two different manners. First in the form of information nuggets within the TREC Temporal Summarization track during 2013 to 2015. Second as information clusters within the TREC Real-time Summarization track during 2016 and 2017. We choose to use the TREC Temporal Summarization implementation as the basis for the study in this paper as it is the more complex/costly to deploy of the two [5]. We discuss this implementaton below. For those interested in differences between the two tracks we recommend the comparison by Baruah et al. [5].

2.3 TREC Temporal Summarization Track

In 2013 the Text Retrieval Conference (TREC) introduced the Temporal Summarization (TREC-TS) track that examined how to extract sentences from high volume streams of news and social content

to return to the user as updates for large events [4]. TREC-TS is a timeline generation task, as defined above, where each topic is an event (represented by an event query, e.g 'costa concordia disaster'), the text items are sentences extracted from a stream containing news articles, blogs and other Web documents. To avoid differences in what might be considered a 'sentence', each document in the stream was pre-segmented. For a set of events, TREC-TS systems processed the high volume stream of sentences and emitted a subset of those sentences into a timeline summary.

For evaluation, TREC-TS adopted an atomic information unit-based evaluation methodology, where an information unit was referred to as a 'nugget'. This methodology was inspired by earlier work developed for question answering [29] and applied in a series of evaluations in the early 2000s. Nuggets in the TREC-TS context represented atomic facts relevant to an event, represented by short natural language phrases. For example, for the event 'Costa Concordia shipping disaster', the ground truth might contain nuggets such as 'occurred on Friday 13th January 2012', 'ran aground on a reef' and 'the hull was punctured'. Under this evaluation methodology a perfect summary is one that covers all of the information nuggets for an event while being as short (contains as few sentences) as possible. Summaries containing redundant (repeated) information are penalized and were also evaluated in terms of timeliness (was the information relevant at the time it was retrieved?).

As a TREC track dedicated to supporting standardized evaluation, TREC-TS produced three test collections for evaluating timeline generation systems, one for each of the years that the track ran (2013, 2014 and 2015). These test collections each contain a number of topics (events), a high-volume stream of sentences for each topic, and a ground truth label set comprised of the information nuggets along with a <text item,information unit> (i.e. <sentence,nugget>) mapping that describes what sentences contain the information represented by each nugget. Creating the ground truth label set for each test collection was a three-step process [2, 3]:

(1) **Nugget Extraction ('nuggetization')**: Human assessors manually defined the information nuggets for each topic. This was achieved by having TREC assessors read the edit stream from the Wikipedia page for each topic (the page describing the event). The assessors defined new information nuggets as they encountered novel information about the event that they considered important enough to be included in a "good" summary about the topic.
(2) **Sentence Pooling**: Each participating system submitted a timeline summary comprised of sentences for each topic (event). The systems assign each sentence a priority score indicating how confident they are that those sentences are of high-quality. The top-k sentences by priority score were then selected and added into a pool to be assessed.
(3) **Nugget Matching**: Given the ground truth nuggets extracted from Wikipedia, assessors then manually checked each sentence in the pool, recording whether those sentences contained any of the information represented by the nuggets. A sentence that contains a nugget's information is referred to as 'covering' that nugget. The result of this is a set of <sentence,nugget> pairs, specifying which sentences contain the information represented by each nugget. It is worth noting that nuggets represent concepts, hence the matching process often requires the assessor to do more than match the text of a concept to the text of a sentence, e.g. accounting for synonyms.

Table 2: Statistics of the TREC Temporal Summarization test collections from 2013, 2014 and 2015.

	Year		
Statistic	2013	2014	2015
Number of Events	9	15	21
Number of Nuggets	1,168	1,394	996
Number of Matches	5,071	13,635	24,823
Number of Updates	10,377	14,652	33,483

The statistics of the resultant TREC-TS test collections for each year are provided in Table 2. Both the nugget extraction and nugget matching steps involved significant human effort (by NIST assessors). According to a study by Baruah et al. [5], the total assessment time spent to create the 2013 and 2014 TREC-TS test collections was around 375 hours, where over 80% of that time was spent on nugget matching.

2.4 Questions on TREC-TS Robustness

The test collections produced by TREC-TS have been used for a range of research papers since their original release [5, 13, 14, 19, 20]. However, a number of these works reported needing to add more nuggets/matches to the provided ground-truth sets to make the test collections usable. In particular, McCreadie et al. [19] reported in their paper that there was very low overlap between the sentences included in the TREC-TS pool and the top sentences selected by their system, i.e. assessment completeness [6] was low. To tackle this, they performed additional pooling and matching based on the TREC-TS guidelines, adding 22,424 sentences to the pool at a significant cost. This was then echoed in their later study [20] where they found almost no overlap between their diversification-focused system and the TREC-TS pool (see Annex A from [20] for details), again requiring the pooling and assessment of the new summaries. On the other hand, Ekstrand-Abueg et al. [11] performed a correlation study examining whether removing individual systems from the pool adversely affected the system ranking under the official track metrics. They reported high correlations between system rankings pre and post pooling, indicating that the test collections are reusable. However, they also noted that there were outlier systems that were severely affected (i.e. were miss-ranked) when they were removed from the pool.

These prior studies lead us to question to what extent the TREC-TS test collections are in fact robust when evaluating unpooled systems. The studies reported in [20] and [19] required significant additional pooling and assessment effort before the collections could be used. Having to perform reassessment for each new system or summary to be evaluated reduces the value that these test collections bring to IR evaluation. Hence, in this paper, we quantify how robust these collections are for evaluating unpooled systems and also propose and evaluate automatic techniques aimed at increasing the robustness of these test collections.

2.5 Incompleteness of Relevance Judgments

Apart from the initial examination by Ekstrand-Abueg et al. [11], the robustness of timeline generation test collections have not been

explored in the literature. However, there have been a number of past works in the wider information retrieval domain (typically for search tasks) examining the effect that relevance assessments (or lack thereof) has on test collection robustness. For instance, early work by Voorhees [28] investigated how different relevance assessment sets for a test collection impacted on the evaluation of retrieval results for the TREC-4 and TREC-6 test collections. That study showed that while the effectiveness metrics were impacted by using assessments created by different groups (e.g. NIST assessors vs. Waterloo assessors), the resultant ranking of the retrieval runs (systems) were highly correlated. Meanwhile, Zobel [34], examined the fairness of top k pooling methods for selecting documents to assess, showing that a pooling depth of 100 appeared to be adequate for search over the TREC-5 test collection. These early studies support the idea that smaller collections are indeed robust in the face of incomplete assessments.

However, over time, the size of test collections used by evaluation campaigns like TREC grew, but the pool depth (the number of judged documents per topic) across years has remained constant, increasing the relative degree of incompleteness (e.g. due to the varied nature of documents retrieved by systems contributing to the pools for these large corpora). Hence, later studies such as that by Buckley and Voorhees [6] examined the effect that further relaxing the completeness assumption has on the Cranfield evaluation methodology in larger test collections. In contrast, they showed that the Cranfield methodology was not robust in the face of massively incomplete relevance judgments. Moreover, works such as [24] have also questioned how robust different IR metrics are when using pooling at different k values. Parallels can be drawn between these works in the search domain and the questions investigated in this paper. The TREC-TS test collections are built on a corpus containing over a billion items (sentences).[1] However, only between 60 and 100 (depending on year [3, 4]) of the top k sentences were pooled from participating systems. Hence, assessment completeness is a valid concern when working with collections at this scale.

3 METHODOLOGY AND SETUP

To examine the extent that the TREC-TS test collections are robust, we need a standardized setting and evaluation methodology with which to quantify 'robustness'. To create such a setting, we first define what we mean by 'robustness' below:

Robustness: In a timeline generation context, a truly robust test collection is one which can be used to accurately estimate the quality of a timeline summary, regardless of whether that summary was included within the initial pool or not. A robust test collection should correctly rank different timeline generation systems in order of the quality of the timeline summaries they produce.

Given this definition, to evaluate the robustness of a test collection we can identify three main requirements: 1) a series of systems that produce summaries of known quality (such that we have a known ordering of systems); 2) an evaluation metric that reflects the quality of a system according to the test collection; and 3) we need to have the ability to compare systems when included in the pool and when excluded from the pool. Below we discuss how we design our experimental setup to meet these three requirements.

[1] http://trec-kba.org/kba-stream-corpus-2014.shtml

3.1 Synthetic System Generation

The first requirement for our evaluation is to have a series of timeline generation systems that can produce timeline summaries of known quality. This is so that we have a gold standard ranking of systems that reflects their actual performance. Initially, one might consider using the systems originally submitted to the TREC track in each year. However, this has some notable limitations. First, the systems that participated in TREC are different from year-to-year and the sources for those systems are not always available, hence we cannot deploy each TREC system across all years. This is potentially problematic, as there are relatively few topics ('events') in each test collection (between 9 and 21, see Table 2), which is less than the recommended number of topics for an IR experiment [7]. Second, the TREC systems only represent a subset of the range of possible system performances, e.g. in the first year, all participating systems were rather poor in terms of effectiveness. It would be preferable to be able to deploy single set of systems across all years such that we can compare across a larger number of events and have those systems represent the full range of system effectiveness (poor to perfect).

To achieve this, we instead take an alternative approach inspired by prior work in the Web and expert search domains [18, 27], where we generate synthetic systems with known performances. This is possible, since we are using the TREC-TS test collections as the subject of our investigations, which have sentence-level labels that quantify how much value is added by any sentence. Hence, we can define a synthetic system that takes in the sentence-level labels along with a target effectiveness level, and generates summaries with (approximately) that effectiveness level for each topic.

In particular, as discussed in Section 2.3, the TREC test collections contain atomic information items (nuggets) that form a ground truth for measuring summary quality. More precisely, the test collections contain <sentence,nugget> pairs that specify which individual sentences contain the information represented by each nugget. Following the atomic information nugget evaluation paradigm, a simple way to represent the quality of a timeline summary with k sentences is to calculate the proportion of nuggets it covers. As long as all summaries are of the same length k for a topic, then nugget coverage is a fair representation of timeline summary quality (it measures the volume of information contained).

Given the above, we specify a series of target effectiveness levels in terms of nugget coverage from 95% to 5% in 5% increments. For each target effectiveness level, we generate one summary per topic within the three TREC-TS test collections. For a topic, we first randomly select a subset of the information nuggets that matches the target effectiveness level, e.g. for 60% coverage, we select 60% of the nuggets for that topic. We then iterate over all <sentence,nugget> pairs for that topic in the ground truth label set, selecting one sentence that matches each nugget in a greedy manner. For instance, if a topic contained 100 nuggets and our target effectiveness level was 40%, we would first randomly select 40 of those 100 nuggets, and then attempt to select one sentence matching each of those 40 nuggets. When considering real timeline summarization systems, not all sentences are equally likely to be selected (some are easier to find than others, e.g. because they contain the event query terms) and most nuggets have multiple sentences we might select. To capture this, instead of selecting any of the available sentences that

Table 3: Synthetic Run Statistics.

Synthetic System	Target Coverage	Actual Coverage	TREC-TS Metrics		
			ELG	*LC*	$\mathcal{H}$*(ELG,LC)*
Synth-C95	95%	72%	0.3590	0.6989	0.4589
Synth-C90	90%	68%	0.3337	0.6474	0.4249
Synth-C85	85%	64%	0.3336	0.6277	0.4216
Synth-C80	80%	61%	0.3404	0.5901	0.4165
Synth-C75	75%	57%	0.3241	0.5676	0.3996
Synth-C70	70%	53%	0.3202	0.5289	0.3834
Synth-C65	65%	49%	0.3189	0.5105	0.3792
Synth-C60	60%	46%	0.3057	0.4466	0.3488
Synth-C55	55%	41%	0.2922	0.4233	0.3314
Synth-C50	50%	38%	0.2997	0.4181	0.3371
Synth-C45	45%	34%	0.2888	0.3596	0.3047
Synth-C40	40%	30%	0.2919	0.3398	0.3002
Synth-C35	35%	25%	0.2989	0.2961	0.2824
Synth-C30	30%	22%	0.2737	0.2515	0.2501
Synth-C25	25%	18%	0.2555	0.2012	0.2108
Synth-C20	20%	14%	0.2785	0.1523	0.1869
Synth-C15	15%	10%	0.3053	0.1199	0.1623
Synth-C10	10%	7%	0.3122	0.0755	0.1121
Synth-C05	5%	3%	0.2478	0.0274	0.0473

match a nugget randomly, we instead use a probabilistic selection of sentences, based on the likelihood of each sentence having been selected by the original TREC systems (the more TREC systems that selected a sentence the more likely our synthetic systems will similarly select that sentence). As a sentence may cover multiple nuggets, we exclude a sentence from being selected if it covers any nuggets not in our target set. Furthermore, only around 70% of nuggets have associated matching sentences, i.e. in the remaining cases no systems in the pool found sentences that covered that nugget. In all cases we select as many sentences as possible and then 'fill' the remaining slots (to maintain a consistent length k) with redundant sentences. In practice, this means that the actual nugget coverage for a synthetic summary is lower than the target coverage, e.g. a 90% coverage target results in 68% actual coverage when averaged across topics. We summarize the statistics of our generated synthetic runs in Table 3. As can be observed from Table 3, this synthetic system generation approach produces a range of systems that span the range of effectiveness levels attainable in terms of nugget coverage.

3.2 Evaluation Metrics

Having produced a set of systems with known performances, we now need to define metrics to capture how effective the summaries produced by those systems are. One possible option would be to simply use nugget coverage averaged across topics as an estimation of summary quality, as we did in the previous section. However, the TREC-TS track also considered factors beyond nugget coverage, such as novelty, brevity and latency [12]. For this reason, as well as to maintain compatibility with the track, we use the official TREC-TS target evaluation metric, which itself is the harmonic mean between two metrics: Expected Latency Gain and Latency Comprehensiveness. Expected Latency Gain (ELG) is a precision-like metric, calculated as the sum of the relevance of each nugget that a sentence covered, computed as:

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

where $\mathcal{S}$ is the stream of sentences returned by the system, $\mathbf{M}(u)$ is the set of gold standard nuggets matching sentence u (as determined by an assessor) and $\mathbf{g}(u, n)$ measures the utility of matching sentence u with nugget n. Latency Comprehensiveness (LC) is the proportion of all nuggets matched by the system updates, computed as:

$$LC(\mathcal{S}) = \frac{1}{|\mathcal{N}|} \sum_{u \in \mathcal{S}} \sum_{n \in \mathbf{M}(u)} \mathbf{g}(u, n) \tag{2}$$

where $\mathcal{N}$ is the set of nuggets for the current event. For both ELG and LC, the $\mathbf{g}(u, n)$ component contains built-in penalties to capture sentence brevity and latency. We refer the reader to the TREC track metrics documentation[2] for a detailed explanation on how these are calculated. To provide a target metric, an F-like measure was also defined, which we denote $\mathcal{H}$*(ELG,LC)*. This is the harmonic mean of ELG and LC,

$$\mathcal{H}(ELG, LC)(\mathcal{S}) = 2 * \frac{ELG(\mathcal{S}) * LC(\mathcal{S})}{ELG(\mathcal{S}) + LC(\mathcal{S})} \tag{3}$$

We report the performance of our synthetic systems under the TREC-TS Metrics *ELG*, *LC* and $\mathcal{H}$*(ELG,LC)* in Table 3. As we can see, the performance as reported by the TREC-TS *LC* and $\mathcal{H}$*(ELG,LC)* and metrics are highly correlated with the actual nugget coverage of the systems.

3.3 Depooling Methodology

Finally, to evaluate the robustness of the test collections, we need to be able to evaluate the difference in performance of systems when they are included within the pool and when excluded from it. The core idea is that if a test collection is robust, then the estimated performance (under $\mathcal{H}$*(ELG,LC)*) of a pooled system with known coverage X should be similar to the estimated performance for that same system when it is not pooled. In this case, an unpooled system represents a hypothetical new system that did not participate in the original TREC track and hence was not pooled.

TREC-TS followed a top k pooling methodology, where the sentences with the k highest confidence scores were added to the pool and later assessed (i.e. they took part in the nugget matching phase resulting in the <sentence,nugget> pairs $\mathbf{M}(u)$). From the TREC-TS pool statistics, we know the number of the original TREC-TS participating systems that contributed each sentence. We refer to sentences contributed by multiple systems as *common sentences* and sentences that were only contributed by a single system as *uncommon sentences*.

Building on past work examining the effect of unpooled systems on IR system performance [11], we simulate the state of the TREC-TS test collections in scenarios where a particular system was not pooled. For ease of reference, we refer to this as *depooling*. depooling involves removing one copy of each of the top k sentences contributed by that system from the pool, along with any associated <sentence,nugget> pairs that resulted from the subsequent nugget matching phase. By definition, common sentences would not be affected by removing only a single system (that system's sentences would still be contributed by some other system). However,

[2] http://www.trec-ts.org/metrics-10242013.pdf

Table 4: Synthetic Run Performances under $\mathcal{H}$(ELG,LC) when pooled or depooled. ▼ indicates statistically significant decreases in estimated performance (t-test p<0.01) between the run when in the pooled and when depooled.

Synthetic System	$\mathcal{H}$(ELG,LC) When Pooled	$\mathcal{H}$(ELG,LC) When Depooled
Synth-C90	0.4249	0.3850▼
Synth-C80	0.4165	0.3823▼
Synth-C70	0.3834	0.3480▼
Synth-C60	0.3488	0.3196▼
Synth-C50	0.3371	0.3105▼
Synth-C40	0.3002	0.2617▼
Synth-C30	0.2501	0.2259▼
Synth-C20	0.1869	0.1627▼
Synth-C10	0.1121	0.0687▼

uncommon sentences would be at risk from being eliminated from the pool entirely. If a sentence is eliminated from the pool, then that loss will impact the scoring of all systems. As we used the sentences in the TREC-TS pool previously to produce our synthetic systems, those systems behave as though they have been pooled. Hence, by depooling one of the synthetic systems we can investigate whether its estimated performance would have been adversely impacted had it not been pooled (i.e. due to uncommon sentences not being assessed). As such, we create an evaluation scenario for each of our 19 systems, each representing the case where that system was depooled. In the next section, we use these depooling scenarios to answer our first research question, i.e. RQ1 'To what extent can the TREC Temporal Summarization track test collections accurately rank unpooled systems?'.

4 RQ1: TO WHAT EXTENT ARE THE TREC-TS TEST COLLECTIONS ROBUST?

To answer RQ1, we first examine whether the estimated performance scores for systems change when pooled and when depooled. The ideal outcome is that the scores would not change, however this would only occur in cases where the system being depooled was totally comprised of common sentences. Hence, we can expect some score variance due to uncommon sentences being eliminated from the pool, but we would hope such score variance is minimal. Table 4 reports the estimated performance of the synthetic systems under $\mathcal{H}$(ELG,LC) in the pooled and depooled scenarios (for brevity we only list performances for half the systems, the observations are the same for the other systems). As we can observe from Table 4, in all scenarios, depooling a synthetic system causes a statistically significant decrease in its estimated performance under the official TREC metric ($\mathcal{H}$(ELG,LC)). This is a first indication that the test collections may not be as robust as we would like, as the effectiveness scores estimated for a system is shown to vary greatly depending on whether it was included in the pool or not. Hence, it is likely that *new* systems that were not originally pooled will have their true performance underestimated.

On the other hand, some error when estimating the performance of depooled systems is to be expected, as this is a known issue with pooling-based evaluation scenarios [6, 11]. From an evaluation perspective, what researchers and developers care about is whether the test collection is able to distinguish between systems with different effectiveness levels, i.e. whether we get the ordering of systems correct (particularly in the top ranks) is more important than whether individual system scores are underestimated [5]. Indeed, while we might expect that underestimations of a system's performance will cause that system to be miss-ranked, evidence from the search domain indicates that IR metrics tend to have some degree of robustness against incompleteness effects [24], i.e. the error in the score for a system may not be sufficient to cause a ranking swap.

Table 5: Effect of depooling a single system in terms of ranking stability.

		Rankings Pooled Vs. Depooled	
Synthetic System	# Rank Swaps	Kendall's τ	τ_{AP}
Synth-C95	3	0.9649	0.8129
Synth-C90	2	0.9766	0.8752
Synth-C85	6	0.9298	0.8771
Synth-C80	0	1.0000	1.0000
Synth-C75	4	0.9532	0.9081
Synth-C70	1	0.9883	0.9914
Synth-C65	3	0.9649	0.9579
Synth-C60	1	0.9883	0.9915
Synth-C55	4	0.9532	0.9614
Synth-C50	1	0.9883	0.9914
Synth-C45	2	0.9766	0.9864
Synth-C40	1	0.9883	0.9870
Synth-C35	1	0.9883	0.9946
Synth-C30	0	1.0000	1.0000
Synth-C25	1	0.9883	0.9006
Synth-C20	0	1.0000	1.0000
Synth-C15	0	1.0000	1.0000
Synth-C10	0	1.0000	1.0000
Synth-C05	0	1.0000	1.0000
Average	1.5790	0.9815	0.9597

As our synthetic systems have known effectiveness levels (based on nugget coverage), we know the correct ordering of systems. We also showed previously in Table 3 that the official TREC-TS metric ($\mathcal{H}$(ELG,LC)) reflects this correct ranking when all systems are pooled. However, given that we know that depooling a system causes statistically significant changes in $\mathcal{H}$(ELG,LC), it is possible that these changes are severe enough to result in that system (and other systems) being miss-ranked. In Table 5, we report the effect that depooling each system individually has on the overall ranking of synthetic systems in terms of number of rank swaps (miss-rankings) and overall rank correlation under Kendall's τ. Additionally, as we are often more interested in distinguishing between systems near the top of the ranking than those at the bottom, we also report τ_{AP} [31] values, which place higher weight on rank correlations occurring in the top ranks. Ideally, we should preserve the original correct ranking, i.e. the number of rank swaps should be 0, while Kendall's τ and τ_{AP} would be 1. From Table 5, we can see that for the majority of scenarios, depooling a single system results in the miss-ranking of at least one pair of systems. Indeed, the average number of system swaps needed to restore the correct ranking across depooling scenarios is 1.579, while the rank correlation on average was around 0.9815. This result is similar to that reported by Ekstrand-Abueg et al. [11], who observed correlation values around 0.97 when holding out individual TREC-TS systems in their study. However, while these rank correlations appear high, it is important to remember that systems are still being ranked incorrectly. Moreover, from Table 5, we see that rank swaps are

more common when highly effective systems are depooled, i.e. if you have a new (unpooled) system that is very effective, it is likely to be miss-ranked (see the top of Table 5). On the other hand, it appears that systems at the lower end of the effectiveness scale have little impact on the overall ranking of systems when not pooled.

To answer RQ1, we conclude that the TREC-TS test collections are likely robust when evaluating systems that were not pooled at the lower end of the effectiveness scale, i.e. systems equivalent to or worse than Synth-C30, that has an actual nugget coverage level of 22%. On the other hand, systems that were not pooled that push the upper-end of the effectiveness envelope are more likely to be miss-ranked, and hence using the TREC-TS test collections out-of-the-box is subject to more risk. The issue is that a researcher or developer has no way of knowing which case they fall into. As such, it would be advantageous to improve the test collections to reduce the risk of ranking error for unpooled/depooled systems.

5 RQ2: CAN WE USE AUTOMATIC MATCHING TO INCREASE ROBUSTNESS?

In the previous section we observed that systems that are depooled (i.e. representing new systems that did not participate in the original TREC tracks) are at risk from being miss-ranked. In particular, when comparing the ranking of the same 19 systems when all were pooled vs. when only 18 of them were pooled, we observed that ranking errors start to occur (average Kendall's τ and τ_{AP} values of 0.9815 and 0.9597, respectively).

These ranking errors stem from a system identifying sentences that are: 1) highly important (e.g. they cover a nugget that it is very difficult to find sentences for), 2) uncommon (no other system contributed those sentences to the pool) and 3) the system assigned them a high confidence score (so they would have been added to the pool if the system had been part of the initial evaluation). The result of such a system not being included in the pool is that a portion of its top k documents will not have been assessed,[3] and unassessed sentences are assumed to not be relevant to any nuggets. Hence, that system's performance estimation is likely to be an underestimation.

From a test collection perspective, such a system not being included in the pool leads to missing <sentence,nugget> pairs. These pairs could be recovered by pooling all new systems and then re-assessing, however, this additional cost eliminates much of the value that these test collections bring to IR evaluation. Indeed, if we use the total amount of time it reportedly took the TREC assessors to perform nugget matching [5] then each additional sentence assessed takes around 50 seconds on average. Moreover, this does not factor in time taken to set up the assessment system and recruit assessors. As such, it would be advantageous to have an automatic means to generate the missing <sentence,nugget> pairs without resorting to more human assessment.

In the remainder of this section we examine methods for automatically generating missing <sentence,nugget> pairs using the initial set of <sentence,nugget> pairs from the TREC-TS pool as a base, which we refer to as *match expansion*. In particular, we first discuss our experimental methodology (Section 5.1) as well as evaluation metrics (Section 5.2). Then we propose two approaches to generate new <sentence,nugget> pairs (Section 5.3). Finally, we report our experimental results in Section 5.4.

5.1 Matching Expansion Methodology

To evaluate matching expansion, we need two sets of sentences for which we know what the correct <sentence,nugget> pairs are. The first set we need represents the sentence pool pre-expansion, while the second set represents the sentences and <sentence,nugget> pairs that are the correct expansions our proposed system should produce. Previously in Section 3.3 we created such a setting via depooling, which we re-use here. In particular, for each of our synthetic systems, by depooling that system we create a scenario where some sentences and associated <sentence,nugget> pairs will be eliminated from the pool. The goal of a matching expansion algorithm in this case is to then to restore as many of those sentences and associated <sentence,nugget> pairs as possible, while avoiding introducing erroneous <sentence,nugget> pairs.

5.2 Matching Expansion Metrics

Next, we need to define metrics that can tell us how effective an expansion attempt is. As we are proposing automatic methods to expand the ground truth, we could do more harm than good by introducing false <sentence,nugget> pairs if not careful. As such, we define three primary metrics to capture expansion effectiveness, namely: Expansion Recall (*E-Recall*), Avg. Expansion Pair F_1 (*aEP-*F_1) and Avg. τ/τ_{AP} ($a\tau/a\tau_{AP}$).

Expansion Recall (*E-Recall*): As discussed above, from a test collection perspective a system not being included in the pool leads to missing <sentence,nugget> pairs, i.e. pairs that would have been added if the top k sentences for that system had been assessed. The goal of the match expansion is to recover these missing <sentence,nugget> pairs. We define E-Recall to be the proportion of missing sentences that were correctly matched to one or more nuggets, calculated as:

$$\text{E-Recall}(\mathcal{S}_p^{\mathcal{M}}, \mathcal{S}_{up}^{\mathcal{M}}, \mathcal{S}_{up\to e}^{\mathcal{M}}) = \frac{|(\mathcal{S}_p^{\mathcal{M}} \cap \mathcal{S}_{up}^{\mathcal{M}}) \cap (\mathcal{S}_{up\to e}^{\mathcal{M}} \cap \mathcal{S}_{up}^{\mathcal{M}})|}{|\mathcal{S}_p^{\mathcal{M}} \cap \mathcal{S}_{up}^{\mathcal{M}}|} \tag{4}$$

where $\mathcal{S}_p^{\mathcal{M}}$ is the set of sentences that correctly matched one or more nuggets if system $\mathcal{S}$ was pooled, $\mathcal{S}_{up}^{\mathcal{M}}$ is the set of sentences that correctly matched one or more nuggets even if system $\mathcal{S}$ was not pooled (i.e. derived from sentences contributed by other pooled systems) and $\mathcal{S}_{up\to e}^{\mathcal{M}}$ is the set of sentences that would correctly match one or more nuggets if system $\mathcal{S}$ was not pooled but after a matching expansion technique (see Section 5.3) has been applied. This is analogous to recall from a classification perspective, representing the proportion of all sentences that we were able to correctly restore through automatic expansion. Note that we are not interested in a precision-like metric here, as 'false positives' represent sentences that do not exist in $\mathcal{S}_p^{\mathcal{M}}$. Indeed, as $\mathcal{S}_p^{\mathcal{M}}$ results from top k pooling, we know that it is incomplete, meaning that a 'false positive' might represent a relevant sentence that does cover one or more nuggets, but no other system contributed it to the pool.

Avg. Expansion Pair F_1 (*aEP-*F_1): On the other hand, a simple recall estimation is also insufficient to determine the quality of an

[3] Recall that the top k pooling methodology guarantees that all pooled systems will have had their top k documents assessed.

expansion, as the goal is to restore all missing <sentence,nugget> pairs correctly. E-Recall only specifies the proportion of missing sentences for which expansion generated at least one correct match (<sentence,nugget> pair). Hence, we need a second metric that for each restored sentence, which has ≥1 correct <sentence,nugget> pairs, measures the *proportion of matches* for the sentence that are *correct*. Extending the idea of precision and recall for this setting, we start by defining Expansion Pair Precision (*EP-P*) and Expansion Pair Recall (*EP-R*) for a sentence u as follows:

$$\text{EP-P}(\mathcal{M}^u_p, \mathcal{M}^u_{up\to e}) = \frac{|\mathcal{M}^u_p \cap \mathcal{M}^u_{up\to e}|}{|\mathcal{M}^u_{up\to e}|} \tag{5}$$

$$\text{EP-R}(\mathcal{M}^u_p, \mathcal{M}^u_{up\to e}) = \frac{|\mathcal{M}^u_p \cap \mathcal{M}^u_{up\to e}|}{|\mathcal{M}^u_p|} \tag{6}$$

where $\mathcal{M}^u_p$ is the set of <sentence,nugget> pairs for sentence u that resulted from matching after being pooled and $\mathcal{M}^u_{up\to e}$ is the set of <sentence,nugget> pairs for sentence u that were produced by expansion when u was not pooled. *EP-P* measures the proportion of nugget matches for sentence u produced by expansion that were correct, while *EP-R* measures the proportion of all nugget matches for sentence u that were restored. We can average both *EP-P* and *EP-R* across all sentences with matches from the TREC-TS pool (i.e. where we know $\mathcal{M}^u_p$) and that were subject to expansion (i.e. $\mathcal{M}^u_p$ and $\mathcal{M}^u_{up\to e}$ are different), which we denote as *aEP-F* and *aEP-R*, respectively. In our later experiments, to have a single metric representing pair generation quality, we report the harmonic mean of *aEP-P* and *aEP-R* across sentences, denoted *aEP-F*$_1$:

$$\text{aEP-F}_1 = 2 \cdot \frac{\text{aEP-P} \cdot \text{aEP-R}}{\text{aEP-P} + \text{aEP-R}} \tag{7}$$

Using these two metrics, we can express how effective an expansion attempt is. For instance, if an expansion method achieved an *E-Recall* of 0.1456 and an *aEP-F*$_1$ of 0.9621, then we can read this follows. First, the expansion method managed to find 14.6% of the missing sentences. Second, of the sentences found, matching F_1 was high at 0.9621, meaning that nearly all missing <sentence,nugget> pairs were restored and very few erroneous <sentence,nugget> pairs were introduced in the process.

Avg. τ/τ_{AP} ($a\tau/a\tau_{AP}$): Finally, while the above metrics capture the effectiveness of an expansion attempt from the perspective of the sentences and <sentence,nugget> pairs restored, our end-goal is to reduce the likelihood that systems which are depooled will be miss-ranked. Previously in Table 5 we reported the average correlation between the correct ranking of systems and the ranking of systems after each individual system was removed from the pool. In a similar way, we can also calculate the average correlation between 1) the correct ranking of systems and 2) the ranking of systems produced after an individual system has been removed and then expansion has been attempted. Intuitively, if expansion is successful, then average correlation should increase, i.e. expansion should reduce the number of ranking errors introduced when each system is depooled. Hence, we report the average τ and τ_{AP} both before and after expansion, as well as the average number of rank swaps needed to restore the correct ranking (i.e. the number of miss-rankings).

5.3 Matching Expansion Techniques

There are two ways one might attempt to generate expansions for <sentence,nugget> pairs. Either we start with a sentence and we try to find all related nuggets (i.e. attempt to simulate in an automatic manner the process that the TREC assessors perform during matching). Alternatively, we start with a <sentence,nugget> pair and try to find other good matches in the collection. To achieve either of these approaches we need effective representations for both the sentences and nuggets. In the case of a sentence, the only representation we have for it is its text. However, an open question is how we represent a nugget. For our experiments here, we choose to use existing <sentence,nugget> mappings to form a series of textual representations for each nugget. Here, each previously matched sentence in the pool is a representation for its associated nuggets. To generate new <sentence,nugget> pairs, for each nugget representation (a sentence *A*), we search for other textually similar sentences. If another sentence *B* is sufficiently similar to *A*, then we infer that for all <*A*,nugget> pairs, there should also be <*B*,nugget> pairs. To calculate similarity, we experiment with two ways to estimate the distance between texts, namely: raw text similarity and semantic similarity, which we summarize below.

Raw Text Similarity: For this expansion method, we take the simplest of approaches, using textual similarity with Levenshtein distance between each sentence that has one or more matches in the pool and every other sentence in the collection. If a sentence *A* and a sentence *B* have a similarity above a threshold θ, then for each <*A*,nugget> pair we add an associated <*B*,nugget> pair.

Semantic Similarity: As sentences in the TREC-TS test collections are drawn from an article stream comprised of news articles, blogs and forum posts, we can expect significant vocabulary miss-match between different articles discussing the same information. For this reason, it is reasonable to expect raw text similarity will fail to identify some similar sentences due to vocabulary miss-matches. To tackle this issue, we also experiment with the identification of similar sentences based on semantic rather than textual similarity. In this case, we represent each sentence as a high-dimensional sentence embedding and then calculate similarity in terms of that semantic space. Sentence embeddings have previously been shown to be an effective sentence representation when calculating similarity [15]. In particular, given a sentence, for each word in that sentence, we first convert that word into a high-dimensional word embedding. To produce a sentence embedding we calculate a single position per dimension by averaging the word positions per dimension. For reproducibility, we use Word2Vec [21] along with pre-trained word embeddings from the Google News dataset (about 100 billion words).[4] We use Cosine similarity for calculating the distance between the sentence embedding vectors. If a sentence *A* and a sentence *B* have a similarity above a threshold θ, then for each <*A*,nugget> pair, we then add an associated <*B*,nugget> pair.

For both similarity metrics we also need to define a similarity threshold, above which a nugget and sentence will be considered a match. The correct similarity threshold will vary between techniques, e.g. what might be considered an acceptable threshold for raw text similarity will differ from semantic similarity. For brevity,

[4] Available from https://code.google.com/archive/p/word2vec/

Table 6: Comparison of match expansion and resultant correlation with the correct system ranking on average across depooling scenarios and topics when performing sentence to sentence similarity expansion. Statistically significant changes (paired t-test p<0.05) in $a\tau$ and $a\tau_{AP}$ against no expansion (None) are denoted △ and ▽ for increases and decreases respectively.

		Expansion Metrics		Ranking Metrics		
Expansion Technique	Threshold θ	E-Recall	aEP-F_1	Avg Rank Swaps	$a\tau$	$a\tau_{AP}$
None	-	-	-	1.5789	0.9815	0.9597
Item-Item Similarity Expansion	0.99	0.0714	0.9401	1.4211	0.9834	0.9597
Item-Item Similarity Expansion	0.90	0.1096	0.8336	1.0526	0.9877△	0.9619
Item-Item Similarity Expansion	0.80	0.1199	0.8349	1.0526	0.9876△	0.9610
Item-Item Similarity Expansion	0.70	0.1199	0.8349	1.0526	0.9876△	0.9610
Item-Item Semantic Expansion	1.00	0.0679	**0.9662**	1.1579	0.9865	0.9625
Item-Item Semantic Expansion	0.98	0.1239	0.8653	0.8421	**0.9902**△	0.9640
Item-Item Semantic Expansion	0.96	0.1666	0.7436	**0.6842**	0.9920△	**0.9878**△
Item-Item Semantic Expansion	0.94	0.2383	0.4819	1.9473	0.9772	0.9334
Item-Item Semantic Expansion	0.92	0.3678	0.2253	5.8421	0.9317▽	0.8361▽
Item-Item Semantic Expansion	0.90	**0.5034**	0.1341	8.2105	0.9040▽	0.7359▽

in the following section we only report performances from around the peak threshold θ observed based on experimentation with different θ ranges. We refer to expansion with the raw text as *item-item similarity expansion* and expansion with semantic similarity as *item-item semantic expansion*.

5.4 Matching Expansion Results

Table 6 reports the effectiveness of our different proposed expansion techniques in terms of the metrics discussed in Section 5.2. We can divide our metrics into two classes, Expansion Metrics that measure how effectively we are restoring missing <sentence,nugget> pairs; and Ranking Metrics that report how well correlated with the correct system ranking we are after expansion has occurred. E-Recall performances indicates the proportion of the missing sentences that were restored during expansion (higher is better), while aEP-F_1 indicates how effectively we restored the matches for those sentences (higher is better). Under the ranking metrics, # Rank Swaps is the number of swaps needed on average to re-create the correct ranking (lower is better), while $a\tau$ and $a\tau_{AP}$ indicate the resultant correlation between the ranking produced post-expansion and the correct ranking (higher is better). In the case of the Ranking Metrics $a\tau$ and $a\tau_{AP}$, we also report statistically significant changes (paired t-test $p < 0.05$) against no expansion (None).

From Table 6, examining the two approaches that use the sentences texts alone for expansion, we observe different behaviours depending on whether raw text similarity or semantic similarity was applied. In the case of using the raw text for identifying similar sentences, we see that the E-Recall scores range between 7% and 12%. This indicates that regardless of the threshold selected, only a small proportion of the missing sentences could be found by comparing the raw text of the sentences. In turn, this indicates that uncommon sentences predominantly do not use the same language as the more common sentences. On the other hand, we also see high aEP-F_1 performances ranging from 0.9401 to 0.8349. This shows that although only a few of the missing sentences are found, the matches derived from those sentences were almost always correct (i.e. all matches for each sentence were restored and we did not introduce many erroneous matches in the process). If we then examine the effect that using sentence text expansion with raw text similarity has on the ranking of systems, we see that this type of expansion results in fewer ranking errors than observed prior to expansion (None). Indeed, the average number of swaps needed to restore the correct ranking drops by 30% from 1.5789 to 1.0526 (threshold=0.9) and an average Kendall's τ correlation against the correct ranking increases by a small but statistically significant margin (0.9815 to 0.9877) across the scenarios tested. Importantly this shows that the restoration of a relatively small proportion of all sentences that should have been pooled (e.g. around 10%) can eliminate around 30% of the ranking errors (1.5789 rank swaps to 1.0526 rank swaps).

Next, we examine how effective sentence expansion is when is when using semantic similarity rather than raw text similarity from Table 6. Examining the expansion metrics first, we observe a much wider range of E-Recall values as we vary the similarity threshold. In particular, exact vector matching (threshold=1.0) results in 6.8% of the missing sentences being found, while a lower similarity threshold of 0.9 results in half (50.34%) of the missing sentences being found. This shows that semantic similarity comparisons are more effective for finding uncommon sentences than raw text matching. However, as we lower the similarity threshold, we also observe a rapid decline in aEP-F_1 (0.9662 when the θ=1.0 to 0.1341 when the θ=0.9). This illustrates that semantic expansion is much more prone to introducing erroneous matches as we relax the selection constraint. If we examine how this affects the system ranking performance, we observe that even though semantic expansion introduces a higher proportion of erroneous matches than raw text expansion, it can be more effective. In particular, we see that when the vector similarity threshold θ is set to 0.98 and 0.96, we further reduce the average number of rank swaps (ranking errors) to 0.8421 and 0.6421, respectively. If we compare this to the number of ranking errors prior to expansion, then semantic expansion can reduce the number of errors by up to 50% (1.5789 rank swaps to 0.6842 rank swaps). This also increases the correlation between the system ranking post-expansion and the correct ranking by a statistically significant margin under both $a\tau$ and $a\tau_{AP}$.

To answer RQ2, we have shown that automatic <sentences,nugget> expansion techniques that find textually or semantically similar sentences and use those sentences to infer matches can be effective for increasing the robustness of the TREC-TS test collections. In particular, we have shown that expansion using raw text similarity can reduce the number of miss-rankings by 30%, while semantic similarity-based expansion can reduce the number of miss-rankings

by 50%. We conclude that these expansion techniques improve the robustness of the TREC-TS test collections, reducing the risk of miss-ranking new systems that were not pooled.

6 CONCLUSIONS

In this paper, we quantified the extent to which the TREC Temporal Summarization (TREC-TS) test collections are able to robustly rank different timeline generation systems in scenarios where one of those systems was not included in the sampled pool. We also proposed new automatic approaches aimed at improving the robustness of those test collections. Through a leave-one-out (depooling) experiment, we observed a mixed picture in terms of test collection robustness. In particular, the TREC-TS test collections appear to be robust when evaluating systems that were not pooled at the lower end of the effectiveness scale, i.e. the correlation with the correct system ranking is perfect when the system is depooled. On the other hand, when more effective systems were depooled, we started to observe ranking errors and lower correlations, particularly towards the top ranks. Hence, we conclude that using the TREC-TS test collections out-of-the-box is subject to some risk.

To reduce this risk, we studied two approaches that use the available sentence pool and associated labels (<sentence,information unit> matches) to automatically generate new labels that might have been missed due to sentences not being pooled. In particular, we evaluated item-item similarity expansion and item-item semantic expansion approaches. Our experiments using these two proposed expansion techniques showed that automatically adding even a small number of <text item,information unit> pairs can markedly reduce the number of ranking errors observed when using the TREC-TS test collections. In particular, item-item similarity expansion can reduce the number of ranking errors by up to 30% while item-item semantic expansion can reduce the number of ranking errors by up to 50%. Since our results show that matching expansion techniques enhance robustness of the TREC-TS test collections, we recommend the use of expanded match sets when evaluating new systems, particularly in cases where low completeness levels are observed.

Our work highlighted the limitations of shallow pooling for creating test collections from large streaming data, particularly for tasks where item selections are not independent (in this case selection of an item for inclusion within the summary is dependent on past selections of other sentences), resulting in low overlap between systems when pooling summaries for labeling. In these cases, new labeling approaches are needed that provide increased stream coverage, without dramatically increasing labeling time/cost.

7 DATA RELEASE

To support future researchers working with the TREC-TS test collections, we provide both the synthetic systems that we used in this study as potential new baselines for the community, as well as the expanded assessment matches produced by the best performing expansion approach (Item-Item Semantic Expansion $\rightarrow \theta = 0.96$), which we recommend that researchers use in scenarios where they are experiencing low completeness levels under the official labels. These can be freely downloaded at:

http://dx.doi.org/10.5525/gla.researchdata.613

REFERENCES

[1] James Allan, Rahul Gupta, and Vikas Khandelwal. 2001. Temporal summaries of new topics. In *ACM SIGIR 2001*.

[2] Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. 2014. TREC 2014 Temporal Summarization Track Guidelines. In *TREC 2014*.

[3] Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. 2015. TREC 2015 Temporal Summarization Track Overview. In *TREC 2015*.

[4] Javed A Aslam, Matthew Ekstrand-Abueg, Virgil Pavlu, Fernando Diaz, and Tetsuya Sakai. 2013. TREC 2013 Temporal Summarization. In *TREC 2013*.

[5] Gaurav Baruah, Richard McCreadie, and Jimmy Lin. 2017. A Comparison of Nuggets and Clusters for Evaluating Timeline Summaries. In *ACM CIKM 2017*.

[6] Chris Buckley and Ellen M. Voorhees. 2004. Retrieval Evaluation with Incomplete Information. In *ACM SIGIR 2004*.

[7] Chris Buckley and Ellen M. Voorhees. 2017. Evaluating Evaluation Measure Stability. *SIGIR Forum* 51, 2 (Aug. 2017), 235–242.

[8] John M. Conroy, Judith D. Schlesinger, and Dianne P. O'Leary. 2011. Nouveau-ROUGE: A Novelty Metric for Update Summarization. *Computational Linguistics* 37 (2011), 1–8.

[9] Hoa Dang. 2005. Overview of DUC 2005. In *DUC 2015*.

[10] Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 Opinion Question Answering and Summarization Tasks. In *TAC 2008*.

[11] Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Fernando Diaz. 2016. A Study of Realtime Summarization Metrics. In *CIKM 2016*.

[12] Qi Guo, Fernando Diaz, and Elad Yom-Tov. 2013. Updating Users about Time Critical Events. In *ECIR 2013*.

[13] Chris Kedzie, Fernando Diaz, and Kathleen McKeown. 2016. Real-Time Web Scale Event Summarization Using Sequential Decision Making. *arXiv preprint arXiv:1605.03664* (2016).

[14] Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2015. Predicting Salient Updates for Disaster Summarization.. In *ACL 2015*.

[15] Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *ACM CIKM 2015*.

[16] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *ACL Workshop On Text Summarization 2004*.

[17] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. 2014. Overview of the TREC-2014 Microblog Track. In *TREC 2014*.

[18] Craig Macdonald and Iadh Ounis. 2011. The influence of the document ranking in expert search. *Information Processing & Management* 47, 3 (2011), 376–390.

[19] Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2014. Incremental Update Summarization: Adaptive Sentence Selection based on Prevalence and Novelty. In *CIKM 2014*.

[20] Richard McCreadie, Rodrygo Santos, Craig Macdonald, and Iadh Ounis. 2017. Explicit diversification of event aspects for temporal summarization. *ACM TOIS* (2017).

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[22] Ani Nenkova and Kathleen McKeown. 2011. Automatic Summarization. *FnTIR* 5, 2–3 (2011), 103–233.

[23] Paul Over. 1997. TREC-6 Interactive Report. In *TREC 1997*.

[24] Tetsuya Sakai. 2008. Comparing metrics across TREC and NTCIR:: the robustness to pool depth bias. In *ACM SIGIR 2008*.

[25] Rodrygo LT Santos, Iadh Ounis, and Craig Macdonald. 2015. Search result diversification. *Foundations and Trends in Information Retrieval* 9, 1 (2015), 1–90.

[26] Luchen Tan, Adam Roegiest, Charles L. A. Clarke, and Jimmy Lin. 2016. Simple Dynamic Emission Strategies for Microblog Filtering. In *ACM SIGIR 2016*.

[27] Andrew Turpin and Falk Scholer. 2006. User performance versus precision measures for simple search tasks. In *ACM SIGIR 2006*.

[28] Ellen M. Voorhees. 1998. Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness. In *SIGIR 1998*.

[29] Ellen M. Voorhees. 2003. Overview of the TREC 2003 Question Answering Track. In *TREC 2003*.

[30] Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011. Evolutionary Timeline Summarization: a Balanced Optimization Framework via Iterative Substitution. In *ACM SIGIR 2011*.

[31] Emine Yilmaz, Javed A. Aslam, and Stephen Robertson. 2008. A New Rank Correlation Coefficient for Information Retrieval. In *ACM SIGIR 2008*.

[32] ChengXiang Zhai, William W. Cohen, and John Lafferty. 2003. Beyond Independent Relevance: Methods and Evaluation Metrics for Subtopic Retrieval. In *ACM SIGIR 2003*.

[33] Chunyun Zhang, Zhanyu Ma, Jiayue Zhang, Weiran Xu, and Jun Guo. 2015. A Multi-level System for Sequential Update Summarization. In *QSHINE 2015*.

[34] Justin Zobel. 1998. How Reliable Are the Results of Large-scale Information Retrieval Experiments?. In *ACM SIGIR 1998*.