The University of Amsterdam (ILPS.UvA) at TREC 2015 Temporal Summarization Track

Cristina Gârbacea and Evangelos Kanoulas

University of Amsterdam, Amsterdam, The Netherlands
g.c.garbacea@uva.nl, e.kanoulas@uva.nl

Abstract. In this paper we report on our participation in the TREC 2015 Temporal Summarization track, aimed at encouraging the development of systems able to detect, emit, track, and summarize sentence length updates about a developing event. We address the task by probing the utility of a variety of information retrieval based methods in capturing useful, timely and novel updates during unexpected news events such as natural disasters or mass protests, when high volumes of information rapidly emerge. We investigate the extent to which these updates are retrievable, and explore ways to increase the coverage of the summary by taking into account the structure of documents. We find that our runs achieve high scores in terms of comprehensiveness, successfully capturing the relevant pieces of information that characterize an event. In terms of latency, our runs perform better than average. We present the specifics of our framework and discuss the results we obtained.

1 Introduction

In recent years, the continuous growth of online information calls for mechanisms able to find and present the textual content efficiently to the end user. Multi-document summarization techniques aim at producing high quality summaries of text buried inside large collections of related documents by condensing the high volume of information into a short, human comprehensible synopsis. In general, there are two main approaches to the task of summarization – extraction and abstraction. Extractive summarization methods determine which are the most important words, phrases or sentences inside the input documents, and select a subset of these to form a summary. On the other hand, abstractive summarization methods build an internal representation of the original documents and exploit semantics and natural language generation techniques to create a summary close to what a human would output. In this paper we focus on extractive summarization techniques of multiple documents during unexpected news events, such as natural disasters, cataclysms, and mass protests.

The TREC 2015 Temporal Summarization task runs for the third consecutive year and is focused on the development of systems that can summarize emerging events in a real-time fashion. It consists of three subtasks: *i) Filtering and Summarization*, *ii) Pre-Filtered Summarization*, and *iii) Summarization Only*. All subtasks involve summarization of high volume streams of news articles and

blog posts crawled from the web . Before the actual summarization, subtasks *i)* and *ii)* require an additional preprocessing step aimed at filtering the relevant documents to be summarized for a specific event. However, our participation in this competition focuses mainly on addressing subtask *iii)*, i.e. we aim to explore ways of identifying potential update sentences by assuming that all documents received as input are relevant to our query event.

The remainder of this paper is organized as follows: Section 2 describes prior initiatives and methods for temporal summarization of events, Section 3 discusses the experimental design of our study, Section 4 describes the experimental results of the performed experiments, and provides an analysis of these results around the limitations of the methods being tested, and last Section 5 outlines the conclusions of our work as well as future directions informed by these conclusions.

2 Related Work

Previous TREC 2013 and TREC 2014 Temporal Summarization campaigns released a series of events that could be used by participant systems to develop algorithms for text summarization and model information reliability in a dynamic setting. There were 10 test events released for the 2013 collection, and 15 test events released for the 2014 collection respectively. Each event is characterized by an event query, the time period the event spans, and the type of the event – can be one of the following: accident, bombing, conflict, earthquake, hostage, protest, riot, storm, and shooting. The corpus, namely the TREC KBA Stream Corpus[1], consists of a set of timestamped documents from a variety of news and social media sources. The use of external information is allowed as long as this information existed before the event start time, or is time-aligned with the KBA corpus.

Most participants employed a pipeline where information is first pre-processed (this involves decrypting, decompressing and indexing the corpus), retrieved (using a wide range of methods for document and sentence retrieval), and finally processed (ranking the retrieved sentences by time and similarity to any prior emitted sentences). Almost all participant systems used query expansion techniques as a way to improve recall, given the short length of the query and the typical mismatch between query terms and the terms found inside relevant updates. Both supervised and unsupervised methods have been used to generate sequential update summarizations. Latent Dirichlet Allocation was used to find that latent semantic topics of documents and generate lists of weighted keywords that could help in sentence scoring and ranking. Discriminative methods for extracting keywords (χ^2) have been employed to collect relevant terms describing an event, and later used as features in training an SVM classifier for sentence update detection. Other participants tried to model events by employing a generic event model, leaving from the assumption that event updates share a common vocabulary of terms independent of the event type. Clustering methods were

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

used to group similar sentences, and from each cluster the cluster centroids were picked as the most salient sentences to output. Finally, sentences are tested for novelty and only the ones passing this filter are emitted as updates.

3 Experimental Design

In this section we describe the experimental design that we used in our analysis. In retrieving relevant updates we consider different information retrieval based approaches that have been adopted in text and document summarization. Ideally, an emitted update should be significant, timely, non-verbose, and novel. We aim to incorporate these all these qualifiers in our summarization framework. In what follows we explain how we account for such characteristics of a sentence update, and illustrate the main components of the summarization system we developed.

1. **Corpus preprocessing:** TREC KBA Stream Corpus is an encrypted file whose decryption requires an authorized key provided to each participant team by the TREC organizers. Flat files have been serialized into batches of documents called Chunks, and further compressed. In order to retrieve the content of a document, we need to perform these operations in reverse order: after decompressing each file, we use the tools provided by the StreamCorpus[2] toolbox to extract large streams of text from each StreamItem and store its content in a custom format.
2. **Document retrieval:** Given the large volume of data, we proceed to indexing the extracted documents into multiple ElasticSearch indices. This makes it convenient in terms of scalability, searching for documents in almost real-time, and enhances the repeatability of our experiments. For each event we issue the event query specified in the description of the event and retrieve relevant documents that constitute the input to our sentence extraction and summarization module. We discard all documents which are outside the time range of a given event.
3. **Query expansion:** The query describing an event is typically very short (2-3 words in legth), and this makes the retrieval of relevant sentences prone to word mismatch problems in cases when the vocabulary of the query differs significantly from the vocabulary of an update. To prevent this, we rely on query expansion techniques to augment a query word with similar terms. We use two methods: *i) Wordnet* - for each query term we retrieve its Wordnet synonyms [9], and augment the original query with these terms, and *ii) Word2Vec* - we train our model [8] on sentences from the relevant documents in TREC Temporal Summarization 2013 and 2014 collections, retrieve the most similar terms to a query term, and add them to the expanded query.
4. **Sentence extraction and summarization:** We employ a variety of sentence selection methods for finding relevant updates inside the relevant documents. In particular, we are interested in finding whether an event update is central in the documents that contain it, and to what extent event updates

[2] `https://github.com/trec-kba/streamcorpus`

are retrievable by means of the shared vocabulary between the language of an event query and the language of an event update. To this end, we probe the utility of the following well-established information retrieval methods:

(a) *Term Frequency:* We rank sentences by the number of matching event query terms found inside a sentence. We set a predefined threshold for the least number of times query terms should be present in a sentence based on empirical observations on the TREC Temporal Summarization 2014 collection. This enables our method to perform in a real time streaming scenario.

(b) *TF.ISF:* Similar to the traditional term frequency - inverse document frequency (tf.idf) method used for document retrieval, the vector space model for sentence retrieval uses the term frequency - inverse sentence frequency (tf.isf) method [4]. Using tf.isf, we rank sentences with the following formula:

$$R(s|q) = \sum_{t \in q} \log(tf_{t,q} + 1) \log(tf_{t,s} + 1) \log\left(\frac{n+1}{0.5 + sf_t}\right) \quad (1)$$

where
- $tf_{t,q}$ is the number of occurrences of term t in query q,
- $tf_{t,s}$ is the number of occurrences of term t in query s,
- sf_t is the number of sentences that contain term t,
- n is the number of sentences in the collection.

To compute the number of sentences that contain a query term t, we treat each document as a collection of sentences. We infer the rest of the counts from documents at the time of emission. We rank sentences in the document according to their corresponding tf.isf values, and keep the ones with a tf.isf score higher than a pre-set threshold.

(c) *Query Likelihood:* The query likelihood model for sentence retrieval ranks sentences by the probability that the query was generated by the same distribution of terms the sentence is from. Since it retrieves sentences that contain exact words as the query, this makes it appropriate for exact similarity match:

$$P(S|Q) \propto P(S) \prod_{i=1}^{|Q|} P(q_i|S) \quad (2)$$

where Q is the query, $|Q|$ is the number of terms in the query, q_i is the i^{th} term in the query, and S is a sentence. The effectiveness of the query likelihood model is demonstrated in prior work on sentence retrieval research [2], [7], [10] where it outperforms word overlap and tf.isf based measures. In addition to the regular query likelihood model, we are using query likelihood linear interpolation smoothing.

(d) *Log-Likelihood Ratio (LLR):* We aim to extract discriminative terms that can distinguish an update sentence from a non-update sentence.

We model the characteristics of an event as the set of the most discriminative LLR terms, which we infer for each event type by building two distinct corpora: a foreground corpus consisting of all relevant event updates, and a background corpus to estimate the importance of a word made up of all non-update sentences per event type from the relevant documents. To build these two corpora we use data from past TREC Temporal Summarization tracks. We use a slight variation of the original method [11], log-likelihood ratio with cut-off and query sensitivity LLR(CQ), to inform the summarizer to make the output more focused [6]. We rank terms by their LLR score and consider the top-N retrieved for each event type when selecting which sentences to include in the summary.

(e) *Latent Dirichlet Allocation (LDA):* We use LDA to capture events covered by the relevant documents. These documents typically have a central theme or event, and other sub-events which support or revolve around this central event. The central theme and the sub-events altogether determine the set of topics covered by the relevant documents. LDA [3] is a generative hierarchical probabilistic model which represents documents as finite mixtures over an underlying set of topics. These topics are modeled in turn as an infinite mixture over an underlying set of topic probabilities. We follow [1] and we weight sentences using a purely statistical approach of capturing the events documents are based on:

$$P(S|T_j) = \sum_{W_i \in S} P(W_i|T_j) * P(T_j|D_B) * P(D_B) \tag{3}$$

where
- $P(S|T_j)$ is the probability that the sentence S represents topic T_j,
- $\sum_{W_i \in S} P(W_i|T_j)$ is the probability that the words of the sentence S belong to topic T_j,
- $P(T_j|D_B)$ is the probability that topic T_j belongs to document D_B,
- $P(D_B)$ is the probability of document D_B (we assume the probability of each relevant document as uniform).

Additionally, we score and rank sentences by the weight of topic words .

(f) *Language Modeling:* We use TREC Temporal Summarization historical data (2013 and 2014) to build a unigram language model from all relevant event updates. We hypothesize that event updates share a common crisis-related vocabulary, that distinguishes them from other non-update sentences. In our implementation we use SRILM[3], an extensible language modeling toolkit which supports the creation and evaluation of a variety of language model types based on N-gram statistics [12].

(g) *Cosine Similarity:* We rank sentences by the cosine of the angle between the document vector and the query vector. We compute the vector representation of each query and each sentence in turn, using tf.idf term

[3] `http://www.speech.sri.com/projects/srilm/`

weights. We compute if values on prior documents. After getting the corresponding vectors, the distance between two vectors is simply defined by:

$$\cos\theta = \frac{\boldsymbol{a} \times \boldsymbol{b}}{||\boldsymbol{a}|| \times ||\boldsymbol{b}||} \tag{4}$$

where θ is the angle between the two vector representations $\boldsymbol{a}$ and $\boldsymbol{b}$.

(h) *Sentence Centrality:* We test across document centrality by running LexRank [5], a state-of-the-art graph based summarization algorithm. LexRank assesses the centrality of each sentence in a cluster (centrality of a sentence is defined in terms of the centrality of the words contained inside the sentence), and extracts the most salient sentences to include in the summary by building a weighted graph with nodes that represent sentences and edges that represent the cosine similarity between pairs of sentences.

5. **Novelty detection:** We rank the sentences retrieved by each of the above methods by time, and at each point in time we ensure we are not adding duplicate content to the summary. To this end, we use the cosine similarity metric presented above to check the degree of redundancy between a new sentence we are about to output and all prior sentences already added to the summary. If the similarity measure is higher than the 0.5 threshold, we discard the sentence as the information contained inside the sentence has already been captured by a more timely update.

4 Results and Analysis

Dataset. We test our methods on the TREC Temporal Summarization 2015 dataset, which is a subset of the TREC KBA 2014 Stream Corpus (4.5 TB). The corpus spans the time period October 2011 – April 2013, and includes timestamped documents collected from a variety of news articles and social media sources. For the *Summarization Only* sub-task we use the filtered corpus of on-topic documents (TREC-TS-2015F-RelOnly) with data for a set 21 crisis events. We submit 15 runs to this sub-track based on the methods presented above, or variations thereof. For the *Pre-filtered Summarization* sub-task, we use the pre-filtered corpus of news articles and blog posts (TREC-TS-2015F), and submit one additional run.

Results. In Table 1 we report on the results we obtained using the official evaluation metrics for the task. We observe that the performance of our runs is very good in terms of recall, and that we manage to retrieve relevant updates covering the important nuggets using our methods. Except for the query likelihood with smoothing run, the coverage of our summaries in terms of *comprehensiveness* is above average as illustrated in Figure 3, and culminates in a maximum of 0.8415 when we identify updates using simple query term frequency. This implies that the summaries we generate identify a great part of the

Table 1. TREC Temporal Summarization 2015 results (average for each submitted run across all test events).

Run	nE[Gain]	nE[Latency Gain]	Comprehen-siveness	Latency	HM
Query likelihood no smoothing	0.0200	0.0145	0.7541	0.5381	0.0277
Query likelihood with smoothing	**0.0798**	**0.0453**	0.4222	0.2687	**0.0618**
Query likelihood with smoothing + higher threshold	0.0359	0.0204	0.6662	0.4664	0.0375
Cosine similarity	0.0428	0.0260	0.5708	0.3655	0.0471
Cosine similarity expanded query (Word2Vec)	0.0281	0.0197	0.7325	0.5118	0.0372
Term frequency	0.0223	0.0160	**0.8415**	0.6289	0.0310
Term frequency expanded query (Wordnet)	0.0200	0.0147	0.8326	0.6209	0.0285
Term frequency expanded query (Word2Vec)	0.0264	0.0172	0.7992	0.5865	0.0330
TF.ISF	0.0234	0.0166	0.8196	0.6080	0.0321
TF.ISF expanded query (Wordnet)	0.0221	0.0158	0.8260	0.6169	0.0306
TF.ISF expanded query (Word2Vec)	0.0212	0.0153	0.8301	0.6107	0.0297
LexRank	0.0224	0.0157	0.7490	0.5111	0.0299
Language modeling	0.0195	0.0135	0.6871	0.4737	0.0258
LLR	0.0173	0.0130	0.8348	**0.6533**	0.0248
LDA	0.0222	0.0131	0.7036	0.4271	0.0250
LDAv2	0.0202	0.0126	0.7423	0.4778	0.0241
TREC TS 2015 Average	0.0595	0.0319	0.5627	0.3603	0.0472

essential information that could have been retrieved for a particular event, and that our methods are effective in terms of recall for the given task.

In terms of precision, however, our scores are comparable to average or lower, as shown in Figure 1 for the *normalized expected gain* metric. The query likelihood with smoothing presents the best precision among our runs, ranking on-topic and novel updates better than average. According to this custom precision metric, systems are penalized not only for an incorrect ranking of the retrieved updates, but also for "verbosity" – a characteristic of a system when it retrieves unreasonably long and difficult to read updates. For example, it could be the case

that our sentence updates do cover relevant nuggets, but they are too long and therefore get penalized for the additional reading effort they introduce. Compared to the normalized expected gain, the *normalized expected latency gain* metric adds an extra time dimension to the evaluation of a summary. When this time component is further considered, our scores understandably drop as cascading errors can propagate throughout the system; this effect can be seen in Figure 2. Interestingly, the query likelihood method with term smoothing is still the top performer. From Figure 4 we can infer statistics for the *latency component* metric. Contrary to its name, a higher value for latency is better because it means that a system does not delay the emission of sentences to collect more information before issuing updates. There is a lot of variation in the performance of our runs with respect to latency, but overall we observe all runs are doing better than average. Finally, the *harmonic mean* between the normalized expected latency gain and latency comprehensiveness is used for ranking the systems participating in the track. Figure 5 shows that not all of our runs surpass the average for this combined metric, however our best systems score above the average value. Query likelihood with smoothing and cosine similarity achieve the best results overall, and are our highest ranked submissions to the *Summarization Only* task.

We now turn to an event-level comparison of our methods. Out of the total 21 test events released, there are 8 events of type bombing, 7 events of type accident, 2 events of type protest, 2 events of type earthquake, 1 event of type conflict, and 1 event of type storm. For event types "accident" and "bombing", term frequency alone seems to identify many of the relevant updates, which implies that events in discussion share a common vocabulary with the query. This fact is confirmed by the presence of terms like "explosion", "bombing", "arson", "bomber", "bomb", "fire" as query terms for events of type bombing. Log-likelihood ratio, TF.ISF and query likelihood are close performers in retrieving updates that match the gold standard nuggets.

5 Conclusions

We have presented a variety of approaches for addressing the task of identifying relevant sentence-level updates that characterize an event for the purpose of extractive document summarization. We observe that traditional information retrieval algorithms present decent performance in detecting these updates, however often times we report an event with considerable time lag after the event has emerged. In future work we would like to focus on improving event detection in real time, and on event summarization at different granularities, possibly through the use of online hierarchical clustering algorithms and event modeling techniques.

Bibliography

[1] Arora, R., Ravindran, B.: Latent dirichlet allocation based multi-document summarization. In: Proceedings of the second workshop on Analytics for noisy unstructured text data. pp. 91–97. ACM (2008)

[2] Balasubramanian, N., Allan, J., Croft, W.B.: A comparison of sentence retrieval techniques. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 813–814. ACM (2007)

[3] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. the Journal of machine Learning research 3, 993–1022 (2003)

[4] Doko, A., Štula, M., Stipaničev, D.: A recursive tf–isf based sentence retrieval method with local context. International Journal of Machine Learning and Computing 3(2), 195–200 (2013)

[5] Erkan, G., Radev, D.R.: Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of Artificial Intelligence Research pp. 457–479 (2004)

[6] Gupta, S., Nenkova, A., Jurafsky, D.: Measuring importance and query relevance in topic-focused multi-document summarization. In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions. pp. 193–196. Association for Computational Linguistics (2007)

[7] Metzler, D., Bernstein, Y., Croft, W.B., Moffat, A., Zobel, J.: Similarity measures for tracking information flow. In: Proceedings of the 14th ACM international conference on Information and knowledge management. pp. 517–524. ACM (2005)

[8] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)

[9] Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J.: Introduction to wordnet: An on-line lexical database. International journal of lexicography 3(4), 235–244 (1990)

[10] Murdock, V.G.: Aspects of sentence retrieval. Ph.D. thesis, University of Massachusetts Amherst (2006)

[11] Rayson, P., Garside, R.: Comparing corpora using frequency profiling. In: Proceedings of the workshop on Comparing Corpora. pp. 1–6. Association for Computational Linguistics (2000)

[12] Stolcke, A., et al.: Srilm-an extensible language modeling toolkit. In: INTERSPEECH (2002)

Fig. 1. Results for the normalized expected gain metric, i.e. the degree to which the updates within the summary are on-topic and novel.

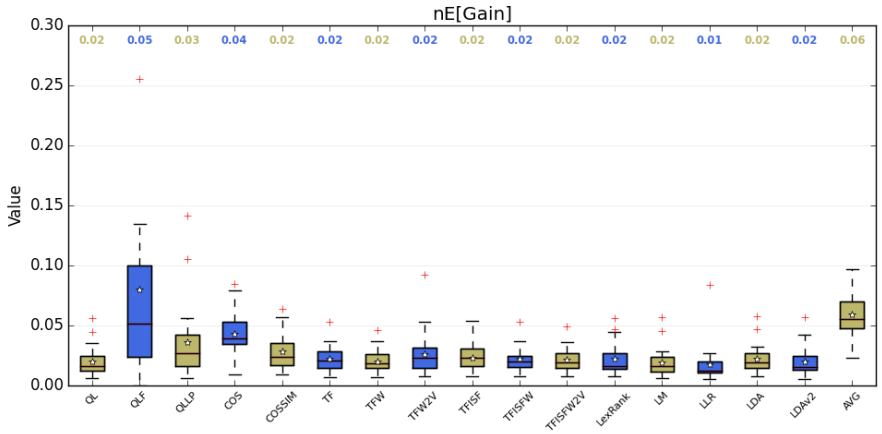

Fig. 2. Results for the normalized expected latency gain metric, i.e. i.e. the degree to which the updates within the summary are on-topic, novel and timely.

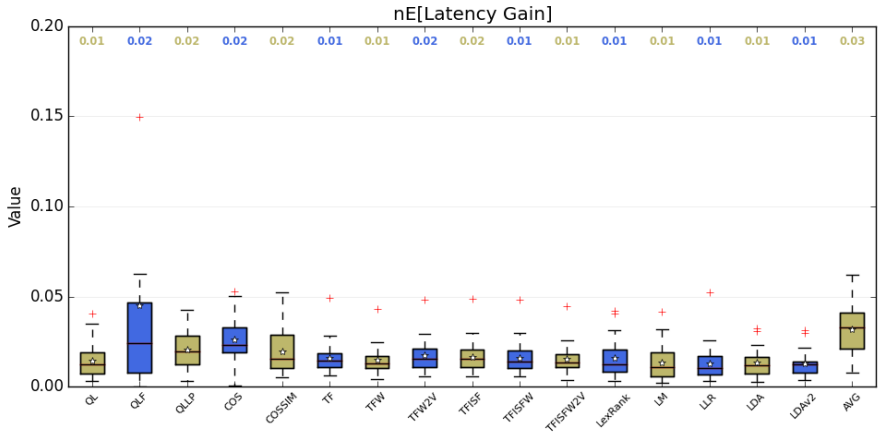

Fig. 3. Results for the comprehensiveness metric, i.e. how many nuggets the system covers. Comprehensiveness is similar to the traditional notion of recall in information retrieval evaluation.

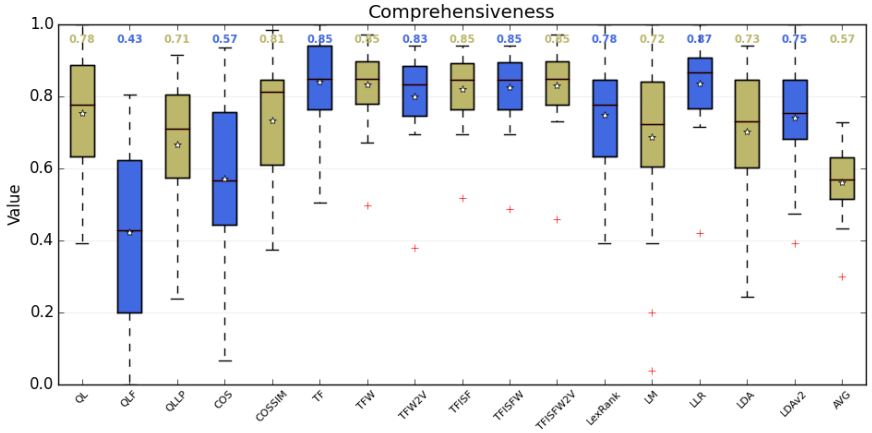

Fig. 4. Results for expected latency metric, i.e. the degree to which the information contained within the updates is outdated (a high value for latency denotes timely performance).

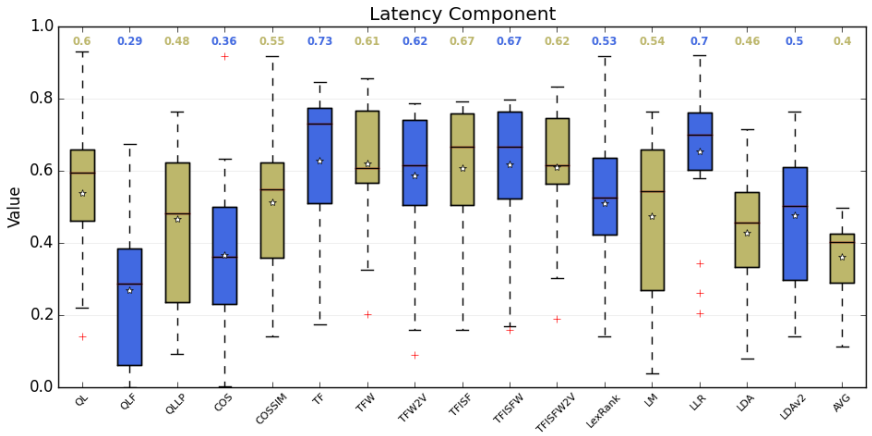

Fig. 5. Results for HM (nE[Latency Gain], Latency Component) - the harmonic mean of normalized Expected Latency Gain and Latency Comprehensiveness. This is the official target metric for the TREC Temporal Summarization 2015 track.