Sequential Clustering and Contextual Importance Measures for Incremental Update Summarization

Markus Zopf, Eneldo Loza Mencía, and Johannes Fürnkranz
Research Training Group AIPHES / Knowledge Engineering Group
Department of Computer Science, Technische Universität Darmstadt
Hochschulstraße 10, 64289 Darmstadt, Germany
{zopf@aiphes,eneldo@ke,juffi@ke}.tu-darmstadt.de

Abstract

Unexpected events such as accidents, natural disasters and terrorist attacks represent an information situation where it is crucial to give users access to important and non-redundant information as early as possible. Previous work uses either a fast but inaccurate pipeline approach or a precise but slow clustering approach. Instead, we propose to use sequential clustering for grouping information so that we are able to publish sentences at each time step. By doing so, we combine the best of both clustering and pipeline approaches and create a fast and precise real-time system. Experiments on the TREC Temporal Summarization 2015 shared task dataset show that our system achieves better results compared to the state-of-the-art.

1 Introduction

Events such as accidents, natural disasters and terrorist attacks provide an important and challenging information problem. Shortly after such an event has occurred, the information situation is usually unclear. Initially, only vague information about the event may become available, for example that an earthquake has occurred, but details such as magnitude, epicenter, and whether a tsunami has to be expected are not known at this early point in time. Such information becomes only available as the event develops over time. In such situations, it is crucial for responders, crisis management organizations, and victims to get information as soon as possible. However, it is impossible for humans to monitor *vast amount of textual information* contained in sources such as news articles, tweets, social media posts, forum discussions, and live blogs. In incremental update summarization (IUS) (McCreadie et al., 2014b) the *detection of important information in a timely manner* is a major challenge. Since the information sources are also *highly redundant*, detecting and filtering redundancy is a second important challenge in IUS.

In the past, two types of systems were used for IUC: pipeline systems (McCreadie et al., 2014a; McCreadie et al., 2015) and ordinary clustering systems (Kedzie et al., 2014; Zhao et al., 2014; Yingzhe Yao and Fan, 2015; McCreadie et al., 2015). Pipeline systems process document or sentence and decide about importance of information and novelty immediately and achieve therefore a good timeliness. However, they publish a large amount of unimportant information (low precision) or miss important information (low recall) since they do not make use of the redundancy of information as signal for importance, which is a usually a very good feature in newswire documents (Nenkova et al., 2006). Traditional clustering systems on the other hand are able to exploit redundancy and therefore produce better summaries according to precision and recall. Since they collect documents (for example all documents within one hour) before they decide whether or not to publish information, timeliness is a major issue of these approaches.

Instead of choosing between one of these architectures, we propose to use sequential clustering (Section 3.1) for incremental update summarization. With a sequential clustering, we are able to combine best of both worlds: clustering for redundancy avoidance to achieve high precision and recall and the ability to publish information at every time step to achieve a good timeliness. Hence, sequential clustering seems to be a natural fit for the task of IUS. In the sequential clustering, we jointly identify importance

This work is licenced under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons.org/licenses/by/4.0/

and redundancy by using contextual importance measures (Section 3.2). They are used to assign utility scores to clusters, sentences, and tokens depending on already selected information. An additional redundancy avoidance methodology, as used by the other systems, becomes therefore obsolete.

We show in an experimental evaluation by using data of the TREC 2015 shared task on temporal summarization (Aslam et al., 2015) that our system `SeqCluSum` is able to outperform state-of-the-art. A substantial improvement is achieved even though the computed scores can only be considered a lower bound on the performance since the original competition used a manual and individual evaluation which cannot be reconstructed without loss of precision.

2 Related Work

Traditional extractive multi-document summarization approaches (Nenkova and McKeown, 2011) extract unmodified sentences from source documents to produce a summary. Graph-based approaches (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Parveen and Strube, 2015) represent source documents as graph and use algorithms such as HITS (Kleinberg, 1999) and PageRank (Brin and Page, 2012) to find important information. Centroid-based summarization (Carbonell and Goldstein, 1998; Radev et al., 2000) systems estimate sentences importance by computing their centrality in the source documents. Similarly to these systems, our approach extracts unmodified sentences and uses centrality as a signal for importance. The systems above perform a retrospective summarization, meaning that the systems analyze all source documents at once independently from their publication date. In IUS however, this is not possible, since important information has to be published as soon as possible. Furthermore, the mentioned systems create summaries of fixed lengths. In IUS we observe a situation where it is not clear in advance how long a summary has to be for a proper summarization of an event. Standard extractive summarization systems are therefore not suited for IUS.

In update summarization (Dang and Owczarzak, 2008), a summary is presented to the systems in addition to source documents which contain new information. The task is to summarize the source documents without repeating information which is already contained in the summary. Since this task is very similar to extractive summarization, methods, which are successfully applied to extractive, were also applied to update summarization.

Research in incremental update summarization (McCreadie et al., 2014b) is strongly influenced by the TREC Temporal Summarization (TREC-TS) shared task (Guo et al., 2013). The TREC-TS 2014 shared task (Aslam et al., 2014) provided a high-volume, pre-filtered version of the TREC KBA 2014 dataset.[1] The best performing run (Kedzie et al., 2014) in the challenge according to the official target metric uses affinity propagation clustering. Zhao et al. (2014), best performing system according to expected latency gain, applies a query expansion and information retrieval step in addition to a k-means clustering and sentence selection step. McCreadie et al. (2014a) proposes a real-time summarization system which achieved best comprehensiveness scores. They use a processing pipeline to filter out irrelevant documents, to classify sentences according to their relevance, and to filter out redundant sentences.

In the TREC-TS 2015 shared task (Aslam et al., 2015) systems competing in the ”summarization only” subtask 3 were provided with a low-volume stream of documents.[2] The best system according to the official evaluation score in 2015 was proposed by Raza et al. (2015). The authors use BM25 for sentence scoring and redundancy avoidance based on cosine similarity values. McCreadie et al. (2015) focuses on entity importance and entity-entity interaction to identify important entities in an event. Kedzie et al. (2015) processes documents in hourly batches and combine salience prediction for sentences with affinity propagation clustering.

Our approach aims to combine the timeliness of a pipeline approach such as McCreadie et al. (2014a) with the importance detection strategy of a clustering approach such as Zhao et al. (2014).

[1]`http://trec-kba.org/kba-stream-corpus-2014.shtml`
[2]`http://dcs.gla.ac.uk/~richardm/TREC-TS-2015RelOnly.aws.list`

3 Approach

In this section, we propose sequential clustering in combination with contextual importance measures for incremental update summarization. The algorithm, `SeqCluSum`, consists of two subsystems, which are alternatingly applied to the documents in the stream. In the first subsystem, a sequential clustering algorithm clusters all sentences in a document according to similarity measures (Section 3.1). The second subsystem, a contextual importance measure, estimates scores for clusters (Section 3.2). Both subsystems work hand in hand in order to satisfy the three objectives to publish (i) *important information* while (ii) *avoiding redundancy* in a (iii) *timely manner*.

To detect important information, we make use of the property that important information occurs frequently in newswire data (Nenkova et al., 2006). We define the utility of a cluster as the sum of the utilities of the contained sentences the size of a cluster is an important factor for the cluster importance. By this definition, larger clusters tend to obtain higher utility scores than smaller clusters. Furthermore, we use normalized temporal TF-IDF scores in the contextual importance measure to estimate the utility of a sentence (cf. Section 3.2). Frequent terms in the input documents tend to obtain higher scores than infrequent terms. A cluster containing sentences with frequent terms will therefore obtain higher scores than a cluster with infrequent terms. We prevent publishing redundant information by allowing at most one update per cluster. This is reasonable since one cluster is assumed to represent one particular information which is not represented by another cluster. If a cluster is selected to publish a sentence, this cluster will be marked as *published* and will not be considered for publishing a sentence again. Since the system processes each document sequentially, it is able to publish sentences after each processing step which enables it to satisfy the third objective of timeliness. The pseudo-code of the complete system is shown in Algorithm 1. Line references are given in parenthesis with respect to this pseudo-code.

Algorithm 1 Sequential Clustering and Contextual Importance Measuring

```
    $documents$ = ordered list of documents, $clusters \leftarrow \emptyset$, $updates \leftarrow \emptyset$
 1: for each $document \in documents$ do
 2:     $document \leftarrow$ preprocess $(document)$                 ▷ remove boilerplate content and stem words
 3:     for each $sentence \in document$ do                           ▷ add each new sentences to a clusters
 4:         $nearestCluster = \arg\max_{c \in clusters}$ similarity$(sentence, c)$
 5:         if $clusters = \emptyset$ or (similarity$(sentence, nearestCluster) > \Theta$) then
 6:             $clusters \leftarrow clusters \cup \{\{sentence\}\}$;                      ▷ create a new cluster
 7:         else
 8:             $nearestCluster \leftarrow nearestCluster \cup sentence$                 ▷ add to nearest cluster
 9:         end if
10:     end for
11:     for each $cluster \in clusters$ do                        ▷ evaluate clusters and generate updates
12:         if $cluster$ is not published and $w(cluster) > \mu$ then
13:             $bestSentence = \arg\max_{s \in cluster} v(s)$                  ▷ find best sentence in $cluster$
14:             $updates \leftarrow updates \cup (document.timestamp, bestSentence)$
15:             set $cluster$ to $published$
16:         end if
17:     end for
18: end for
19: return $updates$
```

Before we apply our system for incremental update summarization to the document stream, we apply three preprocessing steps. We remove duplicate occurrences of web pages, utilize a boilerplate removal, and stem all words in the source documents with the well-known Porter-stemming algorithm (Porter, 1980).

3.1 Sequential Clustering

For the first part of our proposed incremental update summarizer, we adapt the sequential clustering algorithm (Theodoridis and Koutroumbas, 2009) to IUS. The algorithm iterates over all sentences in the currently processed document (line 3) and searches for the nearest existing cluster for each sentence using a similarity measure (line 5), which is described below. If the similarity to all existing clusters is lower than a fixed threshold Θ or no cluster has been created yet, a new cluster is created and the sentence is added to the new cluster (line 6). Otherwise, the sentence is added to the nearest existing cluster (line 8).

The similarity measure reduces the similarity between a sentence s and a cluster c to the similarity between the sentence s and the first sentence of the cluster c. This has several advantages in comparison to considering all sentences in the cluster. First, the center of the cluster is fixed when we compare only with the first sentence of the cluster. This prevents the cluster from a topic drift. Adding more and more sentences and also using the newly added sentences in the similarity calculations could instead change the initial notion of the cluster significantly since the center of the cluster could move. We observed that this can be a serious issue which leads to a poor clustering. Second, the approach is computationally efficient in comparison to an approach where we would compute the similarity by considering all sentences in a particular cluster. The number of maximal comparisons is reduced from the number of all sentences in a topic to the number of clusters created for a topic. Third, it emphasizes the notion of a cluster as a set of sentences, each of which contains the same information. The system chose the first sentences of each cluster to be its representative since each first sentence has a special role. This special role derives from the fact that the first sentence of each cluster was the reason why the cluster was created. The sentences did not fit to another cluster and were the reason for creating its own, new cluster.

The actual similarity measure (line 4+5) is based on Cosine similarity based on TF-IDF vectors (Salton and McGill, 1986). To calculate the TF-IDF vectors, we use a background corpus B created from 100,000 randomly sampled English Wikipedia articles in addition to all Wikipedia articles which are longer than 10,000 characters (5,794 documents). We use DKPro Similarity (Bär et al., 2013) to calculate the Cosine similarity. A detailed analysis of similarity measures is given in Section 5.2.

3.2 Measuring Importance

After the sentences in a document are clustered, the system evaluates the current cluster landscape to detect important information. We introduce the utility functions u, v, and w to measure the utility of tokens, sentences, and clusters, respectively. The score of a cluster c is measured with a cluster utility function w and equals to the sum of the scores of all sentences s_i in the cluster. We define $w(c) = \sum_{s_i \in c} v(s_i)$, where v is a utility function for sentences. Summing the scores of all sentences in a cluster addresses the property of the corpus that more important information is repeated more frequently in the source documents since larger clusters obtain higher utility scores (c.f. Section 1). The score of a sentence s is defined as the sum of the weights of the tokens t_i contained in sentence s. Thus, we define $v(s) = \sum_{t_i \in s} u(t_i)$, where u is a utility function for a token. In the following, we define a contextual importance measure to estimate the utility of a single token. Since we summarize an ongoing event and do not know which documents will appear later in the stream, we cannot compute TF-IDF scores over all documents. Nevertheless, we want to estimate how salient a token is relatively to the already observed documents D_τ at time τ. This will provide us with a signal about the relative relevance of the tokens very similarly to the well-known TF-IDF. To measure the importance of a token, we first define a context-free utility function u_{cf} in Equation 1. This gives us an impression on how salient a token is in a document collection D_τ. Since the document collection D_τ changes after each time step τ, the utility scores are constantly updated when new documents are processed. We define

$$u_{cf}(t, D_\tau) = \sum_{D \in D_\tau} \frac{n_D(t)}{|D|} \cdot log\left(\frac{|B|}{n_B(t)}\right) \tag{1}$$

where $n_B(t)$ denotes the number of occurrences of the token t in the background corpus B and $|B|$ refers to the total number of tokens in B. Unknown tokens, which do not appear in the background corpus (i.e. $n_B(t) = 0$), are ignored in the calculation of sentences importance. The contextual similarity measures defined in Equation 2 takes already published tokens into account when estimating the utility of a token. It captures therefore the importance of a token with respect to already published tokens. Hence, it avoids publishing sentences which are similar to already published sentences by discounting the scores for already published tokens and therefore provides redundancy avoidance implicitly. We discount the context-free values by dividing the context-free score $u_{cf}(t, D_\tau)$ by the number of occurrences of token

t in the already published updates. We define $u_c(t, D_\tau)$ in Equation 2, where $n_U(t)$ denotes the number of occurrences of token t in the current list of updates U.

$$u_c\left(t, D_\tau\right) = \frac{v_{cf}\left(t, D_\tau\right)}{1 + n_U(t)} \tag{2}$$

3.3 Publishing an Information Update

A cluster is considered as sufficiently important if a fixed threshold μ is exceeded (line 12). As a consequence, one sentence from the cluster is published (line 14). The threshold can easily be adapted on demand to produce more or less verbose summaries. If a cluster exceeds the threshold, the best sentence according to the sentence utility score v is published. The cluster is marked as *published* in this case (line 15), which means that the cluster will not be selected in the future.

4 Evaluation

In this section, we describe the evaluation of our approach. First, we provide detailed information about the used evaluation data in Section 4.1. The evaluation methodology is described in Section 4.2. We use a trace-driven simulation which simulates the course of events to evaluate our system. Results are presented in Section 6 after a detailed analysis of our system in Section 5.

4.1 Data

As evaluation dataset, we use the TREC-TS-2015RelOnly[3] dataset, which was created by the organizers of the 2015 Temporal Summarization shared task (Aslam et al., 2015). The corpus is a filtered version of the KBA Stream Corpus 2014[4], contains documents from various text genres like mainstream news articles, blogs/microblogs, and forum posts, and is divided into 21 clusters of documents. Each cluster represents one topic and has additional meta-data about the start and the end of the event as well as a query term indicating the topic of the event. Each document has sentence segmentation annotations and is guaranteed to contain at least one relevant sentence for the topic. All documents in the corpus have an associated timestamp which equals the crawling time of the document.

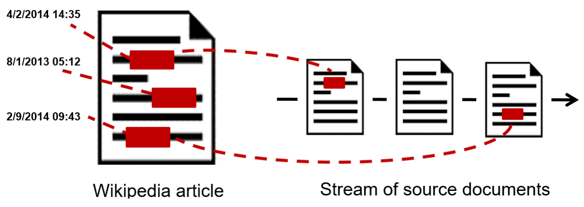

Figure 1: Illustration of the TREC-TS dataset. Information nuggets were retrieved from Wikipedia articles and annotated with timestamps at which this information became publicly available. Sentences in the source documents are matched against the extracted information nuggets.

For the evaluation in the shared task, the organizers extracted time-stamped information nuggets based on the Wikipedia article revision histories of the corresponding events. In a first step, information nuggets were extracted from the revision histories by the track organizers. The nuggets were classified with an importance score of 1, 2, or 3, and tagged with a timestamp, which represents the time when this information became publicly available. In a second step, sentences were pooled from each submission in the TREC-TS 2015 shared task. For each submission, at most 60 sentences were pooled per topic and were manually matched with the extracted information nuggets. A sentence can match multiple nuggets as well as no nugget. Figure 1 illustrates the annotation. Due to the huge annotation effort, only a small set of sentences in the corpus is labeled. This is problematic, since one cannot train, validate, or evaluate a system accurately without additional annotation effort.

[3]`http://dcs.gla.ac.uk/~richardm/TREC-TS-2015RelOnly.aws.list`
[4]`http://trec-kba.org/kba-stream-corpus-2014.shtml`

4.2 Methodology

In the task of IUS, the documents have to be processed in temporal order. The result of a system is a list of updates. When a system decides that a sentence from the stream should be published, this particular sentence is marked with the timestamp of the currently processed document according to Algorithm 2.

Algorithm 2 Incremental Update Summarization (IUS)

```
   $S$ = summarizing system; $documents$ = time-ordered list of documents; $updates = \emptyset$
1: for each $document \in documents$ do
2:     $t \leftarrow document.time$
3:     $S.process(document)$
4:     $updates_t \leftarrow S.getUpdates()$
5:     for each $u \in updates_t$ do $updates \leftarrow updates \cup \{(t, u)\}$ end for
6: end for
7: return $updates$
```

The systems are not allowed to use information from the future to make this decision. Incorporating information that only became available after the current timestamp is not allowed. One example for such an improper use of information would be the computation of TF-IDF values over the whole corpus since this would provide information about which words will become important in the future.

In our evaluation, we use the same metrics as in the TREC-TS 2015 shared task. The first metric is *normalized expected gain* $n\mathbf{EG}(\mathcal{S})$. This metric is comparable to precision since it measures whether a system publishes updates which are on-topic and novel. It will give a low score to systems which publish irrelevant or redundant information. The second metric $\mathbf{C}(\mathcal{S})$ measures the *comprehensiveness* of a summary. It is similar to recall since it measures how many of the information nuggets that could have been retrieved are covered by the summary. Systems which miss a large amount of important information will obtain a low score according to $\mathrm{C}(\mathcal{S})$. A third metric, the *expected latency* metric E[Latency], measures to which degree the information in the summary is outdated.[5] This metric reflects the requirement that systems are supposed to publish sentences as early as possible to produce the maximal benefit for the users. The later a system identifies important information, the lower its expected latency score. The final measure $\mathcal{H}$ combines the three individually measured properties precision, recall, and timeliness and served as official target measure for the task and is also our main metric to evaluate the systems. It computes the harmonic mean of a latency discounted version of $n\mathbf{EG}(\mathcal{S})$ and a latency discounted version of $\mathbf{C}(\mathcal{S})$. We refer to Aslam et al. (2015) for more detailed information about the metrics.

5 Analysis

In the following, we provide a detailed analysis of our system. We investigate the impact of three parts of our system: the boilerplate removal (Section 5.1), the used similarity measure (Section 5.2), and the clustering (Section 5.3).

5.1 Boilerplate Removal

In this section, we evaluate the boilerplate removal preprocessing step in terms of precision and recall. Although preprocessing is not part of the contribution of this paper, it has an impact on the system performance and is therefore discussed briefly. False positive errors made by the preprocessing cannot be corrected by the subsequent clustering and therefore limits the performance of the overall system irrevocably. It gives also an impression about the dataset itself which is otherwise hard to grasp. For the analysis, we use only non-duplicate documents (i.e. only the first version of each document). Since we do not have gold standard boilerplate removal data for the dataset, we use the nugget annotations instead. Table 1 provides the results of this analysis. Since both boilerplate systems work on the document level, we had to map the result of the boilerplate removal to single sentences. To do this, we used the Jaccard

[5]The name for the metric is a little counter-intuitive, since it used as a discount factor for nugget importance. A value of 1 means that the update is in-line with the Wikipedia article and results in no discount. Values smaller than 1 mean that an update was only published after it was included in the reference. Values bigger than 1 mean that a system published a nugget before it occurred in the Wikipedia article the first time.

Boilerplate system	Similarity	Thr	Max length	tp	tn	fp (critical)	fn
-	-	-	5	294,993	8,472	180	474,017
-	-	-	6	341,444	8,396	256	427,566
-	-	-	7	372,331	8,272	380	396,679
-	-	-	8	394,354	8,112	540	374,656
-	-	-	9	416,763	7,908	744	352,247
-	-	-	10	436,579	7,641	1,011	332,431
Kohlschütter et al. (2010)	Jaccard	0.6	5	684,059	4,266	4,386	84,951
Kohlschütter et al. (2010)	Jaccard	0.8	5	714,653	3,194	5,458	54,357
Kohlschütter et al. (2010)	Cosine	0.6	5	639,374	5,706	2,946	129,636
Kohlschütter et al. (2010)	Cosine	0.8	5	695,350	3,859	4,793	73,660
Habernal et al. (2016)	Jaccard	0.4	5	682,985	4,392	4,260	86,025
Habernal et al. (2016)	Jaccard	0.6	5	721,033	3,187	5,465	47,977
Habernal et al. (2016)	Cosine	0.6	5	686,671	4,340	4,312	82,339
Habernal et al. (2016)	Cosine	0.8	5	729,665	2,820	5,832	39,345

Table 1: We report the performance according to true positives (tp), true negatives (tn), false positives (fp), and false negatives (fn) of the length cutoff baseline and two boilerplate removal system. False positives (falsely pruned valuable sentences) are critical, since the pruned sentences are not processed by the following sequential clustering.

and the Cosine similarity. We tagged the sentence as boilerplate if it was less similar to the most similar sentence in the text retrieved by the boilerplate system according to a threshold and had a maximum length of 5. We observe that we can discard a lot of sentences with only a lengths criterion without too many false positives. Both boilerplate removal systems, Boilerpipe (Kohlschütter et al., 2010) and the tool used in Habernal et al. (2016), do not perform very well in combination with the two similarity measures since both generate a large number of false positives.

5.2 Similarity Measures

One key component of summarization systems are similarity measures which estimate the semantic similarity of two texts or sentences. We therefore performed experiments with different measures in the TREC-TS dataset. We evaluate the similarity measure by comparing the score of similar sentences, which contain exactly the same nuggets and for dissimilar sentences which do not share any nuggets. We define sets of sentence $ss_1, \dots ss_L$ according to the information nuggets contained in the sentences. Since each sentence can contain multiple nuggets, we first define nugget sets $ns_1, \dots, ns_M$ which represent all sets of nuggets similarly to label powersets in multi-label classification. Sentences in a particular sentence set ss_i contain exactly the same information nuggets, i.e. each sentence is contained in exactly one sentence set. For sets $ss_1, \dots ss_L$, and sentences $S = \{s_1, \dots, s_K\}$ we define the sim score of a similarity measure σ as the average similarity of all similar sentences

$$\mathrm{sim}(\sigma) = \sum_{l=1}^{L} \frac{1}{|ss_l|} \sum_{l_1,l_2=1,l_1<l_2}^{|ss_l|} \sigma(s_{l_1}, s_{l_2}) \tag{3}$$

and the dissim score of a similarity measure σ as the average similarity of all dissimilar sentence

$$\mathrm{dissim}(\sigma) = \sum_{l_1,l_2=1,l_1<l_2}^{L} \frac{1}{|ss_{l_1}| \cdot |ss_{l_2}|} \sum_{\substack{s_1 \in ss_{l_1} \\ s_2 \in ss_{l_2}}} \sigma(s_1, s_2). \tag{4}$$

An appropriate similarity measure can perform both, assigning high scores to similar sentences and low scores to dissimilar sentences. Therefore, we measure both by computing $\mathrm{diff}(\sigma) = \mathrm{sim}(\sigma) - \mathrm{dissim}(\sigma)$. We report the scores of various similarity measures in Table 2.

We observe that the TF-IDF-based Cosine similarity with $\log +1$ IDF weighting and L2 normalization performs best.[6] For details regarding the similarity measures, we refer to Bär et al. (2013). Cosine similarity based on the sum of GloVe (Pennington et al., 2014) word embeddings to represent a sentences

[6] The results provided in the table were computed for the complete dataset. Results on the splitted datasets, which were used for the evaluation of the system, confirmed the results.

		TF-IDF-based Cosine			Jaccard-n		Word embedding-based Cosine	
	random	log+1 L1	log+1 L2	binar L2	$n = 1$	$n = 2$	unweighted	weighted
$\mathrm{sim}(\sigma)$	0.50	0.16	0.41	0.40	0.29	0.19	0.79	0.69
$\mathrm{dissim}(\sigma)$	0.50	0.01	0.15	0.20	0.10	0.01	0.70	0.52
$\mathrm{diff}(\sigma)$	0.00	0.15	**0.26**	0.20	0.19	0.18	0.09	0.17

Table 2: Results of $\mathrm{sim}(\sigma)$ and $\mathrm{dissim}(\sigma)$ for different similarity measures.

yields the lowest results.[7] A weighted version, where we multiply the word vectors with the corresponding IDF score achieves better results than the unweighted Cosine based on the word embeddings.

5.3 Clustering

In this section, we evaluate the cluster landscape by computing cluster purity (Manning et al., 2008) and nugget concentration. Cluster purity measures how many different nugget sets $(ns_1, \ldots, ns_M)$ are contained in the clusters. According to the underling idea that one cluster represents one nugget set, one cluster should only contain sentences that belong to the same information nuggets. With the definition of a cluster as a set of sentences (i.e. $c \subset S$), the cluster purity of the clusters $C = \{c_1, \ldots c_I\}$, sentence sets $ss_1, \ldots ss_L$, and K sentences (Manning et al., 2008) is defined as

$$\mathrm{purity}(C) = \frac{1}{K} \sum_{i=1}^{I} \max_{l} |c_i \cap ss_l| \tag{5}$$

Similarity to purity, we also define a purity purity^+ where we do not consider the sentence set which contains all sentences without any nuggets as majority class in $\max_l |c_i \cap ss_l|$. This provides a better insight into the mixture of non-empty nugget sets. In a perfect clustering, we get a score of 1, i.e. that all clusters contain only sentence which belong to a particular nugget set. Since a purity of 1 is easy to achieve with K clusters, we introduce the term nugget concentration (nc), which is defined as

$$\mathrm{nc} = \frac{1}{L} \sum_{l=1}^{L} \frac{\max_i |c_i \cap ss_l|}{|ss_l|} \tag{6}$$

Boilerplate system	Θ	# cluster	purity	purity$^+$	nc	nc (after bpr)
max length 8	1.0	1	0.4874	0.1779	0.5134	1.0000
max length 8	0.9	563	0.5759	0.4310	0.4780	0.8914
max length 8	0.8	800	0.6261	0.5151	0.4745	0.8791
max length 8	0.7	893	0.6764	0.6191	0.4717	0.8670
max length 8	0.6	950	0.7259	0.7019	0.4718	0.8648
Kohlschütter et al. (2010) + Cosine < 0.6	1.0	1	0.4466	0.1923	0.4729	1.0000
Kohlschütter et al. (2010) + Cosine < 0.6	0.9	229	0.5328	0.4336	0.4473	0.9038
Kohlschütter et al. (2010) + Cosine < 0.6	0.8	390	0.5784	0.5186	0.4402	0.8871
Kohlschütter et al. (2010) + Cosine < 0.6	0.7	534	0.6613	0.6316	0.4422	0.8780
Kohlschütter et al. (2010) + Cosine < 0.6	0.6	618	0.7360	0.7361	0.4404	0.8693
Habernal et al. (2016) + Cosine < 0.8	1.0	1	0.4406	0.1815	0.3853	1.0000
Habernal et al. (2016) + Cosine < 0.8	0.9	107	0.5299	0.4294	0.3671	0.9291
Habernal et al. (2016) + Cosine < 0.8	0.8	189	0.6058	0.5519	0.3675	0.9253
Habernal et al. (2016) + Cosine < 0.8	0.7	254	0.7078	0.6909	0.3663	0.9195
Habernal et al. (2016) + Cosine < 0.8	0.6	323	0.7826	0.7900	0.3558	0.9219

Table 3: Results of the cluster evaluation for difference boilerplate versions and difference values of Θ.

We use for the analysis the TF-IDF-based Cosine similarity with $log + 1$ IDF weighting and L2 normalization (according to Section 5.2). Since the boilerplate removal can have a significant impact, we report the results based on different boilerplate removal versions. In Table 3 we see that more clusters (due to a lower Θ) lead to a higher purity as expected. Nugget concentration decreases, because it becomes more likely that one nugget set is distributed across multiple clusters. If we only consider

[7]We used pre-calculated word embeddings from `http://nlp.stanford.edu/projects/glove`

sentences which passed the boilerplate removal step (column nc (after bpr)) we see that the system performs best with the stricter boilerplate removal. This can be explained due to the fact that the nugget sets are smaller and a distribution across multiple clusters is less likely. If we consider all sentences in the data, the system based on the first boilerplate removal performs best (column nc). A strict boilerplate removal with many false positives harms the overall performance of the systems. However, processing fewer sentences in the rather computational expensive clustering (compared to the boilerplate removal) leads to better computational performance. A better boilerplate removal would therefore be desirable to improve the performance in IUS.

6 Results

We present in Table 4 the evaluation scores of our system as well as the official evaluation scores of subtask 3 of the TREC-TS 2015 challenge. The results for our systems are computed with the original evaluation script provided by the TREC-TS organizers. The evaluation results of the other systems are taken from the overview paper of the TREC-TS shared task (Aslam et al., 2015). Due to per-task normalization, metric values across the different subtasks at TREC-TS are not comparable. We only provide a lower bound evaluation of our system, generated by only using the annotations provided by the TREC-TS organizers. As described in Section 4.1, only a small subset of sentences in the corpus is annotated. All non-annotated sentences are considered to be unimportant by the evaluation script, which means that valuable sentences might be misinterpreted as noise when using this dataset without additional annotations. This disadvantage does not apply to the reference systems, since at least the top 60 sentences of each reference systems were labeled for each topic. In total, 57.99% of the sentences selected by our system were not annotated. We report this number for a better assessment of the performance of our system. Although our system might get penalized for valuable sentences, we outperform the other approaches. As described in Section 1, we expect that a sequential clustering approach should be able to detect important information very early. The timeliness of our approach (column E[Latency]) confirms this expectation and is an important factor for the superior performance of our system. We also observe that our system is able to generate a reasonable balance of precision ($n\mathbf{EG}(\mathcal{S})$) and recall ($\mathbf{C}(\mathcal{S})$).

System	$\mathcal{H}$	E[Latency]	$F_1(n\mathbf{EG}(\mathcal{S}), \mathbf{C}(\mathcal{S}))$	$n\mathbf{EG}(\mathcal{S})$	$\mathbf{C}(\mathcal{S})$
`SeqCluSum` (lower bound)	**0.1526**	**0.8013**	**0.1842**	0.1485	0.2426
Raza et al. (2015)	0.0853	0.3983	0.1773	0.1840	0.1710
McCreadie et al. (2015)	0.0639	0.5335	0.1189	0.0667	0.5459
McCreadie et al. (2015)	0.0508	0.6741	0.0758	0.0402	0.6590

Table 4: System results sorted by descending $\mathcal{H}$, the main metric used in the TREC-TS shared task. The columns $n\mathbf{EG}(\mathcal{S})$, $\mathbf{C}(\mathcal{S})$, and E[Latency] show the results according to evaluation metrics described in Section 4. Since the main metric is computed by building a harmonic mean of latency discounted versions of $n\mathbf{EG}(\mathcal{S})$ and $\mathbf{C}(\mathcal{S})$ there is no point in achieving high scores in one of these measures while achieving only a low score according to the other metric.

To find reasonable values for the parameters Θ and μ, we manually evaluated our system with a cross-validation. We did this by randomly splitting the topics in two evenly sized sets of topics and used one of the sets as test set while using the other one for parameter optimization. As described in Section 4.1, we did this on the basis of a sparsely labeled dataset. We expect better results if we could use more densely labeled data. We use the Cosine similarity with $\log +1$ IDF weighting and L2 normalization (cf. Section 5.2). We could not determine a superior boilerplate removal in Section 5.1 and Section 5.3 and therefore apply (Kohlschütter et al., 2010) since it is considered to be a well-known standard boilerplate removal system. During the cross-validation, we found $\mathcal{H}$ scores of 0.1664 and 0.1858 in the validation sets which resulted in $\mathcal{H}$ scores of 0.1550 and 0.1501 in the test sets. We report the averaged scores of both test sets in Table 4.

7 Discussion

The sequential clustering depends strongly on the accuracy of the used similarity measure. Since we only use predefined standard similarity measures, we assume that a more sophisticated selection of similarity measures could lead to an improvement of the performance. More abstract semantic methods or additional knowledge resources may model the semantic similarity of words and sentences better. We therefore assume that the performance of the similarity and redundancy detection can be further improved.

Another improvement of the similarity measure would be to utilize the weights of the tokens, which are used during the contextual measuring of importance also in the clustering. If we weight important tokens higher in the similarity measure, we could achieve that the clusters would be more focused on a particular information nugget.

The clustering can also be improved by introducing a re-clustering step, which could split a cluster into multiple clusters when the system detects that one cluster contains too diverse sentences. This is currently prevented by the parameter Θ. By enabling the system to execute a re-clustering step, this parameter could become obsolete. Furthermore, the system could merge multiple clusters to one cluster if it detects that the content of the clusters are more similar as the seed sentence suggested.

Our model relies heavily on centrality as a features to detect important information timely. Kedzie et al. (2015) mentioned that using centrality as signal for importance is problematic in IUS since it requires some time until an important information is central enough in the news stream. However, the burstiness of news streams can diminishes this issue. A system which is independent from the centrality features would nevertheless be able to detect new important information in big data streams earlier than our systems.

To evaluate our system, we used a cross-validation to optimize the parameters μ and Θ in validation sets and applied the values to test sets. This limits the performance of our system, since the topics have different sizes and different amounts of interesting information. Therefore, an extension of our system where the parameters μ and Θ are adaptive could further improve the performance. μ for example, could depend on the time since no update has been published. If there is a long time span with no updates, μ could be lowered to increase the probability that there will be a new message soon.

8 Conclusions

In this paper, we proposed `SeqCluSum`, a system for incremental update summarization based on sequential clustering and contextual importance measures. It combines the strength of the two prior proposed techniques, namely real-time processing pipelines or clustering approaches. The sequential clustering arranges similar sentences together while the contextual importance measures score the clusters and sentences according to their contextual importance. The contextual importance measures estimate the importance of information as well as their novelty jointly. The evaluation shows that even the lower bound scores of our system outperforms previous state-of-the-art systems. We obtain better latency and higher $\mathcal{H}$ scores, which means that our system detects important information earlier and with a better trade-off between precision and recall. We therefore conclude that sequential clustering in combination with contextual importance measuring is well-suited for the task of IUS. Furthermore, our system does not use the provided query and is hence able to detect important information without using query expansion, which was often used by previous systems to improve recall. We do not use any handcrafted knowledge bases to get additional domain-knowledge, which is another advantage of our system.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

References

Javed Aslam, Fernando Diaz, Richard McCreadie, and Tetsuya Sakai. 2014. TREC 2014 temporal summarization track overview. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2014)*. National Institute of Standards and Technology (NIST), November.

Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. 2015. TREC 2015 temporal summarization track overview. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2015)*. National Institute of Standards and Technology (NIST), November.

Daniel Bär, Torsten Zesch, and Iryna Gurevych. 2013. DKPro Similarity: An open source framework for text similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 121–126, Sofia, Bulgaria, August. Association for Computational Linguistics.

Sergey Brin and Lawrence Page. 2012. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833.

Jaime Carbonell and Jade Goldstein. 1998. The use of DKPro Similarity, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR 1998)*, pages 335–336. ACM.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 update summarization task. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Günes Erkan and Dragomir R Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Qi Guo, Fernando Diaz, and Elad Yom-Tov. 2013. Updating users about time critical events. In *Proceedings of the 35th European Conference on Advances in Information Retrieval (ECIR 2013)*, pages 483–494. Springer Berlin Heidelberg.

Ivan Habernal, Omnia Zayed, and Iryna Gurevych. 2016. C4Corpus: Multilingual web-size corpus with free license. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 914–922. European Language Resources Association (ELRA), May.

Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2014. Summarizing disasters over time. In *Proceedings of the Twenty-Third Text REtrieval Conference*.

Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2015. Predicting salient updates for disaster summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL 2015)*, pages 1608–1617, July.

Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, pages 441–450.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Richard McCreadie, Romain Deveaud, M-Dyaa Albakour, Stuart Mackie, Nut Limsopatham, Craig Macdonald, Iadh Ounis, Thibaut Thonet, and Bekir Taner Dinçer. 2014a. University of Glasgow at TREC 2014: Experiments with terrier in contextual suggestion, temporal summarisation and web tracks. In *Proceedings of the Twenty-Third Text REtrieval Conference (TREC 2014)*. National Institute of Standards and Technology (NIST), November.

Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2014b. Incremental update summarization: Adaptive sentence selection based on prevalence and novelty. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 301–310. ACM.

Richard McCreadie, Stuart Mackie, Jarana Manotumruksa, Graham McDonald, Saúl Vargas, M-Dyaa Albakour, Craig Macdonald, and Iadh Ounis. 2015. University of glasgow at TREC 2015: Experiments with terrier in contextual suggestion, temporal summarisation and dynamic domain tracks. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2015)*. National Institute of Standards and Technology (NIST), November.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 404–411. Association for Computational Linguistics, July.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3):103–233.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 573–580. ACM.

Daraksha Parveen and Michael Strube. 2015. Integrating importance, non-redundancy and coherence in graph-based extractive summarization. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1298–1304. AAAI Press.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, pages 21–30. Association for Computational Linguistics.

Ahsan Raza, Devin M. Rotondo, and Charles L. A. Clarke. 2015. WaterlooClarke: TREC 2015 temporal summarization track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2015)*. National Institute of Standards and Technology (NIST), November.

Gerard Salton and Michael J McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill.

Sergios Theodoridis and Konstantinos Koutroumbas, 2009. *Pattern Recognition*, chapter 12, pages 633–634. Elsevier Ltd, Oxford.

Zhen Yang Yingzhe Yao and Kefeng Fan. 2015. BJUT at TREC 2015 temporal summarization track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2015)*. National Institute of Standards and Technology (NIST), November.

Yun Zhao, Fei Yao, Huayang Sun, and Zhen Yang. 2014. BJUT at TREC 2014 temporal summarization track. In *Proceedings of the Twenty-Third Text REtrieval Conference*.