Multi-TimeLine Summarization (MTLS): Improving Timeline Summarization by Generating Multiple Summaries

Yi Yu[1], **Adam Jatowt**[2], **Antoine Doucet**[3]
Kazunari Sugiyama[1], **Masatoshi Yoshikawa**[1]
[1]Kyoto University, Japan
[2]University of Innsbruck, Austria, [3]University of La Rochelle, France
yuyi@db.soc.i.kyoto-u.ac.jp
adam.jatowt@uibk.ac.at, antoine.doucet@univ-lr.fr
{kaz.sugiyama, yoshikawa}@i.kyoto-u.ac.jp

Abstract

In this paper, we address a novel task, *Multiple TimeLine Summarization* (MTLS), which extends the flexibility and versatility of TimeLine Summarization (TLS). Given any collection of time-stamped news articles, MTLS automatically discovers important yet different stories and generates a corresponding timeline for each story. To achieve this, we propose a novel unsupervised summarization framework based on the two-stage affinity propagation process. We also introduce a quantitative evaluation measure for MTLS based on the previous TLS evaluation methods. Experimental results show that our MTLS framework demonstrates high effectiveness and MTLS task can provide better results than TLS.

1 Introduction

Nowadays, online news articles are one of the most popular Web documents. However, due to a huge amount of news articles available online, it is getting difficult for users to effectively search, understand, and track the entire news stories. To solve this problem, a research area of TimeLine Summarization (TLS) has been established, which can alleviate the redundancy and complexity inherent in news article collections thereby helping users better understand the news landscape.

After the influential work on temporal summaries by Swan and Allan (2000), TLS has attracted researchers' attention. Most of works on TLS (Martschat and Markert, 2018; Steen and Markert, 2019; Gholipour Ghalandari and Ifrim, 2020) have focused on improving the performance of summarization. However, their drawbacks are as follows: (a) the methods work essentially on a homogeneous type of datasets such as ones compiled from the search results of an unambiguous query (e.g., "BP Oil Spill"). The requirements imposed on the input dataset make it hard for TLS systems to generalize; (b) the output is usually a single timeline regardless of the size and the complexity of the input dataset.

We propose here the Multiple TimeLine Summarization (MTLS) task that enhances and further generalizes TLS. MTLS automatically generates a set of timelines that summarize disparate yet important stories, rather than always generating a single timeline as is in the case of TLS. An effective MTLS framework should: (a) detect key events including both short- and long-term events, (b) link events related to the same story and separate events belonging to other stories, and (c) provide informative summaries of constituent events to be incorporated into the generated timelines.

MTLS can also help to deal with the ambiguity, which is common in information retrieval. For example, suppose that a user wants to get an overview of news about a basketball player, *Michael Jordan*, from a large collection of news articles. However, when a search engine over such a collection takes "Michael Jordan" as a query, it would likely return documents constituting a mixture of news about different persons having the same name. Then, how can a typical TLS system return meaningful results if only a single timeline can be generated? Similarly, ambiguous queries such as "Apple", "Amazon", "Java" require MTLS solutions to produce high quality results.

To address this task, we further propose a Two-Stage Affinity Propagation Summarization framework (2SAPS). It uses temporal information embedded in sentences to discover important events, and their linking information latent in news articles to construct timelines. 2SAPS has several advantages: firstly, it is entirely unsupervised which is especially suited to TLS-related tasks as there are very few gold summaries available for training supervised systems; secondly, both the number of events and the number of generated timelines are

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 377–387
August 1–6, 2021.

self-determined. This allows our framework to be dependent only on the input document collection, instead of on human efforts.

Furthermore, the current TLS evaluation measures allow only 1-to-1 comparison (system- to human-generated timeline), which is not suitable for MTLS task where multiple timelines must be compared to (typically) multiple ground-truth timelines. Therefore, we also propose a quantitative evaluation measure for MTLS based on the adaptation of the previous TLS evaluation framework.

Given these points, our contributions in this work are summarized as follows:

1. We propose a novel task (MTLS), which automatically generates multiple, informative, and diverse timelines from an input time-stamped document collection.
2. We introduce a superior MTLS model that outperforms all TLS-adapted MTLS baselines.
3. We design an evaluation measure for MTLS systems by extending the original TLS evaluation framework.

2 Related Work

2.1 Timeline Summarization

Since the first work on timeline summarization (Swan and Allan, 2000; Allan et al., 2001), this topic has received much attention over the years (Alonso et al., 2009; Yan et al., 2011a; Zhao et al., 2013; Tran et al., 2013; Li and Li, 2013; Suzuki and Kobayashi, 2014; Wang et al., 2016; Takamura et al., 2011; Pasquali et al., 2019, 2021). In the following, we review the major approaches.

Chieu and Lee (2004) constructed timeline by directly selecting the top ranked sentences based on the summed similarities within n-day long window. Yan et al. (2011b) proposed *evolutionary timeline summarization* (ETS) to return the evolution trajectory along the timeline, consisting of individual but correlated summaries of each date. Shahaf et al. (2012) created information maps (Maps) to help users understand domain-specific knowledge. However, the output consists of a set of storylines that have intersections or overlaps, which is not appropriate for a dataset that may contain quite different topics. Nguyen et al. (2014) proposed a pipeline to generate timelines consisting of date selection, sentence clustering and sentence ranking.

Recently, Martschat and Markert (2018) adapted a submodular function model for TLS task, which is originally used for multi-document summarization (MDS). Duan et al. (2020) introduced the task of *Comparative Timeline Summarization* (CTS), which captures important comparative aspects of evolutionary trajectories in two input sets of documents. The output of the CTS system is, however, always two timelines generated in a contrastive way. Then, Gholipour Ghalandari and Ifrim (2020) examined different TLS strategies and categorized TLS frameworks into the following three types: *direct summarization approaches*, *date-wise approaches*, and *event detection approaches*.

To the best of our knowledge, the idea of multiple timeline summarization has not been formally proposed yet. Table 1 compares the related tasks.

2.2 Timeline Evaluation

Some works (Yan et al., 2011b; Chen et al., 2019; Duan et al., 2020) evaluate timeline by only computing ROUGE scores (Lin, 2004). This way ignores the temporal aspect of a timeline, which is important in timeline summarization. Martschat and Markert (2017) then proposed a framework, called *tilse*, to assess timelines from both textual and temporal aspects. Subsequently, TLS works (Steen and Markert, 2019; Gholipour Ghalandari and Ifrim, 2020; Born et al., 2020) have followed this framework to evaluate their models. Some researches (Tran et al., 2015; Shahaf et al., 2012; Alonso and Shiells, 2013) also involved user studies, in which users are required to score system-generated timelines based on varying criteria such as relevance and understandability. In Section 5, we will adapt the *tilse* framework to MTLS task.

3 Problem Definition

We formulate MTLS task as follows:

Input: A time-stamped news article collection $\mathcal{D} = \{d_1, d_2, ..., d_{|\mathcal{D}|}\}$. The collection can be standalone or compiled from search results returned by a news search engine.

Output: A set of timelines, $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ is generated based on $\mathcal{D}$, so that each timeline T_i includes a sequence of time/date[1] and summary pairs $(t_1^{T_i}, s_1^{T_i}), \ldots, (t_l^{T_i}, s_l^{T_i})$ where $s_j^{T_i}$ $(i = 1, \ldots, k)$ are the summary sentences for the time $t_j^{T_i}$ $(j = 1, \ldots, l)$ and l is the length of T_i. Each timeline in $\mathcal{T}$ should be consistent and coherent, yet different from other timelines.

[1] In this paper, time and date are used as synonyms.

Tasks	Output 1 timeline	Output $\geq$ 2 timelines	Automatically Determine k	Input Heterogeneous Collection	Quantitatively Evaluate
TLS (Most of which in Section 2.1)	✓				✓
CTS (Duan et al., 2020)		✓ (always 2)			✓
ETS (Yan et al., 2011b)	✓				✓
Maps (Shahaf et al., 2012)		✓		✓	
MTLS (Proposed task)	✓	✓	✓	✓	✓

Table 1: Comparison between different TLS related tasks (k is the number of generated timelines).

We note that while the traditional TLS task is limited as a document collection for it is typically coherent and homogeneous, MTLS is more flexible as the input news collection can be diverse. For example, the input collection can be generated using a search query q composed of multiple entities or concepts like $q = \{\text{“}egypt\text{”}, \text{“}h1n1\text{”}, \text{“}iraq\text{”}\}$ or by using an ambiguous query like $q = \{\text{“}michael\text{”}, \text{“}jordan\text{”}\}$, or it can also consist of news articles crawled over a certain time span from multiple news sources. Generally, the more heterogeneous $\mathcal{D}$ is, the more timelines could be produced. The intuition behind this idea is that users will need more structured information to help them understand a relatively complex document collection.

4 Framework

Next, we present two key components of our framework: event generation module (Sec. 4.1) and timeline generation module (Sec. 4.2).

We first make the following two assumptions:

Assumption 1: *News articles sometimes retrospectively mention past events for providing necessary context to the target event, for underlying continuation, causality, etc.*

Assumption 2: *Sentences mentioning similar dates have higher probability to refer to the same event than sentences with different dates.*

4.1 Event Generation Module

In this module, we extract important historical events from a document collection. Gholipour Ghalandari and Ifrim (2020) constructed events by simply grouping articles with close publication dates into clusters, resulting in lower accuracy. Note that Assumption 1 implies that a single news article may contain multiple events. Accordingly, in our work, the concept of event is more fine-grained. We define *event* as a set of sentences that describe the same real-world occurrence, typically using the same identifying information (e.g., actions, entities, locations). This information is captured by sentence-BERT (Reimers and Gurevych, 2019): a pre-trained model on a transformer network where similar meanings are positioned nearby in semantic vector space. We then employ Affinity Propagation (AP) (Frey and Dueck, 2007) following Steen and Markert (2019) for clustering similar sentences. AP algorithm groups data points by selecting a set of exemplars along with their followers due to message passing. It operates over an affinity matrix $\boldsymbol{S}$, where $S(i,j)$ denotes similarity between data points x_i and x_j.

We observe that high semantic similarity does not always guarantee that sentences refer to the same event. Especially, for some periodic events, similar happenings might have occurred several times. For example, a news article could include sentences reporting that Brazil won the gold medal in the World Cup (in 2002) while some other sentences in this document could recall that Brazil has won the first place in the World Cup in 1994. It is clear that those sentences describe two distinct events, which would be grouped into one *event* if only semantic similarity is considered.

Therefore, based on Assumption 2, we introduce another key factor, temporal similarity, which enhances the confidence of how likely two sentences will refer to the same event. We define each element $S_1(v_i, v_j)$ of affinity matrix $\boldsymbol{S_1}$ as follows:

$$S_1(v_i, v_j) = \alpha_1 \cdot S_{date}(t_i, t_j) + (1 - \alpha_1) \cdot S_{cos}(v_i, v_j), \tag{1}$$

where v_i and v_j denote different sentences, and t_i and t_j denote dates mentioned by v_i and v_j, respectively.[2] In addition, S_{date} and S_{cos} denote the temporal and semantic similarities, respectively. While we employ cosine similarity for the semantic similarity, we define temporal similarity $S_{date}(i,j)$ to quantify how similar two dates are using Equation (2):

$$S_{date}(t_i, t_j) = \frac{1}{\exp^{\gamma \cdot |t_i - t_j|}}, \tag{2}$$

where γ[3] is the decay rate of the exponential func-

[2] We use Heideltime (Strötgen and Gertz, 2013) for resolving temporal expressions. If a sentence does not explicitly mention any date, we assume it refers to the publication date of the article.

[3] We set $\gamma = 0.05$ in the experiments.

tion. The larger the time gap between two dates, the smaller the value of S_{date}.

By passing messages of both semantic and temporal information between sentences, clusters consisting of exemplar and non-exemplar sentences are constructed to form the candidate *event* set E. Each cluster represents an *event*.

Event Selection. In a timeline, it is not necessary to show all events of a story as users usually care about the most important events only. We design an *event* selection step that is helpful for handling excessive number of *events*. The selection relies on two measures: *Salience* and *Consistency* defined by Equations (3) and (4), respectively:

$$Salience(e) = \frac{log(\mid e \mid)}{log(\mid \mathcal{D} \mid)}, \quad (3)$$

$$Consistency(e) = \frac{\sum_{v_i \in e, v_i \neq v_e} S_{cos}(v_i, v_e)}{\mid e \mid -1}, \quad (4)$$

where v_e is the exemplar sentence in *event* e; $\mid e \mid$ and $\mid \mathcal{D} \mid$ denote the number of sentences in e and document collection $\mathcal{D}$, respectively.

Intuitively, important historical events would often be mentioned by future news reports. *Salience* of *event* is used to evaluate such importance and is computed as the relative frequency of sentences about that *event* compared with all sentences in the collection. On the other hand, *Consistency* ensures high quality of *events*. We then rank all candidate *events* based on the weighted summed score of these two measures. Hereafter, we denote the weight of *Event Salience* as ζ_1 and that of *Event Consistency* as $1 - \zeta_1$.

We select the top-scored *events* obtaining a new *event* set E^* by setting a threshold. To avoid tuning its value, we set the value to one standard deviation from the mean (lower end).

4.2 Timeline Generation Module

While TLS systems directly link all the identified events, MTLS requires their deeper understanding. As described in Section 1, an effective MTLS framework should link events related to the same story and separate other unrelated events to different timelines. To achieve this, we explain the following steps in this module: *Event Linking*, *Timeline Selection*, and *Timeline Summarizing*.

Event Linking. According to Assumption 1, current events can refer to related past events. We thus define a reference matrix $\boldsymbol{R}$, in which each element $R(e_i, e_j)$ denotes the degree of reference between two *events* e_i and e_j. As *events* in our work are represented by sentences and a sentence belongs to a single *event*, $R(e_i, e_j)$ can be reflected by counting patterns of sentence co-occurrences in documents. Formally, $R(v_i, v_j)$ represents the case where two sentences v_j and v_i refer to each other as defined by Equation (5):

$$R(v_i, v_j) = \begin{cases} 1 & v_i, v_j \in d \wedge v_i \in e_k, v_j \in e_l, e_k \neq e_l \\ 0 & \text{otherwise,} \end{cases} \quad (5)$$

where d is an article, e_k and e_l are elements in E^*.

The degree of reference between e_i and e_j is then defined as follows:

$$R(e_i, e_j) = \frac{\sum_{v_1 \in e_i} \sum_{v_2 \in e_j} R(v_1, v_2)}{\mid e_i \mid \cdot \mid e_j \mid}, \quad (6)$$

where $|e_i|$ and $|e_j|$ are sizes of e_i, e_j, respectively.

We then construct a graph of *events* where each node is an $e \in E^*$, and the value of an edge reflects the connection degree between a pair of two *events*. We reuse AP algorithm to detect the community of *events* over the affinity matrix $\boldsymbol{S_2}$ defined by Equation (7):

$$S_2(e_i, e_j) = \alpha_2 \cdot R(e_i, e_j) + (1 - \alpha_2) \cdot S_{cos}(e_i, e_j), \quad (7)$$

where $S_{cos}(e_i, e_j)$ denotes cosine similarity between e_i and e_j to capture semantic similarity. Based on the affinity matrix $\boldsymbol{S_2}$, AP finally generates clusters, i.e., the initial timeline set, $\mathcal{T}$.

Timeline Selection. In order to ensure the quality of constructed timelines, we define criteria to select high-quality timelines from $\mathcal{T}$. Similar to *event selection* described in Section 4.1, we also use two indicators to evaluate the quality of a timeline. We define *Timeline Salience* as the average score of *Event Salience* of all events within the timeline, and *Timeline Coherence* as the average of semantic similarity scores between any chronologically[4] adjacent *events* defined by Equation (8):

$$Coherence(T) = \frac{\sum_{e_i, e_{i+1} \in T} S_{cos}(e_i, e_{i+1})}{\mid T \mid -1}, \quad (8)$$

where $\mid T \mid$ is the size of a timeline, i.e., the number of *events* in this timeline.

Intuitively, important timelines, which reflect important stories in the document collection, are more likely to be preferred by users. *Timeline Salience* captures this importance by passing the importance of its components (i.e., *events*), while *Timeline Coherence* ensures that the story expressed by the timeline is consistent.

[4]The time of an event e is given by its exemplar sentence.

We rank timelines based on a weighted sum of *Timeline Salience* and *Timeline Coherence*. The weight of *Timeline Salience* is denoted as ζ_2; thus the weight of *Timeline Coherence* is $1-\zeta_2$. We then select the top-scored elements from the timeline set $\mathcal{T}$ based on a threshold. Same as before, we set the value to one standard deviation from the mean.

Timeline Summarizing. By previous steps, we have now obtained multiple timelines $\{T_1, T_2, ...\}$, where T is a list of *events* $\{e_1, e_2, ...\}$. However, it is not feasible to show all contents of each e as it usually contains many sentences. We use only the exemplar sentence in *event* since exemplar is the most typical and representative member in the group.

In addition, it is possible that two *events* e_i and e_j occur on the same day. In this case, we concatenate their exemplar sentences.

Timeline Tagging. This step is an add-on to MTLS systems. To better understand the stories of constructed timelines, we believe that it should be helpful for users to also obtain a *label* for each timeline. As described in Section 1, the input document collection may be composed of different topics or of one topic discussed through different aspects. For example, among the timelines generated based on the topic *syria*, one timeline might summarize the story about *Syrian civil war* while another might be about *Syrian political elections*. A label should then help people understand the story of the timeline. We simply select the 3 most frequent words among *events* (excluding stopwords) for each timeline as its label.

5 Evaluation Framework

5.1 TLS Evaluation

TLS evaluation relies on ROUGE score and its variants as follows:

Concatenation-based ROUGE (*concat*). It considers only the textual overlap between concatenated system summaries and ground-truth, while ignoring all date information of timeline (Yan et al., 2011b; Nguyen et al., 2014; Wang et al., 2016).

Date-agreement ROUGE (*agreement*). It measures both textual and temporal information overlap by computing ROUGE score only when the date in the system-generated timelines matches the one of the ground-truth timeline (Tran et al., 2013). Otherwise, its value is 0.

Alignment-based ROUGE. It linearly penalizes the ROUGE score by the distances of dates or/and summary contents. Martschat and Markert (2017) proposed three types of this metric: ***align***, ***align+***, ***align+m:1*** (align by date, align by date and contents, align by date and contents where the map function is non-injective, respectively).

Date selection (*d-select*). It evaluates how well the model works in selecting correct dates in the ground-truth (Martschat and Markert, 2018).

5.2 MTLS evaluation

The evaluation methods for TLS cannot directly assess the performance of MTLS systems as there are multiple output timelines and multiple ground-truth timelines. Concretely, given an input collection $\mathcal{D}$, corresponding ground-truth timeline set $\boldsymbol{\mathcal{G}} = \{G_1, G_2, ...G_{k_1}\}$ $(k_1 \geq 1)$, and system-generated timeline set $\mathcal{T} = \{T_1, T_2, ..., T_{k_2}\}$ $(k_2 \geq 1)$, evaluation metrics need information to automatically "match" the ground-truth timeline when evaluating T_i. Therefore, we make the system find the closest ground-truth G^* to timeline T as follows:

$$G^* = \underset{G \in \mathcal{G}}{\arg\max} f_m(T, G), \tag{9}$$

where f_m is the TLS evaluation function to compute the score between T and G based on metric m, which can be either *concat*, *agreement*, *align*, *align+*, *align+m:1*, or *d-select*. Then, the overall performance of the MTLS models is computed by taking the average of all the members in $\mathcal{T}$.

6 Experimental Setup

The goal of our experiments is to answer the following research questions (RQs):

RQ1: Do MTLS models produce more meaningful output than TLS models?

RQ2: How does 2SAPS framework perform on MTLS task compared with other MTLS baselines?

RQ3: How effective are the components of the modules in 2SAPS? How do parameter changes in the model affect the results?

6.1 Datasets

We note that there is no available dataset for MTLS task, thus we construct MTLS datasets[5] extending existing TLS datasets. Tran *et al.* released *Timeline17* (Binh Tran et al., 2013) and *Crisis* (Tran et al., 2015) datasets for TLS over news articles.

[5]The datasets are now available at https://yiyualt.github.io/mtlsdata/.

Name	#Topics	#Groundtruth	Avg.Timespan	#Docs.	#Sents.
Timeline17	9	17	250 days	4,650	183,782
Crisis	4	22	343 days	9,242	331,044

Table 2: Statistics on TLS datasets.

L=1	$\mathcal{D}_1$:egypt $\mathcal{D}_2$:finan $\mathcal{D}_3$:haiti $\mathcal{D}_4$:h1n1 $\mathcal{D}_5$:libya
L=2	$\mathcal{D}_6$:egypt+libya $\mathcal{D}_7$:haiti+iraq $\mathcal{D}_8$:h1n1+haiti $\mathcal{D}_9$:finan+mj $\mathcal{D}_{10}$:egypt+mj
L=3	$\mathcal{D}_{11}$:egypt+h1n1+iraq $\mathcal{D}_{12}$:finan+iraq+syria $\mathcal{D}_{13}$:egypt+ iraq+mj $\mathcal{D}_{14}$:finan+h1n1+mj $\mathcal{D}_{15}$:finan+libya+mj
L=4	$\mathcal{D}_{16}$:egypt+finan+haiti+iraq $\mathcal{D}_{17}$:finan+h1n1+ iraq+mj $\mathcal{D}_{18}$:h1n1+haiti+iraq+mj $\mathcal{D}_{19}$:finan+ h1n1+haiti+mj $\mathcal{D}_{20}$:egypt+haiti+iraq+mj
L=5	$\mathcal{D}_{21}$:finan+h1n1+haiti+iraq+mj $\mathcal{D}_{22}$:h1n1+haiti+iraq+mj+syria $\mathcal{D}_{23}$:egypt+finan+haiti+mj+syria $\mathcal{D}_{24}$:egypt+finan+ iraq+mj+syria $\mathcal{D}_{25}$:egypt+finan+h1n1+haiti+mj

Table 3: MTLS datasets used for our experiments.

Table 2 shows their statistics. To assure high complexity of data, we generate multiple datasets from TLS datasets by varying degree of story mixtures. We construct MTLS datasets based on combining TLS datasets, according to the following procedure: (1) set the number of topics L used to generate a new dataset; (2) from TLS datasets, randomly choose L topics, then merge their document collections into a new dataset $\mathcal{D}$ along with grouping their associated ground-truth timelines into $\mathcal{G}$.[6] (3) repeat steps (1) and (2). Here, the value of L reflects the complexity of the dataset. The more topics the dataset contains, the more complex it is.

We repeated the steps (1)~(3) on *Timeline17*[7] and finally created 25 datasets as shown in Table 3. *Timeline17* contains 9 document collections, covering the following topics: "BP Oil Spill" (bpoil), "Influenza H1N1" (h1n1), "Michael Jackson death" (mj), "Libyan War" (libya), "Egyptian Protest" (egypt), "Financial Crisis" (finan), "Haiti Earthquake" (haiti), "Iraq War" (iraq), "Syrian Crisis" (syria).

6.2 Baselines

As there are no ready models for MTLS task, we design the baselines as "divide-and-summarize" approaches. The underlying idea is: first segment the input dataset into sub-datasets (subsequently called segments) by partition/division algorithms; then adopt TLS techniques to generate a timeline for each sub-dataset (segment). We now describe the choices for each step.

Dataset Division Approaches:

- Random. We randomly decide the number of segments from 1 to 10. Then, we assign a news article to a random segment.
- LDA (Latent Dirichlet Allocation) (Blei et al., 2003). Given a dataset, we first use LDA to detect the main topics in the dataset. Then, we assign each news article to its dominant topic.
- K-means (MacQueen et al., 1967). We use k-means algorithm in *scikit-learn*.[8]

TLS Approaches:

- CHIEU2004 (Chieu and Lee, 2004): It is a frequently used unsupervised TLS baseline which selects the top-ranked sentences based on summed similaries within n-day window.
- MARTSCHAT2018 (Martschat and Markert, 2018): It is one of the state-of-the-art TLS models and is also the first work to establish formal experimental settings for TLS task. We use the implementation given by the authors.[9]
- GHALANDARI2020 (Gholipour Ghalandari and Ifrim, 2020): It constructs timeline by first predicting the important dates via a simple regression model and then selecting important sentences for each date.[10]

We combine the above 3 dataset division approaches and 3 TLS approaches and thus yield 9 baselines.

6.3 Experimental Settings

Concerning the characteristics of MTLS task and our datasets, the experimental settings differ from the TLS settings applied in Martschat and Markert (2018). In particular, the settings are:

- When generating timelines, none of the compared models knows the actual value of L (i.e., L is not an input data). The stratification given in Table 3 is shown only for the reader to explain the datasets' construction method.

[6] If a topic has multiple ground-truth timelines, we pick one that has length closest to the average length of the timelines for that topic.

[7] We note that *Crisis* contains only 4 topics, resulting in few possible combinations, so we finally decided to skip it.

[8] https://scikit-learn.org/

[9] https://github.com/smartschat/tilse.

[10] https://github.com/complementizer/news-tls.

- For the dataset-division algorithms, LDA and k-means, we use different techniques to find optimal number of segments. For LDA, we evaluate topic coherence measure (C_v score) (Röder et al., 2015) for topic numbers ranging from 1 to 10, and then choose the optimal number. For k-means, we use silhouette value (Rousseeuw, 1987) to determine the optimal number of segments.

- All the compared methods do not take the information of the ground-truth as input. That is, the number of dates, the average number of summary sentences per date, the total number of summary sentences, the ground-truth start dates, and end dates are all unknown.

- We set the length of timelines to 20 and summary length to 2 sentences per date.

7 Results and Discussion

7.1 MTLS vs. TLS

We first address RQ1 to show the necessity of MTLS and to demonstrate that TLS performs poorly when an input dataset contains mixture of documents on different stories. To achieve this, we compare results of MTLS baselines with a standard TLS approach. Table 4 shows the performance comparison between TLS and MTLS baselines based on MARTCHAT2018. For fair comparison in this first experiment, we select only one timeline from MTLS outputs that is most similar to the timeline generated by TLS. We observe that when $L = 1, 2$, MTLS underperforms TLS by 15.1%, 4.8% in terms of *align+m:1* ROUGE-1, respectively. However, it outperforms TLS by 150%, 117.1%, and 94.7% when L equals 3,4,5, respectively. This indicates that as the complexity of input document collection increases (higher L values), TLS systems do not produce good results when compared to MTLS ones. In real world scenarios, it is rather rare that the input dataset is clean enough to contain only a single topic. Thus, these results suggest that MTLS approach should in practice be more useful than TLS. The results for the other two TLS algorithms introduced in Section 6.2 show a similar trend, too. Furthermore, the example outputs of TLS and MTLS systems are also available as supplementary materials.

7.2 Performance of 2SAPS

We now investigate the performance of our framework to answer RQ2. Table 5 shows the overall performance of MTLS systems. We observe that 2SAPS achieves the best performance in terms of all ROUGE metrics. In particular, when compared with CHIEU2004, MARTSCHAT2018 and GHALANDARI2020 in terms of *concat* ROUGE-1 score, it outperforms them by 52.9%, 12.2%, and 16.4%, respectively. We also observe that GHALANDARI2020 method still achieves the best performance among baselines except for *concat* ROUGE-1. Furthermore, it is worth noticing that k-means works best in dividing datasets. On average, k-means outperforms Random and LDA by 15% and 7.2%, respectively, in terms of *concat* ROUGE-1. Finally, compared with the best-performing baseline, k-means-GHALANDARI2020, our 2SAPS outperforms it by 9.9%, 15.1%, 0%, 10%, 4.7%, 3.6%, 19.1%, in terms of *concat* (ROUGE-1,ROUGE-2), *align+m:1* (ROUGE-1,ROUGE-2), *agreement* (ROUGE-1,ROUGE-2) and *d-select*, respectively.

7.3 Ablation Study

We turn to the first part of RQ3. We conduct ablation tests on *Event Selection* (ES) and *Timeline Selection* (TS) components. Table 6 shows the changes of different models. We observe that without ES, *d-select* and *align+m:1* ROUGE-2 scores decrease 14.6% and 42.2% compared with 2SAPS. The plausible reason is that without ES, many unimportant dates and events are included in a timeline, resulting in low recall of correct dates. On the other hand, without TS component, the generated timeline set tends to contain noisy timelines, causing low ROUGE-1 as the performance drops by 18.8%.

7.4 Parameter Impact

We now analyze the impact of key parameters, α_1, α_2, ζ_1, ζ_2. α_1 and α_2 directly influence the quality of generated *events* and timelines, while ζ_1 and ζ_2 indirectly affect the model's performance by controlling the selection steps. Figure 1 shows the performance of 2SAPS under *concat* ROUGE-1, *align+m:1* ROUGE-1, and *agreement* ROUGE-1.

In particular, we observe that: a smaller value of α_1 (from 0.1 to 0.4) gives better results than a larger value (Figure 1a). When α_1 turns to 1, AP algorithm does not converge, and the values of all measures become 0. The plausible reason for this could be that when sentence dates are very

Model	Metric	L=1	L=2	L=3	L=4	L=5
TLS (MARTSCHAT2018)	concat (ROUGE-1)	**0.287**	0.310	0.214	0.261	0.202
	concat (ROUGE-2)	**0.061**	0.069	0.038	0.044	0.035
	align+m:1 (ROUGE-1)	**0.053**	**0.063**	0.032	0.041	0.038
	align+m:1 (ROUGE-2)	**0.011**	**0.017**	0.011	0.007	0.007
MTLS (k-means-MARTSCHAT2018)	concat (ROUGE-1)	0.272	**0.364**	0.362	**0.400**	**0.390**
	concat (ROUGE-2)	0.056	**0.084**	0.085	**0.100**	**0.084**
	align+m:1 (ROUGE-1)	0.046	**0.063**	**0.082**	**0.097**	**0.082**
	align+m:1 (ROUGE-2)	0.009	0.014	0.026	**0.034**	**0.024**
MTLS (LDA-MARTSCHAT2018)	concat (ROUGE-1)	0.274	0.332	**0.363**	0.335	0.273
	concat (ROUGE-2)	0.054	0.074	**0.089**	0.079	0.059
	align+m:1 (ROUGE-1)	0.043	0.057	0.078	0.080	0.065
	align+m:1 (ROUGE-2)	0.007	0.009	**0.027**	0.024	0.018

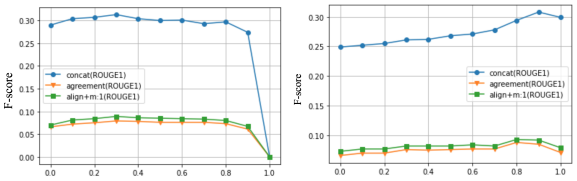
Table 4: Performance comparison between TLS and MTLS systems. For fair comparisons, we compare the single timeline generated by TLS model with the most related timeline generated by MTLS models.

MTLS Methods		**concat**		**align+m:1**		**agreement**		**d-select**
		ROUGE-1	ROUGE-2	ROUGE-1	ROUGE-2	ROUGE-1	ROUGE-2	F1
Baselines								
CHIEU2004	Random	0.191	0.027	0.019	0.004	0.010	0.002	0.075
	LDA	0.192	0.035	0.023	0.005	0.013	0.004	0.089
	k-means	0.229	0.046	0.027	0.006	0.014	0.004	0.096
MARTSCHAT2018	Random	0.254	0.049	0.044	0.009	0.037	0.007	0.352
	LDA	0.289	0.068	0.062	0.017	0.052	0.015	0.387
	k-means	0.291	0.071	0.061	0.017	0.051	0.015	0.376
GHALANDARI2020	Random	0.253	0.048	0.068	0.015	0.058	0.013	0.414
	LDA	0.268	0.062	0.085	0.025	0.076	0.024	0.440
	k-means	0.284	0.073	**0.096**	0.030	0.085	0.028	0.467
Our method								
2SAPS		**0.312**	**0.084**	**0.096**	**0.033**	**0.089**	**0.029**	**0.556**

Table 5: Overall performance obtained by the baselines and the proposed methods over D_1 ~D_{25} datasets.

	d-select	ROUGE-1	ROUGE-2
2SAPS w/o ES	0.475	0.085	0.019
2SAPS w/o TS	0.502	0.078	0.023
2SAPS	0.556	0.096	0.033

Table 6: Ablation results of 2SAPS model, showing changes of *align+m:1* ROUGE and *d-select* F1 scores.



(a) α_1: Temporal similarity (b) α_2: Reference relation

(c) ζ_1: Event salience (d) ζ_2: Timeline salience

Figure 1: Impact of parameters on F1 score.

close, the elements of transition matrix differ only slightly, resulting in non-convergence.

Figure 1b shows the impact of the reference relation in linking *events*. The values of all metrics increase as α_2 increases. It makes sense that reference relation exerts an important role in linking *events* into timelines, thus a higher value is necessary. However, when α_2 is over 0.9, the performance drops because when news articles provide few contextual *events* (e.g., background events, related events, etc.), then the reference relation between *events* becomes unreliable.

ζ_1 controls the impact of *Event Salience* described in Section 4.1. Another corresponding factor is *Event Consistency*, which is weighted by 1-ζ_1. Figure 1c shows that the model with larger values of ζ_1 underperforms the ones with relatively small values of ζ_1 (from 0.2 to 0.4), indicating that consistency of *event* matters more than its salience in selecting high-quality *events*. Finally, in Figure 1d, we observe that along with the increase of ζ_2, the performance of all metrics decrease, suggesting that the coherence of timeline is more effective than salience in selecting good timelines.

7.5 Limitations

Our 2SAPS model works essentially on the unit of sentences and constructs a graph where each sentence is a node and edge is the relation between

sentences. It has then a complexity of $O(n^2)$. Future work could address this by simplifying graph structure and providing approximate solutions to cover also the cases of processing large datasets. Another solution is to select only important sentences from news articles using the combination of classification, summarization or filtering.

8 Conclusions

We introduced MTLS task to generalize the timeline summarization problem. MTLS improves the performance of timeline summarization by generating multiple summaries. We conducted experiments to first show that given a heterogeneous time-stamped news article collection, TLS usually does not produce satisfactory result. We further proposed 2SAPS, a two-stage clustering-based framework, to effectively solve MTLS task. Furthermore, we extended TLS datasets to MTLS datasets, as well as introduced a novel evaluation measure for MTLS. Experimental results show that 2SAPS outperforms MTLS baselines which follow the "divide-and-summarize" strategy. Our work significantly improves the generalization ability of timeline summarization and can provide users with easier access to news collections. As an unsupervised approach that does not require costly training data, it can be applied to any potential datasets and languages.

In future work, we plan to test our approach on additional MTLS datasets. We will also investigate scenarios in which MTLS can enhance information retrieval systems operating over news article collections. For users searching over large temporal collections, structuring the returned results into a series of timelines could prove beneficial, instead of returning a usual list of interwoven documents that relate to different stories or periods.

Acknowledgments

We greatly appreciate the authors in CoNLL'18 paper (Martschat and Markert, 2018) for making their data public. In particular, we wish to thank Sebastian Martschat for his great support in discussions about the experiment setup and reproduction. We also want to thank anonymous reviewers for their invaluable feedback.

References

James Allan, Rahul Gupta, and Vikas Khandelwal. 2001. Temporal Summaries of New Topics. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*, pages 10–18.

Omar Alonso, Michael Gertz, and Ricardo Baeza-Yates. 2009. Clustering and Exploring Search Results Using Timeline Constructions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*, pages 97–106.

Omar Alonso and Kyle Shiells. 2013. Timelines as Summaries of Popular Scheduled Events. In *Proceedings of the 22nd International Conference on World Wide Web (WWW '13)*, pages 1037–1044.

Giang Binh Tran, Mohammad Alrifai, and Dat Quoc Nguyen. 2013. Predicting Relevant News Events for Timeline Summaries. In *Proceedings of the 22nd International Conference on World Wide Web (WWW '13)*, pages 91–92.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Leo Born, Maximilian Bacher, and Katja Markert. 2020. Dataset Reproducibility and IR Methods in Timeline Summarization. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC'20)*, pages 1763–1771.

Xiuying Chen, Zhangming Chan, Shen Gao, Meng-Hsuan Yu, Dongyan Zhao, and Rui Yan. 2019. Learning Towards Abstractive Timeline Summarization. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19)*, pages 4939–4945.

Hai Leong Chieu and Yoong Keok Lee. 2004. Query Based Event Extraction Along a Timeline. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04)*, pages 425–432.

Yijun Duan, Adam Jatowt, and Masatoshi Yoshikawa. 2020. Comparative Timeline Summarization via Dynamic Affinity-Preserving Random Walk. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI'20)*, pages 1778–1785.

Brendan J Frey and Delbert Dueck. 2007. Clustering by Passing Messages Between Data Points. *Science*, 315(5814):972–976.

Demian Gholipour Ghalandari and Georgiana Ifrim. 2020. Examining the State-of-the-Art in News Timeline Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL'20)*, pages 1322–1334.

Jiwei Li and Sujian Li. 2013. Evolutionary Hierarchical Dirichlet Process for Timeline Summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, pages 556–560.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics (ACL'04)*, pages 74–81.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

Sebastian Martschat and Katja Markert. 2017. Improving Rouge for Timeline Summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL'17)*, pages 285–290.

Sebastian Martschat and Katja Markert. 2018. A Temporally Sensitive Submodularity Framework for Timeline Summarization. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CONLL'18)*, pages 230–240.

Kiem-Hieu Nguyen, Xavier Tannier, and Véronique Moriceau. 2014. Ranking Multidocument Event Descriptions for Building Thematic Timelines. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1208–1217.

Arian Pasquali, Ricardo Campos, Alexandre Ribeiro, Brenda Santana, Alípio Jorge, and Adam Jatowt. 2021. TLS-Covid19: A New Annotated Corpus for Timeline Summarization. In *Proceedings of the 43rd European Conference on Information Retrieval (ECIR 2021)*, pages 497 – 512.

Arian Pasquali, Vítor Mangaravite, Ricardo Campos, Alípio Mário Jorge, and Adam Jatowt. 2019. Interactive System for Automatically Generating Temporal Narratives. In *Proceedings of the 41st European Conference on Information Retrieval (ECIR 2019)*, pages 251–255.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP 2019)*, pages 3982–3992.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the Space of Topic Coherence Measures. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining (WSDM '15)*, pages 399–408.

Peter J Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. 2012. Trains of Thought: Generating Information Maps. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*, pages 899–908.

Julius Steen and Katja Markert. 2019. Abstractive Timeline Summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization (NewSum'19)*, pages 21–31.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and Cross-Domain Temporal Tagging. *Language Resources and Evaluation*, 47(2):269–298.

Satoko Suzuki and Ichiro Kobayashi. 2014. On-line Summarization of Time-Series Documents Using a Graph-Based Algorithm. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing (PACLIC'14)*, pages 470–478.

Russell Swan and James Allan. 2000. Automatic Generation of Overview Timelines. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '00)*, pages 49–56.

Hiroya Takamura, Hikaru Yokono, and Manabu Okumura. 2011. Summarizing a Document Stream. In *Proceedings of the 33rd European Conference on Information Retrieval (ECIR 2011)*, pages 177–188.

Giang Tran, Mohammad Alrifai, and Eelco Herder. 2015. Timeline Summarization From Relevant Headlines. In *Proceedings of the 37th European Conference on Information Retrieval (ECIR 2015)*, pages 245–256.

Giang Binh Tran, Tuan A Tran, Nam-Khanh Tran, Mohammad Alrifai, and Nattiya Kanhabua. 2013. Leveraging Learning to Rank in an Optimization Framework for Timeline Summarization. In *Proceedings of SIGIR 2013 Workshop on Time-aware Information Access (#TAIA'13)*.

William Yang Wang, Yashar Mehdad, Dragomir Radev, and Amanda Stent. 2016. A Low-Rank Approximation Approach to Learning Joint Embeddings of News Stories and Images for Timeline Summarization. In *Proceedings of the 15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 58–68.

Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. 2011a. Timeline Generation Through Evolutionary Trans-temporal Summarization. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, pages 433–443.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011b. Evolutionary Timeline Summarization: a Balanced Optimization Framework via Iterative Substitution. In *Proceedings of the 34th international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*, pages 745–754.

Xin Wayne Zhao, Yanwei Guo, Rui Yan, Yulan He, and Xiaoming Li. 2013. Timeline Generation with Social Attention. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '13)*, pages 1061–1064.