Abstractive Timeline Summarization

Julius Steen and **Katja Markert**
Department of Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
(steen|markert)@cl.uni-heidelberg.de

Abstract

Timeline summarization (TLS) automatically identifies key dates of major events and provides short descriptions of what happened on these dates. Previous approaches to TLS have focused on extractive methods. In contrast, we suggest an abstractive timeline summarization system. Our system is entirely unsupervised, which makes it especially suited to TLS where there are very few gold summaries available for training of supervised systems. In addition, we present the first abstractive oracle experiments for TLS. Our system outperforms extractive competitors in terms of ROUGE when the number of input documents is high and the output requires strong compression. In these cases, our oracle experiments confirm that our approach also has a higher upper bound for ROUGE scores than extractive methods. A study with human judges shows that our abstractive system also produces output that is easy to read and understand.

1 Introduction

Many newsworthy events are not isolated incidents but part of long-lasting developments. For example, the events of the Syrian civil war in 2019 are intrinsically linked to events that happened during the beginning of that war in 2011. As the amount of reporting grows, it can be difficult to keep track of important events that may have happened a long time ago. Timeline summarization (TLS) alleviates this problem by providing users with automatically generated timelines that identify key dates in a larger development along with short summaries of the events on these dates. Table 1 shows an example of a timeline.

Prior TLS systems are *extractive*, i.e. they identify important sentences in a corpus and copy them directly to the timeline (Nguyen et al., 2014; Chieu and Lee, 2004; Yan et al., 2011b,a; Wang et al., 2016; Tran et al., 2015, 2013b,a; Martschat and Markert, 2018). However, TLS aggregates information from input corpora that are orders of magnitude larger than for traditional multi-document summarization (MDS) tasks. In addition, documents typically come from many different sources. In this setting, it might be advantageous to generate *abstractive* summaries that combine information from different sentences. While the state of the art in abstractive summarization is achieved by neural networks (Celikyilmaz et al., 2018), these systems require many document/gold summary pairs for training. TLS datasets, on the other hand, have many input documents, but only contain very few gold-standard timelines (between 19 and 22) (Tran et al., 2015, 2013b). Thus, very few input/gold timeline pairs are available for training.

2011-03-15
First protests after calls on Facebook for a "Day of Dignity."
2011-08-18
US President Barack Obama and his allies urge Assad to quit. Western and Arab states later impose sanctions on his regime.
2011-10-02
Creation of the opposition Syrian National Council SNC.

Table 1: Beginning of an example timeline about the Syrian civil war. (Source: Crisis dataset (Tran et al., 2015))

We therefore introduce an *unsupervised abstractive TLS system* that is inspired by the abstractive MDS system in Banerjee et al. (2015). We make the following contributions:

1. We introduce the first abstractive system for TLS and show that it outperforms current ex-

Proceedings of the 2nd Workshop on New Frontiers in Summarization, pages 21–31
Hong Kong, China, November 4, 2019.

tractive TLS systems such as Martschat and Markert (2018) when the input corpora are large with low compression rate.[1]

2. We show that our system delivers significantly better performance than an abstractive neural model not adapted for TLS.

3. We conduct the first abstractive oracle experiments for TLS. Our abstractive approach improves the ROUGE upper bound on large corpora with low compression rate.

A human evaluation confirms that our system outputs readable sentences. Our system does not need any supervision and only requires lightweight preprocessing. This makes it easy to adapt to other languages. The source code for our system is available online.[2]

2 Task

2.1 Definition

We follow the formalization of TLS of Martschat and Markert (2018). Given a collection of news documents D about the topic for the timeline (such as the *Syrian civil war*), we seek to generate a timeline that summarizes the most important events related to the topic in D. The timeline is a sequence of dates $d_1, \ldots d_n$ and their associated daily summaries $v_1, \ldots v_n$. As in most prior work, we require that $d_1, \ldots d_n$ refer to a specific day.

We constrain the maximum number of dates that may be included in the timeline, the maximum number of sentences or tokens per daily summary, and the time span the timeline is supposed to cover. We discuss how we set these constraints in Section 4.2.

2.2 Differences to MDS

While both TLS and Multi-Document Summarization (MDS) generate summaries from multiple input documents, there are substantial differences between the two tasks. Specifically, Martschat and Markert (2018) cite the following differences:

1. MDS does not have a temporal dimension.

2. Typical MDS datasets do not require systems to summarize multiple events instead focusing on non-event topics or singular events. Even where MDS systems are evaluated on corpora with multiple events, evaluation does not consider the temporal dimension.

3. TLS corpora are larger than MDS corpora with lower compression rates, making content selection and scalability more important.

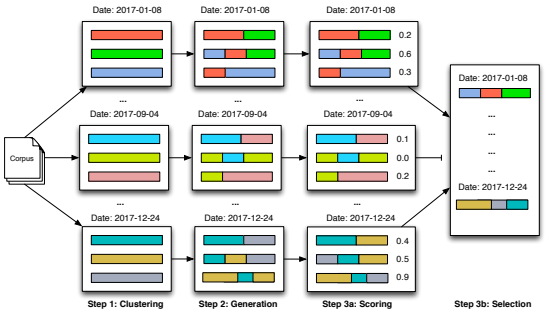

Figure 1: A graphical overview of our system. We can see that not all clusters are included in the timeline.

3 Architecture

We generate timelines in a three step process, outlined in Figure 1. We first cluster sentences that are likely to describe the same event. We then use Multi-Sentence-Compression (MSC) to generate candidate sentences to summarize each cluster. Finally, we score the candidates and select the best ones up to a length limit. Each of our steps is completely unsupervised, which allows us to sidestep the lack of training data in TLS and also makes our system readily adaptable to different datasets.

3.1 Clustering

We need to cluster sentences that describe the same event (such as the formation of the Syrian national council in Table 1) so that the MSC system can generate concise summaries from the resulting clusters. We use Affinity Propagation (AP) clustering (Frey and Dueck, 2007) for this purpose. AP is able to automatically determine the appropriate number of clusters for a dataset. This is advantageous, as different inputs contain different numbers of events. By choosing the number of clusters dynamically, our system can adapt to that without supervision.

AP selects a set of exemplars from the input data points, which can be understood as the centers of the clusters. Non-exemplar points select one of the exemplars to form a cluster with. The algorithm operates over an affinity matrix A, where A_{ij} expresses the appropriateness of item i pick-

[1] In summarization, a low compression rate means that a long input must be condensed to a short summary.

[2] github.com/julmaxi/Abstractive-Timeline-Summarization

ing item j as an exemplar. The diagonal of the matrix A, the so-called *preference values*, determines how suitable an item is to become an exemplar and thus regulates the number of exemplars.

We construct A using TF-IDF vector cosine similarity between the input sentences, which has been shown to be a useful similarity metric for TLS (Martschat and Markert, 2018; Chieu and Lee, 2004). However, sentences in the same cluster should not only be similar but also describe the same dates. To determine which date a sentence refers to, we make the following assumptions:

- Every sentence can refer to the document creation time (DCT).
- Sentences with one or more time expressions can refer either to one of the dates in the expressions or to the DCT
- Time expressions that refer to a range of days, such as a month, may refer to any date within that range.

The set of possible references for a sentence s is called $dates(s)$. A date reference d_1 *contains* another reference d_2 if one of the following holds:

1. d_1, d_2 refer to the same exact day.
2. d_1 refers to a range of dates which contains d_2, and d_2 is an exact date.

A sentence s_2 may select a sentence s_1 as an exemplar if there is a $d_1 \in dates(s_1)$ and a $d_2 \in dates(s_2)$ so that d_2 contains d_1. We set $A_{ij} = \cos(\vec{s_i}, \vec{s_j})$, if s_i may select s_j, and $A_{ij} = -\infty$ otherwise. Preference values are the median of incoming similarities (Frey and Dueck, 2007).

This procedure can still form "incorrect" clusters. If an exemplar sentence contains two or more incompatible date references d_1, d_2, the resulting cluster can contain sentences tagged with only d_1 or only d_2. However, this is an infrequent problem as sentences need to be similar to be clustered.

To determine the date of the event a cluster C describes, we let

$$date(C) = \underset{d \in \bigcup_{s \in C} \{dates(s) | d \text{ exact}\}}{\arg\max} cnt(C, d) \quad (1)$$

where $cnt(C, d)$ is the frequency of d being mentioned as a time expression in the sentences in C.

3.2 Sentence Generation

Following Banerjee et al. (2015), we use the unsupervised, low-cost MSC-system by Filippova (2010) to generate summary candidates for each cluster.

Given the sentence cluster C, the algorithm constructs a word-adjacency graph. The nodes are POS-tagged tokens and directed edges indicate adjacency of these tokens in one of the sentences. Occurrences of the same content word in different sentences are mapped to the same node. Given an edge e_{ij}, its weight $w(e_{ij})$ is:

$$w(e_{ij}) = \frac{freq(i) + freq(j)}{freq(i) * freq(j) * \sum_{s \in C} diff(s, i, j)} \quad (2)$$

where $freq(i)$ is the number of tokens that have been mapped to node i and $diff(s, i, j)$ indicates whether the words that were mapped to the nodes i, j in the sentence s appear close together. This is defined in terms of the position $pos(s, i)$ of a token i in the sentence s:

$$diff(s, i, j) = \max(0, (pos(s, j) - pos(s, i))^{-1}) \quad (3)$$

We generate new sentences from this graph by finding paths from the sentence start node to the sentence end node. We use the shortest path algorithm of Yen (1971) as implemented in the networkx-library (Hagberg et al., 2008) to generate up to 2500 candidate summary sentences per cluster. Following Filippova (2010), we filter out sentences that do not contain a verb or are shorter than eight tokens. We also include the original sentences in the selection candidates. Each candidate g is assigned the date of the cluster it was generated from: $date(g) := date(cluster(g))$.

To prevent ungrammatical or spurious sentence merges, we introduce additional filtering based on dependency parses. Specifically, we only accept a path P through the word-adjacency graph if for every node $i \in P$ at least one of the following holds:

1. i is a stopword node
2. At least one token mapped to i is the root node in its dependency tree
3. The head of at least one token mapped to i is contained in the path

Consider the following two input sentences:

> An armed attack on a government building was met with international shock .
>
> The people responsible for the attack have yet to be determined.

Without the constraint, it is valid to generate *An armed attack have yet to be determined*. The constraint prevents this as none of the heads of *attack* (i.e. *met* and *for*) are in the path.

3.3 Sentence Scoring and Selection

Given the set of generated sentences, we wish to find sentences that are well-formed and informative about important dates and events. We encode these aspects into multiple scoring functions.

3.3.1 Linguistic Quality

To encourage a readable output, we compute a linguistic quality score for each candidate sentence g by using the average probability of the tokens according to a 3-gram language model (Banerjee et al., 2015). We use the KenLM library (Heafield, 2011) with a pretrained model[3]. We compute the LM-score f_{LM} as follows:

$$f_{LM}(g) = \frac{1}{1 - \sum_{w_i \in g} \log(p(w_i|w_{i-1:i-3}))/|g|} \quad (4)$$

Additionally, we include information from the MSC system by preferring sentences which were generated from shorter paths. We let $f_{path}(g) = (1 + w(g))^{-1}$ where $w(g)$ is the length of the weighted path that generated candidate g.

3.3.2 Date Importance

We determine the importance $dimp(d)$ of a date d by counting how often it is mentioned in the input (Martschat and Markert, 2018). The score f_{date} of a sentence g is $f_{date}(g) = dimp(date(g))$.

3.3.3 Informativeness

We construct a keyword-based scoring function using TextRank (Mihalcea and Tarau, 2004) to efficiently score the importance of our candidates. TextRank scores keywords by constructing an undirected graph of content words where words are connected if they appear near each other. A score is computed for each node similarly to the PageRank algorithm (Page et al., 1999) using the following iterative formula:

$$TR^{(t+1)}(w_i) = (1-\alpha) + \alpha \cdot \sum_{w_j \in adj(w_i)} \frac{TR^{(t)}(w_j)}{|adj(w_j)|} \quad (5)$$

where $adj(w_i)$ is the set of nodes neighbouring w_i and $\alpha = 0.85$ is the dampening factor.

Let D_d be the set of all sentences s in the input corpus D whose cluster $cluster(s)$ was assigned the date d as per Equation 1. We compute one TextRank vector TR_d for each date d by running TextRank over all sentences in D_d. To make scores comparable across different D_d, we rescale the scores in TR_d to a 0 to 1 range. The TextRank-score $f_{TR}(g)$ for a candidate g is then defined as the sum of the scores of its tokens.

We also hypothesize that larger clusters are associated with more important events. We thus use the cluster size as a scoring function: $f_{cluster}(g) = \frac{|cluster(g)|}{\max_{C \in \hat{C}} |C|}$ where $\hat{C}$ is the set of all clusters.

3.3.4 Selection

We determine the final score of each candidate g as the product of the scoring functions:

$$score(g) = \prod_{f \in F} f(g) \quad (6)$$

where F is the set of scoring functions, i.e. $F = \{f_{path}, f_{LM}, f_{TR}, f_{date}, f_{cluster}\}$.

We select sentences greedily starting with the highest scoring ones as long as selecting them does not break any constraints. To reduce redundancy, we select at most one candidate from each cluster (Banerjee et al., 2015) and skip sentences with a cosine similarity of more than 0.5 to a previously selected sentence.

4 Evaluation

4.1 Data

We evaluate on the only two publicly available TLS datasets: Crisis (Tran et al., 2015) and Timeline 17 (TL17) (Tran et al., 2013b). Both contain human written timelines about topics such as civil wars or the BP oil disaster, collected from major news outlets. Each topic also has a set of related news articles scraped from the web (see Table 2).

We also report the median compression rate and the median spread of the datasets. The compression rate is the ratio of sentences in a timeline to the number of input sentences. The spread is the

[3] www.keithv.com/software/giga/lm_giga_20k_nvp_3gram.zip

Corp.	#Tops	#TLs	#Sen/Top.	Comp.	Spr.
TL17	13	19	27,222	0.0019	0.279
Crisis	4	22	150,429	0.0002	0.081

Table 2: Dataset statistics, including number of topics, timelines and the average number of sentences to be summarized for each topic. We also report the median compression and spread of the timelines.

ratio of dates with summaries in the timeline to the number of dates in the timeline span. Low compression rate and spread are typically indicative of a more difficult TLS instance (Martschat and Markert, 2018). We find that the datasets have very different characteristics, with Crisis having lower compression rate and spread.

4.1.1 Corpus Cleaning and Preporcessing

We found that some of the news articles to be summarised in both datasets contained full or partial gold timelines. This might cause TLS systems to inadvertently "cheat" by using the leaked gold timelines. We have manually removed 19 such documents in TL17 and 28 in Crisis.[4]

We preprocess all corpora with Stanford CoreNLP (Manning et al., 2014) and use Heideltime (Strötgen and Gertz, 2013) for resolving time expressions. Unlike several other TLS systems (Martschat and Markert, 2018; Chieu and Lee, 2004), we do not filter sentences with topic-specific keywords (e.g. *war* or *Syria*) to be less dependent on additional human input.[5]

4.2 Experimental Setup and Constraints

Like Martschat and Markert (2018), we generate one timeline per reference. We limit the number of dates to that in the reference, while the number of sentences per summary is set to the average number of sentences per summary in the reference.

As abstractive systems generate new text, they could exploit sentence limits by generating very long sentences. We control for this by limiting the number of tokens instead in one algorithm variation. We estimate the maximum number of tokens in the same way as for the sentence constraint.

4.3 Evaluation Metrics

Summarization is usually evaluated with ROUGE (Lin, 2004). This, however, ignores the temporal dimension of TLS. We thus use the two TLS measures proposed by Martschat and Markert (2017):

agree Compute ROUGE only between daily summaries which have the same dates.

align Align summaries in the output with those in the reference based on similarity and the distance between their dates, then compute the ROUGE score between aligned summaries. Distant alignments are punished.

We also report ROUGE *concat*, where we concatenate all entries in gold and system timeline and compute ROUGE between the results discarding all date information. While this measure is suboptimal for TLS (Martschat and Markert, 2017), it has been previously used as an evaluation measure (Yan et al., 2011b,a; Wang et al., 2016). We report the F1 score for all ROUGE metrics. To assess how well the systems are at date selection, we compute the F1 score between the dates that have a summary in the gold timeline and in the system timeline. Finally, we report the *copy rate* as the proportion of sentences copied directly from the corpus into the summary. We use an approximate randomization test (Noreen, 1989) to check statistical significance and the Bonferroni correction to correct for comparing on two datasets (Dror et al., 2018).

4.4 Oracle Summaries

One advantage of abstractive summarization is its potential to increase the maximum attainable scores by forming more succinct sentences. We investigate this potential with an oracle to establish an upper bound on summary scores, following similar work for generic summarization (Hirao et al., 2017). As an oracle over all summaries is intractable, we approximate it by replacing the scoring function (Equation 6) with an oracle that predicts the ROUGE-1-agree F1-score of sentences. The rest of our pipeline remains unchanged.

For the extractive oracle, we greedily select from all sentences in the input documents instead. The date of a sentence is the first exact time expression that appears in the sentence, or its DCT if there is none (Chieu and Lee, 2004).

4.5 Comparison Systems

4.5.1 Extractive Systems

We compare our full system with three extractive comparison systems. The first two are from a col-

[4]The corresponding document ids can be found at www.cl.uni-heidelberg.de/~steen/tls/docids.txt.

[5]However, we do let the competitor systems use filtering.

lection of TLS systems created by Martschat and Markert (2018).[6]

Chieu is a reimplementation of Chieu and Lee (2004), which uses the average cosine similarity of a sentence in a time-window around its date to determine importance and greedy selection. This system is often seen as a baseline for TLS systems (Martschat and Markert, 2018; Tran et al., 2015).

Submod is the state-of-the-art submodular system in Martschat and Markert (2018). Additionally, we have created a version of *Submod* with a token constraint. The same is not possible for *Chieu*, as it always selects one sentence per date.

Extractive is an extractive version of our system. It uses f_{TR} and f_{date} to score sentences. Dates are determined as for the extractive oracle

4.5.2 Neural Baseline

As an abstractive comparison, we use the popular Pointer Generator (See et al., 2017) (*Neural*). It was trained on the CCN/Daily Mail single document summarization corpus (Hermann et al., 2015). We adapt it to TLS as follows:

1. We select the dates for the timeline by ranking them by their frequency $dimp(d)$.

2. For each selected date d, we collect all sentences S_d from the corpus that refer to d.

3. For each collection S_d, we construct a pseudo document for the summarizer. Following Zhang et al. (2018) we use the LexRank score (Erkan and Radev, 2004) to rank the sentences in S_d. We add the top sentences to the document until we reach the maximum input size for the pointer generator (400 tokens).[7]

During our experiments, we found that the self-stopping nature of the pointer generator causes it to generate daily summaries that exceed the token length constraint described in Section 4.2 in 83% of daily summaries. To see if this disadvantages the pointer generator, we tried applying this token constraint to its output. However, this results in lower scores, so we only report results without length constraint.

[6] github.com/smartschat/tilse

[7] In an alternative setup, we tried selecting the centroid document for each date and then summarize it. This performs comparably or worse, depending on the corpus.

5 Results

5.1 Oracle Results

While both the extractive and the abstractive oracle perform equally on TL17, the abstractive oracle outperforms the extractive oracle significantly on Crisis. The abstractive copy rate on TL17 is also much higher than on Crisis. (73.7% vs 38.3% for sentence constraints). We hypothesize that this is related to the lower compression rate and greater size of Crisis (see Table 2). Abstractive TLS can only achieve its full potential when a variety of different texts needs to be compressed to short summaries. We investigate this in Section 5.5.

5.2 System Results

5.2.1 Extractive Systems

Our system outperforms *Extractive*, demonstrating the importance of our abstractive components. While *Chieu* performs better than our system in ROUGE-1 concat on Crisis, it is much worse in all date-sensitive measures and on TL17.

When comparing *Submod* and our abstractive system, we see behaviour similar to the oracles. On TL17, *Submod* achieves higher scores, though the differences are mostly not significant. On Crisis, however, we outperform *Submod* across all date-sensitive metrics and almost double the score in ROUGE-2 for agree and align. All improvements are significant except for ROUGE-1 align.

5.2.2 Neural

Neural performs slightly better than our system on the ROUGE-1 concat metric on Crisis, but performs significantly worse than us on almost all other content measures. This underlines the importance of TLS specific approaches.

5.2.3 Effect of Length Constraints

The token constraint has a small positive influence on our system while resulting in lower results for *Submod*. This shows that our system does not unfairly exploit the sentence constraint.

5.3 Example Timeline

Table 4 shows an example timeline generated by our system. Most entries describe events that are directly relevant to the civil war, though only two appear in the corresponding reference timeline. This demonstrates the difficulty of content selection in TLS, where even human timelines on the

	Date	Concat F1		Agree F1		Align F1		Copy
	F1	R1	R2	R1	R2	R1	R2	Rate
Timeline 17								
Chieu	0.195	0.223	0.049	0.024	0.008	0.046	0.012	1.000
Neural	0.518	0.320	0.055	0.061	0.012	0.069	0.013	0.000
Submod (s)	**0.543**^{3}	0.364	**0.087**	**0.092**^{3}	**0.021**	**0.103**^{3}	**0.024**	1.000
Extr. (s)	0.514	0.294	0.063	0.071	0.018	0.081	0.020	1.000
Abstractive (s)	0.512^{1}	0.349^{12*}	0.081^{12*}	0.075^{12}	0.020^{12}	0.087^{12}	0.022^{2}	0.446
Extr. Oracle (s)	*0.893**	0.501	0.180	0.317	0.143	0.320	0.144	1.000
Abs. Oracle (s)	0.883	0.504	0.179	0.322	0.142	0.324	0.142	0.729
Submod (t)	0.530	0.353	0.087	0.088	0.020	0.100	0.022	1.000
Extr. (t)	0.514	0.321	0.066	0.071	0.018	0.081	0.020	1.000
Abstractive (t)	0.512^{a}	**0.366**^{ab*}	0.084^{ab*}	0.081^{ab}	**0.021**^{a}	0.093^{ab*}	**0.024**^{b}	0.424
Ext. Oracle (t)	0.891	0.511	*0.183*	0.317	0.143	0.320	0.144	1.000
Abs. Oracle (t)	0.885	*0.513*	*0.183*	*0.325**	*0.147*	*0.327**	*0.147*	0.704
Crisis								
Chieu	0.146	**0.348**	0.065	0.026	0.006	0.047	0.010	1.000
Neural	0.279	0.343	0.047	0.049	0.008	0.064	0.010	0.000
Submod (s)	0.288	0.333	**0.071**	0.056	0.012	0.076	0.015	1.000
Extr. (s)	0.273	0.225	0.046	0.037	0.009	0.052	0.011	1.000
Abstractive (s)	**0.297**^{1}	0.324*	0.070^{2*}	0.066^{12*}	0.024^{123*}	0.080^{1*}	0.026^{12*}	0.382
Extr. Oracle (s)	0.934	0.509	0.167	0.359	0.142	0.359	0.142	1.000
Abs. Oracle (s)	*0.936*	*0.530**	*0.190**	*0.396**	*0.168**	*0.397**	*0.168**	0.475
Submod (t)	0.264	0.331	0.065	0.053	0.012	0.072	0.015	1.000
Extr. (t)	0.273	0.229	0.045	0.036	0.009	0.051	0.011	1.000
Abstractive (t)	**0.297**^{a}	0.333*	**0.071**^{b*}	**0.069**^{abc*}	**0.025**^{abc*}	**0.082**^{ab*}	**0.027**^{abc*}	0.331
Ext. Oracle (t)	0.933	0.503	0.164	0.352	0.139	0.353	0.139	1.000
Abs. Oracle (t)	*0.936*	*0.530**	0.186*	0.386*	0.163*	0.387*	0.163*	0.472

Table 3: Result of our system, the oracles, and comparison systems. (s) and (t) indicate sentence or token constraint where applicable. * indicates statistically significant difference between abstractive and extractive oracle and our abstractive system and *Extractive* respectively. ^{123} indicate significant differences between our system with sentence constraint and *Chieu*, *Neural*, and *Submod* with sentence constraint respectively. ^{abc} indicate the same for the token constraint ($p < 0.05$). Bold entries indicate best non-oracle results, italic ones best oracle results.

same topic can vary widely (Martschat and Markert, 2018; Tran et al., 2013b).

Most sentences have been edited by the MSC algorithm. We can observe some minor ungrammaticalities resulting from this process, like the phrase "on march" in the first daily summary. The timeline also exhibits some redundancy as the statement about the Red Cross is repeated twice.

5.4 Ablation Experiments

To study the effects of our scoring functions, we conduct an ablation study where we remove one scoring function at a time and rerun our system. The results can be found in Table 5.[8] We find all features contribute to ROUGE scores. Removing f_{TR} and f_{path} has a small negative effect on date F1 but a big effect on ROUGE, while f_{date} mostly affects date F1. It appears that content and date selection can to some extent be improved independently even with date-sensitive metrics. This might warrant future investigation.

5.5 Utility analysis

Our experiments show that the usefulness of our system is corpus-dependent. We investigate three factors that might explain this difference in performance: The number of input sentences, the compression rate, and the spread (see Section 4.1).

We compute the Spearman-correlation of all three factors with the difference in ROUGE-2-align F1 score between the two oracles as well as between our system and *Submod*. The result can be found in Table 6. For the oracles, we observe a strong negative correlation with compression (plus a weaker one with spread) and a positive one with the number of sentences. With more material the MSC system can generate more new sentences. In the same vein, a lower compression rate makes fusing sentences more useful. The difference be-

[8]To preserve space, we focus on the Crisis dataset with sentence constraint. Results are similar on TL17, but removing features there has generally a smaller effect.

2011-03-15
the conflict erupted on march 2011 when protesters inspired by arab world uprisings took to the streets to call for democratic change.
2012-02-04
russia and china vetoed a draft resolution that backed an arab plan to facilitate political transition in syria.
2012-06-13
talk of civil war in syria is not consistent with reality... what is happening in syria is a war against armed groups that choose terrorism, ”syrian state news agency sana quoted a foreign ministry statement as saying.
2012-07-15
red cross said sunday it now considers the conflict a civil war, meaning international humanitarian law applies throughout the country.
2012-07-16
the international committee of the red cross declared the conflict a civil war.
2012-07-18
on july blast at the syrian national security building in damascus during a high - level government crisis meeting killed four top regime officials, including the defense minister.

Table 4: Beginning of the timeline generated by our abstractive system with sentence constraint for the timeline in Table 1. Red color indicates sentences that were copied directly from the input corpus. Blue color indicates events which can also be found in the reference timeline.

Feat.	Date	R1	R2
$-f_{date}$	-0.049	-0.007*	-0.002
$-f_{TR}$	+0.004	-0.024*	-0.008
$-f_{path}$	+0.002	-0.028*	-0.015*
$-f_{cluster}$	-0.004	-0.012*	-0.010*

Table 5: Ablation results on Crisis, showing changes of ROUGE align and date selection F1. * indicates significant differences to the full model ($p < 0.05$).

Comparison	♯sents	Compr.	Spread
Abs. Or. - Ext. Or.	0.48	-0.61	-0.31
Abs. - Submod	0.45	-0.41	-0.24

Table 6: Spearman correlation of the score difference between systems and timeline properties.

tween our system and *Submod* exhibits similar, although less extreme behaviour. These results, together with the difference in size and compression rate between the datasets observed in Table 2, explain why our system outperforms the state of the art only on the more compressive Crisis dataset.

6 Readability Analysis

We assess the readability of the summaries generated by our abstractive system, the abstractive oracle (both with sentence constraint) and *Neural*.

We sampled 100 daily summaries for each system and from the gold summaries. We ensured that an approximately equal number of summaries was sampled from each generated timeline. Additionally, we sampled another 100 gold summaries and randomly deleted 25% of their tokens to simulate a compressive system without regard for linguistic quality. We call these summaries *Delete25*. We asked annotators from Amazon Mechanical Turk[9] to rate how well they are able to understand the summaries on a scale from 1 (completely ununderstandable) to 5 (easily understood). The descriptions of the rating scale presented to the workers can be seen in Table 7. Items were grouped in randomly ordered batches, so that each batch had one summary from each system.

Table 8 shows readability results. Unsurprisingly, Gold receives the highest score. Delete25 receives an unexpectedly high score, though notably lower than other systems. We find many sentences remain understandable even after deletions as in the following example: *Saif al-islam has been detained several bodyguards near the town obari by fighters in town of zintan, the justice minister and other officials said. He not wounded.*

Among the systems, ours receives the highest score. The oracle performs slightly worse. We speculate that this is due to the fact that the oracle does not include language model information. In both cases, over 80% of the sentences are easily understood (4 or 5). We also outperform *Neural*. This might be a result of its higher abstractiveness, which allows more errors.

7 Related Work

7.1 TLS

To the best of our knowledge, all systems proposed specifically for TLS have been extractive

[9] mturk.com

5	I can understand the text without problems. It does not have any grammaticality or fluency issues.
4	The text has some minor grammaticality or fluency issues but I can still understand it without problems.
3	I can understand the entire text, but it is difficult to do so.
2	I can understand the text only partially.
1	I can not understand the text at all.

Table 7: Rating scale for the readability task.

System	Avg.	#5	#4	#3	#2	#1
Neural	4.02	102	131	41	23	3
Abs. Or.	4.27	142	112	34	10	2
Abs.	4.40	165	103	21	9	2
Delete25	3.43	38	117	90	46	9
Gold	4.52	187	89	17	6	1

Table 8: Results of the readability evaluation. We also report the number of times each category was chosen.

(Nguyen et al., 2014; Chieu and Lee, 2004; Yan et al., 2011b,a; Wang et al., 2016; Tran et al., 2015, 2013b,a; Martschat and Markert, 2018). Several of these evaluate on corpora that are not publicly available (Chieu and Lee, 2004; Yan et al., 2011a,b) so that we cannot compare to their results. Since the advent of TL17 and Crisis, several evaluations have been performed on these datasets (Tran et al., 2015, 2013b,a; Martschat and Markert, 2018; Wang et al., 2016), but only Martschat and Markert (2018) evaluate with appropriate TLS measures. As code and original output are mostly unavailable, it is difficult to compare to them.

7.2 TLS-related Tasks

TLS is related to the TREC real-time summarization task (Lin et al., 2016). Unlike TLS, this task focuses on detecting novel information in a stream of social media posts in real time. TLS, on the other hand, assumes an offline setting and generates timelines for much longer timespans, focusing on the challenges of date selection and dating of information, which are not present in TREC.

There are also several papers that produce timelines by generating a summary for every single date in a given timespan, thus timeline generation without date selection (Wang et al., 2015; Allan et al., 2001). In these cases, the overall compression rate is not as low as for our setting and not comparable to the human timelines in our corpora.

TLS is also related to Task 4 in SEMEVAL 2015 (Minard et al., 2015). In this task, systems need to extract all events a query entity participates in. Unlike TLS the output is not a textual summary but a complete collection of the events in the input. Barros et al. (2019) have proposed narrative abstractive timeline summarization (NATSUM) in which they generate abstractive textual descriptions for the events in the SEMEVAL dataset. However, their work is markedly different from TLS in that "NATSUM [...] aims to generate narrative summaries and not timelines" (Barros et al., 2019, page 15). As a consequence, they do not perform any date selection and do not evaluate with appropriate date-sensitive metrics.

7.3 Generic Summarization

We have already described the differences between TLS and MDS and the limited direct applicability of MDS systems to TLS in Section 2.2. However, our methodology is inspired by the MDS system of Banerjee et al. (2015). We made major adaptations to this system for TLS by (i) using AP clustering to cluster sentences in a date-sensitive way that dynamically adapts to the corpus size and (ii) augmenting sentence scoring and selection to the needs of TLS. Our system is also related to neural abstractive summarization (See et al., 2017; Gehrmann et al., 2018; Cohan et al., 2018; Paulus et al., 2018). However, these methods require large training corpora unavailable for TLS.

8 Conclusion

We have presented a system for abstractive TLS which outperforms the state-of-the-art extractive TLS system when corpora are large and need substantial compression. Our analysis reveals a correlation between the difficulty of a TLS instance (as measured by compression and spread) and the advantage of an abstractive over a purely extractive approach.

Our system requires no supervision, which makes it well suited for TLS where the low number of available timelines makes training supervised systems difficult. We also require only lightweight annotations on the input, which allows for easy adaption to other settings and languages.

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, New York, NY, USA. ACM.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1208–1214. AAAI Press.

Cristina Barros, Elena Lloret, Estela Saquete, and Borja Navarro-Colorado. 2019. Natsum: Narrative abstractive summarization through cross-document timeline generation. *Information Processing & Management*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675. Association for Computational Linguistics.

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*, pages 425–432. ACM.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621. Association for Computational Linguistics.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

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.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 322–330, Stroudsburg, PA, USA. Association for Computational Linguistics.

Brendan J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science*, 315:972–976.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109. Association for Computational Linguistics.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1693–1701, Cambridge, MA, USA. MIT Press.

Tsutomu Hirao, Masaaki Nishino, and Masaaki Nagata. 2017. Oracle summaries of compressive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 275–280. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jimmy Lin, Adam Roegiest, Luchen Tan, Richard McCreadie, Ellen Voorhees, and Fernando Diaz. 2016. Overview of the trec 2016 real-time summarization track. In *Proceedings of the 25th text retrieval conference, TREC*, volume 16.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

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: Volume 2, Short Papers*, volume 2, 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*, pages 230–240. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, and Ruben Urizar. 2015. SemEval-2015 task 4: TimeLine: Cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 778–786, Denver, Colorado. Association for Computational Linguistics.

Kiem-Hieu Nguyen, Xavier Tannier, and Véronique Moriceau. 2014. Ranking multidocument event descriptions for building thematic timelines. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1208–1217, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking bringing order to the web. Technical report, Stanford InfoLab.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298.

Giang Tran, Mohammad Alrifai, and Eelco Herder. 2015. Timeline summarization from relevant headlines. In *European Conference on Information Retrieval*, pages 245–256. Springer.

Giang Binh Tran, Mohammad Alrifai, and Dat Quoc Nguyen. 2013a. Predicting relevant news events for timeline summaries. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 91–92. ACM.

Giang Binh Tran, Tuan A Tran, Nam-Khanh Tran, Mohammad Alrifai, and Nattiya Kanhabua. 2013b. Leveraging learning to rank in an optimization framework for timeline summarization. In *SIGIR 2013 Workshop on Time-aware Information Access (TAIA*.

Lu Wang, Claire Cardie, and Galen Marchetti. 2015. Socially-informed timeline generation for complex events. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1055–1065, Denver, Colorado. Association for Computational Linguistics.

William Yang Wang, Yashar Mehdad, Dragomir R. 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 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 58–68, San Diego, California. Association for Computational Linguistics.

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 Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 433–443, Stroudsburg, PA, USA. Association for Computational Linguistics.

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*, pages 745–754. ACM.

Jin Y. Yen. 1971. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716.

Jianmin Zhang, Jiwei Tan, and Xiaojun Wan. 2018. Towards a neural network approach to abstractive multi-document summarization. *CoRR*, abs/1804.09010.