Placing (Historical) Facts on a Timeline: A Classification cum Coref Resolution Approach

Sayantan Adak[0000−0001−5307−8811], Altaf Ahmad[0000−0002−6211−8237], Aditya Basu[0000−0003−1004−6507], and Animesh Mukherjee[0000−0003−4534−0044]

Indian Institute of Technology Kharagpur
{sayantanadak.skni}@kgpian.iitkgp.ac.in, {altafahmad3037045}@iitkgp.ac.in, {aditya.basu1}@iitkgp.ac.in, {animeshm}@cse.iitkgp.ac.in

Abstract. A timeline provides one of the most effective ways to visualize the important historical facts that occurred over a period of time, presenting the insights that may not be so apparent from reading the equivalent information in textual form. By leveraging generative adversarial learning for important sentence classification and by assimilating knowledge based tags for improving the performance of event coreference resolution we introduce a two staged system for event timeline generation from multiple (historical) text documents. We demonstrate our results on two manually annotated historical text documents. Our results can be extremely helpful for historians, in advancing research in history and in understanding the socio-political landscape of a country as reflected in the writings of famous personas. The dataset and the code are available at https://github.com/sayantan11995/Event-Timeline-Generation-from-Documents.

1 Introduction

Timeline serves as one of the most effective and easiest means to contextualize and visualize a complex situation ranging from grasping spatio-temporal facts in historical studies to critical decision making in businesses. With the stupendous increase of textual resources for many historical contents in several online platforms it has become imperative for the history researchers to understand the chronological orderings of the incessant historical phenomenon. The fact timeline can be an extremely useful aid to highlight the temporal and causal relationships among several facts and the interactions of the characters over time, that results in identifying common themes that arise over the period of interest in a historical document (see Figure 2 in Appendix A.1).

In this paper we present a full pipeline to build a chronology of facts extracted from historical text. Our contributions are as follows.

– We curate a first of its kind dataset from two different historical texts – the *Collected Works of Mahatma Gandhi* (CWMG) and the *Collected Works of Abraham Lincoln* (CWAL) for our experiments. For each of these datasets we manually annotate sentences that correspond to important facts. Next for

each of these annotated sentences we also further annotate the coreferences to the same fact; we call these fact coreferences. Upon acceptance we shall release this data for future research.
- We introduce a novel divide-and-conquer based approach to generate fact timeline from timestamped historical texts. In the first step, we classify sentences as containing facts or not using a generative adversarial learning setup. In the subsequent step we compute fact coreferences using both unsupervised and supervised methods. The main novelty here is that inclusion of world knowledge in the form of tag embeddings results in higher performance gains.
- We present a rigorous evaluation of both the steps as well as the full system which was absent in previous literature [7]. Further we compare our results to the closely related fact timeline summarization tasks by suitably adapting them so that the comparison is fair.
- In order to determine the readability and usefulness of the timeline, we conduct an online crowd-sourced survey. 93% survey participants found it to be effective in summarizing historical timeline of facts.
- We also show that our method is generic by evaluating it against a COVID-19 news related dataset which is not a historical text per se.

2 Related work

Important sentence classification & sentence coreference resolution: Our proposed approach combines important sentence classification, filtering historically important sentences from a bunch of texts, and sentence coreference resolution, merging factually similar sentences. [39] used CNN to analyse sensitivity for text classification. [27] and [38] introduced virtual adversarial training methods for robust text classification from a small number of training data points.
Recent works like [10], [18] have used neural network based architecture to train their model on benchmark coreference dataset (ECB+ [12]). [21] attempted to create an end-to-end event coreference resolution system based on the standard KBP dataset[1].

Timeline of historical facts: [5] proposed an unsupervised generative model to construct the timeline of biographical life-facts leveraging encyclopaedic resources such as Wikipedia. [3] also uses Wikipedia for timeline construction of historical facts. [7] attempted to construct a fact timeline from history textbooks considering the sentences having temporal expressions. [29] proposed an automatic approach to capture and visualize temporal ordering of interactions between multiple actors. [2] created an AI-enabled web portal based on CWMG dataset.

Timeline summarization (TLS): The timeline summarization task aims to summarize time evolving documents. [15] evaluated existing state-of-the-art methods for news timeline summarization and proposed *datewise* and *clustering*

[1] https://www.ldc.upenn.edu/collaborations/past-projects/tac-kbp

based approaches on the TLS datasets. [8] demonstrated the potential of employing several IR methods on TLS tasks based on a large news dataset. [20] proposes a new approach by generating date level summaries, and then selecting the most relevant dates for the timeline summarization.

The present work: Our paper is closest in spirit to the work done by [7]. In this paper the authors outlined the challenges related to fact coreference for timeline generation; however, they did not suggest ways to effectively tackle these challenges and, thereby, solve the problem. We close this gap in our paper by proposing an efficient approach to resolve fact coreference. Our work has also close parallels with the fact timeline summarization (TLS) task. Nevertheless, previous TLS researchers mostly worked on the documents containing multiple news articles, which are rich in facts. These works have not focused much on prior fact detection and have not addressed how they can be effectively generalized in historical text documents such as biographies. Our work for the first time shows that fact detection could largely benefit TLS tasks in the context of historical texts.

3 Data preparation

In this section we present the details of the datasets that we prepare for our experiments. We also outline the overall annotation process of these datasets.

3.1 Datasets

Collected works of Mahatma Gandhi: We leverage the Collected Works of Mahatma Gandhi (CWMG) available at [32], an assortment of 100 volumes consisting of the books, letters, telegrams written by Mahatma Gandhi and also the compiled writings of the speeches, interviews engaging Gandhi. This data covers many important historical facts within the time period of 1884-1948 in British colonised India.

Collected works of Abraham Lincoln: The second dataset we have use to demonstrate our system is based on the life-long writings of the 16^{th} president of the United States, Abraham Lincoln, formally known as the Collected Works of Abraham Lincoln (CWAL)[2] comprising a total of 8 volumes.

COVID-19 fact dataset: In addition, to establish the generalizability of the approach, we collect 140 major facts, that happened in India during the COVID-19 pandemic from different sources such as *Wikipedia*[3], *Who.int*[4] to be placed on a timeline for elegant visualisation using our system.

[2] https://quod.lib.umich.edu/l/lincoln/
[3] https://en.wikipedia.org/wiki/COVID-19_pandemic_in_India
[4] https://www.who.int/india/emergencies/coronavirus-disease-(covid-19)/india-situation-report

3.2 Pre-processing

From the 100 volumes of text files from CWMG we first extract all the letters containing the publication dates and recipients name. There were a total of 28531 letters in the entire CWMG. We primarily use the letters for our experiments as we observe that they contain the best temporal account of the facts. From the overall set of letters, we select the year range 1930–1935 since this range has the largest collection of letters. In order to further choose the right data sample, we categorize the letters into *formal* and *informal* types based on the recipients of the letters. A simple heuristic that we follow is – the letters written to government officials and famous historic personalities can be categorized as formal while those written to the family members can be classified as informal ones. We collect the list of Mahatma Gandhi's family member names from Gandhian experts for identifying the informal letters. We manually notice that the formal letters contain much more useful historic information than the informal ones. We therefore only consider the formal letters for manually annotating the useful sentences. In addition, we only consider the letters which have more than 1000 words in its content. This results in 41 letters with substantial content.

Table 1. Sample list of sentences from CWMG after the sentence classification. The explicit temporal expression inside the sentence is highlighted.

Doc creation time (Initial reference time)	**Important sentences**	**Updated reference time**
May 4, 1930	He was arrested at 12.45 a.m. on May 5.	May 5, 1930
May 4, 1930	In Karachi, Peshawar and Madras the firing would appear to have been unprovoked and unnecessary.	May 4, 1930

3.3 Annotation

In this section we outline the data annotation procedure for the two phases. Recall that our method has two important steps – fact classification and coreference resolution. While the fact classification phase is supervised (Level I annotations), the coreference resolution is done using both unsupervised and supervised techniques. The annotations for the coreference resolution (Level II annotations) are therefore required to (a) train the supervised approach and (b) test the efficacy of both the unsupervised and the supervised approaches.

Level I – Important sentences: Finally, out of these filtered letters we manually annotate all the sentences of 18 letters (i.e., 979 sentences in all). The remaining sentences (i.e., 1689 in total) from the rest of the letters were left unlabelled. Both of these labelled and unlabelled sentences were used for

training the classifier. The classes in which the sentences were classified were based on their historical importance. In specific, we identify two such important classes – (a) the *facts* or factful sentences, which typically represent that some important historical phenomena or event [33] happened or took place , e.g., '*A vegetable market in Gujarat has been raided because the dealers would not sell vegetables to officials*'[5], (b) the *demands*, which represent the demands Mahatma Gandhi had made to the British government through his writings, e.g., '*The terrific pressure of land revenue, which furnishes a large part of the total, must undergo considerable modification in an independent India.*' and (c) others (i.e., not important). As the examples suggest, each individual sentence is annotated as important (i.e., containing a fact/demand) or not. In order to further enrich the dataset we collect gold standard facts related to Mahatma Gandhi from an additional reliable and well maintained resource[6]. We obtain 86 additional sentences thus making a total of 1065 (i.e., 979 + 86) important sentences (see Table **??** for the classwise distribution.).

Table 2. Sample list of sentences from CWMG after the sentence classification. The explicit temporal expression inside the sentence is highlighted.

Classes	**Count**	
	CWMG	**CWAL**
fact	716	242
demand	81	96
other	268	382

x

For the CWAL we simply extract all the sentences from volume 2 and follow similar approaches to annotate important sentences as in the case of CWMG. Without considering any filtering criteria we consider all the 111 articles of volume 2 including his letters and propositions which consist of a total of 1386 sentences. Out of these 720 sentences were manually annotated (see Table **??**).
Annotator details and annotation guidelines: For both the datasets three annotators annotated the sentences. The annotation process was led by one PhD student along with two undergraduate students. The PhD student had substantial experience in historical text analysis and will be referred to as the expert annotator henceforth. The first level of annotation was carried out for each of the sentences and based on the assumption that a full sentence corresponds to a fact/demand. All the annotators annotated the sentences independently. For the training of the two undergraduate annotators, they were provided with the examples of 25 gold standard facts and demands each. The gold standard facts were collected from the reliable resource mentioned in the earlier paragraph and the gold standard demands were collected from the formal letters of Mahatma Gandhi which were first annotated by the expert annotator and verified by a

[5] Such sentences would typically consist of participants and locations.
[6] https://www.gandhiheritageportal.org/

Gandhian scholar (see Table 9 in Appendix A.2 for example annotations). The inter-annotator agreements, i.e., Cohen's κ were 0.66 and 0.58 for the former and the latter datasets respectively. Table **??** shows the category distribution for both the datasets. The Level I annotation was not carried out for the COVID-19 dataset because, each sentence collected were presented as facts in the mentioned portals and thus we considered all the sentences as important facts.

Level II – Coreference resolution: The second round of annotation was carried out for evaluating the fact coreference detection task on the same dataset. For this case we only annotate the texts which were marked important during the Level I annotation. In addition, the Level II annotation was also carried out for the COVID-19 fact dataset.

Annotator details and annotation guidelines: The same annotators annotated for the Level II phase. The annotators were provided with sentences, the reference documents (letters) from which the sentences were extracted and the reference time (document publication date). Based on the perception of the annotators, the sentences that potentially referred to the same fact were placed in the same cluster. The coreferences have been placed by the annotators in different clusters based on different factors like the commonness of the mentioned times, entities and the fact name/composition. Consider these two sentences - '*The crowd that demanded restoration of the flag thus illegally seized is reported to have been mercilessly beaten back.*' and '*Bones have been broken, private parts have been squeezed for the purpose of making volunteers give up, to the Government valueless, to the volunteers precious salt*'. Although there is no explicit mention of time in either of the sentences, both of them are from the same document and thus their reference dates would be the same as the publication date of the document. Also both of them refer to similar types of atrocities. So these two sentences should be placed in the same cluster. We first carried out a trial round for the two undergraduate annotators by using 100 randomly chosen important sentences from the Level I phase and the trial annotations were verified by the expert annotator. Finally for the complete Level II annotations, the inter-annotator agreements were 0.74, 0.61, and 0.78 for the CWMG, the CWAL and the COVID-19 dataset respectively using MUC [37] based F1-score [14] (see Table 10 in Appendix A.2 for example annotations and Appendix A.3 for other agreement metrics.).

4 Methodology

Our method consists of three major components (see Figure 1): (i) important sentence extraction, (ii) sentence coreference resolution, and (iii) timeline visualization. The arrows represent the direction of data flow. In this section we describe in detail the methods used for each of these components.

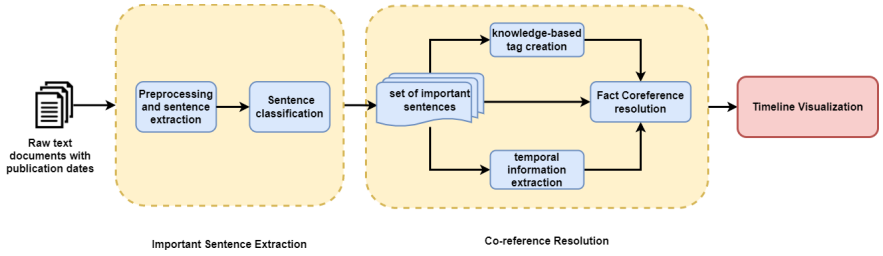

Fig. 1. The overall architecture for generating the timeline.

4.1 Important sentence extraction

Baselines: As baselines, we use *SVM* [16] and *Multinomial Naïve Bayes* [19] on simple bag-of-words feature. For *SVM* we use linear kernel. For the evaluation of the classifiers we use a 70:30 train-test split of the annotated data.

Fine-tuned BERT: Apart from the above two baselines, we try BERT [13] neural network based framework for the classification. We train the model using the PyTorch [30] library, and apply *bert-base-uncased* pre-trained model for text encoding. We use a batch size of 32, sequence length of 80 and learning rate of $2e-5$ as the optimal hyper-parameters for training the model.

GAN-BERT text classifier: In search for further enhancement of the performance based on our limited sets of labelled data, we employ the *GAN-BERT* [11] deep learning framework for classifying the important sentences. It uses generative adversarial learning to generate augmented labelled data for semi-supervised training of the transformer based BERT model. It improves the performance of BERT when training data is scarce and is therefore highly suited for our case. Here we also feed the unlabeled data sample, as discussed in section 3.3, to help the network to generalize the representation of input texts for the final classification [11].

4.2 Sentence coreference resolution

Once the classification was done we end up with 'factful' sentences linked to its corresponding document creation time in the format noted in Table 2.

Time within sentences: For generating the accurate fact timeline we need to assign a valid date to a particular sentence (i.e fact/demand). For example, in the first sentence in Table 2, although the document publication time is mentioned to be `May 4, 1930`, the sentence clearly has embedded in it the exact fact date `May 5, 1930` apparent from the snippet '*arrested on May 5*'. Therefore, if the explicit time is present in the sentence we use it directly, else we use the creation/publication date of the document. We extract the explicit mention of time in the text using the *HeidelTime* [36] tool. This tool is capable of identifying embedded mentions of temporal expressions such as *'yesterday', 'next day' etc.*

Tag generation from world knowledge: An individual sentence does not always contain much information about the fact/demand which it is getting referred to. So we attempt to incorporate world knowledge for each individual sentence. By using each sentence as a query we gather the top five *Google* search results using the *googlsearch* api[7] and also consider the document from which the sentence was being extracted. Next we analyse the search result using *TextRank*[8], *Rake*[9] and *pointwise mutual information*[10] to generate top keywords present in the search result. Although these methods produce reasonably good results, in many cases we needed to manually filter out certain noisy tags. For each sentence we therefore land up with one or more tags. We retain the top ten tags for every sentence which means that the number of tags for a sentence could vary between one and ten. The details of the tag generation procedure mentioned in Appendix A.4. We do not use encyclopaedic resources such as Wikipedia to get the search results because the datasets we are using, are only available in a few very specific websites. We fed the list of keyword(s) or tag(s) obtained for a sentence to the pre-trained *sentence-bert* model for obtaining a 768 dimensional embedding representation of the keywords.

Unsupervised sentence clustering: We employ several unsupervised approaches for sentence coreference resolution. As baselines, we choose two commonly used approaches for coreference resolution – (a) *Lemma*: It attempts to put the sentence pairs in same coreference chain which share the same head lemma, (b) *Lemma-δ*: In addition to same head lemma as a feature, it also computes the cosine similarity (δ) between the sentence pair based on *tf-idf* features, and only places the sentence pairs in the same coreference chain if δ exceeds some threshold. Then the sentence clusters were created using agglomerative clustering method. To extract the head lemma of a sentence, we use the *SpaCy* dependency parser.

Apart from these two common baselines, we vectorize the sentences using *tf-idf* vectorization technique and then apply different clustering techniques such as *Gaussian-Mixture*[11] model, *agglomerative clustering* to cluster the sentences corresponding to similar facts. We also use the pre-trained *sentence-bert* [35] model to encode the sentences and apply similar clustering techniques. Finally, we concatenate the sentence embedding with the tag embedding generated from that particular sentence. We again cluster the sentences based on this new representation. This, as we shall later see, significantly improves the performance of the clustering phase. We evaluate the clustering results on the basis of the annotated data which had been obtained in the second phase of data annotation. We used the *elbow* method to find the optimal number of clusters in case of Gaussian-Mixture and used *dendogram* to select the optimal distance threshold for the suitable number of clusters in case of agglomerative clustering. The

[7] https://github.com/MarioVilas/googlesearch
[8] https://github.com/DerwenAI/pytextrank
[9] https://pypi.org/project/rake-nltk/
[10] https://www.nltk.org/howto/collocations.html
[11] https://scikit-learn.org/stable/modules/mixture.html

distance threshold we selected were 0.25, 0.6 and 0.6 for CWMG, CWAL and COVID-19 data respectively.

Supervised fact mention-pair model: A *fact mention* is a sentence or phrase that defines a fact and one fact may contain multiple *fact mentions* [9]. We first create a dataset containing all the possible pairs of *factful* (i.e., fact or demand) sentences from the ground-truth annotations. We set the coreference label to 1 if the sentence pair is contained in the same cluster as per the Level II annotation and 0 otherwise. Here we again use a 70:30 split to generate training and test instances. The overall architecture is inspired from [6] (see Appendix A.5). The inputs to the model are the two sentences (i.e. S_1 and S_2) and their corresponding *actions* (i.e., A_1 and A_2), *time* (i.e., T_1 and T_2) and *tags* (i.e., K_1 and K_2). We extract *actions* (i.e., A_i) for each of the sentences using *SpaCy* dependency parser[12].

Mention pair construction: We used *Tensorflow* [1] tokenizer to vectorize each feature (i.e., sentences, actions, time and tags) to convert it into sequence of integers after restricting the tokenizer to use only the top most common 5000 words. For the sentences we limit the sequence length to 64. For the other features - actions, time and tags - we limit the sequence length to 10. We always use zero padding for smaller sequences. We next encode the words present in each of these sequences using a pre-trained *GloVe* [31] embedding (100 dimensions). Thus each sentence comes out as a $64 * 100$ size vector representation while each of the other features come out as a $10 * 100$ size vector representation. Now each of these vectors are separately passed through a LSTM [17] layer with default hyperparameters to transform them into 128 size vectors each. Next each of these 128 size vectors are passed through separate dense layers to obtain 32 size vectors. Finally, these 32 size vectors are concatenated using a concatenation layer. The output of the concatenation layer is what we term as a *mention representation*. Two mention representations are concatenated to get a pairwise representation (i.e., an *fact mention pair*) and passed through a feed forward network to return a score denoting the likelihood that two mentions are coreferent (see Figure 3 in Appendix A.5). Based on the predicted pairwise score on the test instances we used a threshold (0.5 in our case) to generate a similarity matrix of the mentions, and then applied agglomerative clustering to partition the similar mentions into the same clusters.

4.3 Timeline visualization

Once the sentence coreference resolution phase was successfully executed, we generated visualization for the given fact/demand sequence using *vis-timeline*[13], a dynamic, browser based visualization library.

[12] We consider the root verb as action for a sentence
[13] https://visjs.github.io/vis-timeline/docs/timeline/

Table 3. Results (accuracy and macro F1-score) for the important sentence classification using our approaches on the two datasets. MNB: Multinomial Naïve Bayes. Best results are marked in boldface and highlighted in green cells.

Dataset	Model	Evaluation Metric	
		Accuracy	F1
CWMG	MNB	0.74	0.45
	SVM	0.79	0.5
	Fine-tuned BERT	0.8	0.57
	GAN-BERT	**0.9**	**0.69**
CWAL	MNB	0.6	0.3
	SVM	0.6	0.34
	Fine-tuned BERT	0.61	0.56
	GAN-BERT	**0.7**	**0.65**

5 Experiments

5.1 Evaluation metrics

We have used separate evaluation metrics for the two phases.

Important sentence classification: In this case we use the standard *accuracy* and *F1-score* values.

Sentence coreference resolution: Here we conduct the evaluation based on the widely used coreference resolution metrics – (a) *MUC* [37], (b) B^3 [4], (c) *CEAF* [22], and (d) *BLANC* [34]. Due to the inconsistency of each of these evaluation metrics [28] we shall also report the average outcomes of all the metrics.

5.2 Results

We evaluate the two different phases separately. Ground-truth data was used from each phase for respective evaluations.

Important sentence classification: The key results for the two datasets (CWMG and CWAL) are summarised in Table 3. Our approach based on GAN-BERT by far outperforms the standard baselines. For the CWMG dataset, the macro F1-score shoots from 0.50 (SVM) to 0.69 on the three class classification task. Likewise for the CWAL dataset, the macro F1-score shoots from 0.34 (Naïve Bayes) to 0.65.

Evaluation of coreference resolution: For the evaluation of coreference resolution we use several coreference resolution metrics to analyse the model performance. It is apparent from Table 4 that the approach based on clustering with *sentence-bert* embeddings by far outperforms the baselines *lemma* and *lemma-δ*. For the CWMG dataset, *sentence-bert* + agglomerative clustering is the best overall; for the other two datasets no single method is a clear winner. However, the primary point that we wish to emphasize in the table is the result after incorporating tag embedding. It can be clearly observed that this intuitive, albeit hitherto unreported, technique almost always produces better

Table 4. Sentence coreference results before and after tag embedding. GM: Gaussian Mixture based clustering; AC: Agglomerative Clustering; s-bert: sentence-bert; m-pair: supervised mention-pair model. Best results including the tag embedding are marked in boldface and highlighted in green cells. Best results excluding the tag embedding are marked by underline and highlighted in blue cells.

Dataset	System	MUC	B^3	CEAF_E	BLANC	Avg (overall)			Time taken
		F1	F1	F1	F1	Recall	Precision	F1	
CWMG	Lemma	0.45	0.38	0.20	0.49	0.39	0.38	0.38	45 sec
	Lemma-δ	0.53	0.41	0.19	0.48	0.48	0.40	0.41	7 min 22 sec
	tf-idf + GM	0.53	0.53	0.36	0.60	0.49	0.52	0.50	26 min 14 sec
	tf-idf + AC	0.55	0.50	0.42	0.57	0.50	0.53	0.51	5 min 13 sec
	s-bert + GM	0.61	0.54	0.41	0.60	0.54	0.54	0.54	29 min 34 sec
	s-bert + AC	0.63	0.57	0.40	0.61	0.55	0.56	0.55	7 min 42 sec
	+ tag embedding								
	tf-idf + GM	0.64	0.57	0.45	0.64	0.57	0.60	0.58	28 min 19 sec
	tf-idf + AC	0.62	0.61	0.51	0.66	0.58	0.63	0.60	6 min 57 sec
	s-bert + GM	0.65	0.62	0.48	0.66	0.60	0.60	0.60	30 min 28 sec
	s-bert + AC	0.75	**0.70**	0.52	**0.73**	0.65	**0.71**	0.68	8 min 36 sec
	m-pair model	**0.91**	0.59	**0.83**	0.53	**0.83**	0.69	**0.72**	2 hr 10 min 32 sec
CWAL	Lemma	0.28	0.11	0.17	0.49	0.26	0.27	0.27	58 sec
	Lemma-δ	0.31	0.15	0.14	0.48	0.28	0.27	0.18	9 min 41 sec
	tf-idf + GM	0.53	0.37	0.35	0.49	0.42	0.45	0.43	41 min 25 sec
	tf-idf + AC	0.57	0.42	0.38	0.49	0.45	0.49	0.46	8 min 5 sec
	s-bert + GM	0.43	0.39	0.40	0.54	0.43	0.46	0.44	46 min 18 sec
	s-bert + AC	0.51	0.42	0.40	0.54	0.46	0.48	0.47	11 min 15 sec
	+ tag embedding								
	tf-idf + GM	0.74	0.52	0.40	0.63	0.56	0.59	0.57	43 min 23 sec
	tf-idf + AC	0.72	0.51	0.48	0.64	0.57	0.61	0.59	9 min 27 sec
	S-bert + GM	0.74	0.41	0.34	0.67	0.51	0.57	0.54	47 min 12 sec
	s-bert + AC	0.82	**0.53**	0.44	**0.72**	0.60	**0.66**	0.63	11 min 42 sec
	m-pair model	**0.96**	0.42	**0.78**	0.35	**0.82**	0.65	**0.64**	2 hr 11 min 40 sec
COVID-19	Lemma	0.55	0.39	0.28	0.55	0.51	0.42	0.44	9 sec
	Lemma-δ	0.34	0.29	0.25	0.51	0.35	0.34	0.35	1 min 8 sec
	tf-idf + GM	0.56	0.41	0.36	0.60	0.47	0.50	0.48	6 min 37 sec
	tf-idf + AC	0.59	0.45	0.36	0.62	0.49	0.54	0.51	1 min 44 sec
	s-bert + GM	0.63	0.45	0.32	0.57	0.47	0.51	0.49	8 min 41 sec
	s-bert + AC	0.61	0.44	0.35	0.57	0.48	0.50	0.49	2 min 25 sec
	+ tag embedding								
	tf-idf + GM	0.44	0.33	0.28	0.54	0.39	0.40	0.39	7 min 31 sec
	tf-idf + AC	0.44	0.34	0.32	0.44	0.4	0.42	0.41	2 min 38 sec
	s-bert + GM	0.57	0.41	0.35	0.59	0.47	0.49	0.48	9 min 35 sec
	s-bert + AC	0.63	0.46	0.39	0.59	0.51	0.52	0.52	3 min 19 sec
	m-pair model	**0.86**	**0.80**	**0.97**	**0.65**	**0.80**	**0.84**	**0.82**	29 min 18 sec

results (see Appendix A.4 and the Table 12 therein describing the tag generation process in more details). In fact, the assimilation of the tag embeddings with the *sentence-bert* embeddings boosted the overall F1-score by 13%, and 16% for the

CWMG and the CWAL datasets respectively. Note that these results hold even if the manual filtering step in the tag generation is completely omitted (see Table 7). An interesting observation is that the benefit of the tag embedding is best leveraged by the sentence-bert + agglomerative clustering. For the COVID-19 dataset, since search results are generic, the benefit of tag embedding is less. Furthermore, the supervised model consistently outperforms the unsupervised results across all three datasets. Note that the tag generation is done only once and therefore takes a fixed amount of time. It took 3.26 seconds, 3.47 seconds, and 1.96 seconds per sentence on average to generate knowledge-based tags for CWMG, CWAL, and COVID-19 datasets respectively. The time that the model takes to inference in presence of the tag embeddings is negligible as compared to the model without these embeddings (see the last column of Table 4). For the supervised models though, the major chunk of time is required for the mention pair generation.

Full system evaluation: So far, the assessment for the two components was carried out separately, i.e., the evaluation for the important sentence extraction was based on Level I annotated data while the evaluation for sentence coreference resolution was on the basis of Level II annotations independently. We also conduct the full system evaluation for CWMG and CWAL datasets, i.e., the complete evaluation was only dependent on Level II annotated data. For this case we trained the GAN-BERT classifier with 30% of the labeled data along with the unlabeled data (discussed in section 3.3), and had predictions for the rest of 70% data. Now, we consider only the *true positives* (labeled as important, and also predicted important), before performing the coreference resolution. This task is evaluated based on the Level II annotated data. The primary reasons for considering only true positive samples are - (1) we do not have ground-truth Level II annotated data for the non-important sentences (i.e., the false positives), (2) for all practical purposes we are only interested in the coreferences present in the positive predictions (i.e., in the predicted important sentences). Table 5 shows the comparison between the full system evaluation result and the standard result (see Appendix A.8 for results w/o tags). The results shown here are the average value of the four different standard metrics (MUC, B^3, CEAF_E and BLANC) corresponding to the best performing unsupervised model as well as the mention-pair based supervised model.

Comparison with TLS: Since our method has some parallels with TLS, in this section we perform a thorough comparison with state-of-the-art TLS systems. Note that the output of our system is not similar to that of the standard TLS output. In order to make the comparison possible and fair we added a simple summarization step at the end of our pipeline. We used the BERT extractive summarizer [26] to extract the two most important sentences as the summary for each of the fact clusters generated by our method. We evaluated the summaries using the alignment-based ROUGE (AR) F-Score [24]. Unlike [15], we did not use any date ranking method to rank the dates of the predicted timeline and compared the ground-truth with the top-k predicted timeline. We tested all the approaches using our Level I annotated data as the ground-truth reference.

Table 5. Full system evaluation result. Type: Coref-resolution type, MA: Important sentences obtained through manual annotation, MP: Important sentences obtained from model prediction, Su: Supervised, Un: Unsupervised. Appendix A.8 shows the same results without using tag embeddings.

Dataset	Type	M	R	P	F1
CWMG	Su	MA	0.83	0.69	0.72
		MP	0.74	0.63	0.64
	Un	MA	0.65	0.71	0.68
		MP	0.62	0.65	0.63
CWAL	Su	MA	0.82	0.65	0.64
		MP	0.74	0.59	0.60
	Un	MA	0.60	0.66	0.63
		MP	0.55	0.59	0.57

Table 6 shows the detailed comparison of our approach with few of the existing state-of-the-art TLS approaches on two of our datasets. In order to perform these experiments we considered pre-selected 41 formal letters from CWMG in the time period 1930-1935 with more than 1000 words and all the documents of volume 2 from CWAL (from which the Level I annotations were performed) and directly passed through the TLS pipeline using the codes provided by the respective authors. In order to make the comparison further fair, we also performed an experiment by first carrying out important sentence classification using our method and then feeding the filtered data into the TLS pipeline provided by the authors. In order to benefit the TLS models the fact detection for this pre-filtering was performed using the model fine-tuned on our dataset. This modification results in superior performance of the TLS. In fact, fact detection prior to summarization always helps – our method as well as one of the baseline methods [15] where fact detection can be easily incorporated show significantly[14] improved performance. In Table 13 of Appendix A.6 we also show that this fact detection step brings benefits to a standard TLS dataset which has not been built from historical text. The reason for this inferior performance could be that the summary in the standard TLS approaches are highly sensitive to the keywords used for the particular dataset and generating quality keywords for a dataset consisting of diverse facts like ours requires domain-expertise (see Table 14 in Appendix A.7).

6 Ablation study

We performed two ablation studies - first, to check the effectiveness of manual filtering of noisy tags, second, to assess the added value of each component in the mention-pair model.

Sentence coreference resolution results without manual filtering of tags: Table 7 shows result obtained from different coreference resolution

[14] Statistical significance were performed using Mann–Whitney U test [23]

Table 6. Comparison of our method for the with the existing state-of-the-art TLS methods - (1) MM (submodularity based method): [25] and (2) DT: datewise and (3) CLUST: clustering based TLS by [15], FD: Fact detection. †, *, • show that our results are significantly different from MM, FD + DT, FD + CLUST respectively. In turn, any method with FD (*, •) is significantly better than MM.

System	CWMG Dataset		CWAL Dataset	
	AR1-F	AR2-F	AR1-F	AR2-F
MM	0.023	0.001	0.052	0.024
DT	0.008	0.001	0.022	0.002
FD (our) + DT	0.015*	0.006*	0.026*	0.002
CLUST	0.028	0.02	0.055	0.040
FD (our) + CLUST	0.034•	0.025•	0.086•	0.071•
Our method	**0.062†*•**	**0.043†*•**	**0.069†*•**	**0.042†*•**

techniques when we do not include any manual filtering steps to the generated tags. It can be noticed that there is not much difference in the results even when we omit this step.

Added value of each element in the mention-pair model: Table 8 shows the added value of each feature in the mention-pair model. For both the historical texts we observe that inclusion of each feature improves the overall performance. The best improvement is observed on the inclusion of the external knowledge in the form of tag embeddings.

7 Timeline visualization

Generating a timeline would not be that impactful unless it is visualized in an interpretable and convenient way. We incorporate an elegant visualization for the generated fact/demand timelines using *vis-timeline* javascript library (Appendix A.9 shows an example timeline).

Survey: In order to understand the effectiveness of the interface we ran an online crowd-sourced survey. Out of 33 participants with different educational backgrounds, overall 93% agreed that the interface was very useful for summarization of historical timeline of facts. 88% participants found some information which would have been hard for them to fathom just by reading the CWMG plaintext (more results in Appendix A.10).

8 Conclusion

In this work we presented a framework to generate fact timeline from any timestamped document. The entire pipeline has two parts – important sentence detection and sentence coreference resolution. We achieve very encouraging results for

Table 7. Sentence coreference results without using manual filtering for the tags. D: dataset, M: model, GM: Gaussian Mixture based clustering; AC: Agglomerative Clustering; s-bert: sentence-bert, m-pair: mention-pair model, B: BLANC, C: CEAF_E. The results mostly remain unaffected.

D	M	MUC	B^3	C	B	Avg (overall)		
		F1	F1	F1	F1	R	P	F1
CWMG	tf-idf+GM	0.61	0.55	0.51	0.58	0.62	0.57	0.56
	tf-idf+AC	0.64	0.59	0.51	0.66	0.58	0.64	0.60
	s-bert+GM	0.68	0.61	0.44	0.63	0.62	0.60	0.59
	s-bert+AC	0.76	0.71	0.50	0.72	0.65	0.72	0.67
	m-pair	0.92	0.61	0.85	0.53	0.85	0.70	0.73
CWAL	tf-idf+GM	0.76	0.51	0.44	0.65	0.55	0.59	0.59
	tf-idf + AC	0.75	0.50	0.49	0.65	0.56	0.63	0.59
	S-bert+GM	0.76	0.40	0.35	0.69	0.51	0.59	0.55
	s-bert+AC	0.81	0.59	0.47	0.70	0.63	0.72	0.64
	m-pair	0.95	0.43	0.76	0.36	0.81	0.67	0.62
COVID-19	tf-idf+GM	0.40	0.33	0.26	0.55	0.39	0.44	0.38
	tf-idf+AC	0.42	0.35	0.34	0.43	0.41	0.39	0.38
	s-bert+GM	0.56	0.43	0.36	0.57	0.44	0.49	0.48
	s-bert+AC	0.65	0.44	0.37	0.59	0.52	0.50	0.51
	m-pair	0.84	0.80	0.95	0.66	0.79	0.82	0.81

both these tasks. While it is true that our evaluations are based on two historical texts, our methods are generic and can be easily extended to other datasets. The system that we developed is not limited to any actor specific fact (human or location) which, in fact, made the coreference resolution task even more challenging. We believe that our work will open up new and exciting opportunities in history research and education.

References

1. Abadi, M., Agarwal, A., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. Adak, S., Vyas, A., Mukherjee, A., Ambavi, H., Kadasi, P., Singh, M., Patel, S.: Gandhipedia: A one-stop ai-enabled portal for browsing gandhian literature, life-events and his social network. In: JCDL. p. 539–540. New York, NY, USA (2020)
3. Aprosio, A., Tonelli, S.: Recognizing biographical sections in wikipedia. pp. 811–816 (01 2015)
4. Bagga, A., Baldwin, B.: Entity-based cross-document coreferencing using the vector space model. In: Coling. vol. 1, p. 79 (2000)
5. Bamman, D., Smith, N.A.: Unsupervised discovery of biographical structure from text. Transactions of the Association for Computational Linguistics **2**, 363–376 (2014)
6. Barhom, S., Shwartz, V., Eirew, A., Bugert, M., Reimers, N., Dagan, I.: Revisiting joint modeling of cross-document entity and event coreference resolution (2019)

Table 8. Added value of each component in the mention-pair model for each dataset; F: features, S: considering sentence embedding as the only feature, D: date, A: action, T: tag.

D	F	Avg F1	Inc.
CWMG	S	0.613	-
	S+D	0.657	0.044
	S+D+A	0.688	0.031
	S+D+A+T	0.720	0.038
CWAL	S	0.394	-
	S+D	0.544	0.15
	S+D+A	0.560	0.016
	S+D+A+T	0.640	0.008
Covid-19	S	0.791	-
	S+D	0.778	-0.013
	S+D+A	0.811	0.033
	S+D+A+T	0.820	0.009

7. Bedi, H., Patil, S., Hingmire, S., Palshikar, G.: Event timeline generation from history textbooks. In: Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017). pp. 69–77. Asian Federation of Natural Language Processing, Taipei, Taiwan (Dec 2017)
8. Born, L., Bacher, M., Markert, K.: Dataset Reproducibility and IR Methods in Timeline Summarization. In: LREC 2020 (2020)
9. Chen, Z., Ji, H., Haralick, R.: A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In: Proceedings of the Workshop on Events in Emerging Text Types. pp. 17–22. Association for Computational Linguistics, Borovets, Bulgaria (Sep 2009)
10. Choubey, P.K., Huang, R.: Event coreference resolution by iteratively unfolding inter-dependencies among events. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 2124–2133. Association for Computational Linguistics, Copenhagen, Denmark (Sep 2017)
11. Croce, D., Castellucci, G., Basili, R.: GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 2114–2119. Association for Computational Linguistics, Online (Jul 2020)
12. Cybulska, A., Vossen, P.: Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). pp. 4545–4552. European Language Resources Association (ELRA), Reykjavik, Iceland (May 2014)
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019)
14. Ghaddar, A., Langlais, P.: Wikicoref: An english coreference-annotated corpus of wikipedia articles. In: Chair), N.C.C., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S. (eds.) Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA), Paris, France (may 2016)

15. Gholipour Ghalandari, D., Ifrim, G.: Examining the state-of-the-art in news timeline summarization. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 1322–1334. Association for Computational Linguistics, Online (Jul 2020)
16. Hearst, M.A.: Support vector machines. IEEE Intelligent Systems **13**(4), 18–28 (Jul 1998)
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997)
18. Kenyon-Dean, K., Cheung, J.C.K., Precup, D.: Resolving event coreference with supervised representation learning and clustering-oriented regularization (2018)
19. Kibriya, A.M., Frank, E., Pfahringer, B., Holmes, G.: Multinomial naive bayes for text categorization revisited. In: Proceedings of the 17th Australian Joint Conference on Advances in Artificial Intelligence. p. 488–499. AI'04, Springer-Verlag, Berlin, Heidelberg (2004)
20. La Quatra, M., Cagliero, L., Baralis, E., Messina, A., Montagnuolo, M.: Summarize Dates First: A Paradigm Shift in Timeline Summarization, p. 418–427. Association for Computing Machinery, New York, NY, USA (2021)
21. Lu, Y., Lin, H., Tang, J., Han, X., Sun, L.: End-to-end neural event coreference resolution (09 2020)
22. Luo, X.: On coreference resolution performance metrics. (01 2005)
23. Mann, H.B., Whitney, D.R.: On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. The Annals of Mathematical Statistics **18**(1), 50 – 60 (1947)
24. Martschat, S., Markert, K.: 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. pp. 285–290. Association for Computational Linguistics, Valencia, Spain (Apr 2017)
25. Martschat, S., Markert, K.: A temporally sensitive submodularity framework for timeline summarization. In: Proceedings of the 22nd Conference on Computational Natural Language Learning. pp. 230–240. Association for Computational Linguistics, Brussels, Belgium (Oct 2018)
26. Miller, D.: Leveraging bert for extractive text summarization on lectures (2019)
27. Miyato, T., Dai, A.M., Goodfellow, I.: Adversarial training methods for semi-supervised text classification (2017)
28. Moosavi, N.S., Strube, M.: Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 632–642. Association for Computational Linguistics, Berlin, Germany (Aug 2016)
29. Palshikar, G., Pawar, S., Patil, et al.: Extraction of message sequence charts from narrative history text. In: Proceedings of the First Workshop on Narrative Understanding. pp. 28–36. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019)
30. Paszke, A., Gross, S., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
31. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)

32. Preservation, S.A., Trust, M.: The Collected Works of Mahatma Gandhi. https://www.gandhiheritageportal.org/the-collected-works-of-mahatma-gandhi (2013), [Online; accessed 22-February-2020]
33. Pustejovsky, J., Castaño, J., Ingria, R., Saurí, R., Gaizauskas, R., Setzer, A., Katz, G., Radev, D.: Timeml: Robust specification of event and temporal expressions in text. pp. 28–34 (01 2003)
34. Recasens, M., Hovy, E.: Blanc: Implementing the rand index for coreference evaluation. Natural Language Engineering **17**, 485 – 510 (10 2011)
35. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks (2019)
36. Strötgen, J., Gertz, M.: HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In: Proceedings of the 5th International Workshop on Semantic Evaluation. pp. 321–324. Association for Computational Linguistics, Uppsala, Sweden (Jul 2010)
37. Vilain, M., Burger, J., Aberdeen, J., Connolly, D., Hirschman, L.: A model-theoretic coreference scoring scheme. pp. 45–52 (01 1995)
38. Zhang, W., Chen, Q., Chen, Y.: Deep learning based robust text classification method via virtual adversarial training. IEEE Access **8**, 61174–61182 (2020)
39. Zhang, Y., Wallace, B.: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification (2016)