Timeline as a Knowledge Representation for Retrieving Similar Safety Incidents from Industrial Repositories

Sangameshwar Patil, Nitin Ramrakhiyani, Swapnil Hingmire[*]**,**
Alok Kumar, Girish K. Palshikar, Harsimran Bedi[*]
TCS Research, Tata Research Development and Design Centre, Pune, India 411013
{sangameshware.patil, nitin.ramrakhiyani, k.alok9, gk.palshikar}@tcs.com

Abstract

Minimizing safety incidents and their consequent damage has been a high priority for industries. Incident reports capture many details such as causes, failures as well as consequences. However, automated analysis of repositories of incident reports has remained a challenge. In this paper, we propose to apply knowledge-based NLP algorithms to automatically identify similar safety incidents from repositories of historical industrial incidents. We devise a notion of similarity among industrial incident reports based on their event timelines. A timeline is an informative knowledge representation capturing the incident events and their chronological sequence. We construct an event timeline representation of each incident using a transfer learning based event extraction algorithm as well as a novel event temporal ordering approach that makes use of domain-specific knowledge. We then propose an unsupervised, dynamic programming based *TLSim* algorithm to compute the similarity between two event timelines. Paraphrased natural language descriptions of similar events in different incident reports pose a major challenge for computing event timeline similarity. We explore two variants of approximate matching of the event nodes on the timelines: predicate-arguments (PA) representation and the sentence transformer based representation of event node description (ED). Effectiveness of the proposed techniques is experimentally validated on real-life incidents from two different industries: (i) Construction, and (ii) Aviation.

Introduction

Industrial safety incidents (such as accidents, hazards, or near-misses) even though highly undesirable, are an unavoidable reality. Multiple studies estimate that the cost of industrial incidents runs into multiple billion dollars (InjuryFacts 2021; IEN 2017) per annum. Even more importantly, there is an irreparable human cost due to fatalities and major injuries such as permanent disabilities. In most cases, reports summarizing the incidents as well as their investigation are maintained in incident document repositories (Olivares, Rivera, and Mc Leod 2014). For example,

Sample incident #1: On February 1, 2014, at approximately 11:37 a.m., a 340 ft.-high guyed telecommunication tower, suddenly collapsed during upgrading activities. Four employees were working on the tower removing its diagonals. In the process, no temporary supports were installed. As a result of the tower's collapse, two employees were killed and two others were badly injured.
Sample incident #2: On March 25, 2014, two communication towers owned by Union Pacific Railroad collapsed in Blaine, KS, killing two workers. One employee was engaged in disconnecting the 10 ft. diameter dish and another employee was on the same tower approximately 80 ft. from the top. One worker died at the scene and the other was pronounced dead at the hospital.

Table 1: Sample Incidents

Table 1 shows two sample incident report summaries in the construction domain. Multiple stakeholders spend extensive efforts to analyze incidents, identify root causes, suggest preventive actions and conduct trainings (OSHA 2021) to avoid recurrence. However, most of these investigative studies (Latorella and Prabhu 2000; Chettouh, Hamzi, and Benaroua 2016) are carried out with manual analysis. There is little work towards automated processing of repositories of incident reports which can be useful for the above analysis.

In this paper, we propose techniques to facilitate automated analysis of a repository of incident reports using event timelines. Timelines are an important knowledge representation that capture chronological ordering of the events. A timeline is useful in the process of root cause analysis as the causes temporally precede the effect (the incident in this case). To construct an event timeline, we use a state-of-the-art incident event extraction approach to obtain events from incident reports and propose significant improvements to the Document Level Time-anchoring (DLT) algorithm for temporal ordering. We then propose *TLSim*, an unsupervised algorithm that makes use of neural representations of the events and longest approximate common subsequence

[*]Work done when the author was affiliated to TCS Research

Accepted to Workshop on Knowledge Augmented Methods for Natural Language Processing, in conjunction with AAAI 2023.

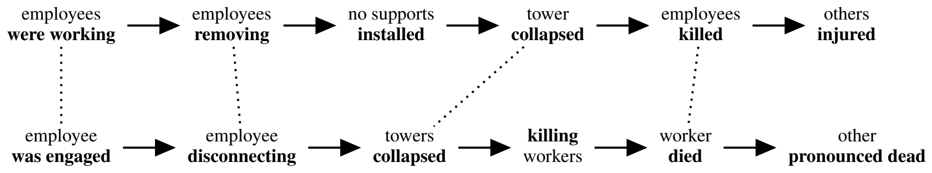

Figure 1: Timelines corresponding to the sample incidents. The similar events are shown using dotted lines.

matching of the timeline representations of incidents to identify similar incident reports. Approximation of the longest common sequence technique is necessary as the mentions of events and their arguments could be paraphrased across different incident reports. Further, the event mentions may be incomplete, and their similarity may be defined in complex, domain-specific ways. To tackle the challenges posed due to variation in describing an event, we explore two variants of approximate matching of the event nodes on the timelines: predicate-arguments (PA) representation and the sentence transformer based representation of event description (ED). As an illustration of the TLSim, Figure 1 shows events, their ordering and desired similarity for the sample incidents presented in Table 1. We show effectiveness of the proposed approach by considering real-life incident reports from two industries, (i) Construction and (ii) Civil Aviation.

Related Work

Several articles such as (Wang et al. 2017; Tanguy et al. 2016; Zhang et al. 2020; Zhong et al. 2020) have focussed on classification of incidents into multiple classes. Wang et al. (2017) use different classifiers to classify reports of incidents happened in a hospital. Tanguy et al. (2016) use SVM for classification and topic models for visualization of aviation safety reports. Rose et al. (2020) provide a clustering and visualization framework for analysis of aviation safety narratives. Zhang et al. (2020) propose a convolutional bidirectional long short-term memory (C-BiLSTM)-based method for classification of construction incident reports obtained from the Occupational Safety and Health Administration website. Zhong et al. (2020) propose a Convolution Neural Network (CNN) model for classification of accident narratives in construction. All these papers classify incident reports of respective domains into multiple classes. However, such document classification based analysis can be considered largely coarse-grained and may not be useful where fine-grained analysis at sentence or token level is necessary. Few other examples of coarse-grained incident report analysis apart from classification are: (i) Tulechki (2015), who proposes a system for measuring similarity between incidents using a bag-of-features approach and, (ii) Liu et al. (2021) employed K-means clustering and co-occurrence network to infer latent causality of pipeline incidents.

Several authors have explored information extraction from incidents in different domains, for example, Osoba (2015) extracts victims, hospitals, police from road accident reports, while Alkadi (2017) extracts crash site and date from air-plane crashes reports. In general, the nuggets of information extracted in these papers are largely independent and would require post-processing such as statistical analysis or further NLP processing such as relation extraction for deriving meaningful insights.

Timelines are an important knowledge representation. Extraction of timelines (Bedi et al. 2017) and its variants (Palshikar et al. 2019a; Hingmire et al. 2020) have been explored in other domains such as history textbooks or software analytics (Palshikar et al. 2019b). However, automated extraction of timeline representation from safety incident reports has not been explored in the literature.

As a progression over the discussed relevant literature, our work considers three important aspects: (i) focus on extraction of fine-grained aspects such as events and their temporal order using state-of-the-art NLP/ML techniques, (ii) generation of the expressive timeline representation and its use for analysis such as incident similarity and, (iii) support for analysis of unstructured textual descriptions of incidents, generally available across industrial domains.

Representing Incidents as Event Timelines

Constructing an event timeline from text involves solving two important information extraction tasks: (i) Identifying event triggers and their arguments, and (ii) Chronologically ordering the extracted events.

Event Trigger and Argument Identification

As *events* are represented as nodes in an event timeline, extracting events from a narrative is an important step in timeline construction. Events in the text of an incident report represent instantaneous occurrences or changes in states of the involved participants.

The task of event-trigger extraction is a sequence classification task and recurrent networks such as BiLSTMs with a CRF decoder require large amounts of data to train. We follow the neural network based transfer learning approach (Ramrakhiyani et al. 2021) for event extraction. First, a BiLSTM-CRF model is pre-trained with event labelled data on general purpose text ECB corpus of generic news reports. This helps the network to learn about “eventiveness” property of verb-based and nominal phrases that describe events. Then, the pre-trained network is fine-tuned on a relatively smaller event annotated dataset of incident reports. This allows us to create a transfer-learnt and effective event extractor focused on industrial incidents.

Arguments of an event predicate define the participant context of an event and are also important for creating an informative label for an event node on a timeline. To obtain arguments of the extracted events, we use the AllenNLP Semantic Role Labelling tool (Gardner et al.

2018). We primarily focus on A0 (agent like) and A1 (Patient/Experiencer/Theme like) arguments.

Temporal Ordering of Events

To construct event timelines, temporal ordering of the identified events is a crucial task. We propose a set of event temporal ordering annotation guidelines that are useful for incident report analysis and are in congruence with the recent research literature (Ning, Subramanian, and Roth 2019; Ning, Wu, and Roth 2018).

Event Ordering Annotation: The event temporal relation extraction literature consists of work on some major datasets - TimeBank (Pustejovsky et al. 2003), TimeBank Dense (Cassidy et al. 2014) and MATRES (Ning, Wu, and Roth 2018). The most important difference among them is the granularity of the tag set. TimeBank Dense (TB-Dense) considers six relations namely BEFORE, AFTER, INCLUDES, INCLUDED_IN, SIMULTANEOUS and VAGUE. However, the more recent work by Ning, Wu, and Roth (2018) for MATRES dataset reduces the inter-event temporal relations to only BEFORE, AFTER, EQUAL (i.e., SIMULTANEOUS) and VAGUE. Han, Ning, and Peng (2019) show that extracting event relations from the denser tag-set of TB-Dense with high accuracy is challenging. We consider the MATRES tag-set to annotate the event relations as it is intuitive to understand for a user consuming the event timeline. We exclude the VAGUE relation from the tagset as it increases the reader's confusion and is not directly usable for the process of timeline construction. We devise a minimal event temporal relation tagging scheme, as follows:

- The annotation scheme in the literature involves tagging relations for every event pair in the same sentence and the consecutive sentence. But this scheme leads to annotation of multiple unnecessary event connections in turn reducing performance and understanding. We restrict the annotator to tag consecutive events in a sentence for a relation. As an example, consider the sentence: `While flying over water, the pilots got distracted by the dolphins splashing in the water`. We focus on annotating temporal relations for the consecutive event pairs (flying, distracted) and (distracted, splashing), only.
- Further, we ask the annotator to identify first and last events temporally in a sentence and only add the annotation between the last event in sentence i and the first event in sentence $i+1$. As an example, consider the consecutive sentences `While flying over water, the pilots got distracted by the dolphins splashing in the water.` and `Inadvertant inputs got fed to the controls leading to the plane yawing.` Here, we observe that the last event for the first sentence is `distracted` and the first event for the second sentence is `fed`. So, we focus on annotating the temporal relation for the event pair (distracted, fed), only.
- One of limitations of the MATRES annotation scheme is that it does not provide any rules for annotating relations with nominal events. We address this limitation in our work. If nominal events occur as arguments to verb-based events, we ask the annotators to mark a SIMULTANEOUS relation between the two. For example, a SIMULTANEOUS relation needs to be marked between `survived` and `excursion` in the sentence `The plane survived the runway excursion`. In all other cases when nominal events occur independently, they need to be treated similar to the verb-based events for relation annotation.
- We also make the annotators ensure that any relations which can be derived through transitivity are not labelled.

INCIDENT: On February 1, 2014, at approximately 11:37 a.m., a 340 ft.-high guyed telecommunication tower, suddenly collapsed during upgrading activities.
BACKGROUND: Four employees were working on the tower removing its diagonals.
BACKGROUND: In the process, no temporary supports were installed.
CONSEQUENCES: As a result of the tower's collapse, two employees were killed and two others were badly injured.

Table 2: Sentence-wise aspect marking - Sample Incident #1

Proposed Event Ordering Approach: Our approach for temporal ordering of events not only makes use of the temporal expressions (TIMEX), but also makes use of insights derived from domain-specific knowledge. We first dwell upon these domain-specific insights which are useful for the temporal ordering of events.

Note that an incident occurs due to a series of events which eventually lead to some aftermath, either serious (e.g., a major injury, a death or damage to equipment or material) or something milder (e.g., a minor injury). We observe that the sentences in an incident report give information on one of the three aspects of the incident: pre-incident background circumstances (referred to as BACKGROUND), description of the incident (INCIDENT) and a post-incident aftermath (CONSEQUENCES).

For instance in Table 2, the Sample incident #1 report is shown with each sentence marked with the corresponding aspect. The first sentence which mentions the collapse of the communication tower is describing the incident. The next two sentences however, give a description of events that were in progress just before the collapse such as the ongoing removal the diagonals. The final sentence describes the aftermath involving death and injury to the employees. In certain cases, a sentence describes more than one aspect (such as the incident and the aftermath) for example, the first sentence in Sample incident #2. However, this mixing of aspects in the same sentences was observed sparingly. We use the following two insightful observations to develop the first step of the temporal ordering component of our approach.

- **Observation 1**: While recording the incident, the author of the incident report would formulate a sentence to encode information about one or more of the three aspects:

(i) BACKGROUND, (ii) INCIDENT, and (iii) CONSEQUENCES. However, to describe the incident she may present these aspects (i.e. the corresponding sentences) in the order she thinks is most important to communicate. Events described as part of the BACKGROUND aspect sentences would have occurred first and would temporally precede the events described in INCIDENT aspect sentences which in turn would precede events described as CONSEQUENCES. This implies that if the sentences can be associated to the aspect they describe, an effective way to order the sentences chronologically can be devised.

- **Observation 2**: These three aspects mentioned above can be the defining dimensions if a comparison needs to be carried out between any two incidents. Similar backgrounds may indicate similar prevalent conditions and possibly causes. Similar incident descriptions may indicate similar series of failure events. More interestingly, similar incident descriptions but dissimilar backgrounds may indicate scenarios with different causes but similar incidents. Similarly, more such scenarios can be deciphered.

The event ordering approach we propose is built on this premise. We divide the task of event temporal ordering into two steps - inter-sentence ordering and intra-sentence ordering. As part of the ***inter-sentence ordering step***, we first make a simplifying assumption that each sentence reports about only one of the three aspects - BACKGROUND, INCIDENT and CONSEQUENCES. We then train a neural network based classifier to perform the classification of each sentence to its corresponding aspect. As output of this step, a list of sentences ordered temporally is obtained by placing the BACKGROUND aspect sentences first, followed by the INCIDENT aspect sentences and then the CONSEQUENCES aspect sentences. The classifier architecture (shown in Figure 2) comprises of

- **a sentence representation layer**: converts every sentence into a vector representation by passing the sentence through a pre-trained BERT-Base model and considering the output CLS representation
- **hidden layer**: reduces the input high-dimensional BERT representation to a smaller dimension, followed by a dropout layer for regularization.
- **output layer**: a softmax activation layer for the three classes output classification

Once the sentences are ordered, events inside each sentence need to be ordered which is carried out as part of the ***intra-sentence ordering step***. We harness the time expressions (TIMEX) as well as temporal cues occurring as lexical markers (`after`, `before`, `when`, etc.) to place intra-sentence events in correct order. The process involves checking the presence of these temporal markers on the lowest common ancestor path between the events and checking their occurrence with respect to the first event in the pair. Nominal events are also handled by establishing a SIMULTANEOUS relation with their dependency parent verbs which are verb-based events. The detailed algorithm for the intra-sentence ordering step is presented in Algorithm 1.

Algorithm 1: *Intra-sentence* Event Temporal Ordering Algorithm

```
Input: Set of events $E$ for a sentence $S$, set of POS tags $P$ for sentence $S$ and set of dependency relations $D$ for sentence $S$
Result: Temporally ordered list of the input events $OE\_LIST$
1  $OE\_LIST := array(0 \dots |E|)$;
2  $order\_number = 1$;
3  for i in $0 \dots |E| - 1$ do
4      for j in $1 \dots |E|$ do
5          if P[i] = 'NOUN' and P[j] = 'VERB' and i in D[j]['children'] and D[j][i]['relation'] in ['nsubj', 'nsubjpass'] then
               // 'SIMULTANEOUS' relation when nominal event is subject of verb-based event
6              $OE\_LIST[order\_number]$.add(i);
7              $OE\_LIST[order\_number]$.add(j);
8              order_number := order_number + 1;
9          else if P[i] = 'VERB' and P[j] = 'NOUN' and j in D[i]['children'] and D[i][j]['relation'] in ['dobj', 'nmod:*'] then
               // 'SIMULTANEOUS' relation when nominal event is object of verb-based event
10             $OE\_LIST[order\_number]$.add(i);
11             $OE\_LIST[order\_number]$.add(j);
12             order_number := order_number + 1;
13         else
14             lca_path = $compute\_lowest\_common\_ancestor(i, j)$
               for k in lca_path do
15                 lca_path_token = $S$[k] if lca_path_token == 'after' and i < k then
                       // event1 happened after event2
16                     $OE\_LIST[order\_number]$.add(j);
17                     order_number := order_number + 1;
18                     $OE\_LIST[order\_number]$.add(i);
19                 else if lca_path_token == 'after' and k < i then
                       // after event1, event2 happened
20                     $OE\_LIST[order\_number]$.add(i);
21                     order_number := order_number + 1;
22                     $OE\_LIST[order\_number]$.add(j);
                   // The above check is made for multiple temporal markers such as 'before', 'while', 'when', 'during', 'at the same time' and appropriate relations are established. The exact pseudocode is skipped for brevity.
23 return $OE\_LIST$;
```

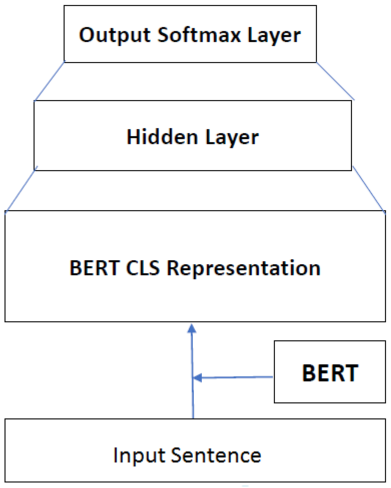

Figure 2: Architecture of the sentence aspect classifier

Using Event Timeline Similarity for Retrieving Similar Incidents

Identifying incidents with similar causes or similar consequences or both can help in devising strategies to minimize the recurrence of similar incidents. It will also help to recommend best practices and better post-incident remedial measures. However, to derive such minute observations and insights, a simple textual query based search of similar incidents may not suffice. We propose that in addition to the index based searching, finding similar event timelines would help in establishing and observing fine-grained facets of incident similarity. The task of finding similar timelines is not straightforward even for human experts owing to the complex nature of the timeline representation. To obtain a quantitative measure of similarity between two event timelines, there is a need to devise an objective definition of the similarity which should allow for an ordinal grading instead of a binary similar/not-similar scoring. Our proposed definition of timeline similarity is grounded in multiple observations we make while manually exploring multiple pairs of timelines to understand their similarity. Some of them are highlighted as follows:

- A higher number of similar events which describe the actual incident, indicate a high similarity between the incident timelines (sim_{actual}).
- If the actual incident is not similar, but there are a high number of similar events which describe the background/context of the incident, a lower value of similarity can be assumed between the incident timelines (say $sim_{context}$ where $sim_{context} < sim_{actual}$).
- If the actual incident and incident contexts are not similar, but there are a high number of similar events which describe the aftermath of the incident, an even lower value of similarity can be assumed between the incident timelines (say $sim_{aftermath}$ where $sim_{aftermath} < sim_{context}$).
- Depending on the number of similar events corresponding to the three facets above, a graded similarity can be assigned on a Likert scale of 0 to 3.

Proposed Approach

We propose an approach to find similar incidents through a combination of an inverted index based similarity and event timeline similarity. As the first filtering step, we build an inverted index on the corpus of incidents and given a query incident, we query the index to retrieve a set of k most similar incidents. We use the standard Information Retrieval (IR) pipeline of stop word removal and lemmatization for both corpus documents and query incidents while creating and querying the inverted index respectively. This index search step is important as the incidents vary in their characteristics widely, and it would be wise to perform the fine-grained timeline comparison only with the incidents having similar overall characteristics (e.g., in aviation domain, an *aircraft fuel starvation* incident may have more similarity with other fuel starvation incidents than, say, *aircraft bird hit* or *runway excursion* incidents). This step also improves computational efficiency of the overall pipeline by reducing the search space of possible matching incidents.

As the second step, we propose the TLSim algorithm to perform an unsupervised timeline similarity matching on the k most similar incidents obtained from the first step. As timelines are ordered sequences of events, a matching algorithm which compares two strings (ordered sequences of characters) can be employed. Further, the matching of items in the sequence need not be contiguous as the timelines may have extra unrelated events between two highly matching events. We can observe such characteristics of event timelines in Figure 1. Considering these factors, we devise a technique based on the Longest Common Subsequence (LCS) (Bergroth, Hakonen, and Raita 2000) approach to match two event timelines. It is important to note that LCS counts a match when the characters under consideration are exactly same, which may not be true for events as similar events may be expressed through different lexical/semantic forms. Hence, we modify the exact matching process to make it “approximate” through embeddings based similarity of the event words and their arguments, while retaining the dynamic programming based computation.

We discuss in detail the event similarity procedure which forms the basis for the TLSim timeline matching algorithm. We consider two ways to represent an event node on the timeline: (i) *“Predicate-Argument” (PA) representation of an event*: We represent an event node as a three-tuple (e, A0, A1) which includes the event trigger e, argument A0 and argument A1. Arguments A0 and A1 represent semantic roles of the event e, such that A0 represents the prototypical agent (i.e., the doer or initiator of the event) and A1 represents patient (i.e., the experiencer, undergoer or affected entity) of the event e. (ii) *“Event Description” (ED) representation of an event*: We construct a sentence-like description of the event by combining the event trigger, argument A0 and argument A1 using syntactic dependencies. The event description is constructed in such a manner that it expresses the event completely as a string or sentence while being succinct. We propose two variants of approximate LCS based TLSim timeline matching which use the two event definitions respectively:

TLSim_PA: In this variant of the TLSim algorithm, we use the predicate and arguments based representation of each event node on the timeline and compute the similarity using word embeddings along with multiple linguistic constraints.

- **Word embedding based similarity**:
  - Similar events should have high cosine similarity between the word embeddings of their respective elements. This implies that two events are similar to each other if there is high word embeddings similarity between the event triggers as well as the entities at their respective semantic roles (A0, A1).
  - For example, in the events `(hit, the beam, the worker)` and `(struck, the beam, the employee)`, apart from the exactly same A0 arguments, both the event triggers and the A1 arguments have high word embeddings based similarity, respectively, leading to prediction of both the events being similar.
- **Linguistic constraints**:
  - Similar events should show negation compatibility. For example, the events `(did not hit, the beam, the worker)` and `(struck, the beam, the employee)` are not similar even though their event phrases (hit and struck) are similar.
  - If two events are antonyms of each other they should not be considered similar. However, if the two events are antonyms but have opposite negation compatibility, then the events are likely to be similar. For example, the events, `(failed to open, the worker, the valve)` and `(did not succeed in opening, the employee, the valve)` are similar to each other.
  - If two events are similar then their respective particles/post-positions should be compatible to each other. For example, in the events `(was climbing up, he, the stairs)` and `(was climbing down, he, the stairs)`, the particles (`up` and `down`) associated with the event phrase `climbing` are antonyms and non-compatible, and hence the events are not similar even though the arguments of the events are same.

We employ two thresholds one for the event trigger similarity (τ_1) and another for argument similarity (τ_2). We find these thresholds empirically and set them as τ_1 = 0.6 and τ_2 = 0.25. If either of the arguments are not extracted, we default on the arguments being similar. Further, for multi-word event or argument phrases, we devise the vector representation as an average of constituent word's embeddings. We use the GloVe based 100 dimensional word embeddings for all the experiments.

TLSim_ED: In this variant of the TLSim algorithm, we use the sentence-level embeddings of the event descriptions from sentence transformers (such as Sentence-BERT (Reimers and Gurevych 2019)). We gauge similarity between event nodes on the two different timelines based on similarity between these sentence representations. Two event nodes are similar if the cosine similarity between the Sentence-BERT representations of their event descriptions is high. For example, the event descriptions `the worker was electrocuted` and `the employee received an electric shock` discuss similar events as they emit highly similar Sentence-BERT representations.

We employ a threshold for the event description similarity (found empirically), setting it as τ_3 = 0.7. We use the sentence transformers model *all-MiniLM-L6-v2* available on HuggingFace for obtaining the sentence representations of the event descriptions.

Algorithm 2: TLSim Timeline Matching

```
Input: Event Sequences $S_1$ and $S_2$,
       variant_type (TLSim_PA or TLSim_ED),
       event similarity threshold $\tau_1$,
       argument similarity threshold $\tau_2$ and
       event description similarity threshold $\tau_3$
Result: Longest subsequence of approximately
        matching events $S_{tlsim}$ and its score
1  $C := array(0 \dots |S_1|, 0 \dots |S_2|)$;
2  for i in $0 \dots |S_1|$ do
3      C[i,0] := 0
4  for i in $0 \dots |S_2|$ do
5      C[0,i] := 0
6  for i in $0 \dots |S_1|$ do
7      for j in $0 \dots |S_2|$ do
8          if approximate_similarity($S_1$[i], $S_2$[j],
             variant_type, $\tau_1$, $\tau_2$, $\tau_3$) then
9              C[i,j] := C[i-1,j-1] + 1
10         else
11             C[i,j] := max(C[i,j-1], C[i-1,j])
12 $S_{tlsim}, score_{tlsim}$ := backtrack(C);
13 return $S_{tlsim}, score_{tlsim}$
```

The TLSim algorithm used to compute the longest approximate common subsequence of events is given in Algorithm 2. The function *approximate_similarity* computes the event level similarities depending on the variant_type specified as input parameter and returns a *True* or *False* match result. Given a pair of event timelines, we compute their matching event subsequence and a match score as the sum of similarity between the matching events. A list of matching incidents, ordered descending on this score, is presented as output.

Datasets

We evaluate the performance of the proposed approach on two real-life incidents datasets from the aviation industry and construction industry. For aviation, we crawl summaries of aircraft incidents from the Skybrary

https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

	LCS (Baseline)	TLSim_PA	TLSim_ED
Aviation			
Object Strikes	0.53	0.56	**0.60**
Equipment Faults	0.58	**0.60**	0.58
Path Excursion	0.65	**0.70**	0.68
Construction			
Electrocution	0.63	**0.65**	0.61
Worker falls	0.53	**0.75**	**0.75**
Vehicle related	0.61	**0.69**	0.68

Table 3: Evaluation - Event Timeline Similarity

repository (https://www.skybrary.aero/index.php/Category: Accidents_and_Incidents) for multiple years, leading to a total of 1225 incidents. For construction industry, we obtain a dataset of 1863 OSHA incident report summaries contributed by Zhang et al. (2020). For evaluating the proposed approach, we select a sample of incidents as query incidents from these datasets and remove them from the rest of the corpus used for finding similar incidents. Considering the wide variety of incidents we group the query incidents based on a common incident type. For aviation, we collect 4 subsets of 10 queries each, relating to waterbody crashes, object hits/strikes (e.g., bird hit), equipment malfunctions and flight path excursions. Similarly, for construction we collect 4 subsets of 10 queries each relating to electrocution, worker falls, asphyxiation and vehicle related accidents.

Evaluation

For incidents in the above datasets, we first construct their event timelines as per the approaches discussed earlier. We execute the index search and TLSim based matching between the query incidents and the corpus incidents and obtain a list of 10 best matching incidents with respect to a query incident based on the match score. To obtain gold standard annotations for incident similarity, we set-up a Likert-scale based grading exercise of the top 5 best incidents for each of the 40 queries in the two datasets. As part of this grading exercise, the annotators were required to annotate each of the top-5 timelines predicted to be most similar by the algorithm for a query timeline on a Likert scale of 0 to 3, where 0 indicates no similarity and 3 indicates very high similarity.

Two annotators were employed and each of them were required to understand the objective definition of incident timeline similarity described earlier. A sample of the queries were repeated between the annotators to compute an agreement score for both Likert-scale based simlarity scores. A Krippendorff's alpha of 0.8941 is observed for the similarity scores assigned by the two annotators indicating high inter-annotator agreement.

We compute a Normalized Discounted Cummulative Gain (NDCG) (Schütze, Manning, and Raghavan 2008) over the scores of the 5 results for each query and then report the average over the scores for queries in each incident domain (Table 3). As a baseline, we compare against an approach which uses the standard longest common subsequence (LCS) algorithm over the index search results. We do not compare with any supervised baseline as no labelled training data on incident similarity is available.

As we can observe, the approximate nature of the TLSim matching based on word/sentence embeddings based similarity, surpasses the performance of the exact matching based LCS algorithm in all of the incident types in both datasets. Also, the similarity based on event description representations from Sentence-BERT (TLSim_ED) seem to work at par with the TLSim_PA variant which works at a fine-grained element level and is equipped with linguistic constraints. In specific cases of Waterbody Crashes and Object Strikes the TLSim_ED variant performs better the predicate argument variant.

It is important to note that incident types such as "Waterbody Crashes" and "Object Strikes" are challenging owing to high embeddings similarity in event arguments such as `sea/water` and `terrain/land` and event triggers such as `crashing` and `striking`. Similarly, for Construction, asphyxiation proves to be a challenging dataset. This is mainly because a major cause of asphyxiation is the trapping of the employee and getting overcome by debris from mud/trench collapses. However, similarity with such collapse like events cause other incidents to crawl up the rank list even though they have no mention of asphyxiation.

Conclusions and Future Work

In this paper, we proposed to represent incident reports through their event timelines, an expressive knowledge representation; and find similar incidents based on their event timelines. In order to compute similarity between two incident reports, we use the TLSim algorithm based on the dynamic programming paradigm. To tackle the challenges posed due to paraphrased natural language descriptions of an event in different incident reports, we explored two variants of approximate matching of the event nodes on the timelines: predicate-arguments (PA) view and the sentence transformer based representation of event node description (ED). The proposed approach was applied to identify similar incidents using two real-world incident datasets from construction and aviation industries. Both variants of the TLSim algorithm work better than the standard longest common subsequence algorithm. The groups of similar incidents identified are useful for identifying recurring patterns in the incident event sequences, root cause analysis as well as to recommend preventive actions to avoid recurrence of similar actions.

As part of future work, we are exploring an ensemble of the TLSim variants to further improve the accuracy. Temporal question generation (Bedi, Patil, and Palshikar 2021) from the timelines of a similar set of incidents could be useful in the incident investigation as well as in root cause analysis process. We also plan to explore the effect of coreference resolution of the entities that appear as the arguments of events (Patil et al. 2018; Gupta et al. 2018) on efficacy of both versions of TLSim algorithm.

References

Bedi, H.; Patil, S.; Hingmire, S.; and Palshikar, G. 2017. Event timeline generation from history textbooks. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA) at IJCNLP, 2017*, 69–77.

Bedi, H.; Patil, S.; and Palshikar, G. 2021. Temporal Question Generation from History Text. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, 408–413.

Bergroth, L.; Hakonen, H.; and Raita, T. 2000. A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, 39–48.

Cassidy, T.; McDowell, B.; Chambers, N.; and Bethard, S. 2014. An Annotation Framework for Dense Event Ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Vol. 2: Short Papers)*.

Chettouh, S.; Hamzi, R.; and Benaroua, K. 2016. Examination of fire and related accidents in Skikda Oil Refinery for the period 2002–2013. *Journal of Loss Prevention in the Process Industries*, 41.

Gardner, M.; Grus, J.; Neumann, M.; Tafjord, O.; Dasigi, P.; Liu, N. F.; Peters, M.; Schmitz, M.; and Zettlemoyer, L. 2018. AllenNLP: A Deep Semantic Natural Language Processing Platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Association for Computational Linguistics.

Gupta, A.; Verma, D.; Pawar, S.; Patil, S.; Hingmire, S.; Palshikar, G. K.; and Bhattacharyya, P. 2018. Identifying participant mentions and resolving their coreferences in legal court judgements. In *International Conference on Text, Speech, and Dialogue*, 153–162. Springer.

H.Alkadi, S. 2017. Rule-based Information Extraction for Airplane Crashes Reports. *International Journal of Computational Linguistics (IJCL)*, 8(1): 1–36.

Han, R.; Ning, Q.; and Peng, N. 2019. Joint Event and Temporal Relation Extraction with Shared Representations and Structured Prediction. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*. Association for Computational Linguistics.

Hingmire, S.; Ramrakhiyani, N.; Singh, A. K.; Patil, S.; Palshikar, G.; Bhattacharyya, P.; and Varma, V. 2020. Extracting Message Sequence Charts from Hindi Narrative Text. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events*, 87–96.

IEN, I. E. N. 2017. The True Cost of Industrial Machinery Accidents. https://www.ien.com/safety/article/20859375/the-true-cost-of-industrial-machinery-accidents. [accessed 1-Dec-2022].

InjuryFacts. 2021. Work Injury Costs. https://injuryfacts.nsc.org/work/costs/work-injury-costs/. [accessed 1-Dec-2022].

Latorella, K. A.; and Prabhu, P. V. 2000. A review of human error in aviation maintenance and inspection. *International Journal of Industrial Ergonomics*, 26(2).

Liu, G.; Boyd, M.; Yu, M.; Halim, S. Z.; and Quddus, N. 2021. Identifying causality and contributory factors of pipeline incidents by employing natural language processing and text mining techniques. *Process Safety and Environmental Protection*, 152: 37–46.

Ning, Q.; Subramanian, S.; and Roth, D. 2019. An Improved Neural Baseline for Temporal Relation Extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Ning, Q.; Wu, H.; and Roth, D. 2018. A Multi-Axis Annotation Scheme for Event Temporal Relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Olivares, R. D. C.; Rivera, S. S.; and Mc Leod, J. E. N. 2014. Database for accidents and incidents in the biodiesel industry. *Journal of Loss Prevention in the Process Industries*, 29.

OSHA. 2021. Occupational Safety and Health Administration. https://www.osha.gov/Publications/3439at-a-glance.pdf. [accessed 1-Dec-2022].

Osoba, O. 2015. *Information Extraction For Road Accident Data*. Master's thesis, The University of Manchester.

Palshikar, G.; Pawar, S.; Patil, S.; Hingmire, S.; Ramrakhiyani, N.; Bedi, H.; Bhattacharyya, P.; and Varma, V. 2019a. Extraction of Message Sequence Charts from Narrative History Text. In *Proceedings of the First Workshop on Narrative Understanding at NAACL 2019*, 28–36.

Palshikar, G.; Ramrakhiyani, N.; Patil, S.; Pawar, S.; Hingmire, S.; Varma, V.; and Bhattacharyya, P. 2019b. Extraction of message sequence charts from software use-case descriptions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2*, 130–137.

Patil, S.; Pawar, S.; Hingmire, S.; Palshikar, G.; Varma, V.; and Bhattacharyya, P. 2018. Identification of alias links among participants in narratives. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, 63–68.

Pustejovsky, J.; Hanks, P.; Sauri, R.; See, A.; Gaizauskas, R.; Setzer, A.; Radev, D.; Sundheim, B.; Day, D.; Ferro, L.; et al. 2003. The timebank corpus. In *Corpus linguistics*, volume 2003, 40. Lancaster, UK.

Ramrakhiyani, N.; Hingmire, S.; Patil, S.; Kumar, A.; and Palshikar, G. 2021. Extracting Events from Industrial Incident Reports. In *Proceedings of the 4th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE 2021)*, 58–67. Association for Computational Linguistics.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in*

Natural Language Processing. Association for Computational Linguistics.

Rose, R. L.; Puranik, T. G.; and Mavris, D. N. 2020. Natural Language Processing Based Method for Clustering and Analysis of Aviation Safety Narratives. *Aerospace*, 7(10).

Schütze, H.; Manning, C. D.; and Raghavan, P. 2008. *Introduction to Information Retrieval*, volume 39. Cambridge University Press Cambridge.

Tanguy, L.; Tulechki, N.; Urieli, A.; Hermann, E.; and Raynal, C. 2016. Natural language processing for aviation safety reports: From classification to interactive analysis. *Computers in Industry*, 78: 80–95.

Tulechki, N. 2015. *Natural language processing of incident and accident reports : application to risk management in civil aviation*. Ph.D. thesis, Université Toulouse le Mirail - Toulouse II.

Wang, Y.; Coiera, E.; Runciman, W.; and Magrabi, F. 2017. Using multiclass classification to automate the identification of patient safety incident reports by type and severity. *BMC medical informatics and decision making*, 17(1): 1–12.

Zhang, J.; Zi, L.; Hou, Y.; Deng, D.; Jiang, W.; and Wang, M. 2020. A C-BiLSTM Approach to Classify Construction Accident Reports. *Applied Sciences*, 10(17): 5754.

Zhong, B.; Pan, X.; Love, P. E.; Ding, L.; and Fang, W. 2020. Deep learning and network analysis: Classifying and visualizing accident narratives in construction. *Automation in Construction*, 113: 103089.