1
Extracting and Aligning Timelines

Mark A. Finlayson[a], Andres Cremisini and Mustafa Ocal

Abstract

Understanding the timeline of a story is a necessary first step for extracting storylines. This is difficult, because timelines are not explicitly given in documents, and parts of a story may be found across multiple documents, either repeated or in fragments. We outline prior work and the state of the art in both timeline extraction and alignment of timelines across documents. With regard to timeline extraction, there has been significant work over the past forty years on representing temporal information in text, but most of it has focused on temporal graphs and not timelines. In the past fifteen years researchers have begun to consider the problem of extracting timelines from these graphs, but the approaches have been incomplete and inexact. We review these approaches and describe recent work of our own that solves timeline extraction exactly. With regard to timeline alignment, most efforts have been focused only on the specific task of cross-document event co-reference (CDEC). Current approaches to CDEC fall into two camps: event-only clustering and joint event-entity clustering, with joint clustering using neural methods achieving state-of-the-art performance. All CDEC approaches rely on document clustering to generate a tractable search space. We note both shortcomings and advantages of these various approaches, and importantly, we describe how CDEC falls short of full timeline alignment extraction. We outline next steps to advance the field toward full timeline alignment across documents that can serve as a foundation for extraction of higher-level, more abstract storylines.

[a] The authors were supported by U.S. Office of Naval Research Grant N00014-17-1-2983 to Dr. Finlayson.

1.1 Introduction

Storylines rarely spring from a text fully formed, neatly and precisely laid out and clear for all to see. Rather, storylines come to us piecemeal, in dribs and drabs, often with multiple storylines intertwined. This is especially evident in news about current events, where a story may unfold across days, weeks, or even years, where specific texts (e.g., news articles written by journalists) often present in detail only the most recent part of the story, or focus on one particular episode, with only quick reviews of prior events included for context. As the chapters of this book discuss, there are numerous processing steps that are necessary to reveal the actual storylines, including general syntactic preprocessing, entity detection, and event and temporal relation extraction **XXX** . In this chapter we focus on two critical steps along the path to revealing storylines, namely, the extraction of *timelines* from texts, and the *alignment* of those timelines with each other.

cite other chapters?

A timeline is a total ordering of the events and times in a text, possibly anchoring some time points to clock or calendar time, and providing metric durations for some intervals. It is important to note that a timeline is not the same as a storyline. A *storyline* is a sequence of interrelated events that tells a specific story of interest, often with a plot or other narrative structure. A specific text might contain part or whole of any number of storylines (including none at all), and those storylines could appear in a wide variety of orders, fragments, or combinations. Indeed, the identity of a storyline is dependent to some degree on the reader, where the storyline might change depending on their goals or interests. In contrast, a *timeline* is a structure that organizes the events and times mentioned in a text into a global ordering. It is one step beyond the *temporal graph* which captures the explicit temporal relationships mentioned or directly implied in the text.

In the timeline extraction and alignment work described in this chapter, we assume that we begin with three basic inputs that themselves have been extracted from a set of texts. First, we assume that we have the events and times mentioned in the texts; this step can be achieved via TimeML event and temporal expression extraction **XXX** . Second, we assume that we have the temporal relationships for each text, which again is a matter of TimeML parsing. Third, we assume that we have the entities and their roles with respect to events. Entity extraction is a well covered topic, and role assignment is covered by SRL or AMR parsing **XXX** .

cite other chapters?

cite other chapters?

Starting from these inputs, in the first half of this chapter (§1.2) we focus on timeline extraction, namely, taking a temporal graph that reflects directly expressed local orders and converting it into a set of timelines that expresses a global ordering. We first define the problem (§1.2.1), and then discuss the significant prior work in this area, which unfortunately are limited and suffer from poor performance (§1.2.2). We then describe our TLEX method, the most recent work on this problem, which solves the timeline extraction problem exactly (§1.2.3).

With timelines in hand, we turn in the second half of the chapter to timeline alignment (§1.3). Timelines extracted from each individual text will need to be aligned globally (§1.3.1) and we concentrate here on the most well-addressed portion of the problem, that of cross-document event co-reference (CDEC; §1.3.2). We review the prior work and state of the art, identifying two main types of approaches: event-only clustering, and joint event-entity clustering. We identify the pros and cons (§1.4), and point toward next steps (§1.5). We conclude the chapter by showing how these two different streams of work can be brought together (§1.6), and summarize the contributions (§1.7).

1.2 Extracting Timelines

1.2.1 Definition of the Problem

A timeline gives a total ordering of events and times, and is useful for a number of natural language understanding tasks. Unfortunately, timelines are rarely explicit in text, and usually cannot be directly read off from the text itself. Instead, texts explicitly reveal only *partial* orderings of events and times. Such information can be used to construct a temporal graph by using a temporal representation language such as a temporal algebra or TimeML, as described below, either through automatic analyzers (Verhagen, 2005), manual annotation (Pustejovsky et al., 2003a), or some combination of the two.

There has been some prior work on extracting timelines from TimeML graphs, but these solutions are incomplete: they do not deal with all possible relations and result in output that can contain ordering errors, a natural result of using supervised machine learning. In contrast, we have showed in our TLEX approach that if we leverage prior work on temporal constraint problem solving and formulate timeline extraction as a *constraint satisfaction problem*, we can achieve a theoretically *exact* solution (Ocal and Finlayson, 2019b). Furthermore, in contrast to prior work, TLEX handles all possible temporal relations.

1.2.2 Prior Work in Timeline Extraction

Temporal Algebras

Allen (1983) proposed a calculus that defines 13 possible relations between time intervals, called Allen's interval algebra. Every interval has a starting time point(I^-) and an ending time point (I^+), and the start point comes strictly before the end point ($I^- < I^+$). Intervals can be related by disjunctive sets of 13 primitive temporal relations (e.g., *before*, *after*, *meets*, etc.). Allen's algebra is especially useful for describing the temporal relationships expressed in text, and using it we can construct temporal graphs by representing events and times mentioned in the texts as intervals, with relationships expressed using Allen's framework.

Allen's algebra is a *qualitative temporal framework* (Barták et al., 2014). In a great deal of later work, Allen's conception of temporal algebras was extended to quantitative cases, such as Simple Temporal Problems (STPs), Temporal Constraint Satisfaction Problems (TCSPs), Disjunctive Temporal Problems (DTPs), and Temporal Networks with Alternatives (TNAs). These types of temporal frameworks allow precise reasoning about the temporal distance between time points as represented in the temporal graphs. Theorists have proved quite a number of formal results regarding both types of frameworks (Barták et al., 2014, §2). While quite useful for planning and scheduling problems, quantitative frameworks are less useful for natural language text (especially narratives and news), which usually do not contain a great deal of precise metric information about time.

Whether quantitative or qualitative, it is possible to *solve* a temporal graph, which means assigning specific time values (or at least, integer order values) to every time point in the graph, which is the same as extracting a timeline.

Temporal Annotation in Language

The gap between formal representations such as Allen's algebra and actual real-world text is bridged by temporal annotation schemes. With regard to time expressions themselves, which includes expressions of when something happened, how often something occurs, or how long something takes, researchers have developed a sequence of TIMEX annotation schemes (Setzer, 2001; Ferro et al., 2001; Pustejovsky et al., 2003b). This allow the annotation of expressions such as *at 3 p.m.* (when), *every 2 days* (how often), or *for 1 hour* (how long). Because events are also involved in temporal relations, these approaches were extended into schemes for capturing both times and events. For example, the Translingual Information Detection, Extraction, and Summarization scheme (TIDES; Ferro et al., 2001) integrates TIMEX2 expressions as well as a scheme for annotating events. TIDES includes annotations for temporal expressions, events, and temporal relations. TIDES uses only six temporal relation types to represent the relationship between events, therefore it gives only a limited view temporal information from texts.

Deficiencies in TIDES led to the development of TimeML (Sauri et al., 2006), another markup language for annotating temporal information, originally targeted at news articles. TimeML added facilities for representing not just Allen's temporal relations, but added several finer-grained temporal relations (such as *immediately before* or *identity*), as well as relations for expressing sub-event structure (aspectual relations) and relations of conditional, hypothetical, or counterfactual nature (subordinating relations). Such an annotation results in a TimeML graph where the nodes represent events, temporal intervals, and time points, and edges represent temporal, aspectual, or subordinating TimeML links between nodes.

Prior Approaches to Timeline Extraction

Kreutzmann and Wolter (2014) showed how to use AND-OR linear programming (LP) to solve *qualitative* (i.e., non-metric) graphs, a set of which includes graphs represented in Allen's algebra. Similarly, Gantner et al. (2008) built the *Generic Qualitative Reasoner* to solve binary qualitative constraint graphs, which takes a calculus description and one or more constraint graphs as input and solves them using path consistency and backtracking. Although these provide approaches for solving qualitative temporal graphs, their methods cannot be applied directly to TimeML graphs because of subordinating relationships.

In contrast to these constraint-based approaches, other NLP researchers have applied machine learning to extract timelines. Do et al. (2012) developed a model to predict associations and temporal relations between pairs of temporal intervals. Combining Integer Linear Programming (ILP) and a collection of local pairwise classifiers, they performed global inference to predict both event-time relations and event-event relations at the same time. Before applying the classifiers, they grouped same events by using event co-reference and they showed event coreference can increase timeline construction performance significantly. Their model attempts to predict both absolute time occurrence for each event in a news article, as well as temporal relations between neighboring events. Because they only look at neighbors, the timeline they extract is necessarily a reflection only of local ordering information. Furthermore, they trained on only three of Allen's temporal relations (BEFORE, AFTER, and OVERLAPS). Their system achieved an accuracy of 73%.

Kolomiyets et al. (2012) proposed a timeline extraction approach using temporal dependency structures over intervals (TDTs, which are trees rather than graphs), again using only a subset of Allen's temporal relations. The main advantage of TDTs are that they can be straightforwardly computed using adapted dependency parsers. This approach took a sequence of event words as input and produced a TDT structure. Although they achieved 70% accuracy in event ordering, the approach only used six temporal relations—BEFORE, AFTER, INCLUDES, IS_INCLUDED, IDENTITY, OVERLAP. Additionally, we have showed that the TDT representation looses significant temporal information relative to temporal graphs (Ocal and Finlayson, 2019a).

Finally, Mani et al. (2006) proposed a machine learning method to partially order events from a qualitative TimeML temporal graph. To order events, their model uses as features the TimeML event class, aspect, modality, negation, event string, and signal. In their experiment they started with gold-standard annotated data, but also expanded their dataset using transitive completions (Verhagen, 2005). They applied Maximum Entropy classifiers to gold standard annotated data. They showed logical axioms for temporal closure can be extremely valuable in addressing data sparseness. However, that method was again demonstrated only with six relations —BEFORE, IBEFORE, BEGINS, ENDS, INCLUDES, and SIMULTANEOUS—and thus excludes large portions of TimeML, and results in roughly only 76% ordering accuracy.

In addition to the fact that all of these systems had imperfect performance, in all cases the methods only consider intervals, rather than start and end points, and so lose much detailed temporal information.

1.2.3 State of the Art in Timeline Extraction

Although the machine learning-based methods mentioned above are useful in terms of generating partial orderings, they suffer from three major problems: they do not handle all possible temporal relations (including subordinating relations); they work only on intervals rather than time points; and their statistical approaches introduce noise into the final result.

In contrast to the above approaches, we designed TLEX (TimeLine EXtraction), a method for extracting a set of exact timelines using all the information available in a TimeML graph (Ocal and Finlayson, 2019b). TLEX achieves perfect accuracy modulo the correctness of an underlying TimeML graph. Like prior work in solving temporal constraint problems, TLEX checks the TimeML graph for consistency, but goes further by automatically identifying inconsistent subgraphs, which allows them to be manually corrected. TLEX outputs one timeline for each temporally connected subgraph, including *subordinated* timelines which represent possible, counterfactual, or conditional situations. These subordinated timelines are connected to the main timeline in a trunk-and-branch structure. We provided a formal argument for TLEX's correctness, as well as an experimental evaluation of TLEX using 385 manually annotated texts comprising 129,860 words across four corpora.

We illustrate TLEX's method using the following example. In this example, each event is underlined and given a numerical subscript for reference.

David's door is knocked$_1$, and he answered$_2$ it. As soon as he opened$_3$ the door, David's neighbor started complaining$_4$ about the noise. He was quickly bored$_5$, but realized$_6$ that if he said$_7$ something, his neighbor would be mad$_8$. So he continued$_9$ to listen$_{10}$.

TLEX takes the full TimeML graph as input, which is shown in Figure 1.1. TLEX first **partitions** the TimeML temporal graph into subgraphs internally connected only with temporal and aspectual links; each of these subgraphs will correspond to an individual timeline (either a *main* timeline or a *subordinated* timeline), and are connected to other subgraphs only via subordinating links. Figure 1.1 shows this partitioning with dashed lines.

TLEX next **transforms** each TimeML temporal subgraph to a temporal constraint graph. As we explained in above, a temporal constraint graph is a graph where nodes are time points and edges are primitive temporal constraints such as $<$ and $=$. We assume every node in the TimeML graph can be represented as an interval I with a starting time point (I^-) and an ending time point (I^+), related by the constraint $I^- < I^+$. Every temporal and aspectual links can then be rewritten as simple conjunctions of temporal primitive constraints. For example, we can rewrite A BEFORE B as $A^+ < B^-$, or A CULMINATES B as

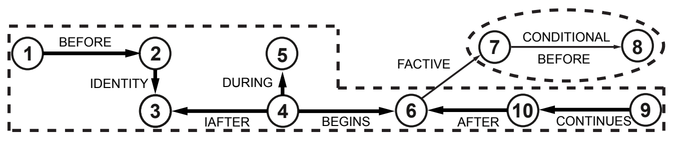

Figure 1.1 Visualization of the TimeML graph from the example. Numbers correspond to the events in the text, and arrows correspond to the temporal, aspectual, or subordinating links. The two temporally and aspectually connected subgraphs are separated by dashed lines, and links on the main timeline are bolded.

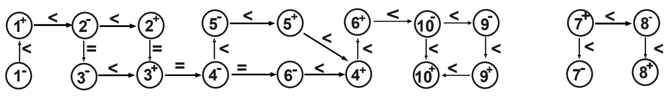

Figure 1.2 The two constraint graphs corresponding to the temporally and aspectually connected subgraphs shown in Figure 1.1. These are produced by replacing each node I with I^- and I^+, and replacing each temporal or aspectual link with the equivalent set of primitive temporal relationships.

$(B^- < A^-) \wedge (A^+ = B^+)$. The temporal constraint graph for the example shown in Figure 1.2.

TLEX next **solves** the temporal constraint graph, using off-the-shelf constraint solvers, to assign integers to interval start and end points. When we order the integers that are assigned for nodes, we will obtain the order of events and times, namely, the *timeline*. The timeline for the example is shown in Figure 1.3.

Barták et al. (2014, p. 20) showed that if there is a solution that satisfies the temporal graph, then the graph must be consistent. If the constraint solver determines that there is no solution, then the TimeML graph must be inconsistent. When TLEX finds an inconsistent graph, it finds all relations that contribute to specific inconsistencies. TLEX merges any inconsistent subgraphs that share relations, and these subgraphs can then be given to annotators to fix those inconsistencies based on the texts.

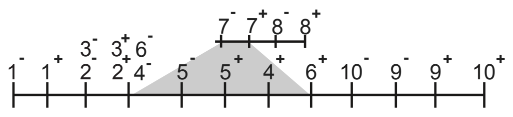

Figure 1.3 Visualization of the timeline extracted from Figure 1.2. The two subgraphs are arranged into a main and subordinated timeline connected by a grey branch.

1.3 Aligning Timelines

1.3.1 Definition of the Problem

Once we have timelines extracted from texts, the next problem is to align those timelines to provide a global, corpus-wide ordering. While anchors to clock and calendar times facilitate such alignment, this information is normally sparse, especially in news and narrative. Aligning timelines thus requires inference of when different timelines refer to the same events. This task is called cross-document event co-reference (CDEC). Even perfect CDEC will not result in a full global alignment, however, because some events will not be mentioned in all timelines. Thus timeline alignment also requires identification of overlapping but otherwise distinct events and time periods—or, at least, identification of indeterminacy of the available information.

Unfortunately, the field has not attacked the full timeline alignment problem. Rather, it has focused primarily on CDEC, and that is the work we will review here. The goal of CDEC is to assign every event mention in a corpus to exactly one set of event mentions, where all the mentions refer to the same event. All existing approaches to CDEC have two steps: first document clustering, followed by event clustering within each document cluster.

CDEC is not restricted exclusively to events that appear in different documents but rather *all* events within a corpus, including those that appear within the same document. Aligning events within a document is a sub-task of CDEC and is called Within Document Event Coreference (WDEC). Although conceptually similar to CDEC, there are some differentiating practical considerations that merit discussion. Most CDEC systems include document information in their feature set when deciding coreference between events, which is not available for WDEC. Additionally, since a document clustering step is not necessary, WDEC reduces solely to event clustering. Importantly, determining pairwise event coreference within-document can actually be a more challenging task than cross-document. For example, our reimplementation of the system of Cybulska and Vossen (2015) shows that while the pairwise event co-reference classifier achieves an F_1 of 0.78 on cross-document event pairs, the same classifier achieves only 0.57 on within-document pairs.

1.3.2 Prior Work and State of the Art in CDEC

ECB & ECB+ Corpus

Most CDEC work has been evaluated on the EventCorefBank (ECB) and EventCorefBank+ (ECB+) corpora, with most using ECB+ because it is larger. ECB was the first corpus developed specifically for CDEC (Bejan and Harabagiu, 2010). It comprises 482 documents selected from GoogleNews, clustered into 43 topics, with each topic containing documents that discuss a specific event, such as the 2009 Indonesian earthquake or the 2008 riots in Greece over a teenager's death. The corpus is annotated using a "bag of events" and entities approach, where co-referring events are all placed into the same group along with their related entities, but relationships between specific entities and events are not recorded. A limitation of this annotation scheme is that it makes it impossible to differentiate events based on their arguments.

ECB+ extends ECB with 500 articles (for a total of 982) that refer to similar but unrelated events across the same 43 topics (Cybulska and Vossen, 2014). For example, the topic with the 2009 Indonesian earthquake was expanded with texts referring to the 2013 Indonesian earthquake. These extra texts were marked with a different sub-topic.

Initial Approaches

As noted, all extant CDEC systems begin with document clustering followed by event clustering. Most CDEC systems approach document clustering with off-the-shelf algorithms, and in the experimental setups used with the ECB+ corpus these algorithms tend to work quite well, though we discuss some subtleties in Section 1.4.

Early CDEC resolution systems used different approaches that were not carried into more recent work. Bejan and Harabagiu (2010) used a Bayesian approach that extends a Dirichlet Process using a mixture model called the Chinese Restaurant Process to find the configuration of event clusters with greatest probability given the data. The authors use gold-standard document clusters, but do not make use of gold-standard event annotations, rather using an event extractor developed in earlier work and augment the predicted events using a semantic parser.

They tested their model on the ECB dataset, and achieved an overall performance of 0.52 CoNLL F_1. This is the only system that reports cross-validation results.

Chen and Ji (2009), in contrast, developed an approach that formulates WDEC as a spectral graph clustering problem. Although this system was tested on the ACE dataset, which only includes WDEC annotations (not CDEC), its performance of 0.836 ECM F-measure is potentially of interest to CDEC work.

Event-Only Clustering

Later approaches to CDEC sub-divide into *event-only* clustering and *joint event-entity* clustering. Cybulska and Vossen (2015) describe a conceptually straightforward, yet strong-performing event-only approach using three decision tree classifiers, one to predict if two documents contain at least one coreferring event pair, one to generate WDEC clusters, and one for merging cross-document WDEC clusters (Cybulska and Vossen, 2015; Vossen and Cybulska, 2018). All the classifiers were trained with features derived from pairs of "event templates," which comprise all event information within a unit of discourse organized into labeled slots (e.g. *action*, *human_participant*, etc.). The attractiveness of this approach lies in is its conceptual uniformity and simplicity, essentially repeating the same process at different levels of granularity.

Kenyon-Dean et al. (2018) describe a different event-only clustering approach that attempts to generate event embeddings for clustering within the hidden layer of a neural network. The paper does not specify if document clustering was performed before CDEC, or if they used gold-standard labels. The authors trained a neural network with a single hidden layer to predict the event cluster of an event given its feature representation (e.g. *word2vec* embeddings). Since their interest was clustering and not classification, however, they constrained the training loss function in such a way as to produce more clusterable event embeddings in the model's hidden layer. As a final step, they use the event embeddings of test set events as input to an agglomerative clustering algorithm.

Joint Event-Entity Clustering

In contrast to event-only clustering, joint event-entity clustering attempts to resolve event and entity coreference concurrently, using information from either step to inform the decisions made by the other.

Lu and Ng (2017) described a system that jointly learns (1) event trigger detection (2) event anaphoricity and (3) event coreference. They only perform WDEC and evaluate their model on the KBP 2016 English and Chinese datasets for event coreference. Their formulation makes explicit use of discourse information within the document to construct a conditional random field (CRF) that performs the classification. Given the vast conceptual differences between KBP 2016 and ECB+ it is difficult to compare results across the two datasets. However, Lu and Ng (2017) report state-of-the-art performance on KBP 2016 at the time.

Lee et al. (2012) described a system that also performs document clustering before computing event and entity clustering. Instead of ingesting gold-standard event and entity labels, they use a publicly available system that performs nominal, pronominal, and verbal mention extraction. After extracting all candidate event or entity mentions, they make use of a publicly available within-document resolution system that applies a series of high precision deterministic rules to decide entity coreference. Using this initial clustering, they trained a linear regressor that predicts the quality of merging two clusters (where quality is defined as the number of correct pairwise links divided by the number of total pairwise links), merging clusters in decreasing order of predicted quality. They did not distinguish between events and entities at clustering time, but rather perform cluster merges using features derived from the relationships between the mentions in two candidate clusters, relying heavily on a semantic role labeler (SRL). They use the ECB dataset, adding a series of event and entity coreference annotations.

Barhom et al. (2019) is a joint entity-event clustering model and is the current state of the art on ECB+. This system performs document clustering using K-means and then uses gold-standard event trigger and entity annotations to generate vector embeddings for events and entities, including both character-, word-, and context-embeddings (ELMo is used for the context embeddings; Peters et al., 2018). Together with these vectors the system uses a *dependency vector*, which is the concatenation of a set of vectors designed to capture inter-dependency between event and entity mentions. For entities, this set includes an embedding for the event head the entity modifies as well as the embeddings for the event heads of all coreferring events. For events, the set includes entity embeddings for each of four event roles (ARG0, ARG1, TMP, LOC) that combine the embedding for the modifying entity mention and the embeddings of all other entity mentions that corefer with the modifying entity. The system computes event and entity clusters iteratively, recomputing the dependency vectors as clusters are merged. They employ an agglomerative clustering algorithm furnished with two trained pairwise prediction functions that output the likelihood that two pairs of events or entities corefer.

1.4 Shortcomings of Current Approaches

Given that all existing CDEC systems use document clustering models to restrict the search space for event clustering, it is important to investigate the role played by errors in document clustering. Importantly, all ECB+ CDEC papers report near-perfect or perfect performance on document clustering on the same test set, topics 36-45 (Barhom et al., 2019; Kenyon-Dean et al., 2018; Cybulska and Vossen, 2015). For example, we confirmed the results reported in Barhom et al. (2019) on topics 36-45 by reimplementing their document clustering method. Yet when we use the same algorithm to perform 5-fold cross validation on document topic clusters of roughly the same size (1-9, 10-18, etc.) on different sections of the corpus, the approach showed performance of anywhere between 2 and 11 percentage points lower across a range of metrics, including homogeneity, completeness, V-measure, and the adjusted Rand index. It is highly likely that this variation in document clustering performance impacts final CDEC performance, as document clustering defines the search space for event clustering.

Only one CDEC paper reports cross-validated performance results, though on the earlier and smaller ECB corpus and using the gold-standard document clustering labels instead of a document clustering model (Bejan and Harabagiu, 2010). No ECB+ papers report on cross-validated CDEC performance.

1.5 Next Steps

Table 1.1 compares performance from a selection of papers from the CDEC literature discussed above. We see a general upward trend in performance over time, with lower performance on ECB than on ECB+; this is of note because ECB+ was intended to be more difficult (Cybulska and Vossen, 2014). The significant performance gains made by Barhom et al. (2019) suggest that a joint event-entity clustering approach is a promising research direction. In fact, taking the joint approach to its logical

Table 1.1 *Performance Comparison of Selected Papers*

Paper	Test Set (ECB or ECB+)	Document Clustering Method	Summary of Approach	Co-NLL F_1
Bejan, 2010	5-fold CV	Gold labels	Bayesian	0.52
Lee, 2012	13-43	EM-based	Joint event/entity	0.56
Kenyon-Dean, 2018	36-45 (+)	?	Event embedding	0.69
Cybulska, 2015	36-45 (+)	DT	Event features	0.73
Barhom, 2019	36-45 (+)	KMeans	Joint event/entity	0.80

conclusion suggests removing the document clustering step altogether and solving the entire task in one shot. Whether or not such an approach would be tractable on large corpora is an open question. Despite this, the performance achieved by Cybulska and Vossen (2015) is noteworthy, given that they describe a conceptually less complex approach than that of Barhom et al. (2019). The performance discrepancies on pairwise event resolution classification within- and cross-document reported in Section 1.3.1 suggest that furnishing the framework introduced by Cybulska and Vossen (2015) with a more sophisticated clustering technique might prove worthwhile. This was one motivation behind the work of Kenyon-Dean et al. (2018), who developed event embeddings specifically tuned for clustering, although they do not report performance improvement on the results of Cybulska and Vossen (2015). Possibly, the more straightforward feature vectors extracted by Cybulska and Vossen (2015) would prove easier to cluster.

In general, a more comprehensive study of how error propagates through the different systems, along with now-standard cross-validation experiments, would seem to be highly informative for future work. This could be at least partly achieved by researchers reporting the results of the subcomponents of their systems (e.g., document clustering, pairwise event classifiers, etc.).

1.6 Bringing it all Together

The motivation bridging timeline extraction and CDEC is the extraction of timelines on a corpus-wide level. Currently, all timeline extraction approaches focus on single documents, with most ingesting gold-standard

WDEC annotations. CDEC identifies clusters of different descriptions of the same event, and as such provides no information about the temporal ordering of the event clusters. Extracting a corpus-wide, cross document event timeline requires merging these two tasks and tackling other as-yet-unaddressed problems. In particular, how do we align event sequences that are not shared between different timelines? CDEC at best provides individual points of alignment, leaving the alignment of interstitial intervals indeterminate.

Beyond corpus-wide timeline alignment, the long-term goal of this work is extracting corpus-wide *storylines*. How do we decide which timelines combine to form a story? How do we define what a story is, and can this definition vary based on the kind of story we wish to extract? These questions lie at the root of the curiosity driving our work on timeline extraction and alignment.

1.7 Conclusion

We have reviewed the literature on both timeline extraction and timeline aligninment, focusing in the latter case specifically on the task of cross-document event coreference (CDEC). Although there has been quite a bit of formal work on temporal representation and problem solving, the most recent approaches to timeline extraction adopted a supervised machine learning paradigm which has fallen short. Our own state-of-the-art TLEX approach has demonstrated an exact solution to the problem, at least in the case of a correct underlying temporal graph. For CDEC, we discussed the two main types of clustering approaches—event-only vs joint event-entity—with systems of the latter form achieving state-of-the-art performance. We outlined several paths forward, noting that document clustering is a key step that has been hitherto not carefully examined. Despite the state of the art, CDEC is not yet fully solved, and even perfect performance on CDEC will not beget the medium-term goal of corpus-wide timeline alignment. Therefore we are still some steps away from the ultimate goal of global, corpus-wide storyline extraction.

References

Allen, James F. 1983. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM*, **26**(11), 832–843.

Barhom, Shany, Shwartz, Vered, Eirew, Alon, Bugert, Michael, Reimers, Nils, and Dagan, Ido. 2019. Revisiting Joint Modeling of Cross-document Entity and Event Coreference Resolution. *arXiv*, **2**.

Barták, R., Morris, R.A., and Venable, K.B. 2014. *An Introduction to Constraint-Based Temporal Reasoning.* Morgan & Claypool Publishers.

Bejan, Cosmin, and Harabagiu, Sanda. 2010. Unsupervised Event Coreference Resolution with Rich Linguistic Features. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 1412–1422.

Chen, Zheng, and Ji, H. 2009. Graph-based event coreference resolution. *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, 54–57.

Cybulska, Agata, and Vossen, Piek. 2014. Using a sledgehammer to crack a nut? Lexical diversity and event coreference resolution. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 4545–4552.

Cybulska, Agata, and Vossen, Piek. 2015. "Bag of Events" Approach to Event Coreference Resolution. Supervised Classification of Event Templates. *International Journal of Computational Linguistics and Applications, IJCLA*, **6**(2), 11–27.

Do, Quang Xuan, Lu, Wei, and Roth, Dan. 2012. Joint Inference for Event Timeline Construction. Pages 677–687 of: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'12).*

Ferro, L., Gerber, L., Mani, I., Sundheim, B., and Wilson, G. 2001. *TIDES Temporal Annotation Guidelines, ver. 1.0.2.* `http://www.timeml.org/terqas/readings/MTRAnnotationGuide_v1_02.pdf`.

Gantner, Zeno, Westphal, Matthias, and Wolfl, Stefan. 2008. GQR - A Fast Reasoner for Binary Qualitative Constraint Calculi. In: *Proceedings of the AAAI'08 Workshop on Spatial and Temporal Reasoning.*

Kenyon-Dean, Kian, Cheung, Jackie Chi Kit, and Precup, Doina. 2018. Resolving Event Coreference with Supervised Representation Learning and Clustering-Oriented Regularization. *arXiv.*

Kolomiyets, Oleksandr, Bethard, Steven, and Moens, Marie-Francine. 2012. Extracting Narrative Timelines As Temporal Dependency Structures. Pages 88–97 of: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL'12).*

Kreutzmann, Arne, and Wolter, Diedrich. 2014. Qualitative Spatial and Temporal Reasoning with AND/OR Linear Programming. Pages 495–500 of: *Proceedings of the Twenty-first European Conference on Artificial Intelligence.* ECAI'14. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Lee, Heeyoung, Recasens, Marta, Chang, Angel, Surdeanu, Mihai, and Jurafsky, Dan. 2012. Joint Entity and Event Coreference Resolution across Documents. *(EMNLP-CoNLL 2012) Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 489–500.

Lu, Jing, and Ng, Vincent. 2017. Joint Learning for Event Coreference Resolution. Pages 90–101 of: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).*

Mani, Inderjeet, Verhagen, Marc, Wellner, Ben, Lee, Chong Min, and Pustejovsky, James. 2006. Machine Learning of Temporal Relations. Pages 753–760 of: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ICCL-ACL'06).* Sydney, Australia.

Ocal, Mustafa, and Finlayson, Mark Alan. 2019a. On the Deficiencies of Temporal Dependency Trees (under review). In: *Proceedings of the 34th Conference of the Association for the Advancement of Artificial Intelligence (AAAI 2020).*

Ocal, Mustafa, and Finlayson, Mark Alan. 2019b. TLEX: A Formally Correct Method for Extracting Exact Timelines from TimeML Temporal Graphs (under review). *Journal of Artificial Intelligence Research.*

Peters, Matthew E., Neumann, Mark, Iyyer, Mohit, Gardner, Matt, Clark, Christopher, Lee, Kenton, and Zettlemoyer, Luke. 2018. Deep contextualized word representations. *CoRR*, **abs/1802.05365**.

Pustejovsky, James, Hanks, Patrick, Saur, Roser, See, Andrew, Gaizauskas, Rob, Setzer, Andrea, Radev, Dragomir, Sundheim, Beth, Day, David, Ferro, Lisa, and Lazo, Marcia. 2003a. The TimeBank corpus. Pages 647–656 of: *Proceedings of Corpus Linguistics Conference.* Lancaster, UK.

Pustejovsky, James, Castaño, José, Ingria, Robert, Saurí, Roser, Gaizauskas, Robert, Setzer, Andrea, and Katz, Graham. 2003b. TimeML: Robust specification of event and temporal expressions in text. Pages 1–11 of: *Fifth International Workshop on Computational Semantics (IWCS-5).*

Sauri, Roser, Littman, Jessica, Gaizauskas, Robert, Setzer, Andrea, and Pustejovsky, James. 2006. *TimeML Annotation Guidelines, Version 1.2.1.* `https://catalog.ldc.upenn.edu/docs/LDC2006T08/timeml_annguide_1.2.1.pdf`.

Setzer, Andrea. 2001. *Temporal Information in Newswire Articles: an Annotation Scheme and Corpus Study.* Ph.D. thesis, University of Sheffield.

Verhagen, Marc. 2005. Temporal Closure in an Annotation Environment. *Language Resources and Evaluation*, **39**(05), 211–241.

Vossen, Piek, and Cybulska, Agata. 2018. Identity and Granularity of Events in Text. Pages 501–522 of: *Lecture Notes in Computer Science*, vol. 9624 LNCS. Springer-Verlag.