Towards Learning Action Models From Narrative Text Through Extraction and Ordering of Structured Events

Ruiqi Li[1], Patrik Haslum[1], and Leyang Cui[2]

[1] Australian National University, Canberra Australia
{ruiqi.li,patrik.haslum}@anu.edu.au
[2] Tencent AI Lab, Shenzhen China
leyangcui@tencent.com

Abstract. Event models, in the form of scripts, frames, or precondition/effect axioms, allow for reasoning about the causal and motivational connections between events in a story, and thus are central to AI understanding and generating narratives. However, previous efforts to learn general structured event models from text have overlooked important challenges raised by the narrative text, such as the complex (nested) event arguments and inferring the order and actuality of mentioned events. We present an NLP pipeline for extracting (partially) ordered, structured event representations for use in learning general event models from three large text corpora. We address each of the challenges that we identify to some degree, but also conclude that they raise open problems for future research.

Keywords: event model acquisition · event extraction · event ordering.

1 Introduction

Narratives are made up of events. Events tell us what happened, who or what was involved, and in what order. Knowledge about typical events, such as their usual participants, dependencies and effects, is crucial to understanding, and creating, narrative text. Event models can take several forms, from simple scripts or schemas – subsequences of events, parameterised by their participants, that frequently occur together [10] – to models that link each schematic known event to the possible preceding or following events [11], which may be ranked by likelihood [38], to state/transition models that associate to each event the conditions on the world state under which it can occur, and the effects of its occurrence on the state. Model-based reasoning about events/actions[3], studied in AI domains such as reasoning about action and change [39], planning [15] and diagnosis of dynamical systems [20], assumes such a symbolic preconditions-and-effects model of events/actions is available.

Hand-crafting event models with the breadth required for narrative understanding or generation is an enormous task. Hence our interest in learning them from text, a medium that hosts an abundance of human-authored narratives in many styles and genres. We aim to explore the hypothesis that by processing a sufficient volume of text we

[3] We will use the terms event and action interchangeably. Although a distinction can be made – actions have at least one intentional participant or actor – from the point of view of the models we consider, they are largely the same.

can extract sufficiently many distinct events with a sufficient number of occurrences to learn meaningful models of them. The problem of learning various kinds of event models from text has been researched for well over a decade [10, 36, 38], and has recently gained renewed interest in AI planning [13, 23, 24, 27] for several reasons, including narrative generation [17, 18]. The latter breaks the problem into two stages: first, extracting structured and ordered (typically as sequences) representations of event mentions from source texts and, second, inducing event models from these examples of event sequences, leveraging existing work on planning model acquisition [e.g., 2, 8]. The work we present in this paper also follows this approach. However, we argue that previous work has overlooked several characteristics of narrative text, which raise challenges both for extraction of event mentions from text and for model acquisition.

Event ordering: In simpler texts, such as recipes or instruction manuals, events/actions are for the most part written in the order they occur [13, 18, 23, 24, 27]. In these cases, it is acceptable to assume that the event sequence corresponds to the sequence of event mentions in the text. In narrative texts, however, such linearity is rarely the case. As one point of reference, in the MATRES [32] dataset, which contains 275 news articles manually annotated with temporal relations between events, approximately 58.2% of the events in each article indeed do not occur before all of the events mentioned later in the text.

Olmo et al. [33] also note that inferring plan order from text "... is a feature which is mostly missing in the previous task-specific state of the art", but test the ability of a model to perform such inferences only with three hand-crafted examples.

Events as arguments: Many verbs can take, or require, a clausal complement, i.e., an argument of the event verb is itself an event. For example, if "she tries to stop them ...", the event verb is "try", and the event "[she] stop them" is its argument. Furthermore, event arguments can be nested: if "she tries to stop them finding out ...", "find out" is an argument of "stop", and the event "[she] stop ([they] find out)" is itself the argument of "try". Events of this kind are frequent in narrative texts: in each of the three datasets we examine, 16.4%, 14% and 14.7%, respectively, of event mentions are arguments or conditions. Distinguishing them is important for both accurately representing the mentioned events, and for identifying those events that actually take place.

Event actuality: Narrative texts very frequently mention events that are not actually part of the sequence of events that is recounted. For example, consider the sentence "King Louie offers to help Mowgli stay in the jungle if he will tell Louie how to make fire like other humans." (from [18] Figure 5). Although it contains many event verbs ("help", "stay", "tell", "make"), only one of them, "offer", is actually said to happen: the remaining are the argument and condition of the offer. Events that are arguments are only one source of event mentions that do not correspond to actual event occurrences.

Scale: Finally, the size of the model required for general narrative understanding and generation is significantly greater than what has been considered in previous work. For example, the Story Cloze dataset [30], which tests a form of narrative understanding, supposes knowledge of over 2,000 distinct verbs.

In this paper, we present an NLP pipeline for extracting (partially) ordered, structured event representations, that to some degree deals with the above challenges. We apply

it to three sets of texts: the movie plot summaries provided by Bamman et al. [3], a subset of articles from the Goodnews dataset [6], and a set of New York Times articles obtained from kaggle[4]. The partially ordered sets of event occurrences produced by our pipeline are meant as a basis for inducing event models. Furthermore, we report a preliminary experiment extracting simple narrative chains (recurring subsequences of connected events). However, our aim is to induce knowledge such as event preconditions and effects. The necessary complexity of the events we extract – nested argument events, conditionals, etc – as well as the scale of the task, are beyond the capabilities of current model acquisition techniques [see, e.g., 2, 8]. We plan to pursue this goal in future work. We will also make the generated dataset public so that other researchers can attempt to tackle the challenge of learning models from it.

2 Related work

Learning of event models from text has been studied for some time. Chambers & Jurafsky [9, 10] proposed a system for unsupervised induction of parameterised scripts from text; applied to news text from the Gigaword corpus, it generated scripts covering around 1800 verbs. But it is restricted to finding scripts in which one entity, i.e., the protagonist, is common to all events. Tandon et al. [38] analysed movie and television scripts, as well as novels, to extract over 2,000 activities (verb, optional preposition, and object, e.g., "chase car") and link them with typical values for location, time and participants, as well as likely preceding and following activities. However, activity participants are ground, i.e., objects rather than parameters (e.g., agents of "loom in distance" are "door" and "island"). The Never-Ending Language Learning (NELL, http://rtw.ml.cmu.edu/rtw/) project extracted over 50M beliefs, in the form of entity–relation triples, from the web, combined with crowdsourced feedback. However, it is focused on acquiring instance-level facts, and mainly about entities, not general activities. Although it has categories for "event" and different types of actions, the majority refer to specific event instances (e.g., product launches, elections, etc) and information about them is sparse: 99% have no relation but a hypernym and a source URL.

Extracting sequences of actions or events from text has recently gained interest in AI planning [13, 23, 24, 27]. In some instances, this is a precursor to learning the kinds of precondition/effect action models that AI planners require, using planning model acquisition tools. Methods of event extraction vary, from using the dependency parse structure to neural language models and reinforcement learning. However, previous work in this line has overlooked several of the complexities of narrative events. As noted by Olmo et al. [33], they have assumed the order of events is their order of mention. Furthermore, none identify events that are arguments or conditions of other events. For instance, the plan representation generated by Hayton et al. [18] from a synopsis of the movie "The Jungle Book" includes many of the events mentioned in the sentence quoted in the Introduction ("stay in jungle", "tell" and "make fire") even though none of these actually happen in the story.

[4] https://www.kaggle.com/datasets/nzalake52/new-york-times-articles

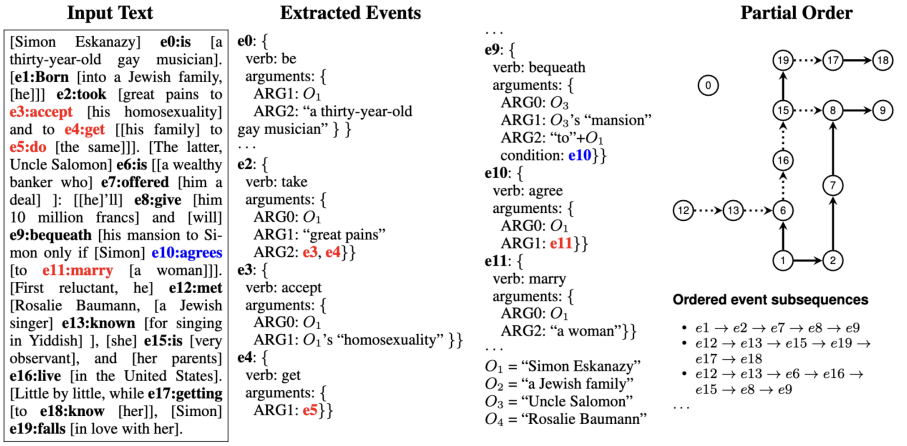

Fig. 1: An example of partially ordered events extracted by the pipeline. Left: Source text. (The mark-up of verbs and arguments of extracted events are not part of the input, just for improving readability) Middle: Extracted event samples. Events that are arguments are in red, conditions are in blue. Right: Predicted partial order between events, and possible event traces. Dashed edges indicate predictions we consider incorrect.

3 Structured Event Extraction

The ordered event extraction process has three main steps: (1) Extracting event verbs and arguments from the source text. This is done using the BERT-based AllenNLP (https://allenai.org/allennlp) semantic role labelling (SRL) system. We also use POS tagging and dependency parse information, obtained using the Stanford CoreNLP toolkit [25], in several ways. First, the SRL system often mistakes adjectives for verbs (e.g., "reloaded" in "the Matrix reloaded"), and we use the words' POS tags to filter these out. We also filter out modal verbs. Second, the SRL system does not detect phrasal verbs, so we use a rule-based method that relies on the dependency parsing to identify these. (2) Identifying argument and condition events. As mentioned, events may be arguments or conditions of other events. We use the argument structure provided by the SRL system, together with a rule-based method that relies on the dependency parse information, to determine which events are arguments or conditions of other events. We term any event that is not an argument or condition *independent*, and we assume that the non-negated independent events are those that, according to the text, actually take place. This is a simplification, as some argument events can also be actual. (3) Ordering the independent events. We use a temporal relation classifier [31] to predict temporal relations between the pairs of independent events, and inference (transitivity) to extend the partial order. Because the classifier is myopic, i.e., predicts each event pair in isolation, it often induces inconsistent orders (with cyclic precedences). We resolve these by deleting the predicted order with the lowest probability in each cycle.

Event Verb and Arguments An event mention e consists of a verb or phrasal verb $V(e)$ and a set of labelled arguments $\mathbb{A}(e)$. Verbs are lemmatised. We call any event whose verb lemma is "be" or "have" a *statement*, since these describe facts or circumstances rather than events occurring. The extraction and ordering of statements is exactly the same as for other events, but it is useful to make the distinction for some results analysis. In the rest of the paper, we use the term "event" to mean both statements and other events except where specifically stated otherwise.

The SRL system follows the PropBank schema [7], which divides argument labels into numbered arguments (ARG0–ARG5), for arguments required for the valency of an action (e.g., agent, patient, etc.), and modifiers of the verb, such as purpose (PRP), locative (LOC), and so on. Argument values are spans of text. Arguments (without their label) of extracted events are shown with brackets in the source text in Figure 1.

Phrasal Verb Detection Phrasal verbs are common in English, and identifying them is important because the meaning of a phrasal verb is often different from that of the verb part of it (e.g., the meaning of "make up" is different from "make"; this is distinct from the fact that "make up" also has several meanings). The SRL system, however, extracts only single verbs. We use the following rule, adapted from [19], to detect phrasal verbs: If a word P either (i) has a *compound:prt* relation with the event verb W, or (ii) is adjacent to the event verb W and has a *case* or *mark* relation with W in the dependency parse tree, then WP is a candidate phrasal verb; it is accepted if it appears in a list of known phrasal verbs[5]. We performed a small-scale evaluation of the accuracy of this method: of 100 randomly sampled extracted events where rules (i) or (ii) apply, 58 of the candidate phrases appear in the known phrasal verbs list. Manually checking these, 47 are actual phrasal verbs, i.e., the precision is 0.81. The false positives are sentences where the phrase is used in a literal sense (e.g., "take off" in "takes off his clothes").

Argument Event Detection Because arguments are spans of text, part or all of an extracted event may lie within the argument of another event. If $V(e_j)$ is within an argument of e_i, we say e_j is *contained* in e_i. This can be nested. Contained events are candidates for being arguments, but are not necessarily so. For example, in Figure 1, ARG2 of **e6** ("is") contains $V($**e7**$)$ ("offer"), but **e7** is not an argument of **e6**.

We designed the following rules: If any of them is satisfied, a contained event e_j is an argument of the containing event e_i: (i) The dependency relation $V(e_i)$ to $V(e_j)$ is clausal complement (*ccomp* or *xcomp*) or clausal subject (*csubj*). (ii) The dependency relation from $V(e_j)$ to $V(e_i)$ is copula (*cop*). (iii) All of e_j is contained in an argument of e_i that is labelled with either ARGM-PRP ("purpose") or ARGM-PNC ("purpose not cause").

Condition Event Detection We commonly find conditional promises, threats, etc, in narrative text. Conditional offers are found in both the example from The Jungle Book quoted in the introduction and the example in Figure 1. The condition event is not an

[5] https://en.wiktionary.org/wiki/Category:English_phrasal_verbs

argument of the offer, but it is also not actual; hence, a different mechanism is required to identify conditions.

We use a method based on the signal words and phrases "if", "whenever", "as long as", "on [the] condition that", and "provided that". For example, in Figure 1 the signal word "if" is in between the consequence **e9** and the condition **e10**. Our method is a modification of that introduced by [34]. Event e_j is determined to be a condition of e_i iff (a) one of the subsequences $V(e_i) \quad S \quad V(e_j)$ or $S \quad V(e_j) \quad V(e_i)$, where S is one of the signal words/phrases, appear in the sentence, with no other event verb appearing in the subsequence; and (b) one of the following holds:

S1: $V(e_i)$'s tense is future simple, $V(e_j)$'s tense is present simple;
S2: $V(e_i)$'s tense is present simple, $V(e_j)$'s tense is present simple;
S3: "must" or "should" or "may" or "might" is adjacent to $V(e_i)$, the tense of $V(e_j)$ is present simple;
S4: "would" is adjacent to $V(e_i)$ and $V(e_i)$'s tense is infinitive, $V(e_j)$ is past simple;
S5: "could" or "might" is adjacent to $V(e_i)$, $V(e_j)$'s tense is past simple;
S6: $V(e_i)$ is preceded by "could" and its tense is infinitive, the tense of $V(e_j)$ is past continuous or past perfect;
S7: "would have" or "might have" or "could have" is adjacent to $V(e_i)$ and the tense of $V(e_i)$ is past participle, the tense of $V(e_j)$ is past perfect;
S8: $V(e_i)$'s tense is perfect conditional continuous, $V(e_j)$'s tense is past perfect;
S9: $V(e_i)$'s tense is perfect conditional, $V(e_j)$'s tense is past perfect continuous;
S10: "would be" is adjacent to $V(e_i)$ and $V(e_i)$'s tense is gerund, $V(e_j)$ is past perfect;

Tenses of event verbs are determined from their POS tags.

Entity Resolution In the event representation that we generate, the exact words used to denote entities in event arguments do not matter, as long as we identify repeat mentions of entities throughout the narrative. (The words that denote an entity may of course carry important information about the entity type, or relation to other entities.) Hence, we apply co-reference resolution, and substitute the first mention of any resolved entity for later mentions. We use a document-level inference-based LSTM model [22] from AllenNLP for the co-reference resolution task. While there are potentially better co-reference resolution methods [18], this aspect of the pipeline is somewhat orthogonal, and we leave its further development for future work.

4 Event Ordering

To determine the relative order of the events narrated within a document, we build a temporal relation classifier. We order only the independent events, i.e., not argument events or conditions, since the independent events are, the ones that are actually said to happen. Although temporal relations can exist between argument events, their actuality is also uncertain. Following the method proposed by Ning et al. [31], we trained an LSTM-based classifier using the temporal relation dataset MATRES [32]. Earlier studies of temporal relation extraction [e.g. 5, 21, 29] have adhered to the TimeML standard

[35], which uses fourteen different temporal relations between events and temporal expressions. In contrast, the MATRES dataset considerably simplifies the task by using only four relations: *before*, *after*, *equal* and *vague* (the last meaning no or unknown relation). However, because these four seem sufficient to establish a partial order between events, and because earlier studies have reported low prediction accuracies for the TimeML relation set, we choose the simpler task variant.

The predictor is trained on pairs of events within the same or adjacent sentences and keeps only relations with a probability of at least 0.5. *after* relations are reversed and converted to *before*, giving a network of precedence and equality relations. We then use transitivity to infer additional precedences. Because the classifier is myopic, i.e., predicts each event pair in isolation, it often induces inconsistent orders (with cyclic precedences). We resolve these by deleting the predicted ordering with the lowest probability in each cycle. An example of the resulting partial order is shown in Figure 1. On average, we find 2.7 subnetworks per document, with an average size of 22 events.

Table 1: Number of distinct event chains that meet the selection criteria, and the number of distinct event chains with more than one occurrence. Numbers are subgrouped by the genre of the document in which chains appear: CT: crime thriller, RC: romantic comedy. "Across" means chains that occur in documents in both genres, while the total includes both chains recurring in documents within one genre only and across both.

		M1			**M2**		
		N=3	**N=4**	**N=5**	**N=3**	**N=4**	**N=5**
# of distinct event chains	CT	191,933	388,455	782,441	197,503	401,840	818,449
	RC	216,752	474,706	1,041,439	223,520	487,491	1,071,732
	Total	404,971	862,925	1,823,004	419,119	888,348	1,889,313
# of recurring chains	CT	1,712	104	5	676	20	1
	RC	2,417	160	8	816	39	1
	Across	2,948	203	14	1,128	40	1
	Total	6,015	453	27	2,358	99	3

5 Narrative Chain Extraction

Finally, we perform a preliminary experiment in acquiring simple scripts, or narrative event chains [9], from the extracted partially ordered events. A narrative event chain is a subsequence of N non-statement events that are all connected by sharing some coreferring argument with another event in the chain. (It does not have to be one entity that is shared by all events. For example, if A is an argument of events 1 and 3, and B is an argument of events 2 and 3, the chain is connected.) Events in the chain must be totally ordered, but not necessarily consecutive. We are interested in event chains that recur, in particular across documents, since these represent general elements of narrative. For two event chains to match, the event verbs and arguments must match. We consider

two different conditions for arguments matching: The first (**M1**) is that there must be a mapping between the labelled arguments of the two instances of the chain that respects argument recurrence within the chain (i.e., if A in one instance maps to B in the other, B must recur in the same argument positions of the corresponding events that A does). The second (**M2**) requires in addition that argument and condition events of events in the chains recursively match, under the same condition.

We applied chain extraction to the partially ordered events extracted from a subset of the Movie summaries dataset, comprising those with genre label "crime thriller" (1678 documents) and "romantic comedy" (2069 documents), for $N = 3, 4, 5$. Results are summarised in Table 1. We count repetitions of event chains only when they are repeated in different documents, because repetitions within a document tend to have large overlap. As expected, increasing the length of the chain or applying the stricter argument matching condition **M2** substantially reduces the number of recurring event chains found. In this experiment, we use a straightforward form of matching events verbs: they match if their lemma is the same. Applying sense disambiguation, i.e., matching verbs only when used in the same sense, could reduce the number of recurring event chains. Meanwhile, applying semantic generalization that matches distinct verbs when their senses satisfy some degree of semantic similarity, would likely increase the number.

6 Challenges for NLP Research

Several tasks in our event extraction pipeline pose open challenges for NLP research.

Event and Argument Extraction Some aspects of the current SRL systems can be improved: First, identifying phrasal verbs, and distinguishing their occurrence from literal uses of the same phrase. The method we have used depends on a given list of known phrasal verbs (without this filter, its precision would be too poor) and also cannot recognise discontinuous phrasal verbs with more than two words, such as "put ... down to", which is different from "put down". Second, arguments identified by the SRL system often capture independent event mentions, particularly when those occur in relative clauses of a nominal argument. Our argument event classifier tries to resolve these cases, but still misses some.

Event Actuality Related to deciding which events are arguments of other events is determining which of the mentioned events are actual, i.e., which have, according to the narrative, taken place. We have considered any non-negated non-argument and non-condition event to be actual. This is, however, an approximation. First, because argument events can also be actual; this depends on the verb they are arguments of. Consider "she thought it rained outside" and "she could see it rained outside": in both, "rain" is an argument, but in the second example it is also actual. Second, some temporal expressions give rise to non-argument events that are also not actual. Consider, for example, "before she falls down, she catches the railing and steadies herself" (implying the "fall" does not actually happen). We are not aware of prior work on resolving event actuality. As the examples above show, this is closely linked with the meanings of the verbs involved. This is a significant research challenge.

Conditional Event Detection The problem of detecting condition–consequence relations between events in text has been studied, motivated in particular by finding causal relationships [34]. We evaluated two recent methods that detect conditional structures, called CNC [14] and CiRA [37], respectively. Both are BERT-based neural networks, but trained with different data. CiRA use an annotated set of requirements documents, while CNC annotated and used a set of news articles, together with the Penn Discourse Treebank 3.0 [40] and CausalTimeBank [28] datasets. However, we also note that both are intended to extract causal relations between events, which do not always coincide with the condition–consequence relation.

	Precision	**Recall**	**EM-rate**
CiRA [14]	0.75	0.79	0.41
CNC [37]	0.80	**0.80**	0.1
Ours	**0.93**	0.71	**0.85**

Table 2: Precision and recall of detecting the existence of conditionals in sentences. EM-rate is the proportion of sentences in which the detected condition and consequence events exactly match our annotation.

We apply these two systems to the same set of 100 randomly selected sentences from the Movie summary dataset that we used to evaluate our rule-based method. Recall that these were selected to include the five signal words or phrases that we use (20 for each) and that 75 of them contain conditionals. 3 sentences have more than one condition–consequence event pair. Both systems detect the presence of conditionals in a sentence in more cases than our method (59 and 60 of the 75 positive cases, respectively, compared to 53 for our method), but also have a much higher number of false positives (20 and 15 of the 25 negative cases, respectively, compared to 4 for our method), leading to their lower precision, as shown in Table 2. Furthermore, in true positive cases identified by each, we compare the events identified as conditions and consequences with our annotation. These results are worse: CNC identifies the correct text spans in only 6 of the 60 cases (EM-rate=0.1), while CiRA does so in 24 of the 59 cases. On the other hand, our method is blind to any conditional expression that does not use one of the five signal words or phrases. We conclude that more research on this aspect of relations between events is warranted.

Event Ordering Predicting the right order of event mentions remains a hard problem. For instance, in the example in Figure 1, six of the thirteen precedence relations are incorrect. (The precision of the classifier is actually better than this would suggest, as it makes many correct predictions that are transitively implied by those shown in the figure.) There are also some event pairs that arguably should be ordered, but which are not detected, e.g., **e7** before **e12**, **e7** before **e0** and **e1** before **e0**. We note that four of the falsely predicted precedences are associated with statements (**e0**, **e6** and **e15**), suggesting that these pose particular difficulty for the classifier. Although Ning et al. [31]

report much better accuracy on the simplified 4-relation prediction task than previously achieved by state-of-the-art predictors for the full TimeML relation set, the difficulty caused by statements, which often describe enduring circumstances, suggests that including a *during* relation may help.

Causal relations between events imply a temporal ordering (i.e., cause precedes effect). However, current state-of-the-art in predicting causal relations [e.g., 14, 29, 37] did not perform well on narrative text in our test. We did not test Mirza & Tonelli's system, because it depends on features such as verb mood and aspect, which are available in the annotated CausalTimebank dataset but not in our unannotated source texts, and because CNC achieved better prediction performance when evaluated on the same data. Another potential source of information is the super–sub relation: an event is defined to be a sub-event of another iff it occurs *during* the super-event and is spatially *contained* by the super-event [16]. The precision of current state-of-the-art sub-event detection methods is, however, mostly low [1, 4].

7 Challenges for Model Acquisition

The problem of inducing action preconditions and effects from observations of plans has been studied in AI planning for some time (e.g., [12, 41]). Approaches vary in the assumptions they make, such as whether observations are complete or partial, precise or noisy, whether only actions/events are observed or also (full or partial) states (i.e., statements), and so on (see, e.g., [2, 8]). However, we are not aware of any method that, off-the-shelf, can successfully exploit the event information that we can extract from text: partially ordered, with partial state observations, and some degree of uncertainty in all parts (i.e., which events actually occur, their arguments, and order). Furthermore, the complex structure of events, with nested argument events and conditions, is not represented in standard planning formalisms (e.g., PDDL [26]), and not supported by current model acquisition methods.

Another challenge raised by learning models from events extracted from text is sense disambiguation and semantic generalization, of both verbs and arguments. While this can be handled as a separate task in between extraction and model acquisition, integrating with the latter, i.e., deciding which general event each occurrence is an instance of, and what level from a hierarchy of types to assign each of its arguments, jointly with inducing the models of general events, may lead to better models.

8 Conclusions

Narrative text exhibits many of the complexities of natural language, but is also a rich source of knowledge about events/actions. We proposed a pipeline for automatically extracting (partially) ordered structured event representations from narrative texts, with the ultimate aim of learning general event models. Applying the pipeline to three large-scale narrative corpora demonstrates several open research challenges. We propose methods to deal with some of these, such as argument and conditional event recognition, and so on. Learning general action models from the event representations will be the next step of our work.

Bibliography

[1] Aldawsari, M., Finlayson, M.A.: Detecting subevents using discourse and narrative features. In: Proceedings of the 57th Annual Meeting of ACL (2019)

[2] Arora, A., Fiorino, H., Pellier, D., Métivier, M., Pesty, S.: A review of learning planning action models. The Knowledge Engineering Review **33** (2018)

[3] Bamman, D., O'Connor, B., Smith, N.A.: Learning latent personas of film characters. In: Proceedings of ACL. pp. 352–361 (2013)

[4] Bekoulis, G., Deleu, J., Demeester, T., Develder, C.: Sub-event detection from twitter streams as a sequence labeling problem. arXiv preprint:1903.05396 (2019)

[5] Bethard, S.: Cleartk-timeml: A minimalist approach to tempeval 2013. In: Proceedings of SemEval 2013. pp. 10–14 (2013)

[6] Biten, A.F., Gomez, L., Rusinol, M., Karatzas, D.: Good news, everyone! context driven entity-aware captioning for news images. In: CVPR (2019)

[7] Bonial, C., Hwang, J., Bonn, J., Conger, K., Babko-Malaya, O., Palmer, M.: English propbank annotation guidelines. Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado (2012)

[8] Callanan, E., De Venezia, R., Armstrong, V., Paredes, A., Chakraborti, T., Muise, C.: MACQ: A holistic view of model acquisition techniques. arXiv preprint:2206.06530 (2022)

[9] Chambers, N., Jurafsky, D.: Unsupervised learning of narrative event chains. In: Proceeding of ACL (2008)

[10] Chambers, N., Jurafsky, D.: Unsupervised learning of narrative schemas and their participants. In: Proc. 47th ACL Meeting and 4th IJCNLP. pp. 602–610 (2009)

[11] Cook, W.W.: Plotto: The Master Book of All Plots. Tin House Books (1928)

[12] Cresswell, S., Gregory, P.: Generalised domain model acquisition from action traces. In: Twenty-First ICAPS (2011)

[13] Feng, W., Zhuo, H.H., Kambhampati, S.: Extracting action sequences from texts based on deep reinforcement learning. In: Proc. of IJCAI. pp. 4064–4070 (2018)

[14] Fischbach, J., Frattini, J., Spaans, A., Kummeth, M., Vogelsang, A., Mendez, D., Unterkalmsteiner, M.: Automatic detection of causality in requirement artifacts: the cira approach. In: REFSQ. pp. 19–36. Springer (2021)

[15] Geffner, H., Bonet, B.: A Concise Introduction to Models and Methods for Automated Planning. Morgan & Claypool (2013), iSBN: 9781608459698

[16] Glavaš, G., Šnajder, J., Kordjamshidi, P., Moens, M.F.: Hieve: A corpus for extracting event hierarchies from news stories. In: Proceedings of 9th language resources and evaluation conference. pp. 3678–3683. ELRA (2014)

[17] Hayton, T., Porteous, J., Ferreira, J., Lindsay, A., Read, J.: StoryFramer: From input stories to output planning models. In: ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (2017)

[18] Hayton, T., Porteous, J., Ferreira, J.F., Lindsay, A.: Narrative planning model acquisition from text summaries and descriptions. In: Proceedings of AAAI (2020)

[19] Komai, M., Shindo, H., Matsumoto, Y.: An efficient annotation for phrasal verbs using dependency information. In: Proceedings of PACLIC. pp. 125–131 (2015)

[20] Lamperti, G., Zanella, M.: Diagnosis of active systems (2003)
[21] Laokulrat, N., Miwa, M., Tsuruoka, Y., Chikayama, T.: Uttime: Temporal relation classification using deep syntactic features. In: SemEval. pp. 88–92 (2013)
[22] Lee, K., He, L., Zettlemoyer, L.: Higher-order coreference resolution with coarse-to-fine inference. arXiv preprint:1804.05392 (2018)
[23] Lindsay, A., Read, J., Ferreira, J., Hayton, T., Porteous, J., Gregory, P.: Framer: Planning models from natural language action descriptions. In: ICAPS (2017)
[24] Manikonda, L., Sohrabi, S., Talamadupula, K., Srivastava, B., Kambhampati, S.: Extracting incomplete planning action models from unstructured social media data to support decision making. In: KEPS (2017)
[25] Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: ACL (2014)
[26] McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: Pddl–the planning domain definition language–version 1.2 (1998)
[27] Miglani, S., Yorke-Smith, N.: Nltopddl: One-shot learning of pddl models from natural language process manuals. In: KEPS (2020)
[28] Mirza, P., Sprugnoli, R., Tonelli, S., Speranza, M.: Annotating causality in the tempeval-3 corpus. In: CAtoCL. pp. 10–19 (2014)
[29] Mirza, P., Tonelli, S.: CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In: Proceedings of COLING. pp. 64–75 (2016)
[30] Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., Allen, J.: A corpus and cloze evaluation for deeper understanding of commonsense stories. In: Proceedings of NAACL. pp. 839–849 (2016)
[31] Ning, Q., Subramanian, S., Roth, D.: An improved neural baseline for temporal relation extraction. In: Proceedings of EMNLP. pp. 6203–6209 (2019)
[32] Ning, Q., Wu, H., Roth, D.: A multi-axis annotation scheme for event temporal relations. In: ACL (7 2018), http://cogcomp.org/papers/NingWuRo18.pdf
[33] Olmo, A., Sreedharan, S., Kambhampati, S.: Gpt3-to-plan: Extracting plans from text using gpt-3. arXiv preprint:2106.07131 (2021)
[34] Puente, C., Sobrino, A., Olivas, J.A., Merlo, R.: Extraction, analysis and representation of imperfect conditional and causal sentences by means of a semi-automatic process. In: International conference on fuzzy systems. pp. 1–8. IEEE (2010)
[35] Saurí, R., Littman, J., Knippen, B., Gaizauskas, R., Setzer, A., Pustejovsky, J.: TimeML Annotation Guidelines Version 1.2.1 (2006)
[36] Sil, A., Yates, A.: Extracting STRIPS representations of actions and events. In: Recent Advances in Natural Language Processing (2011)
[37] Tan, F.A., Hürriyetoğlu, A., Caselli, T., Oostdijk, N., Nomoto, T., Hettiarachchi, H., Ameer, I., Uca, O., Liza, F.F., Hu, T.: The causal news corpus: Annotating causal relations in event sentences from news. arXiv preprint:2204.11714 (2022)
[38] Tandon, N., de Melo, G., De, A., Weikum, G.: Knowlywood: Mining activity knowledge from hollywood narratives. In: Proc. CIKM (2015)
[39] Van Harmelen, F., Lifschitz, V., Porter, B.: Handbook of knowledge representation. Elsevier (2008)
[40] Webber, B., Prasad, R., Lee, A., Joshi, A.: The penn discourse treebank 3.0 annotation manual. Philadelphia, University of Pennsylvania **35**, 108 (2019)
[41] Yang, Q., Wu, K., Jiang, Y.: Learning action models from plan examples using weighted max-sat. Artificial Intelligence **171**(2-3), 107–143 (2007)