Towards Comprehensive Automation: Extracting Action Models from Diverse Scenarios

Ruiqi Li

A thesis submitted for the degree of
Doctor of Philosophy of
The Australian National University

July, 2024

Except where otherwise indicated, this thesis is my own original work.

Ruiqi Li
July, 2024

To my beloved mother

Acknowledgements

This thesis is a result of many people's contributions. I would like to express my deep gratitude to the following people. Without their support, I would not have been able to complete the voyage.

- AsPr. Patrik Haslum. My deepest thanks go to Patrik for his invaluable guidance and insights throughout my research journey. His consistent encouragement and constructive feedback were fundamental to this thesis. His expertise and enduring patience have significantly shaped my academic growth.
- Prof. Jochen Renz. My year of collaboration with Professor Renz was enlightening and educational. The academic skills that he taught me played a key role in achieving my first publication. I will always value his sage advice.
- Prof. Lexing Xie and Dr. Charles Martin. Lexing has been a source of inspiration, both professionally and personally. Her words of encouragement have been a beacon for me during challenging times. Her passion and determination will always inspire me to better myself. Also, I want to express my gratitude to Charles for being such a kind and helpful co-supervisor. Every conversation I had with Charles was insightful.
- Dr. Leyang Cui, Dr. Hua Hua, and Mr. Songtuan Lin. I want to express my sincere gratitude to them for their significant contributions to most of my publications. Their expertise in writing and experimental design has been instrumental to my research work. Their encouraging words carried me through difficult moments, especially when confronted with paper rejections.
- My friends. Thank you, Jingmeng Li, Rong Wang, Yufei Cao, Lei Sun, Ke Tao, Ziyu Chen, Yuxin Cao, Vimukthini Pinto, Chathura Gamage, Ekaterina Nikonova, and Cheng Xue. The meals, trips, and conversations with such cherished friends have consistently lifted my spirits. Without their emotional support, I could not survive this long journey.
- My family. Special thanks to my parents; they are always supportive of every decision I have made. My deepest regret is not being able to be by my grandmother's

side when she passed away, due to the constraints of the COVID pandemic. Her memory remains alive in my heart. Lastly, special thanks to M, whose presence has fulfilled my soul.

Abstract

AI planning depends on the systematic formation of action sequences or plans that transition from a given initial state to a desired objective state. To accomplish this, well-defined action models, which are typically encapsulated in planning languages such as Planning Domain Definition Language (PDDL), are required. However, creating these models by hand requires considerable effort and domain-specific expertise.

This thesis focuses primarily on the automated generation of action models from two crucial domains: the physical and narrative domains. Fundamental to our methodology for the physical domains is the incorporation of qualitative spatial relations to represent states. Regarding the narrative domains, we present NaRuto, an automated system that generates planning action models efficiently from narrative texts. In addition, we present a dataset that focuses on event-event dependency relationships.

The challenge of planning languages is that they must accurately and concisely convey intricate planning problems while guaranteeing the consistency of developed plans. PDDL, a major planning language, is founded on the STRIPS approach and provides a structured method for describing planning problems via domain predicates and actions. An action in PDDL consists of three fundamental elements: parameters, preconditions, and effects, which map out the transition between states and are each represented by a set of predicates describing their properties.

The difficulties intensify when addressing actions in physical domains. In comparison to classical planning domains, such as the 8-puzzle game, physical domains encompass more intricate scenarios. For example, an action like shooting a ball into a goal can have a variety of results depending on the variables involved. To address these complexities, we characterize states using qualitative spatial relations, a well-researched topic in AI. We present a novel model-building technique that combines neighborhood and hierarchical clustering to merge observed state transitions resulting from identical activities. Notably, our method also contributes to detecting and describing novelties in the physical domains, illuminating both their preconditions and effects, thereby facilitating the dissemination of knowledge among agents.

Narrative texts also present a significant challenge. In contrast to structured data, narra-

tive texts, which are rich in events, are by nature unstructured. In existing research, the extraction of action models from such texts has been mostly manual or semi-automated. We present NaRuto, a system designed to bridge this gap. In a two-step procedure, NaRuto derives structured events from narrative texts and then produces action models. By comparing it with action models from the literature and assessing its action identification, action precondition/effects construction, and ability to generate alternative plans, we demonstrate the effectiveness of NaRuto.

We have learned through the construction of NaRuto that events are critical to narrative texts, specifically in terms of accurately identifying action models. Despite advancements in the study of relational extraction between entities, many of the relationships between events remain unexplored. We address this deficiency by introducing our human-annotated Event Dependency Relation (EDeR) dataset, which aims to extract event dependency information. Using the EDeR dataset, we refine the event representations of existing semantic role labeling models and apply them to improve the co-reference resolution task.

The automated generation and representation of planning action models, whether derived from physical domains or narrative texts, are essential to AI planning. Through our methodologies, datasets, and systems, we intend to expedite these processes, resulting in more effective and precise AI planning applications across a variety of domains.

List of Publications

The publications resulting from my doctoral research are provided below. They partially form the foundation of this thesis.

- **Ruiqi Li**, Hua Hua, Patrik Haslum, and Jochen Renz. "Unsupervised Novelty Characterization in Physical Environments Using Qualitative Spatial Relations". In Proceedings of the 18th International Conference on the Principles of Knowledge Representation and Reasoning (KR2021). (Chapter 3).
- **Ruiqi Li**, Patrik Haslum, and Leyang Cui. "Towards Learning Action Models From Narrative Text Through Extraction and Ordering of Structured Events". In Proceedings of the 36th Australasian Joint Conference on Artificial Intelligence (AJCAI2023). (Chapter 4).
- **Ruiqi Li**, Patrik Haslum, and Leyang Cui. "EDeR: Towards Understanding Dependency Relations Between Events". In Proceedings of the 22nd Conference on Empirical Methods in Natural Language Processing (EMNLP2023). (Chapter 5).
- **Ruiqi Li**, Leyang Cui, Songtuan Lin, and Patrik Haslum. "NaRuto: Automatically Acquiring Planning Models from Narrative Texts". In Proceedings of the 38th Annual AAAI Conference on Artificial Intelligence (AAAI2024). (Chapter 4).
- Hua Hua, Dongxu Li, **Ruiqi Li**, Peng Zhang, Jochen Renz, and Anthony Cohn. "Towards Explainable Action Recognition by Salient Qualitative Spatial Object Relation Chains". In Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI2022). Third author. I contributed to paper structure discussions and experiment designs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the domain of artificial intelligence (AI), the planning process is intrinsically focused on the methodical production of action sequences, which are widely known as plans. These sequences serve as pathways, connecting an established initial state and the desired goal states within a problem space. In order to formulate action sequences efficiently, AI planners require action models that are precisely defined. These models are typically encapsulated in a declarative planning language. A prime example of such a language is the Planning Domain Definition Language (PDDL), which has been thoroughly described by McDermott *et al.* [1998a]. One of the prevalent challenges in this domain lies in the manual construction of action models. Crafting these models by hand not only demands an exhaustive amount of effort but also requires a profound level of domain-specific expertise, making it a complex labor.

This thesis focuses on the challenges of automatically generating planning-language-based action models from two important domains, both tightly related to the real world: (1) physical domains and (2) narrative texts. During the research on action model generation in the narrative texts, we also addressed (3) the challenge of identifying the event-event dependency relationships, which are important for accurate event representation as well as action model generation.

For the action model generation for the physical domains, we introduce qualitative spatial relations into the application of planning domain model generation to represent the before- and after-state of the actions. For the narrative domains, we develop NaRuto, an innovative, fully automated two-stage system for generating planning action models from narrative text. Regarding the identification of event-event dependency relations, we create a human-annotated dataset, which brings out a feasible knowledge base for an effective automatic relation classification process. In the rest of this chapter, we briefly describe the challenges, motivations, and contributions to the three aspects.

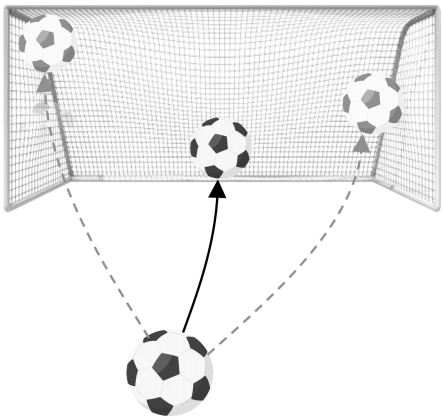
1.1 Action Model Generation in Physical Environments

Planning languages have a significant impact on the illustration of planning problems, as they facilitate the operation of AI planning tools, usually known as planners, in formulating consistent strategies that are realized as plans. In the realm of planning languages, PDDL stands out as a prominent and extensively adopted framework. The fundamental principles of this technique draw inspiration from the known STRIPS [Fikes and Nilsson, 1971; Sabbadin *et al.*, 2020] approach, which is a creative framework for representing planning problems.

When studying the intricate framework of PDDL, the domain definition incorporates crucial elements, particularly domain predicates and actions. As illustrated in Figure 1.1 (right), a PDDL action is defined by three fundamental components: parameters, preconditions, and effects. Parameters highlight the relevant entities associated with the action, while the precondition and effect components collectively describe the change of states, clarifying the necessary conditions before the execution of an action and the resulting changes in state afterwards [Haslum *et al.*, 2019]. In the context of this theoretical framework, each action may be seen as a process of transitioning between different states. These states are characterized by a collection of predicates that specify their properties and characteristics. For example, the predicate (*disconnected* ?ball ?goal) represents a scenario in which the ball is not connected to the goal.

1.1.1 Challenges in Describing Actions in Physical Domains

Modeling classical planning domains with PDDL has seen widespread use, and one example of this is the sliding puzzle game 8-Puzzle. The game 8-Puzzle begins with a frame containing eight square tiles arranged in a jumbled fashion, with one tile absent; the objective is to rearrange the tiles such that they are in the correct sequence. For example, in 8-Puzzle, players are only allowed to move one square tile left, right, up, or down to the empty space, and there are two main state representation strategies: states can be described with predefined predicates like *move-left* [Yang *et al.*, 2007; Mourao *et al.*, 2012; Andersen and Konidaris, 2017] or in the form of vectors [Rodrigues *et al.*, 2010; Asai and Muise, 2020]. The physical domains, on the other hand, are more intricate and grounded in reality. This is due to the vastly different outcomes that the same action can have depending on how it is realized. For instance, the action of kicking a ball into a goal (shown in Figure 1.1, left) can be performed from a wide variety of different positions, resulting in the ball traveling along a variety of different trajectories and landing in a variety of different parts of the goal. Even though the time at which a state transition occurs is ambiguous because state properties such as the relative positions of objects can change by arbitrarily small quantities, only changes that are somehow significant should be considered a transition to a different state. In the computer vision community, tasks such as video scene graph generation (VSGG) pose a direction to creates a visual representation of the video scenes and their visible changes using natural language [Wang *et al.*, 2023; Pu *et al.*, 2023]. VSGG focuses on understanding and representing the state



```
(:action shoot-into-goal
 :parameters (?a - ball ?b - goal)
 :precondition (and (disconnected ?a ?b)
                    (far ?a ?b)
                    (approaching ?a ?b)
                    (stationary ?b))
 :effect (and (not (disconnected ?a ?b))
              (nt_proper_part ?a ?b)
              (not (far ?a ?b))
              (touching ?a ?b)
              (not (approaching ?a ?b))
              (stationary ?a))
```

Figure 1.1: Left: Some possible realizations of the action of shooting a ball into a goal. Right: the corresponding qualitative action model. Predicates are in the form of relation object triples like (disconnected ?a ?b).

of the world from video data, including object detection and relationships but is not able to formulate strategies or sequences of actions to achieve specific goals.

1.1.2 Contributions to Action Description Generation and Novelty Characterization

To address the challenges described above, we create action models describing states in the form of qualitative spatial relations between objects and propose a graph-based hierarchical clustering architecture to extract actions from observations. Most actions in physical domains can be observed by their resultant changes in spatial relations between objects. For example, as shown in Figure 1.1, the action of shooting a ball to a goal leads to several changes in the spatial relations between the ball and the goal. Qualitative spatial relations are spatial relations in a qualitative form. There are a large number of formally defined qualitative spatial relations [Randell *et al.*, 1992; Van de Weghe *et al.*, 2005; Dylla *et al.*, 2017], which have covered most of the spatial aspects in our real world (e.g., location, distance, and speed). Qualitative spatial relations have already been used to model physical states and changes in [Delafontaine *et al.*, 2011; Duckworth *et al.*, 2019]. We present an innovative approach to the construction of the action model. This method integrates neighborhood and hierarchical clustering in a way that has a synergistic effect. The goal of this method is to group observed state transitions that are likely the result of the same activities. According to the results of our research, this combination approach provides superior performance to other methods since it achieves the ideal balance between the model's coverage, validity, and dimensionality.

As an application, we show that our action model generation system can contribute to novelty detection and characterization, particularly in the construction of updates to action models to incorporate observed novelties. In practice, "novelty" refers to events

Input: Bryan hits Jack in the face.

Output:
```
(:action hit
         :parameters   (?x - subject ?o - object)
         :precondition (and (close-to ?x ?o)
                            (angry-at ?x ?o)
                            (in-a-fight ?x ?o))
         :effect (and (yell-at ?o ?x)
                      (injured ?o)
                      (not (close-to ?x ?o))))
```

Figure 1.2: Input: A narrative sentence. Output: The corresponding generated action model.

that challenge either implicit or explicit preconceptions about agents, environmental settings, or their interactions [Langley, 2020]. Our designed characterizations define these novelties through actions, explaining both their preconditions and consequential effects. This approach aligns closely with models commonly utilized in AI planning [Li *et al.*, 2018]. An explicit description of the novelty not only allows the agent to get a deeper understanding of the novelty, but also allows it to distribute the learned insights to other agents, including humans. It would be impossible without such a clear model of the novelty. The ability of our action model generation technique to recognize and define observed novel behaviors within a physical environment at a qualitative level of abstraction distinguishes it from others. Furthermore, this process is carried out automatically and without supervision. Our findings show that varying novelties have significantly diverse properties, with minimal overlap in their novel actions.

1.2 Action Model Generation from Narrative Texts

Manually constructing action models is not only a labor-intensive process but also demands a depth of domain expertise. Besides the physical domains, the task is also complicated when tackling natural-language narrative texts. These texts, often characterized by their massiveness and the myriad of activities they contain, bring extraordinary challenges. Unlike structured data, narrative texts are inherently unstructured, and this sprawling and diverse nature compounds the difficulty in planning-language-based action model creation.

1.2.1 Challenges and Motivations

Recently, researchers have made moves to extract action models from narrative texts, drawing from resources like short stories and movie summaries [Hayton *et al.*, 2017; Huo *et al.*, 2020; Hayton *et al.*, 2020]. A common methodology in their work is the dependency on human intervention, either to complement or improve the outcomes of automated extraction. Consequently, these approaches present challenges when anticipated for large-scale action model generation, primarily due to their semi-automated nature.

Furthermore, there have been some advanced efforts to design fully automatic methods to extract action models from instructional texts, such as culinary recipes or user manuals [Mei *et al.*, 2016; Feng *et al.*, 2018; Olmo *et al.*, 2021]. Nevertheless, it's pertinent to note that these textual materials typically exhibit straightforward syntax and avoid complex linguistic constructs. They seldom mirror the colloquialisms or the complicated phrases and clauses often found in narrative texts, like movie plots.

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 Hayton *et al.* [2020], Figure 5. This example sentence shows the situation in which verbs can take or require a clausal complement, i.e., an argument of the event's verb is itself an event. If "King Louie offers to help Mowgli ...", the event verb is "offer", and the event "[King Louie] help Mowgli" is its argument. Additionally, event arguments can be nested: if "King Louie offers to help Mowgli stay ...", "stay" is an argument of "help", and the event "[King Louie] help ([Mowgli] stay)" is itself the argument of "offer". Besides, " he will tell Louie how to make ..." is a condition of the "offer". It means, that although the sentence contains many event verbs ("help", "stay", "tell", "make"), only one of them, "offer", is indeed said to happen.

Complexities in narrative texts, such as the presence of argument events and conditional events, make narrative texts more difficult to comprehend, creating a gap in AI agents' ability to automatically recognize action models from such texts on a broad scale.

1.2.2 Contributions to Action Model Generation for Story Planning

Recognizing this gap and its associated challenges, we propose an automated two-stage system named NaRuto to generate planning action models from narrative text, as shown by the example in Figure 1.2. In the first stage, it extracts structured representations of event occurrences from the source text, and in the second, it generates action models based on predictions of commonsense event/concept relations. The novelty of NaRuto is that it deals with many of the complexities of narrative text, such as argument and conditional events, uses commonsense knowledge predictions, and is fully automated.

We compare the NaRuto-generated actions with examples from the literature of the out-

puts of other methods applied to the same narrative texts. The advantages of NaRuto-generated action models are demonstrated in three dimensions:

- Action name identification. We assess NaRuto's *coverage* of the actions in the source text, by comparing the sets of extracted action names with those in the ground truth domain, using manual alignment of names. We prove that NaRuto can identify the actions provided in the ground truth planning domain.
- Action preconditions/effects creation. We ask experts to assess the appropriateness of each action's preconditions and effects generated by NaRuto in relation to the predicates defined in its domain model. The assessment results of the domain model's predicates used across its actions reflect the internal consistency, or soundness, of the domain models developed by NaRuto.
- Alternative plan generation. We evaluate the extent to which the NaRuto-generated domain model supports the generation of other story plans. By using the Fast Downward [Helmert, 2006] planner to generate plans, we show the generalization capabilities of the NaRuto-derived domain model based on the number of problems solved and the number of distinct plans produced.

Furthermore, we apply the NaRuto-created action models to guide large language models' story generation. In detail, we first use NaRuto to create a set of action models based on movie plot summaries, due to [Bamman *et al.*, 2013], and news articles from the Goodnews dataset [Biten *et al.*, 2019]. We then generate stories using action models by collecting causally connected action sequences and feeding them into a large language model (LLM). When the results of this procedure are compared to the results of merely giving the LLM an initial prompt (the first event in the sequence), it is clear that stories developed utilizing the action sequence as a guide are significantly more likely to be both topical and cohesive.

1.3 Learning Event Relations for Action Model Representation

In narrative texts, events tell us what happened, who or what was involved, and at where. Because actions are portrayed as events, extracting events becomes critical to creating actions. Event extraction is based on semantic role labeling (SRL) [Gildea and Jurafsky, 2002]. The SRL method locates verbs and their related arguments within text segments and labels them based on their semantic roles (such as agent, patient, modal modifier, and so on). When we look at the SRL-based representation of events, we notice a certain form of event that is important to distinguish but, to the best of our knowledge, hasn't been effectively addressed before – the event dependency relation. This is exemplified when the argument of an event's verb is itself an event. For example, consider this variation of the sentence in Figure 1.2: "Bryan tries to hit Jack". The main event verb is "try", and the associated argument is the event "[Bryan] hit Jack".

The task of representing and extracting dependency relations among events allows the identification of both semantic (e.g., clarifying the object of an event predicate) and syntactic (e.g., elucidating the relative clause(s) linked to an event predicate) connections. It is challenging but interesting for AI researchers. Furthermore, such event-dependency relations provide invaluable insights, which can improve a variety of downstream tasks in information retrieval (IR) and natural language processing (NLP) communities.

1.3.1 Challenges in Learning Event Dependency Relations

Prior research in relation extraction has primarily focused on investigating the interactions between entities [Miwa and Sasaki, 2014; Huguet Cabot and Navigli, 2021; Chen *et al.*, 2020; Ma *et al.*, 2022], leaving the interactions between events relatively unexplored. The variety of relations studied in the comparatively limited research addressing the relations between events is predominantly causal [Mariko *et al.*, 2020, 2022; Tan *et al.*, 2022a], temporal [Bethard, 2013; Laokulrat *et al.*, 2013], and hierarchical [Hovy *et al.*, 2013; Glavaš and Šnajder, 2014]. These relations treat two events as if they are independent, both syntactically and semantically, ignoring potential interdependencies between them. Such omissions may result in narrative interpretations that are either fragmented or erroneous.

Consider the sentence "He tried to forgive his father one time". In this case, the event with the predicate "forgive", represents an action that he "tried" to take. Notably, no temporal, causal, or hierarchical relationship exists between "tried" and "forgive". Instead, the event "forgive" is the object (syntactically depending) and the patient (semantically depending) of the event "tried". In fact, "forgive" did not actually happen. Recognizing such relationships, which include both semantic and syntactic dependencies, is a critical, albeit difficult, task for AI researchers. Accurate recognition of these dependency interactions will enhance language understanding and give vital insights, making them useful for a variety of related NLP tasks.

1.3.2 Contributions to Event Representation and Other NLP Tasks

Argument-event dependency relations frequently occur in natural language texts, a prominence especially noted within narrative contexts like news articles, conversations, and others. In light of this prevalence, we introduce a human-annotated **E**vent **De**pendency **R**elation dataset (named EDeR). This dataset (1) extracts event dependency information based on a sample of 275 documents spanning seven genres from OntoNotes [Pradhan *et al.*, 2013] and integrates with orthogonal annotations of this dataset, and (2) provides refined semantic role-labeled event representations based on this information. The construction of the EDeR dataset is underpinned by 11,852 high-quality annotations, aligned with the proposed event relation taxonomy.

Both human-annotated EDeR and superior baseline model-predicted event dependency relations were deployed to improve a state-of-the-art semantic role labeling (SRL) model [Zhang *et al.*, 2022]'s abilities in generating event representations. Empirical analy-

ses confirm that integrating these relations enhances event extraction, thereby deriving refined representations from our EDeR dataset. These enhanced event representations have been applied to the co-reference resolution (CR) task. Subsequently, a comparative performance evaluation, utilizing both original and our refined event representations as inputs on a state-of-the-art CR model [Jiang and Cohn, 2021], demonstrates the efficacy and reliability of the refined representations provided by EDeR.

1.4 Thesis Outline

The body of this thesis is structured on the basis of the three key contributions outlined, following the same sequence: (1) the creation of action models in physical contexts, (2) the generation of action models in narrative texts, and (3) the modeling of event dependency relations for learning actions. The structure is described thoroughly as follows:

- Chapter 2 gives an overview of the background and related work. In the context of the physical domains, we provide an introduction to the relevant literature on action model extraction and the utilization of qualitative spatial representations (QSRs) for action representation. In the context of narrative domains, we elaborate on the process of modeling planning-language-based actions derived from textual events retrieved from narratives. Furthermore, it is crucial to underline the importance of identifying event-event dependency relationships in order to effectively depict events and subsequently portray actions.
- Chapter 3 describes the process of using QSRs to express state transitions and subsequently generalizing these transitions to create action models from physical environments. As a means of testing and illustrating the application of our method, we have chosen to utilize the well-known physical video game, Angry Birds, as a foundational case study. Through this analysis, we aim to build action models and demonstrate the unique potential of our approach for detecting and characterizing novelties that occur in physical domains.
- Chapter 4 introduces the NaRuto system as a means of extracting action models from natural language narrative texts in an automated manner. The system is equipped with a range of NLP techniques, including semantic role labeling and commonsense knowledge reasoning. The system is assessed based on many factors, such as the comparative analysis of the generated action models against other advanced models from the existing work, the capacity to generate alternative plans, and the application of guiding large language model's story generation.
- Chapter 5 investigates the dependency relationships between events. In this chapter, we present EDeR, a dataset consisting of human-annotated event dependency relations. We present baseline models that have been developed for the purpose of classifying these dependency relations. Additionally, we demonstrate the practicality of the relationships by showing their potential advantages in improving

popular NLP tasks, such as semantic role labeling and co-reference resolution.

- Chapter 6 serves as a conclusion, highlighting the contributions made by this thesis in addressing the challenges presented and also providing a description of potential future works.

Chapter 2

Background

This thesis explores the derivation of planning-language-style action models from diverse sources, including physical environments and narrative texts. This chapter highlights research relevant to the generation of such action models, drawing from physical game-playing video clips as well as narrative stories. During our review of the literature on action model extraction from narrative events, we also delved into the inter-relationships among these events. The relationships between events can influence their description, subsequently impacting the representation of the action model.

The first section reviews the literature concerning the extraction of action models from state transitions in physical environments, emphasizing the use of qualitative spatial relation representations for accurate action characterizations. The following section focuses on extracting action models from events extracted from narrative texts. Additionally, we investigate the relationships between events, underscoring the necessity of determining the event dependency relations – it could affect the precision of event extraction and representation.

2.1 Action Model Generation From Physical Environments

2.1.1 Action Models in Physical Domains

Action model generation in the realm of artificial intelligence (AI), especially for physical environments, has been a subject of extensive research for quite some time. One such study by Pasula *et al.* [2007] experimented with the Block Stacking game, a simulated physical environment. They focused on learning models of actions that have an element of randomness, while also noting that the actions taken (not just the results of these actions) can be observed.

Shifting the lens to video games, two significant contributions come from Konidaris *et*

al. [2015] and Andersen and Konidaris [2017]. Both sets of researchers directed their efforts toward learning about stochastic actions that progress over a duration, which they referred to as "options".

Fast-forward to more recent times, there has been a notable shift towards unsupervised or semi-supervised techniques for generating action models. For instance, Miglani [2020] utilized deep reinforcement learning to extract action models from textual descriptions in natural language. On the other hand, Asai and Fukunaga [2017] and Asai and Muise [2020], offered a novel approach that employed autoencoders. This was done to determine a representation of various states and to categorize transitions between states into specific actions.

In addition to these, a cluster of researchers has dedicated their efforts to understanding how action models can be learned through the observation of symbolic actions and states. This has been well documented in works by Amir and Chang [2008]; Cresswell *et al.* [2013]; Aineto *et al.* [2018].

2.1.2 Action Model Representation by Qualitative Spatial Relations

QSRs represent relations between objects in space using qualitative terms. They represent intuitive, human-understandable concepts, such as *disconnected*, which means two objects appear separate from each other. There is a large number of formally defined QSRs [Randell *et al.*, 1992; Van de Weghe *et al.*, 2005; Renz and Nebel, 2007; Cohn and Renz, 2008; Dylla *et al.*, 2017], which cover most of the spatial aspects relevant to our world, such as relative object location, distance and movement. QSRs have also previously been used to model physical states and state transitions [Delafontaine *et al.*, 2011; Duckworth *et al.*, 2019].

QSRs have been used to recognize action from video in an automatic way [Sridhar *et al.*, 2011a,b; Dubba *et al.*, 2015]. Tayyub *et al.* [2014] use both qualitative and quantitative spatio-temporal representations as joint features to recognize daily activities. Young and Hawes [2015] combined four kinds of qualitative spatial calculi to demonstrate the movements of agents in a soccer game simulator, like *kick* or *shoot*. Duckworth *et al.* [2016] and Duckworth *et al.* [2019] extracted qualitative spatial relations like qualitative distance relation from robotic movement observations and combined them as features for unsupervised clustering algorithms such as K-means [Yuan and Yang, 2019], and Latent Semantic Analysis (LSA) [Deerwester *et al.*, 1990]. Although QSR features are utilized to depict states in these works, their sole purpose is to identify pre-existing actions. In the case of novelties, whose actions may not have been observed previously, actions cannot be instantiated and explained in the absence of a description that specifies their preconditions and effects.

2.2 Action Model Generation From Narrative Texts

Besides physical domains, in recent years, the generation of action models from textual data has received significant attention in the field of AI planning. This section provides a comprehensive review of the existing literature regarding the creation of action models based on events extracted from textual contexts. It is worth mentioning that within the scope of this thesis, the term *action* is utilized to denote the action itself in the planning action models, while *event* is employed to denote the action's occurrence in natural language texts.

2.2.1 Extracting Actions in Natural Language Texts

Sil and Yates [2011] identified web texts that contained words that mirrored target verbs, as determined by textual correlations. They then used supervised learning approaches to determine both the pre- and post-conditions of actions. Branavan *et al.* [2012] pioneered the development of a reinforcement learning model that used surface verbal clues to understand potential action preconditions. Similarly, Manikonda *et al.* [2017] extracted plan traces from a broad range of social media texts, resulting in the creation of partial action models.

The landscape of action extraction methodologies is vast and diverse. Some strategies hinge on the utility of dependency parse structures, while others embark on the realms of neural language models and the nuanced intricacies of reinforcement learning. Of particular note, several researchers, including Branavan *et al.* [2009], Yordanova [2016], Lindsay *et al.* [2017], Feng *et al.* [2018], Miglani and Yorke-Smith [2020], and Olmo *et al.* [2021], have centralized their efforts on instructional texts. These range from the simplicity of recipes to more sophisticated game-playing instructions and user guides. By doing so, they avoid the intricate complexities intrinsic to narrative texts. On the other hand, studies by Hayton *et al.* [2017] and Huo *et al.* [2020] ventured into the narrative texts. However, they did not address the challenges, as their action model generation processes weren't fully automated, and often needed human interventions or even wholly depended on manual additions and refinements. Hayton *et al.* [2020] introduced a novel system that aimed to automate such processes; however, their produced action models primarily mirrored the input narrative sequence without broader generalization.

2.2.2 Learning Action Models from Events

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 – none identify events that are arguments or conditions of other events.

A conceivable alternative to learning actions from events is to construct them from one or more existing semantic knowledge sources. VerbNet [Schuler, 2005] is a curated knowledge base, covering at the time of writing 4570 English verbs in 602 classes, which

correspond roughly to distinct events. Event classes are annotated with semantic information, including roles (parameters for event arguments) with type restrictions, and axioms expressing preconditions and effects. Although impressive in scope, this knowledge base still has gaps: we find over 2100 verbs with 20 or more mentions across two large corpora (the movie plot summaries dataset [Bamman *et al.*, 2013] and the Goodnews dataset [Biten *et al.*, 2019]) we analyzed that are not in VerbNet.

Similarly, ConceptNet [Speer *et al.*, 2019] is a knowledge graph of concepts, which may be verb, noun or adjective phrases, and their relations, which brings together information from many sources, including crowdsourced (e.g., OMCS [Singh, 2002] and DBPedia [Lehmann *et al.*, 2015]) and curated (e.g., OpenCyc, [Lenat and Guha, 1989]). While it includes many relationships that are relevant to modeling actions/events – for example, *Causes*, *HasSubevent*, *HasPrerequisite*, and *MotivatedByGoal* – and over 128,000 concepts classified as verbs, their relation-involved instances amount to only about 1% of ConceptNet. For example, the number of concepts that have both *HasPrerequisite* and *Causes* – i.e., both preconditions and effects – are only 577, and range from very general, e.g., "talking", to highly specific, e.g., "adding up a column of numbers".

Our work identifies previously ignored argument and conditional events and provides an unlimited source of events for action extraction. We utilize commonsense knowledge graphs to infer the pre- and post-conditions of action models because they reflect human experience and reasoning, in an automatic manner. These elements ensure that our model can create reliable action models from narrative texts.

2.3 Learning Event Relations for Action Model Representation

2.3.1 Event and Event Representation

An event mention consists of a verb or phrasal verb as the predicate and a set of labeled arguments [Levin, 1999; Rappaport Hovav *et al.*, 2010]. Events can be extracted from texts by performing SRL to label the roles (i.e., predicates and arguments) of words or word spans in a given sentence. The OntoNotes dataset follows the PropBank annotation schema [Bonial *et al.*, 2012] for semantic role labels, in which arguments are text spans, and which divides argument labels into numbered arguments (ARG0–ARG5), for arguments required for the valency of an action (e.g., agent and patient), and modifiers of the verb, such as purpose (PRP), locative (LOC), and so on.

There are also some public datasets for some shared SRL tasks (e.g., CoNLL 2005 [Carreras and Màrquez, 2005], CoNLL 2009 [Hajič *et al.*, 2009], and CoNLL 2012 [Pradhan *et al.*, 2012] tasks.). CoNLL-2012 has been merged into the OntoNotes V5.0 dataset, so we call this dataset OntnoNotes in the rest of the thesis.

An event can itself be an argument for another event. In the annotated SRL datasets, we find such argument events contained within the span of the event they are an argument

of, but the converse does not hold: events that are contained in the span of another event are not always argument events, but may share part of their arguments (subject or object) with the containing event. For example, in the sentence "The man who works here tells me to get the hell out", the event "The man [who] works here" is contained in the event "The man who works here tells me to get the hell out" but is not an argument of "tells". Such cases reveal that the annotation of events in these SRL datasets does not consider such dependency relations between events. Making the relations explicit helps us establish a more accurate argument span, such as "The man" for ARG0 (agent) of the event with the predicate "tells" in the example above.

2.3.2 Relations Between Events

Table 2.1: Comparisons of event representations and relations between our EDeR dataset and other public event relation datasets.

datasets	**representations**			**relations**
	verb	**argument**	**span**	
ECB [Bejan and Harabagiu, 2008]	✓	✗	✗	hierarchical
Hieve [Glavaš *et al.*, 2014]	✓	✗	✗	hierarchical
IC [Araki *et al.*, 2014]	✓	✗	✗	hierarchical
TimeBank [Pustejovsky *et al.*, 2003]	✓	✗	✗	temporal
TB-Dense [Cassidy *et al.*, 2014]	✓	✗	✗	temporal
MATRES [Ning *et al.*, 2018b]	✓	✗	✗	temporal
ECD [Do *et al.*, 2011]	✓	✗	✗	causal
CiRA [Fischbach *et al.*, 2021]	✗	✗	✓	causal
CNC [Tan *et al.*, 2022a]	✗	✗	✓	causal
FCR [Yang *et al.*, 2022]	✗	✗	✓	causal
CATENA [Mirza and Tonelli, 2016]	✓	✗	✗	causal+temporal
ESC [Caselli and Vossen, 2017]	✓	✗	✗	causal+temporal
TCR [Ning *et al.*, 2018a]	✓	✗	✗	causal+temporal
EDeR (Ours)	✓	✓	✓	conditional+dependency

Several event-event relations have been proposed for decades. The summary of varied relations among the existing datasets is shown in Table 2.1's last column. The TimeBank [Pustejovsky *et al.*, 2003], TB-Dense [Cassidy *et al.*, 2014] and MATRES [Ning *et al.*, 2018b] datasets focus on the temporal relationship which reveals if one event happens BEFORE, AFTER, etc. another. Logical relationships like causality are also explored in datasets like CausalTimeBank [Mirza and Tonelli, 2016], CiRA [Fischbach *et al.*, 2021] and CNC [Tan *et al.*, 2022b]. Such relationships regard two events as syntactically and semantically independent – the occurrence and actuality of each event are isolated and not affected by others.

Besides, several datasets propose hierarchical relationships (i.e., super-event and sub-event relations), where a sub-event should satisfy two conditions: Temporally, the sub-

event occurs during the super-event. Spatially, the sub-event should be contained by the super-event [Hovy *et al.*, 2013; Glavaš *et al.*, 2014; Glavaš and Šnajder, 2014]. Araki *et al.* [2014] introduces a dataset detecting event co-reference relation – whether two event mentions refer to the same event.

Such hierarchical relations cannot deal with cases when an event (especially the verb of the event) requires a clausal complement, i.e., an argument of the event verb is itself an event, as the "tell" and "get" examples shown in the example sentence "The man who works here tells me to get the hell out". Therefore, introducing such event-event dependency relationships is demanded.

Chapter 3

Action Model Generation in Physical Environments

Generating action models from physical domains presents a significant challenge for the AI planning community. This complexity arises because identical actions can yield vastly different outcomes based on their specific execution. Facing this challenge, this chapter demonstrates the following contributions of our work: Based on the qualitative spatial relations, we propose a new method to automatically create the action model, combining neighborhood and hierarchical clustering to group observed state transitions likely to be caused by the same action. We show that this combination yields a better trade-off between the model's coverage, validity and size than other methods. Therefore, our approach is relevant to applications that require the automated generation of a world representation. One such application is detecting and characterizing novelties. The novelty and advantage of our approach, are that it can compute a characterization of observed novel behavior in the environment at a qualitative level of abstraction, and do so automatically, in an unsupervised manner. We show that different novelties mostly give rise to highly distinct characterizations, with very little overlap in their novel actions.

We structure this chapter as follows: Section 3.2 presents the related work on generating action models in physical domains and its application in characterizing novelty; Section 3.3 details the design of our proposed approach: taking the game-play video clips as input and outputting the base and novel action representations; Section 3.4 and 3.5 describe the two experiments on base action model extraction and using the base model for novelty characterization.

3.1 Novelty Detection and Characterization

In the real world, "novelty" refers to situations that violate implicit or explicit assumptions about agents, the environment, or their interactions [Langley, 2020]. As AI agents

such as robots become increasingly popular, they are expected to operate reliably in unconstrained environments where novelties are increasingly likely to occur. Therefore, it is essential for AI agents to be capable of detecting, characterizing and adapting to novelties in their environment [Boult *et al.*, 2020]. Arguably, an agent could adapt to novelty through a process of trial-and-error without an explicit characterization of the novelty. However, an explicit model of the novelty can enable more efficient adaptation, as it allows for reasoning about the novelty. The characterizations we create are expressed in terms of actions, with preconditions and effects, that describe the behavior of the novelty and are thus very close to the kind of model that is used in AI planning [Li *et al.*, 2018]. An explicit model of the novelty also enables the agent to share the knowledge it has learned about the novelty with other agents, including humans. This is key to making the agent's response to the novelty explainable to an observer, such as a human monitoring the agent.

Novelty detection and characterization are based on constructing an update to the agent's action models to reflect the detected novelties. Our approach to characterizing novelty in the environment consists of two steps (See Figure 3.2 for an overview). In the first step, we learn, from observation, a model of the environment without novelty. This model takes the form of a set of actions, defined by their preconditions and effects, which describe the possible behaviors of objects in the environment. We call this the *base model*. Actions are generalized from state transitions observed during training. This step is done offline, which means we can use a large volume of training data. However, because training data can never be guaranteed to be complete, neither is, in general, the model. Also, generalization from observed state transitions means the model may permit some transitions that do not occur in the environment. Note that although we term them "actions", state transitions permitted by the base model are not necessarily actions taken by the agent. They can also represent events that occur spontaneously, or as delayed consequences of the agent's actions.

The second step occurs online, when novelty is encountered. We detect novelty by the recurring presence of state transitions that are not possible in the base model; we call such transitions *uncovered*. Because the base model cannot be guaranteed to be complete, some uncovered state transitions may also occur without novelty, i.e., there is some noise. However, because novel behaviors are persistent, uncovered state transitions caused by novelty are far more likely to be repeated as we observe more of the agent's interaction with the world, and we use this fact to distinguish when novelty appears. The repeated uncovered state transitions also become the basis for our novelty characterization. We create new actions, covering the novel behaviors, from these transitions, by a process similar to that which creates the base model.

Learning action models in physical domains is challenging because of the great variety of realizations of an action that can occur. For example, the action of kicking a ball into a goal can be taken from a myriad of different positions, with different ball trajectories, ending at different locations in the goal. Although the occurrence of a state transition is ambiguous because state properties such as the relative positions of objects can change

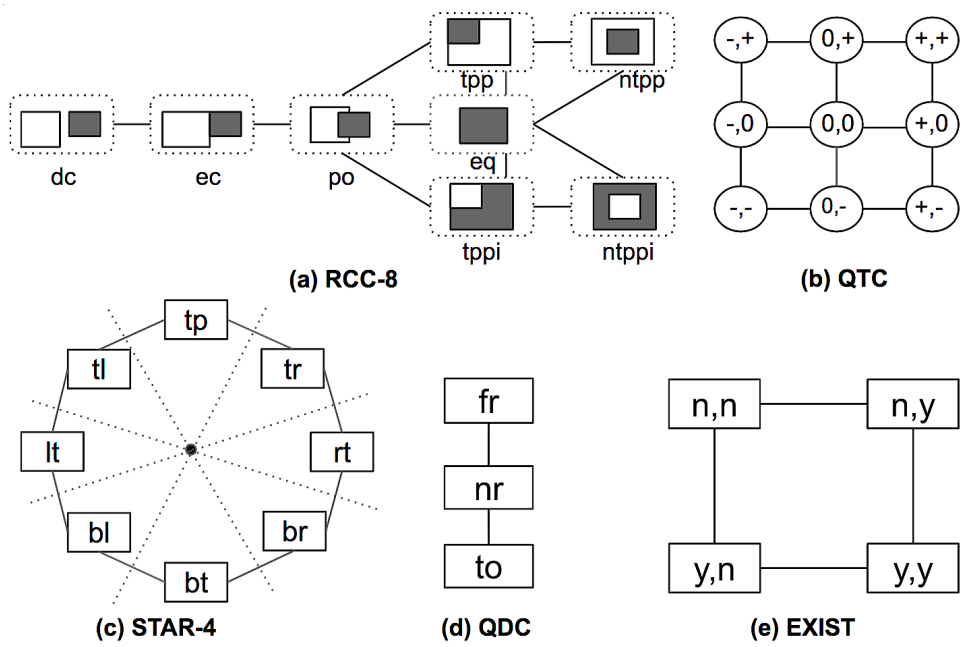


Figure 3.1: (a) RCC-8: topology relations; (b) QTC: movement relations; (c) STAR-4: direction relations; (d) QDC: distance relations; (e) EXISTS: existence relations. Adjacent (connected) relations are conceptual neighbors.

by arbitrarily small quantities, only changes that are somehow significant should be considered a transition to a different state. To address this problem, we abstract the observation of the world state into a set of qualitative spatial relations (QSRs) between observed objects.

The action models we generate are defined by their typed parameters, preconditions and effects, and can be expressed as a planning domain, e.g., in the Planning Domain Definition Language (PDDL) [McDermott *et al.*, 1998b], using the QSRs as predicates. We make one departure from standard PDDL, by allowing for effects that assign an arbitrary relation to one of the spatial aspects. For example, in the action in Figure 1.1, the (center of the) ball can end up with any direction to the (center of the) goal. This is simply a shorthand for a larger number of possible actions.

3.2 Background and Related Work

In this section, we review the literature on generating action models in physical domains based on QSRs and their application in characterizing novelty.

We present action models as state transitions in the physical domain, and states will be described as QSRs between objects. The QSRs are drawn from four calculi, corresponding to four essential spatial aspects of any physical domain: topology, movement,

direction and distance.

Topology

Topological relations refer to the connections between object boundaries. The Region Connection Calculus (RCC) [Randell *et al.*, 1992] stands out as one of the most renowned and extensively used qualitative spatial calculi. Specifically, RCC delineates the topological relationships between regions, with its foundational representation being RCC-8. RCC-8 consists of 8 relations: *dc* (disconnected), *ec* (externally connected), *po* (partially overlapping), *eq* (equal), *tpp* (tangential proper part), *tppi* (tangential proper part inverse), *ntpp* (non-tangential proper part) and *ntppi* (non-tangential proper part inverse) as shown in Figure 3.1 (a).

Movement

Relations in the Qualitative Trajectory Calculus (QTC) [Van de Weghe *et al.*, 2005] describe the qualitative moving direction relations between two moving objects. QTC expresses if one object is approaching $(-)$, moving away from $(+)$, or stationary relative to (0) another object. For example, $(+,+)$ indicates two objects are moving away from each other. There are 9 QTC relations, as shown in Figure 3.1 (b).

Direction

Directional relationships are commonly observed in daily communication. Over the past three decades, numerous qualitative direction calculi have emerged to represent these varied relationships. Cardinal directions work within a consistent, implicit frame of reference. For example, when stating that Object A is south of Object B, it means that Object A is positioned to the south with Object B at the center. STAR-4 [Renz and Mitra, 2004] consists of qualitative direction relations. Given two objects, as shown in Figure 3.1(c), the 2D space is uniformly divided into 8 cones by 4 dotted lines, with the first object in the center and the direction relation is decided by which cone the second object is located in. The 8 cones stand for 8 direction relations: *tp* (top), *tr* (top right), *rt* (right), *br* (bottom right), *bt* (bottom), *bl* (bottom left), *lt* (left) and *tl* (top left).

Distance

The Qualitative Distance Calculus (QDC) [Clementini *et al.*, 1997] indicates the qualitative distance relations between objects. For QDC, the number of space divisions can be flexible and qualitative. The span of each distance division is also configurable in various scenarios. As shown in Figure 3.1 (d), we use the three relations fr (far), nr (near) and to (touching).

Conceptual Distance

The conceptual distance between two QSRs can be defined by *conceptual neighborhood diagrams* [Van de Weghe and De Maeyer, 2005; Dondrup *et al.*, 2015]. In such diagrams, shown in Figure 3.1, nodes are relations, and edges connect conceptual neighbors. Two QSRs are conceptual neighbors if there is no other intermediate relation in a consecutive change. For example, in a scenario where two touching objects move away from each other, their QDC relation changes from to to nr and then to fr. Thus, nr is a conceptual neighbor to both fr and to while to is not a neighbor of fr. Note, however, that all steps of the change may not be observed: if the objects are moving fast enough, we may see it as a transition from to to fr in consecutive video frames. We will use the conceptual distance as a (dis-)similarity measure between QSRs to cluster similar state transitions.

When considering the concept of "novelty", a significant portion of the AI community's efforts have been devoted to identifying or detecting new things, as opposed to comprehending or describing them. Consider a few examples. Sabokrou *et al.* [2018] present a Generative Adversarial Network (GAN)-based system that is designed specifically to detect anomalous or out-of-place objects within images. An additional study, Yahaya *et al.* [2019] attempts to identify novel or distinct human behaviors in the course of routine activities. They accomplished this by analyzing specific attributes that people selected, such as the duration of the activity. Thus, while numerous efforts have been dedicated to instructing machines to identify what is "new" or "different", comparatively fewer have focused on facilitating machines in comprehending or articulating the precise attributes that distinguish something as "new" or "different".

3.3 Proposed Approach

An overview of our approach is shown in Figure 3.2. The inputs are sequences of unlabeled frames (from videos) of the environment without novelty, for base model generation, and with novelty, for detection and characterization. Qualitative states are created by detecting and classifying objects in each frame, and computing their QSRs. Classification assigns each object a type, based on its appearance (e.g., a ragdoll is of type "cat"). If objects of a novel type appear, we have, of course, no knowledge of that type, but we assume that the classifier can recognize different objects with the same appearance as belonging to the same (novel) type. We do not assume that object detection, tracking or classification is perfect, although if they are too unreliable, the quality of the generated model will suffer. Because the four QSR calculi we use are made up of binary relations, a state can be decomposed into a collection of object-pair states. State transitions are created from consecutive object-pair states where a change in at least one qualitative aspect occurs.

To create the base model, we cluster the extracted state transitions. Clustering is done in two stages, as explained later in this section, and results in a set of clusters of similar state transitions. From each cluster, we create one action, which is representative of the

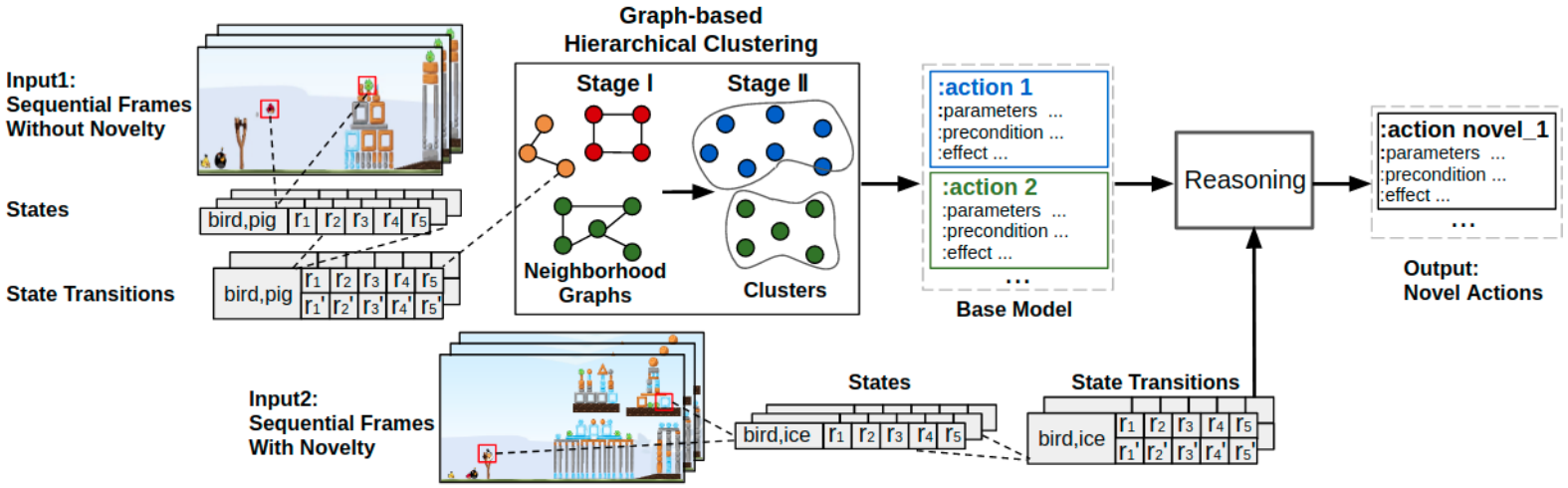


Figure 3.2: An overview of our proposed solution for the problem of novelty characterization by its novel actions in physical domains. The input is two sets of game-playing videos (with or without novelty), and the output is the novel actions (i.e., uncovered state transitions). Our approach consists of four procedures: (1) two sets of states and corresponding state transitions are extracted from both inputs; (2) state transitions are grouped into clusters by our two-stage graph-based hierarchical clustering approach: first, constructing neighborhood graphs, where each node represents a state transition and state transition with similar relation changes are connected; second, merging similar neighborhood graphs into larger clusters; (3) a base model that includes a set of actions is generated from clusters; and (4) repeated uncovered state transitions are selected from Input 2 as the novel actions.

transitions in that cluster.

In the novelty detection step, observed state transitions are matched against actions in the base model; a state transition that is not matched by any action is said to be *uncovered*. Uncovered state transitions that occur repeatedly indicate novelty, once they exceed a certain threshold. From these repeated uncovered state transitions (RUSTs), we extract a set of novel actions that characterize the novel behavior.

3.3.1 State and State Transition

Our qualitative state representation is made up of the QSRs of four calculi, namely RCC-8, QTC, STAR-4 and QDC, and one additional property which describes a change in object existence (EXIST). We refer to each of these five as an *aspect* of the state. Because the relations in each calculus are mutually exclusive, exhaustive and binary, each pair of objects satisfies exactly one relation from each calculus, and a state is the union of the pair-wise states of all object pairs. The possible existence relations are EXIST $= \{(n,n),(n,y),(y,n),(y,y)\}$, i.e., it is the cross product of a binary property of each object. We represent it as a relation for uniformity with the other aspects. Non-existent objects are not detected, and are not represented in the state; a change in existence from y to n means the object is about to disappear (is not present in the next frame) and a change from n to y means the object has just appeared (was not present in the previous frame).

Definition 1 (Object-Pair State)**.** An *object-pair state* is a 2-tuple $s = (op, R)$, where $op = ((o_1, t_1), (o_2, t_2))$ is a pair of typed objects, and $R = (r_1, r_2, r_3, r_4, r_5)$ is the 5-tuple of relations, one for each aspect, that hold between o_1 and o_2, i.e., $r_1 \in$ RCC-8, $r_2 \in$ QTC, $r_3 \in$ STAR-4, $r_4 \in$ QDC, $r_5 \in$ EXIST.

For each frame, we extract one object-pair state for each pair of detected objects. Automatic state extraction can be done by object detection algorithms like Fast-RCNN [Dias *et al.*, 2020] and QSR computation tools like QSRLib [Gatsoulis *et al.*, 2016]. A state transition occurs when the object-pair states refer to the same pair of objects in two consecutive frames that differ in at least one aspect.

Definition 2 (State Transition)**.** Given a sequence of sets of object-pair states $S_1, S_2, \ldots$, where S_i is the set of object-pair states extracted from frame i, a *state transition* is a pair of object-pair states $(s = (op_s, R_S), s' = (op_{s'}, R_{s'})$ such that: (1) $s \in S_i$ and $s' \in S_{i+1}$ for some i; (2) $op_s = op_{s'}$; and (3) $R_s \neq R_{s'}$, We refer to s and s' as the *before* and *after* states of the transition.

In the following, we will use a slightly different representation of state transitions: instead of a pair of object-pair states, (s, s'), we describe a transition as $((t_1, t_2), r_1\text{-}r'_1, \ldots, r_5\text{-}r'_5)$, where (t_1, t_2) are the types of the object pair and r_i and r'_i the relation of the i-th aspect that holds between the objects in the before state s and after state s', respectively. That is, once we have identified state transitions, the individual objects' identities are no longer important; we regard it as a transition of the objects' types. We refer to the

pair r_i-r'_i as the transition in aspect i. Note that r_i may equal r'_i for some i, since a state transition requires only that there is a change in at least one aspect.

3.3.2 Graph-based Hierarchical Clustering

The clustering of state transitions proceeds in two stages. In the first stage, we construct a neighborhood graph based on conceptual similarity, and form initial clusters from the connected components of this graph. In the second stage, we iteratively merge pairs of clusters that satisfy certain conditions to form larger clusters. At both stages, we cluster only state transitions with the same pair of object types. The complete clustering procedure is shown in Algorithm 1.

Neighborhood Graph Construction

Due to the qualitative abstraction, and because state transitions are identified with pairs of object types, not objects, it is common to observe the same state transition more than once in any data set of sufficient size. Hence, we first group all identical state transitions together. These initial sets can be represented as pairs of a state transition and a repetition count, (st, n_{st}).

The neighborhood graph is an undirected graph whose nodes are the unique state transitions. Two state transitions $st_1 = (otp_1, r_{1,1}\text{-}r'_{1,1}, \ldots, r_{1,5}\text{-}r'_{1,5})$ and $st_2 = (otp_2, r_{2,1}\text{-}r'_{2,1}, \ldots, r_{2,5}\text{-}r'_{2,5})$ are connected iff $otp_1 = otp_2$ and $d(st_1, st_2) = \sum_{i=1}^{5}(d_i(r_{1,i}, r_{2,i}) + d_i(r'_{1,i}, r'_{2,i})) = 1$, where $d_i(r_{1,i}, r_{2,i})$ is the conceptual distance between relations $r_{1,i}$ and $r_{2,i}$ in the i:th aspect (i.e., the length of the shortest path between them in the corresponding conceptual distance graph shown in Figure 3.1). In other words, transitions are connected in the neighborhood graph iff they differ in exactly one aspect, and in that aspect either their before or after state relations are conceptual neighbors. The initial clusters are the connected components of this graph.

The first clustering stage improves efficiency by reducing the number of initial clusters for the next stage. It can also improve model generalization since it may cluster some transitions not considered mergeable in the next stage.

Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering starts with an initial set of clusters, and iteratively selects a most similar pair of clusters to be merged into one, until no more mergers are possible [Rokach and Maimon, 2005].

In our application of this idea to clustering state transitions, we impose two conditions that a pair of clusters must meet to be merged: (1) their sets of state transitions must refer to the same object type pair; and (2) for at least δ of the aspects, a fraction τ of transitions in their union must have the same before and after relation. The parameters $\delta \in \{0, \ldots, 5\}$ and $\tau \in [0, 1]$ control how much the actions generated from the clusters are

permitted to generalize from the observed state transitions. To state this more precisely, we need the following definitions:

Definition 3. Let C be a set of state transitions, and $i \in \{1, \ldots, 5\}$. The *representative transition in aspect i for C* is a relation pair r_i-r'_i in aspect i that occurs most frequently in C, taking into account repetitions. If two or more relation pairs in aspect i are tied for the highest frequency, one of them is chosen arbitrarily.

The *agreement in aspect i of C* is the number of state transitions in C whose relation pair in aspect i is the representative transition in that aspect for C divided by the size of C, again taking into account repetitions.

We denote the agreement in aspect i of C by $\eta_i(C)$.

Definition 4. Two sets of state transitions, C_1 and C_2, are *mergeable*, under parameters $\delta \in \{0, \ldots, 5\}$ and $\tau \in [0, 1]$, iff (1) all state transitions in $C_1 \cup C_2$ have the same object type pair, and (2) there are at least δ different aspects $i_1, \ldots, i_\delta$ such that $\eta_i(C_1 \cup C_2) \geq \tau$ for each i in $i_1, \ldots, i_\delta$, i.e., the agreement in each of those aspects of $C_1 \cup C_2$ is at least τ.

Definition 4 tells us which clusters may be merged and therefore also provides the stopping condition for the hierarchical clustering process (when there are no more mergeable clusters). It remains to specify which pair of mergeable clusters are selected in each iteration, i.e., the similarity measure. Given two mergeable candidate clusters C_1 and C_2, their similarity $\theta(C_1, C_2)$ is the average agreement in the δ aspects that have the highest agreement. The cluster pair with the highest similarity score is selected to be merged.

3.3.3 Base Model Formation

The output of clustering is a set of clusters; each cluster is a set of state transitions. By construction, all state transitions in each cluster have the same object type pair. We generate one action from each cluster. How this is done is described in this section. The resulting set of actions is our base model.

Representative State Transition

Given a cluster C, we form a single representative state transition, $RST_C = (otp_C, r_1\text{-}r'_1, \ldots, r_5\text{-}r'_5)$, where otp_C is the object type pair common to all state transitions in C and r_i-r'_i is the representative transition in aspect i for C if $\eta_i(C) \geq \tau$. If $\eta_i(C) < \tau$, the representative state transition has a special value $\star$ in place of a relation pair for aspect i. The meaning of this value is that any transition is considered possible in this aspect. As the example in Figure 3.3 shows, the cluster consists of five state transitions (note that the third and fourth rows are two repetitions of the same state transition). The agreements in each of the five aspects are $1, 0.8, 0.8, 0.6$ and 0.8, respectively. Assuming $\tau = 0.7$, we obtain the representative state transition $((\text{bird}, \text{pig}), \textit{dc-dc}, (0, 0)\text{-}(-, 0), \textit{bl-lt}, \star, (y, y)\text{-}(y, n))$.

Algorithm 1 Graph-based Hierarchical Agglomerative Clustering.

procedure G-HAC
  # Stage 1: Neighbourhood graph clustering
  # Collect unique state transitions with counts:
  nodes$(G) = \{(st_1, n_{st_1}), \ldots, (st_n, n_{st_n})\}$
  for $(st_i, n_{st_i}), (st_j, n_{st_j}) \in$ nodes(G) **do**
    if $st_i \neq st_j \wedge d(st_i, st_j) = 1$ **then**
      connect (st_i, n_{st_i}) and (st_j, n_{st_j}) in G.
  $C_S =$ connected_components(G) # initial clusters
  # Stage 2: Hierarchical clustering
  # mergeable$^{\delta,\tau}(C_S)$ denotes cluster pairs that are
  # mergeable under params δ and τ (cf. Definition 4)
  while $\exists C_i, C_j \in$ mergeable$^{\delta,\tau}(C_S)$ **do**
    # Select pair with highest similarity:
    $C_i, C_j \leftarrow \text{argmax}\{\theta(C_i, C_j) \mid C_i, C_j \in \text{mergeable}^{\delta,\tau}(C_S)\}$
    $C_S = (C_S \setminus \{C_i, C_j\}) \cup \{C_i \cup C_j\}$
  return C_S

Actions

An action consists of three components: parameters, precondition and effect. The action generated from a cluster C has two parameters, whose types are given by the common object type pair of the cluster. The action's precondition and effect are determined by the before and after relations of the representative state transition RST_C. When RST_C has a relation pair r_i-r_i' in aspect i, the relation in the before state, r_i, becomes an action precondition, and the relation in the after state, r_i', becomes an effect if it is different from r_i. This follows the planning convention that the persistence of facts is implicit [McDermott *et al.*, 1998b]. When RST_C has an arbitrary change, $\star$, in aspect i, the action has no corresponding precondition or effect that allows any one of the relations in this aspect to become true. This can be seen as a shorthand, representing one action for each possible relation that can hold in the after-state. The action created in the example above is also shown in Figure 3.3.

The predicates used by these actions are simply one for each possible relation in each of the five aspects that make up our qualitative state representation.

3.3.4 Novelty Characterization

The output of the first step is the base model: a set of actions, A_0, that describe the possible state transitions in the environment without any novelty. In the second step, we observe new state transitions as the agent interacts with the world, and we attempt to match them to actions in this set.

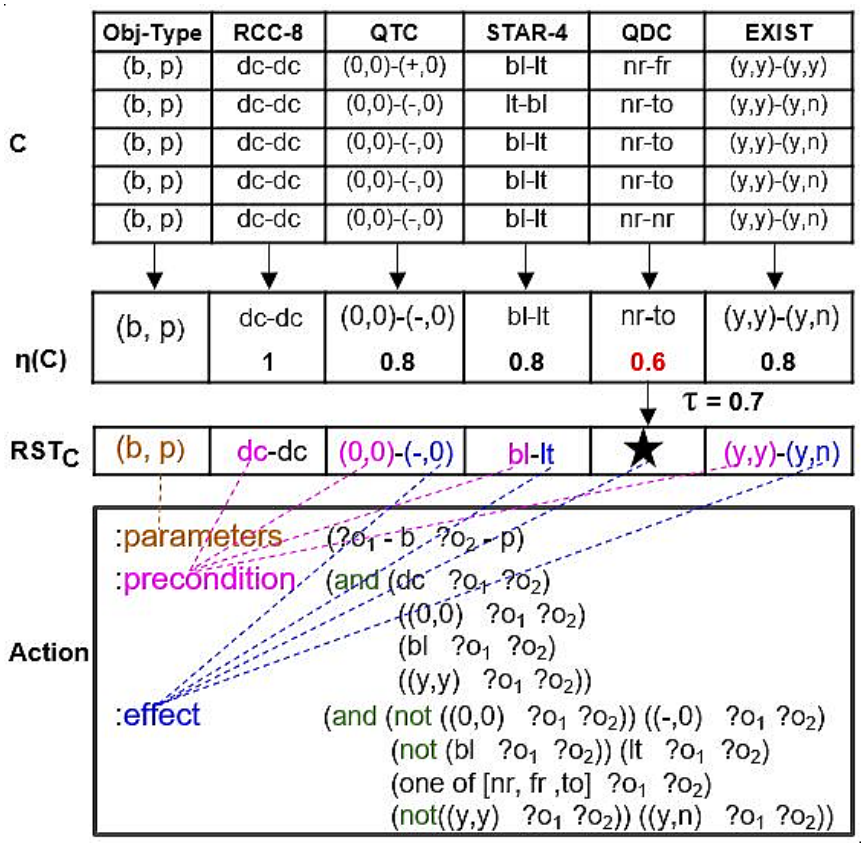


Figure 3.3: Example of how an action is created from a cluster. There are five state transitions in cluster C as shown in the table. These state transitions share the same object type pair (bird, pig). Below the table is shown the most frequent relation pair for each aspect and the corresponding agreement ($\eta(C)$). The representative state transition (RST_C) also has the most frequent transition in each aspect, except for the distance aspect, where it has an arbitrary change effect (marked by a $\star$) because the agreement in this aspect in C is below the threshold $\tau = 0.7$. Finally, the generated action.

Definition 5. A state transition (s, s') for the typed object pair $((o_1, t_1), (o_2, t_2))$ is *covered* by action a iff (1) t_1 and t_2 are the types of a's parameters; (2) the before state s satisfies a's precondition; and (3) a's effects can be applied to s in a way that yields the after state s'.

A state transition is covered by a set A of actions iff it is covered by some action $a \in A$.

If an action has an arbitrary change effect in an aspect, we can choose any of the possible relations for this aspect in the after-state; hence the wording of condition (3) above.

A state transition not covered by A_0 is said to be *uncovered*. Because the base model is

learned from a finite set of observations, it is not perfect, and hence there will normally be some uncovered state transitions when observing the environment without novelty. However, because novelties in a physical environment cause a persistent change to the behavior of the environment, or some types of objects in it, we claim that *repeated* uncovered state transitions (RUSTs) will be more frequent when a novelty is introduced, compared to the non-novel environment, and that this difference is often sufficient to detect novelty. We demonstrate that this claim holds for most of the novelties in our case study in Section 3.5.

The RUSTs are also the basis for generating the characterization of the novelty once detected. From them, we create a set of *novel actions*, following the same process as described in Section 3.3.3. Note, however, that we did not apply any clustering to the RUSTs before generating novel actions.

3.4 Experiment 1: Base Action Model Generation

We conduct two sets of experiments: First, we evaluate how well the generated base action model describes the non-novel physical environment. We compare our method with three other methods from the literature, and evaluate the impact of the δ and τ parameters on the model. In the second set of experiments (Section 3.5), we introduce novelties and measure how well our approach can detect and characterize them.

3.4.1 Dataset

Our testbed is a classical physics-based video game – Angry Birds. In this game, the player uses a slingshot to shoot birds at structures made up of different kinds of building blocks, with the aim of destroying all pigs to clear a level. The game can be described as a "physics-based puzzle": the behavior of objects as they fly, collide, and so on is calculated by an internal physics simulation, and solving a level requires an element of logical thinking. Playing Angry Birds has been a challenge for AI agents since 2012 [Renz *et al.*, 2015].

We use the Science Birds open-source clone of the game[1] [Ferreira and Toledo, 2014], since this allows to introduce novel behaviors. We extract objects and the properties needed to compute the qualitative state in each frame from the game engine directly, instead of indirectly from the rendered video frame. Thresholds for qualitative distances (QDC relations) were set based on the distribution of all object pair distances in the training data set, as follows: touching $to \in [0, 10]$, near $nr \in [10, 50]$, and far $fr \in [50, \infty]$.

Training Data

is the set of qualitative state transitions collected from 170 game levels, each played 10 times. To generate training data exhibiting a broad range of possible object behaviors,

[1] https://gitlab.com/aibirds/sciencebirdsframework/-/tree/release/alpha0.4.1

we use a random game-playing agent to play these levels. This agent shoots each bird randomly, without any strategy or target. This means each time the agent replays a game level, different actions are likely to happen, which helps capture more infrequent actions.

To limit the size of the experiment, we focus on actions related to birds. These are the most interesting from the agent's perspective, since it is the birds that the player controls. This means we keep only the state transitions in which at least one of the two objects is of the type "bird", resulting in 6233 state transitions in total in the training data set.

Test Data

is extracted from playing another 30 unseen game levels, each 3 times. These are played by an agent that is more goal-directed, targeting the pigs. The agent's strategy is not sophisticated: it selects which pig to shoot at randomly, ignoring obstacles. Hence, it is called the "naive" agent. Nevertheless, it is a better representative of how an intelligent agent may act than the random agent used in training. Only state transitions in which at least one of the two objects is of type "bird" are kept, resulting in 520 state transitions in the test data set.

3.4.2 Evaluation Metrics

The performance of the base model is evaluated on three criteria: (1) the validity of its actions, meaning the degree to which they represent state transitions that can actually happen in the game; (2) coverage on the test set, i.e., on unseen game levels without novelty; and (3) size, meaning the number of actions. A perfect model would achieve 100% validity and coverage, with a small number of actions. Actions corresponding exactly to state transitions in the training set are sure to be valid, but their coverage outside the training set may be low. Introducing some generalization increases coverage on the unseen test set, but also risks lowering validity.

Action Validity

Since we cannot know the set of all state transitions that are possible in the environment, we approximate it with the union of the training and test sets; call this set S_U. Actions without arbitrary change effects represent only one state transition, and thus have a validity score of either 1 (if found in S_U) or 0 (if not). An action a that has an arbitrary change effect in one or more aspects can result in a set of state transitions, all $(otp_a, r_1\text{-}r'_1, \ldots, r_5\text{-}r'_5)$ such that either r_i is in the precondition of a and r'_i in the effect of a or equal to r_i, or a has an arbitrary change effect on aspect i. The validity of the action is the fraction (in $[0, 1]$) of these state transitions found in S_U. The validity score of the base model is the average validity of all its actions.

Coverage on Test Data

Coverage on test data measures how well the base model describes game levels outside the training set, i.e., how general it is. This is measured simply as the fraction of state transitions in the test set that are covered (cf. Definition 5) by the base model's action set.

3.4.3 Baseline Approaches

We compare our proposed graph-based hierarchical clustering approach (G-HAC) with three baseline approaches: (1) the Cube-Space Action Auto-Encoder (AAE), proposed by Asai and Muise [2020]; (2) the widely-used clustering method K-means; and (3) Latent Semantic Analysis (LSA). We also apply hierarchical agglomerative clustering (HAC) alone as a comparison. Without the neighborhood graph-based clustering stage, the initial clusters each contain only repetitions of a unique state transition.

AAE is a neural network that produces action models by learning object distribution changes as state transitions. State transitions are represented as numeric vectors by embedding the pixel distribution of consecutive game state images. With the state transitions as the input to AAE, the result is which action the input state transition belongs to, where the total number of actions K is predefined. K-means and LSA were both successfully used by Duckworth *et al.* [2019] to classify human activities based on the similarity of their QSR changes. Given a predefined number of clusters K (the same as in AAE), they both return K clusters (i.e., activities).

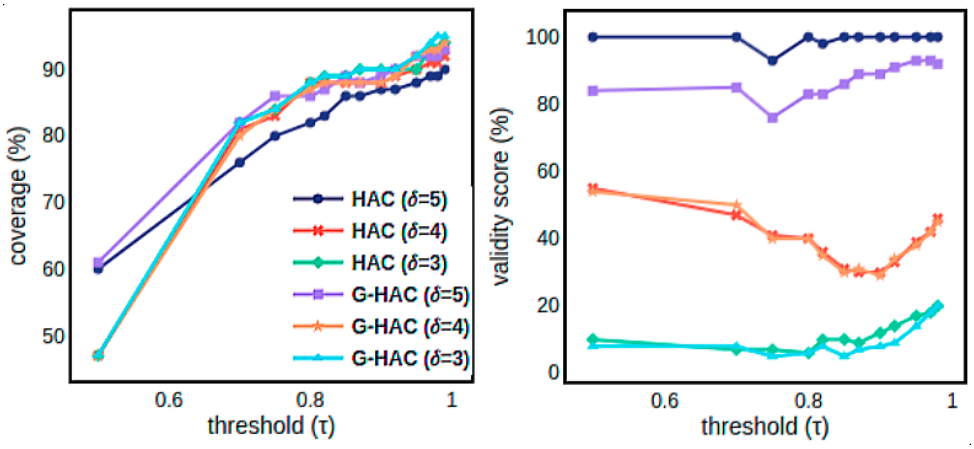


Figure 3.4: Coverage on test data (left) and validity (right) of the base models generated by G-HAC and HAC with different parameter values. $\delta \in \{3, 4, 5\}$, $\tau \in [0.5, 0.99]$.

3.4.4 Experiment Results and Analysis

Both G-HAC and HAC are tested with different combinations of $\tau \in [0.5, 0.99]$ and $\delta \in \{3, 4, 5\}$. Figure 3.4 shows the results. Increasing τ boosts coverage for both methods, while varying δ does not affect coverage much. This is because as τ increases, fewer clusters are mergeable, and more actions are generated. For the same parameter settings, the model generated by G-HAC has slightly better coverage than the one generated by HAC alone. Validity, on the other hand, is strongly affected by δ. When $\delta < 5$, hierarchical clustering can generate actions with arbitrary change effects, and all the possible state transitions represented by those actions are rarely present in the combined training and test sets.

The three other methods all require a number of clusters K as an input parameter. We use values of K equal to the number of clusters generated by G-HAC with all different δ and τ values. Figure 3.5 shows the coverage–validity trade-off achieved by the models generated by all five methods, under all tested parameter settings. The closer a model is to the top right corner, the better its performance. Models generated by AAE and LSA have visibly lower validity scores, though their coverage is high. In other words, these methods generate models that over-generalize.

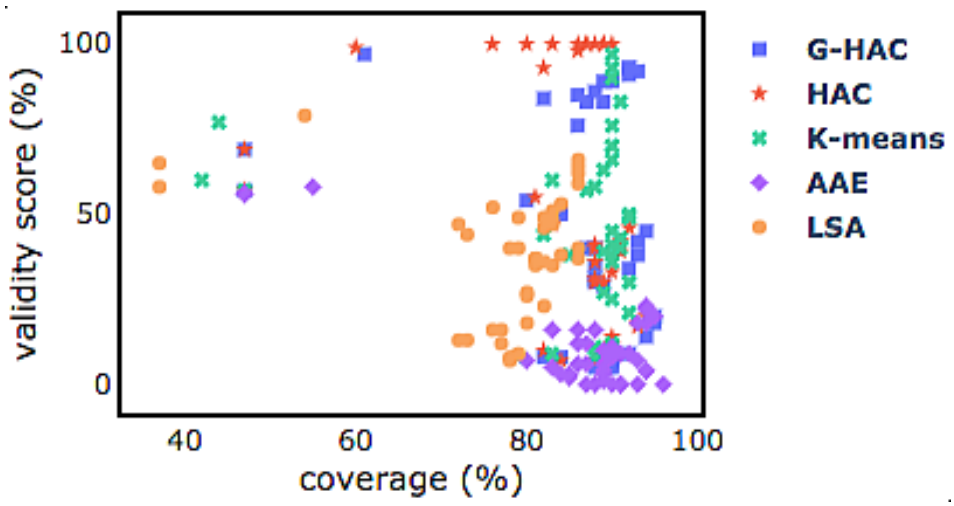


Figure 3.5: Comparison of coverage and validity achieved by all models generated using G-HAC, HAC, AAE, K-means and LSA.

Table 3.1: Size, coverage and validity of the best model generated by each method. These correspond to the points that are closest to the top right corner in Figure 3.5.

	G-HAC	HAC	K-means	AAE	LSA
No. of actions	593	662	663	166	496
Coverage (%)	93.0	90.4	90.4	94.1	91.0
Validity (%)	95.2	100.0	94.6	18.7	66.6

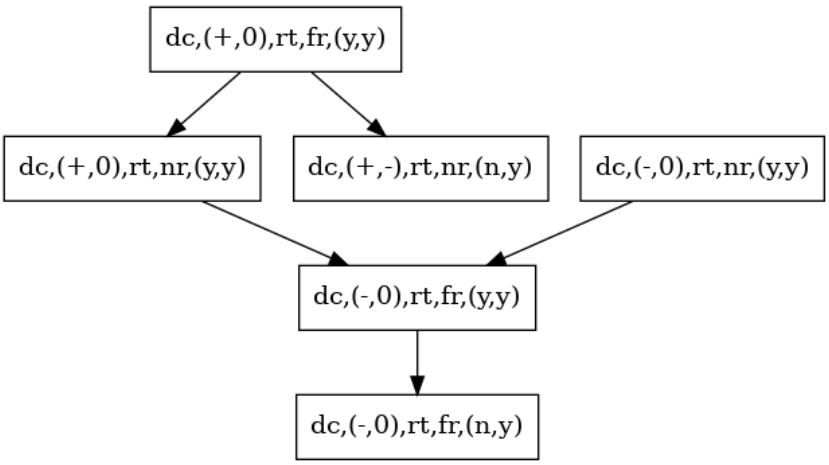


Figure 3.6: Part of the model of interactions between "red bird" and "fat ice rect". The aspects, from left to right, are RCC8, QTC, STAR4, QDC and EXISTS.

Table 3.1 shows the size, coverage and validity of the best (i.e., closest to the top right corner in Figure 3.5) model generated by each method. While no method dominates in all three performance criteria, the G-HAC method offers the best trade-off. The model generated by AAE has the highest coverage, but it is only slightly higher than that of the G-HAC model and its validity is vastly lower.

Clustering with G-HAC and generating action take around 20 seconds. HAC alone takes about 10 times longer.

3.4.5 Using the Model

The generated model cannot be used directly to plan the actions of an Angry Birds playing agent, because it does not distinguish which actions are actions of the agent and which are uncontrollable events that happen as delayed consequences. However, plan existence can be used as a predictor of what future states can occur. As an example, Figure 3.6 shows a small part of the model of interactions between objects of the "red bird" type and the "fat ice rect" type. It can be seen that, for example, these particular starting conditions never result in the ice block being destroyed.

3.5 Experiment 2: Novelty Detection and Characterization

We select the best model generated by G-HAC, as shown in Table 3.1, as the base model for testing novelty detection and characterization.

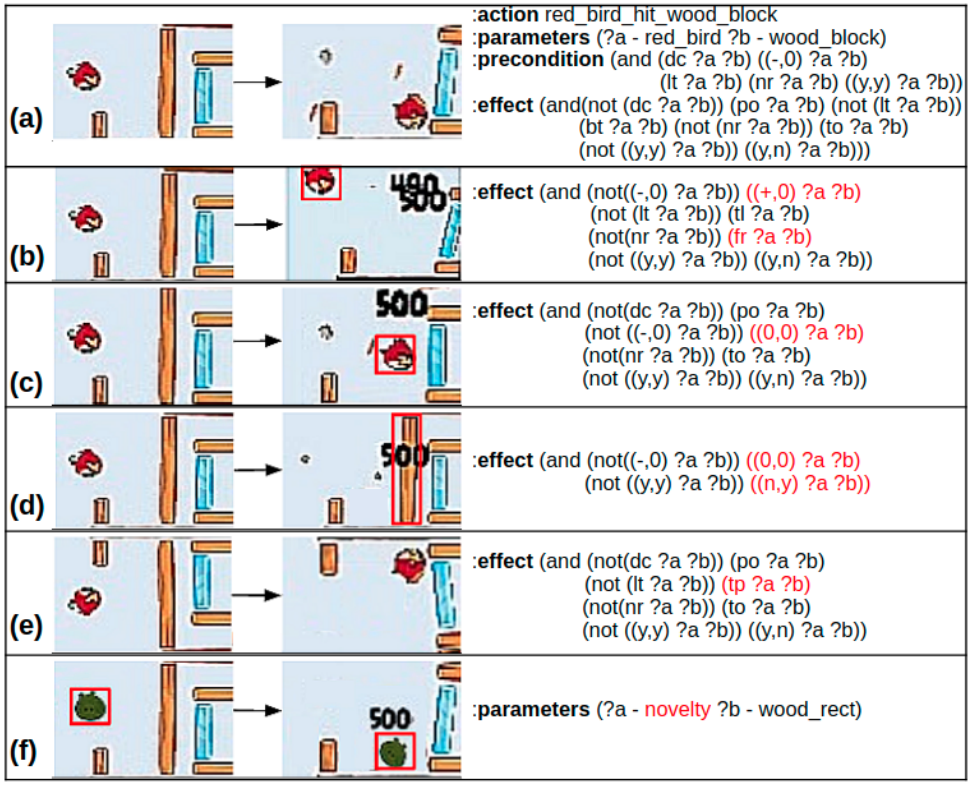


Figure 3.7: (a) A red bird hits a wood block in the environment without novelty; right: the corresponding action model. (b) Novelty 1: Increasing the bounciness of the red bird causes it to bounce back after hitting the wooden block. QTC and QDC in the effect change accordingly. (c) Novelty 2: Increasing the linear drag of the red bird causes it to fall slowly and more vertically. QTC in the effect is different. (d) Novelty 3: By increasing the health points of the wooden block, a single hit from the bird is not sufficient to destroy it. The EXIST relation changes in the effect. (e) Novelty 4: Rotating the view of the environment 180 degrees. The STAR-4 relation in the effect changes to *tp*. (f) Novelty 5: Introducing a new object type with a different appearance but the same properties as the red bird.

3.5.1 Novelties

For this experiment, we introduce five novelties, of three different kinds, in the Angry Birds game. Novelties 1–3 are changes to the properties of some object types, which change their behavior; novelty 4 is a "global" change – rotating the screen 180 degrees – which causes a change in the apparent behavior of almost all objects; novelty 5 introduces a new object type, with a different appearance but which is behaviorally the same as the red bird type in the non-novel game. These correspond to levels 2, 3 and 1 of the classification proposed by DARPA (2019). Figure 3.7(a) shows an example of one action in the standard game, and how this action may change with each of the novelties (b–f).

3.5.2 Novelty Detection

Detecting novelty is a prerequisite for characterization. To determine the detection ability of our approach and the number of levels needed to detect these novelties, we insert the novelties into the 30 unseen game levels used to generate the test data set. For each novelty, and also for the unseen levels without novelty, we randomly sample $n_l = \{5, 10, 20\}$ of the levels, observe the naive agent play each of them three times, and collect all state transitions that involve an object of type red bird. We then calculate the number of repeated uncovered state transitions (RUSTs) in this set using the base model. We repeat the experiment 100 times for each novelty (including no novelty) to obtain a distribution of the number of RUSTs, which is shown in Figure 3.8.

Without novelty, the number of RUSTs is always small (< 10). Novelties cause a greater number of RUSTs. Novelties 4 and 5 are clearly detectable even at $n_l = 5$, while novelties 1–3, which change the behavior of only one object type, are harder to detect. Nevertheless, at $n_l = 20$ we can detect all novelties except #2 with a high recall rate without false positives, by setting the decision boundary at the maximum number of RUSTs seen in the non-novel environment (shown by the dashed line in the graph).

Novelty detection and characterization time is linear in data size (length of the frame sequence), and at $n_l = 20$ takes around 40 seconds, over 99% of which is extracting object-pair states and transitions.

3.5.3 Novelty Characterization

We create characterizations of each of the novelties from the RUSTs collected after observing $n_l = 20$ game levels. The characterization of novelty k is a set of novel actions, NA_k, each corresponding to one of the RUSTs.

We evaluate the specificity of these characterizations by measuring their overlap. The overlap of two novel action sets, $O(NA_i, NA_j)$, is the fraction (in $[0, 1]$) of actions in NA_i that are also found in NA_j, i.e., $|NA_i \cap NA_j|/|NA_i|$. Note that this is not symmetric. The overlap is an indication of the risk, given the characterization of novelty j, subsequent observations of game levels with novelty i would not be recognized as a different novelty, or, in other words, the degree to which novelty i fails to stand out.

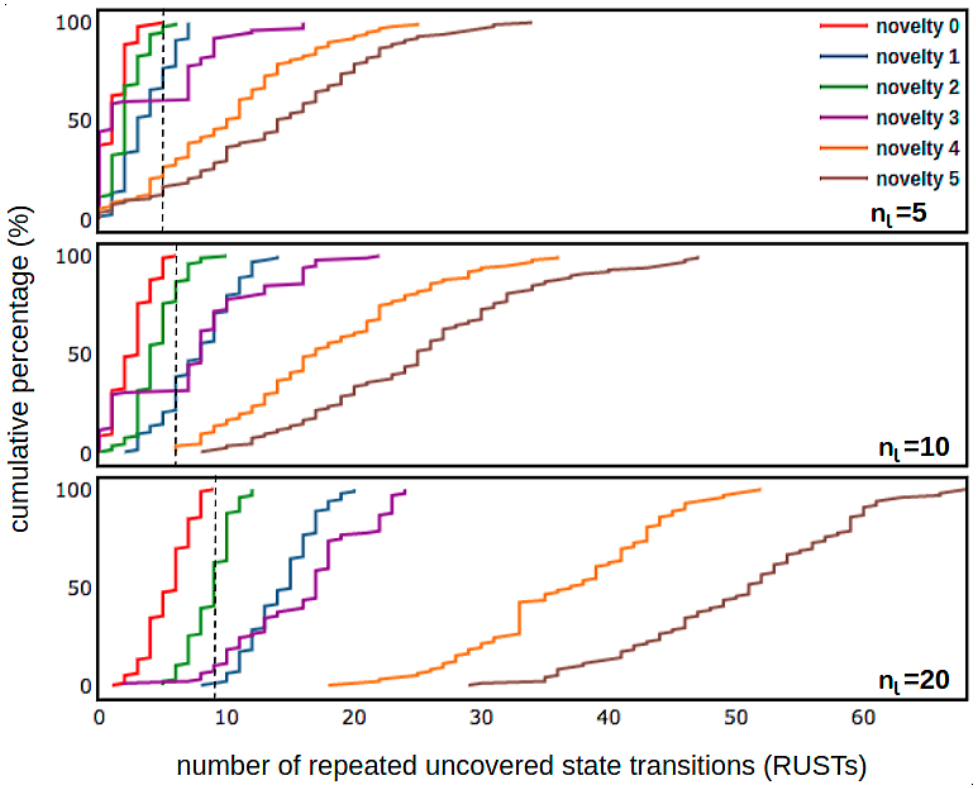


Figure 3.8: Cumulative distribution of the number of repeated uncovered state transitions (RUSTs) for each novelty (including the non-novel environment as #0) under different numbers of game levels observed (n_l) over 100 trials. The vertical dashed line in each graph marks the maximum number of RUSTs in non-novel levels.

Table 3.2: The average overlap rate $OR(NA_i, NA_j)$. NA_i is the row and NA_j is the column. In parenthesis, the standard deviation.

	$\mathbf{NA}_1$	$\mathbf{NA}_2$	$\mathbf{NA}_3$	$\mathbf{NA}_4$	$\mathbf{NA}_5$
$\mathbf{NA}_1$		0.22 (0.06)	0.02 (0.03)	0	0
$\mathbf{NA}_2$	0.36 (0.11)		0	0	0
$\mathbf{NA}_3$	0.02 (0.03)	0		0	0
$\mathbf{NA}_4$	0	0	0		0
$\mathbf{NA}_5$	0	0	0	0	

Table 3.2 shows the average overlap, and standard deviation, obtained over the 100 repeats of the experiment. Except for novelties 1 and 2, the overlaps are zero or close to zero, indicating that the novel actions are highly distinct. Novelties 1 and 2, which both change properties of red bird objects, produce some visually similar behaviors, and therefore a greater overlap. This can be seen in Figure 3.7 (b) and (c), as the position of the bird in one can be an intermediate state compared to the other. This indicates that our action-based novelty characterization can not only distinguish novelties, but also see similarities between them.

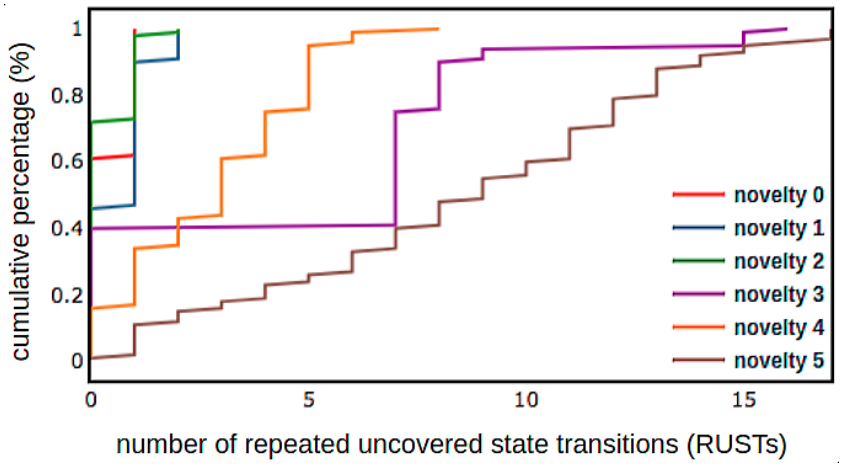


Figure 3.9: Cumulative distribution of the number of repeated state transitions which are uncovered by $A_0 \cup NA_1$. For each novelty, these are taken over the 10 remaining unseen game levels.

To further examine the quality of the characterization of each novelty, we test how well it can help to detect the same novelty and still distinguish other novelties in new, unseen game levels. We take the 10 remaining unseen game levels, after selecting $n_l = 20$ that the characterization is based on, and collect the state transitions from three plays each of these in each environment version, i.e., with each novelty as well as without novelty. From each of these sets, we calculate the number of repeated state transitions that are uncovered by $A_0 \cup NA_i$. Figure 3.9 shows the result for novelty 1. As can be seen, the

number of RUSTs relative to $A_0 \cup NA_1$ for novelty 1 as well as no novelty are now small (smaller in fact than what we observed in the graph for $n_l = 10$ in Figure 3.8). This means that after we incorporate the novel actions into the model, new, unseen levels with novelty 1 no longer appear novel; in other words, the characterization recognizes new instances with the same novelty. The other novelties, except #2, still stand out, and thus would be detected as different novelties after learning of novelty 1. Novelty 2, which is the most difficult to detect and also has a high overlap with novelty 1, does not.

3.6 Conclusion

We propose an approach that automatically generates action models from physical domains in an unsupervised manner. Our approach has been evaluated on Angry Birds, a typical physical domain, and we have proved its ability in automatic action description generation. We consider this to be an important first step for AI agents to solve the problem of action description generation in physical domains, which is the basis of applying planning techniques to solving real-world problems.

Furthermore, our approach can automatically generate a characterization of novelty in physical environments, based on the novel observable behaviors (actions) it creates. We demonstrated, still using the physics-based puzzle game Angry Birds as our testbed, that our approach can detect and distinguish most of the novelties. The more precisely the base action model describes the non-novel environment, the more accurate our novelty detection and characterization will be. Thus, refining the base action model with information not just about what actions are possible, but also what sequences of them can occur, how likely they are, and quantitative delays between them is a direction for future work. We also seek to evaluate how well the novelty characterization can guide the adaptation of an agent's behavior, for example, using the extended model to predict possible action outcomes. Also, applying this approach to other vision-illustrated domains, such as Minecraft and BlocksWorld, in addition to the Angry Birds game, would allow us to evaluate the generalization ability of our method across different domains.

Acknowledgments

The work in this chapter was supported by the DARPA SAIL-ON program under contract W911NF-20-2-0002. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Chapter 4

Action Model Generation from Narrative Texts

Narrative texts stand out as another domain that poses significant challenges for automatic action model generation. In contrast to instructional texts—such as recipes, manuals, and navigational instructions—which have been the primary focus of much past literature, narrative texts exhibit unique characteristics. They are often written in a more colloquial manner and frequently employ intricate clause structures to convey intricate ideas, including conditional events. Given the rich and varied nature of these narratives, it's impractical to rely on the manual creation of action models by human experts. To address this challenge, this chapter presents NaRuto, an innovative system designed to create action models directly from narrative texts. Moreover, NaRuto is capable of rendering these models into structured, planning-language-style action representations. The efficiency and reliability of the NaRuto system are underlined through a comprehensive series of experimental results that are shared in subsequent sections.

The organization of this chapter unfolds as follows: Initially, we provide an overview, background, and motivation behind our NaRuto system in Sections 4.1 and 4.2. Subsequently, we intricately present the system's pipeline and its associated modules in Section 4.3. This contains aspects such as event extraction and representation, along with action model formulation. It's noteworthy to mention two distinguishable event types we've identified and addressed: conditional events and argument events. The latter, argument events, which emerge as a groundbreaking event dependency relationship identified by us, are further detailed in Chapter 5. Moving forward, we design and implement a series of experiments aimed at evaluating the performances of NaRuto. These experiments include: (1) comparative analysis with both fully and partially automated action model generation models, measured based on their generated action representations; (2) alternative plan generation strategies; and (3) an application for guiding the story generation capabilities of large language models.

4.1 Introduction

AI planning comprises generating action sequences, i.e., plans, from the initial state of the problem to the goals. To construct such action sequences, AI planners require action models specified in a declarative planning language, such as the Planning Domain Definition Language (PDDL) [McDermott *et al.*, 1998a]. Building action models by hand is laborious and requires domain expertise. For narrative planning applications in particular (e.g., see the work by Porteous *et al.* [2013]), due to the large variety of activities that occur in unstructured narrative texts, constructing action models for such narrative domains is challenging.

Recently, researchers have attempted to extract action models from narrative texts such as short stories and movie synopses [Hayton *et al.*, 2017; Huo *et al.*, 2020; Hayton *et al.*, 2020]. However, the methods proposed so far either generate quite simple and highly specific action models, or rely on human input to complement or correct automatic extraction. Fully automated approaches have been applied to instructional texts such as recipes, user manuals and navigational instructions [Mei *et al.*, 2016; Lindsay *et al.*, 2017; Feng *et al.*, 2018; Olmo *et al.*, 2021] (or, in some cases, transcriptions of plans generated from a ground truth domain into text). Such texts, however, lack many of the complexities of narrative texts, which are typically more colloquial and use complex clauses to express, e.g., conditional events. Hence, narrative texts (such as movie plots) are more difficult to comprehend. Therefore, how to automatically extract action models from narrative texts is still an open question.

In narrative texts, events tell us what happened, who or what was involved, and at where. Actions are portrayed as events, so we extract events as the basis for generating actions. Event extraction is based on semantic role labeling (SRL) [Gildea and Jurafsky, 2002], which identifies verbs and their arguments, as spans of text, and labels the arguments with their semantic role (e.g., agent, patient, modal modifier, etc.). We complement SRL with several processing steps to refine events and their semantic relations. In particular, there are two types of events that must be distinguished in order to accurately interpret the text:

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, consider this variation of the sentence in Figure 1.2: “Bryan tries to hit Jack”. Here, the event verb is “try”, and its argument is the event “[Bryan] hit Jack”. Clearly, the preconditions and effects of this event may differ from those shown in the example. Event arguments can be nested: If “Daniel sees Bryan try to hit Jack”, then the (complex) event “Bryan try ([Bryan] hit Jack)” is itself the argument of “see”. Since the occurrence of argument events depends on the main event, we regard the main and argument events as a whole by merging their verbs (e.g., “try” and “hit” become “try to hit”).

Events as conditions Narrative texts frequently mention events that happen only if or when some condition(s) happens. For example, in "She will hate me if I tell the truth.", the event with the verb "tell" after "if" is a condition. Unlike argument events, such conditional events are not generally dependent on the conditioned event. Thus, we distinguish such conditional events from their consequent events and generate actions from them separately.

To generate action models from events, we make predictions of the preconditions and effects. In Figure 1.2, for instance, being "close to" Jack is a precondition for Bryan to hit him in the face. We fine-tune a GPT [Radford *et al.*, 2018] and BART [Lewis *et al.*, 2019] based on commonsense knowledge graphs [Sap *et al.*, 2019], resulting in a system called COMET-BM. In COMET-BM, the input consists of an event and a relation type, while the corresponding output includes relevant concepts as preconditions or effects.

In this chapter, we present NaRuto (an abbreviation of "narrative" and "automated"), an innovative, fully automated system that derives planning action models from narrative texts in two stages: extracting structured event representations and constructing action models using commonsense relations. Unique for its capacity to process narrative complexities and leverage commonsense knowledge while remaining fully automated, NaRuto is compared to previous methods based on classical narrative planning domains. Experimental results indicate that NaRuto consistently generates superior action models, surpassing fully automated SOTA methods and even some methods reliant on human input.

4.2 Background and Related Work

Most events, or actions, mentioned in narratives are familiar ones. Their preconditions and physical and emotional effects are largely common-sense knowledge. Thus, we can look to commonsense knowledge (graphs) that incorporate events to infer them. Despite the methods from the above-mentioned action model generation work, the preconditions and effects of actions can be inferred from commonsense knowledge (graphs) based on the events.

ConceptNet [Speer *et al.*, 2019] is a knowledge graph of concepts, which may be verb, noun or adjective phrases, and their relations, which brings together 3.4 million entity-relation tuples information collected from many sources, including crowdsourced (e.g., OMCS [Singh, 2002] and DBPedia [Lehmann *et al.*, 2015]) and curated (e.g., OpenCyc [Lenat and Guha, 1989]). ConceptNet is composed of 36 relations concentrating primarily on taxonomic and lexical knowledge (e.g., RelatedTo, Synonym, IsA) and physical commonsense knowledge (e.g., MadeOf, PartOf).

The ATOMIC [Sap *et al.*, 2019] knowledge graph is made up of 880K tuples linking 24K events to statements using 9 relations, including dynamic aspects of events like causes and effects, if-then conditional statements, social commonsense knowledge and mental states. For example, the relation *xNeed* holds between an event or action and a prerequisite for

the action's subject to perform it, as in, for instance, "X gets X's car repaired" *xNeed* "to have money". Crowdsourcing was used for both data collection and verification of this dataset. Hwang *et al.* [2021] presented ATOMIC-2020, a similar but more robust dataset. It is a new, high-quality commonsense knowledge graph comprising 1.33 million commonsense knowledge tuples throughout 23 relations, encompassing social, physical, and event-based aspects of everyday inferential knowledge.

We utilize these commonsense knowledge graphs to infer the pre- and post-conditions of action models because they reflect human experience and reasoning. These elements ensure that our model can create reliable action models from narrative texts, in an automatic manner.

4.3 Proposed Approach

An overview of our proposed approach is shown in Figure 4.1. We extract event occurrences, consisting of verbs and their arguments, using the AllenNLP [Gardner *et al.*, 2018] semantic role labeling (SRL) system, which is a BERT-based neural network [Shi and Lin, 2019]. We further use heuristic rules, based on dependency parse and POS tagging information, obtained using Stanford CoreNLP [Manning *et al.*, 2014], for detecting phrasal verbs, argument events, and condition events, and update event structures accordingly (Section 4.3.1). We create action models from the events in two steps (Section 4.3.2): The first employs a commonsense event relation predictor to generate candidate precondition and effect phrases; in the second, these are filtered using textual similarity and contradiction, so that both preconditions and effects are distinct and consistent.

4.3.1 Structured Event Representation

Event Verb and Arguments

An event occurrence e consists of a verb or phrasal verb $V(e)$ and a set of labeled arguments $\mathbb{A}(e)$. Verbs are lemmatized. We call any event whose verb lemma is "be" or "have" a *statement*, since these describe facts rather than events occurring, and we do not generate action models from them, except if an argument of the statement is an event.

The SRL system follows the PropBank annotation schema [Bonial *et al.*, 2012], which divides argument labels into numbered arguments (ARG0–ARG5), for arguments required for the valency of an action (e.g., agent, patient, and so on), and modifiers of the verb, such as purpose (PRP), locative (LOC), and so on. Argument values are spans of text. Event Occurrences in Figure 4.1 illustrate extracted event arguments with their respective labels as colored chunks.

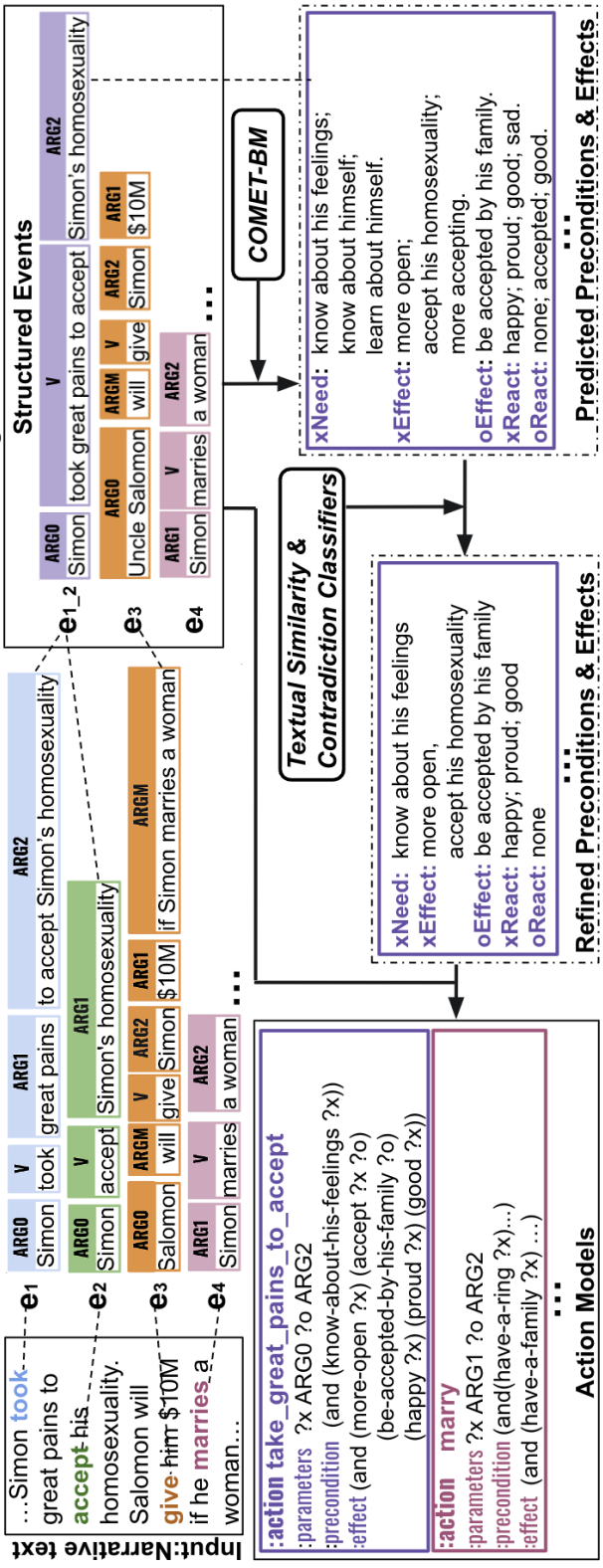


Figure 4.1: Overview and example of our proposed approach for automatically generating action models from narrative text. The example input is part of a plot summary for the movie "Man Is a Woman", from Bamman *et al.* [2013].

Entity Resolution

We apply co-reference resolution to the input narrative texts, and substitute the first mention of any resolved entity for later mentions. For example, in Figure 4.1, "his" in the input text is substituted with the referenced entity "Simon's" in the events $e1$ and $e2$. We use a document-level inference-based LSTM model [Lee *et al.*, 2018] from AllenNLP for the co-reference resolution task.

Phrasal Verb Detection

Phrasal verbs are common in English, and identifying them is important because their meaning is often different from that of the verb part (e.g., "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 apply the following rule, adapted from [Komai *et al.*, 2015], 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[1].

Argument Event Detection

We use the argument structure provided by the SRL system, together with a rule-based method that relies on dependency parsing information, to determine which events are arguments of other events.

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 4.1, ARGM of e_3 contains $V(e_4)$ ("marries"), but e_4 is not an argument of e_3.

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*) or auxiliary (*aux:pass*). (iii) All of e_j is contained in an argument of e_i that is labeled with either ARGM-PRP ("purpose") or ARGM-PNC ("purpose not cause").

If an event e_i has argument event(s), we take the span of all the event verbs and words within the range of the verbs as a verb phrase to be the updated $V(e_i)$. As shown in Figure 4.1, e_2 is an argument event of e_1, so they are combined to e_{1_2}, and $V(e_{1\_2})$ becomes "take great pains to accept".

To test the general applicability of our designed argument detection rules, we performed a small-scale evaluation of this method. We manually annotated 114 pairs of events

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

with containing-contained relations, which were extracted from 35 randomly selected sentences within a movie plot summary dataset from Bamman *et al.* [2013], finding that in 34 cases, the contained event is indeed an argument of the containing event. Based on this sample, the proposed rules achieve a precision of 1 (i.e., no false positive) but a recall of 0.44; thus, they are somewhat conservative but precise.

Condition Event Detection

Conditional promises, threats, etc., are common in narrative text, as e_4 in Figure 4.1 shows. The condition event e_4 is not an argument of e_3, and should be removed from e_3. 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 4.1, the signal word "if" is in between the consequence e_3 and the condition e_4. Our method is a modification of that introduced by Puente *et al.* [2010]. Event e_j is determined to be a condition of e_i iff (a) one of the sub-sequences $V(e_i) \quad S \quad V(e_j)$ or $S \quad V(e_j) \quad V(e_i)$, where S is one of the signal words or phrases, appear in the sentence, with no other (non-argument) event verb appearing in the sub-sequence; and (b) one of the following holds:

S1: $V(e_i)$ is future simple tense, $V(e_j)$ is present simple;

S2: $V(e_i)$ is present simple tense, $V(e_j)$ is future simple;

S3: "must" or "should" or "may" or "might" is adjacent to $V(e_i)$, and $V(e_j)$ is present simple tense;

S4: "would" is adjacent to $V(e_i)$ and $V(e_i)$ is infinitive, $V(e_j)$ is past simple tense;

S5: "could" or "might" is adjacent to $V(e_i)$, and the tense of $V(e_j)$ is past simple;

S6: $V(e_i)$ is preceded by "could" and infinitive, and the tense of $V(e_j)$ is past continuous or past perfect;

S7: "would have", "might have" or "could have" is adjacent to $V(e_i)$ and $V(e_i)$ is a past participle, and $V(e_j)$ is past perfect tense;

S8: the tense of $V(e_i)$ is perfect conditional continuous, the tense of $V(e_j)$ is past perfect;

S9: the tense of $V(e_i)$ is perfect conditional, the tense of $V(e_j)$ is past perfect continuous;

S10: "would be" is adjacent to $V(e_i)$ and the form of $V(e_i)$ is gerund, $V(e_j)$ is past perfect;

The tenses of event verbs are determined by their POS tags. The updated event structures after detecting argument and conditional events are shown in Figure 4.1 Structured Events.

Table 4.1: 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.

	Precision	**Recall**	**EM-rate**
Fischbach *et al.* [2021]	0.75	0.79	0.41
Tan *et al.* [2022b]	0.80	**0.80**	0.1
Ours.	**0.93**	0.71	**0.85**

To evaluate the method, we annotated 100 randomly selected sentences that contain the signal words/phrases (20 for each), finding 75 of them have at least one condition-event pair. Our method correctly classifies 74 of the 100 sentences (53 true positive and 21 true negative cases). Furthermore, it correctly identifies all condition–consequence event pairs in 45 of the 53 true positive cases. Furthermore, we report the results of experiments with other systems for detecting conditional events.

The problem of detecting condition–consequence relations between events in texts has been studied, motivated in particular by finding causal relationships [Puente *et al.*, 2010]. We evaluated two recent methods that detect conditional structures, due to Fischbach *et al.* [2021] and Tan *et al.* [2022b], respectively. Both are BERT-based neural networks, but trained with different data. Fischbach *et al.* [2021] use an annotated set of requirements documents from Fischbach *et al.* [2020], while Tan *et al.* [2022b] annotated and used a set of news articles, together with the Penn Discourse Treebank 3.0 [Webber *et al.*, 2019] and CausalTimeBank [Mirza *et al.*, 2014] 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.

We apply these two systems to the same set of 100 randomly selected sentences from the movie plot summary dataset. 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. Three 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 4.1. Furthermore, in true positive cases identified by each, we compare the events identified as conditions and consequences with our annotation. These results are worse: Tan *et al.* [2022b] identifies the correct text spans in only 6 of the 60 cases (EM-rate = 0.1), while Fischbach *et al.* [2021] 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. (For example, the sentences “Go away or I’ll call the police!” and “Come back and I’ll call the police!” both express conditional using conjunction, while “I’ll call the police and come back” does not.) We contend that further investigation into this particular aspect of event relationships is merited.

If a conditional event is detected, we will make it separate from its consequence and regard them as two individual events for further action generation. As Figure 4.1 e_4 and e_5 in the updated event structures show, we update e_4 by removing e_5 from it.

4.3.2 Action Model Creation

An action model consists of four components: the action name, parameters, preconditions, and effects. We generate an action from each structured event, using the event verb as the action name, or the combined verb phrase if the event has argument events, as in the example in Figure 4.1. Each action has one or two parameters, x and o, representing the subject and (direct) object of the event, selected from the event argument based on the SRL argument labels (Section 4.3.2). We trained a commonsense event/concept relation predictor (COMET-BM) to generate candidate preconditions and effects from the event text (Section 4.3.2). Candidate preconditions and effects are filtered to remove semantic duplicates and contradictory phrases, to ensure they are consistent. We also use textual entailment to infer negated effects and preconditions (Section 4.3.2).

Precondition and Effect Prediction

Our event/concept relation predictor is trained on the three datasets ATOMIC [Sap *et al.*, 2019], ATOMIC-2020 [Hwang *et al.*, 2021] and ConceptNet [Speer *et al.*, 2019], introduced in Section 4.2.

We adopt COMET [Bosselut *et al.*, 2019], which is a GPT model [Radford *et al.*, 2018] that is pre-trained on the ATOMIC and the ConceptNet knowledge graphs, and build a BART [Lewis *et al.*, 2019]-based variation (named COMET-BM) by fine-tuning it on a subset of the ATOMIC-2020 dataset. Specifically, we select triples involving the relations *xNeed* (precondition for the subject, x, to undertake or complete the event), *xEffect* (effect on the subject, x, of the event), *oEffect* (effect on the object, o, of the event), *xReact* and *oReact* (reaction, i.e., emotional effect, on the subject, x, and object, o, respectively). Together, there are over 472K instances of these relations in the ATOMIC-2020 dataset. In the COMET-BM training procedure, we construct the tuple $< I, T >$ from the dataset. I denotes the concatenation of e and r, where e is the text describing the event (verb and argument spans) and r is the relation. T is the textual description denoting the commonsense knowledge inferred from I. Following Bhagavatula *et al.* [2019]'s work, we add two special tokens `<s>` and `</s>` to mark the beginning and the end for each I. The conditional probability of the nth token of T is defined as:

$$P(T_n|T_{[0,n-1]}) = Softmax(W * D(H_{T_{[0,n-1]}}, E(I)) + b),$$

where T_n and $T_{[0,n-1]}$ are the nth token and all preceding $(n-1)$ tokens in T; E and D are the encoder and decoder of the COMET-BM model (details refer to the BART model structure); $H_{T_{[0,n-1]}}$ is the decoded hidden states of all $n-1$ tokens; and W and b are learnable weight and bias parameters, respectively. During training, the objective

Table 4.2: The evaluation results of different metrics on the generated commonsense knowledge inferences using different fine-tuning models and the same beam search algorithm.

	ROUGE-L	**METEOR**	**BERT Score**
GPT2-M	0.41	0.30	0.57
GPT2-L	0.45	0.31	0.58
GPT2-XL	0.45	0.34	0.64
T5	0.48	0.36	0.64
BART (Ours.)	**0.50**	**0.37**	**0.68**

function for COMET-BM to minimize is the negative log-likelihood:

$$\mathcal{L} = -\sum_{n=1}^{|T|} \log P(T_n | T_{[0,n-1]})$$

Phrases that have the *xNeed* relation with the event become candidate preconditions, and the other candidate effects. Because action can have multiple preconditions and effects, COMET-BM works as a text generation model: it takes the event text e and relation r as input and outputs the candidate phrase T with an unfixed number of words. For each e, r, we generate up to K different outputs with the highest probabilities, from which we retain the ones with a normalized probability greater or equal to a threshold θ_r. Based on the probability distributions of each relation's predictions, we set $K = 6$, θ_{xNeed} to 0.7, $\theta_{xEffect}$ and $\theta_{oEffect}$ to 0.5, and θ_{xReact} and θ_{oReact} to 0.2. If any of the predicted phrases is "none", all lower-probability predictions are omitted. As an example, Figure 4.1 Predicted Preconditions & Effects shows the top predictions for each relation, given e_{1_2} as input.

For this precondition and effect prediction, we investigated not only BART but also various other publicly available and freely accessible generative large language models: GPT2-Medium, GPT2-Large, GPT2-Extra Large [Radford *et al.*, 2019], and T5 [Raffel *et al.*, 2020]), to fine-tune the COMET model. We assess their performance on the ATOMIC-2020 test set (a subset comprising exclusively the five selected relation tuples) by employing various well-established evaluation metrics pertinent to text generation: ROUGE-L [Lin, 2004], METEOR [Banerjee and Lavie, 2005] and BERT Score [Zhang *et al.*, 2019]. For the hyper-parameter tuning of each model, we set the candidate sets of the learning rate, batch size and number of epochs as [1e−5, 5e−5], [16, 32, 64], and [2, 3, 4], respectively. The results that achieve the best performance of each model are shown in Table 4.2. The BART model outperforms all the others, thus providing the rationale for our selection.

Precondition and Effect Selection

Because COMET-BM generates each candidate precondition and effect separately, when taken together, they are not necessarily semantically distinct or consistent. In Figure 4.1,

for example, the two predicted preconditions "know about his feelings" and "know about himself" for event e_{1_2} are semantically similar, and including both is redundant.

Hence, we filter COMET-BM outputs by deleting lower-probability preconditions that contradict or are similar to ones of higher probability, and likewise for effects. We pair the phrases output by COMET-BM for each of the (up to) six relations, and use two sentence-transformer [Reimers and Gurevych, 2019] models to predict if they are similar or contradictory. If yes, we eliminate the one with a lower probability.

The similarity predictor[2] is based on a large pre-trained model for natural language understanding and fine-tuned on over 1B phrase pairs. It computes the cosine similarity of 384-dimensional phrase embeddings. We judge two phrases to be redundant if their similarity is 0.5 or higher. The second model[3] is fine-tuned on over 4M sentence pairs from the SNLI [Bowman *et al.*, 2015] and MultiNLI [Williams *et al.*, 2018] datasets, and ranks whether the relation between two phrases (a, b) is most likely to be entailment (i.e., a implies b), neutral (a and b are logically independent) or contradiction. Both models have a reported test accuracy of over 90%.

We also use the textual contradiction classifier to generate negated action effects and preconditions. If a literal (p ?x ?o) is contradicted by a positive effect (resp. precondition), then (not (p ?x ?o)) is a candidate negative effect (resp. precondition). We have tried two generating strategies: in full negation (named *global*), we apply this test to all predicates defined in the domain; in restricted negation (named *local*), we generate only effects that are negations of literals that appear in the action's precondition, and no negated preconditions.

Parameter Selection

The commonsense relation predictor expects each event to have a subject, x, and optionally an object, o. Hence, each generated action has one or two corresponding parameters. These are selected from the event arguments, based on the SRL labels, following Algorithm 2. Only arguments with numbered labels ($ARG0$–$ARG5$), which represent the event's agent, patient, and so on, are considered; event modifiers detected by SRL can not instantiate parameters.

The action parameter ?x becomes an argument of each predicates obtained from the *xNeed*, *xEffect* and *xReact* relations; likewise ?o becomes an argument of predicates obtained from the *oEffect* and *oReact* relations. However, if the argument that instantiates the other parameter in the event appears, literally, in the predicate, it is also removed and replaced with the other parameter. For example, given the prediction "X gets X's car repaired" *xReact* "X doesn't like X's car", the generated effect will be (doesnt_like ?x ?o).

To evaluate the efficacy of our proposed event SRL argument name-based algorithm to extract the subject and object of the event as the parameters of the action, we randomly selected 120 events from the movie plot summary dataset. Among them, 65 events have no contained or

[2] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[3] https://huggingface.co/cross-encoder/nli-deberta-v3-base

Algorithm 2 Extracting Parameters from Event Arguments.

Input: e # event

procedure
  $x = \text{null}$ # subject
  $o = \text{null}$ # object
  $\mathbb{A}(e) = \{(a_1, lbl_1), \ldots, (a_k, lbl_k)\}$ # numbered arguments in e
  # Stage 1: Find subject
  if $(a, ARG0) \in \mathbb{A}(e)$ **then**
    $x = a$
  else if $(a, ARG1) \in \mathbb{A}(e)$ **then**
    $x = a$
  # Stage 2: Find object
  if $x \neq \text{null}$ **then**
    for $i \in [s+1, 5]$ **do**
      if $(a, ARGi) \in \mathbb{A}(e)$ **then**
        $o = a$
        break
  return x and o

argument events, and 55 have argument events detected. For all 120 events, we manually annotate the parameters for evaluation. For the 65 events, the accuracy is 46/65=70.8%, and the number is 65.5% for the 55 cases. Furthermore, as we observed, in most of the cases, the event's $\langle subject, object \rangle$ are as $\langle ARG0, ARG1 \rangle$, or $\langle ARG1, ARG2 \rangle$, or $\langle ARG1, none \rangle$, or $\langle ARG0, none \rangle$ (for instance, over 78.3% of all the events' parameters in the Movie plot summary dataset are determined as one of the 4 combinations by our algorithm). The precision on the 60 cases that are predicted as one of the 4 mentioned combinations is 41/60=68.3%. These findings suggest that our algorithm possesses potential for further refinement, as it faces more challenges in detecting parameters within the more intricate main-event-contain-argument-event structure.

4.4 Evaluation

To evaluate NaRuto, we want to determine how well the action models generated from a narrative text describe the intended meaning of the actions, and compare this with models obtained by previous systems that address the same task[4]. Let us start with why this is not easy. First, action verbs extracted from text often have multiple meanings; even the same action may have different conditions or effects depending on the narrative context. For example, in a fantasy story, a knight who slays a dragon will often become a hero – unless the story is told from the point-of-view of a den of dragons. Second, the same meaning can be expressed in different ways, using different predicates. For example, the condition "?x and ?y are in the same place" could be a binary predicate (in-same-place ?x ?y), or the two facts "(at ?x ?place)" and "(at ?y ?place)". Third, although quite a few approaches to action sequence extraction and model generation from text have been proposed, only a small number target narrative text, and have results available

[4]Evaluations of the performance of some individual components of the system can be found in the descriptions of the components in Section 4.3.

for comparison.

For input, we use two short stories that have appeared in work on narrative planning: Riedl and Young [2010]'s Aladdin story, and Ware [2014]'s Old American West story. Both are hand-written textual descriptions of plans generated by narrative planning systems. The advantages of using these texts are that there thus exists a ground truth, in the form of the planning domains used by the respective narrative planners, and that these two stories have also been used for evaluation in previous work on action model learning from narrative text [Hayton *et al.*, 2017; Huo *et al.*, 2020], so that some results are available for comparison. A downside is that these texts are somewhat simpler than what we intend NaRuto to target. For example, neither story mentions any conditional events.

We compare the action domain models generated by NaRuto to those generated by Hayton et al's 2017 StoryFramer system [Hayton *et al.*, 2017], and their 2020 system [Hayton *et al.*, 2020] (abbreviated "H2020"). Neither system is available for use, but the domains produced by StoryFramer for the two example stories were kindly provided by the authors, and we have attempted to replicate the results of H2020 applied to the same stories using the StoryFramer material and the description in their paper. The details are described in Section 4.4.2 below. Huo *et al.* [2020] provide some results for Ware's Old American West story, but not a complete domain model; we compare the aspects of NaRuto that we can with their data.

We focus our evaluation on the set of actions modeled. We do not try to align the predicates defined in each of the different domain models generated from the same text. As noted, models can describe the same actions with different sets of predicates, or equivalent predicates that have different names. Furthermore, the narrative text usually focuses on what happens, i.e., the events, and only infrequently mentions what is, i.e., facts. Thus, we find higher agreement between different approaches in the set of actions. We compare the sets of action names extracted from each story with those in the ground truth domain, using manual alignment of names. The same comparison was also done by Hayton *et al.* [2017] and Huo *et al.* [2020]. The results are presented in Section 4.4.3 below. Furthermore, we conducted a (blind) expert assessment of the appropriateness of each action's preconditions and effects, relative to the predicates defined in the domain. The details of this evaluation process and its results are described in Section 4.4.4 below.

Third, we evaluate the generality of each domain model, by applying a planner to problem instances for each domain with varied initial states and goals and recording the number and variety of plans generated The detailed procedure can be found in Section 4.4.5. In Section 4.4.6 and 4.4.7, we further evaluate the action model generation and generalization ability of NaRuto by applying it to two large narrative text datasets. And we prove the usefulness of the generated action models by improving the performance of the LLM's in the story generation task.

4.4.1 Implementation Settings

NaRuto was run on a computer with 64 CPUs with 126GB of RAM and one RTX-3090 GPU with 24GB of RAM. We use the stanford-corenlp-4.2.0 version toolkit [Manning *et al.*, 2014] for dependency parsing and POS tagging. We train our COMET-BM model on this single GPU, which takes around 4 hours. We set the optimization method as AdamW [Loshchilov and Hutter, 2017] with an initial learning rate of 1e−5, a batch size of 32, and 3 epochs during the training (fine-tuning) process.

4.4.2 Comparison Systems

StoryFramer [Hayton *et al.*, 2017] is a partially automated domain modeling tool. Given a narrative source text, it automatically extracts candidate action verbs, candidate predicates based on properties, and candidate objects (nouns). The remaining modeling task is left to the user, who must assign types to objects and action parameters, identify duplicates, and, crucially, select which predicates to use as preconditions and effects for each action. The system adds some default predicates and preconditions, such as automatic inequality constraints for all action parameters of the same type. The user can also override or edit any system suggestion (e.g., add or remove candidate predicates, objects, etc.). Hayton *et al.* [2017] applied StoryFramer to the Aladdin and Old American West stories. While we cannot replicate their process, both because the system is not available for use, and because we do not know what edits the user(s) in that study made, the resulting domain files were provided by the authors.

Their recent system [Hayton *et al.*, 2020] ("H2020") is automatic. It is also not available for us to use, but since its event extraction mechanism is very similar to that of StoryFramer, apart from using a novel co-reference resolution method, we approximate its result by supposing it would extract the same action signatures (names and parameters) as StoryFramer, and constructing the corresponding action models following the method described in their paper. The action models created by H2020 aim to replicate the input narrative sequence.

Huo *et al.* [2020] propose a partially automated system to learn a planning domain model, applied to natural disaster contingency plans. They use POS tagging to extract (action name, subject, object) triples, representing actions taking place. Their system also involves a human user to refine the action model. Because neither the system nor its outputs are available to us, we can only compare the results included in their paper.

Ware [2014]'s Old American West text consists in fact of several story variants; since H2020 depends on the order of events in the source text, we apply it to one of them (plan G, the longest). The other systems use the concatenation of all story variants as inputs.

4.4.3 Identification of Narrative Actions

Table 4.3 lists the action names extracted by StoryFramer [Hayton *et al.*, 2017], Huo *et al.* [2020]'s system and by NaRuto, and compares them with actions defined in the ground truth, i.e., the hand-written narrative planning domains from Ware [2014]'s thesis (West story) and from [Riedl and Young, 2010] (Aladdin). Results for StoryFramer and Huo *et al.* [2020]'s systems are from the respective papers. Note that Huo et al. did not try their system on the Aladdin story.

There is not a perfect match between action names defined in the ground truth domains and those extracted from the narrative texts because the texts use different words to describe them; for example, the action give in the Aladdin story is described as "Aladdin hands the magic lamp to King Jafar", so the automatically extracted action name is hand. For Ware's Old American West story, the events "use" and "anger" are not actions in the ground truth planning domain, but are implicitly represented in the effects of other actions, e.g., the action heal requires the character who performs it to have medicine, which is used up as part of the action's effects. These events occur in descriptions of those actions, in the story sentences "Carl the shopkeeper healed Timmy using his medicine" and "Hank stole antivenom from the shop, which angered sheriff William". Our detection of argument events is seen in, for example, the actions "get bitten" or "intend to shoot", where StoryFramer only extracts "intended" from the sentence "Sheriff William intended to shoot Hank for his crime" and Huo et al.'s system only extracts

Table 4.3: Action names extracted from the two input stories by StoryFramer [Hayton *et al.*, 2017], Huo et al.'s system [Huo *et al.*, 2020], and NaRuto. The first column (GT) lists the corresponding action names in the ground truth, i.e., the hand-written narrative planning domain files by Ware [2014] (Old American West story) and Riedl and Young [2010] (the Tale of Aladdin). ✗ means the action is not detected; ✓ means the action is partially extracted or incomplete; ∗ means the action is not present in the ground truth planning domain but occurs in the story text.

GT [**Ware, 2014**]/[**Riedl and Young, 2010**]	**StoryFramer** [**Hayton *et al.*, 2017**]	**Huo et al.** [**Huo *et al.*, 2020**]	**NaRuto** [**Ours**]
Domain 1: An Old American West Story			
die	✓died	✓died	✓die
heal	✓healed	✓healed	✓heal
shoot	✓shot	✓shot	✓shoot
steal	✓stole	✓stole	✓steal
snakebite	✓bitten	✓ got	✓get bitten
	✓ intended	✓intended to heal	✓intend to heal
		✓intended to shoot	✓intend to shoot
		∗ using	∗ use
		∗ angered	∗ anger
Domain 2: The Tale of Aladdin			
travel	✓travels		✓travel
slay	✓slays		✓slay
pillage	✓takes		✓take
give	✓gives		✓hand
summon	✗		✓summon
love-spell	✓casts		✓cast
fall-in-love	✗		✓fall-in
marry	✓wed, married		✓wed
	∗ confined		∗ be-confined
	∗ rubs		∗ rub
	∗ sees		∗ see
			∗ make
			∗ be-not-confined

Table 4.4: The average scores over all the respondents for all actions' preconditions and effects within each domain model (**Sco.**); and the average percentage in agreement(**Agg.**) and disagreement (**Disagg.**) scores. **Type** indicates if the domain model is generated manually or by a semi-automated or (fully) automated system. **Bold numbers** indicates the best results and underline denotes the second-best results.

Method	**Type**	**Sco.** ↑	**Agg.** ↑	**Disagg.** ↓
Domain 1: An Old American West Story				
GT [Ware, 2014]	Manual	4.34	82.7%	10.7%
StoryFramer [Hayton *et al.*, 2017]	Semi-auto	3.26	42.5%	26.3%
H2020 [Hayton *et al.*, 2020]	Auto	2.54	17.7%	41.2%
NaRuto$_{(G)}$	Auto	2.98	43.0%	39.3%
NaRuto$_{(L)}$	Auto	**3.57**	**60.8%**	**25.3%**
Domain 2: The Tale of Aladdin				
GT [Riedl and Young, 2010]	Manual	4.84	97.3%	1.2%
StoryFramer [Hayton *et al.*, 2017]	Semi-auto	**4.04**	**74.8%**	**17.3%**
H2020 [Hayton *et al.*, 2020]	Auto	2.81	38.2%	48.8%
NaRuto$_{(G)}$	Auto	3.03	41.0%	38.3%
NaRuto$_{(L)}$	Auto	3.34	52.0%	29.5%

"got" from "Hank got bitten by a snake". Moreover, StoryFramer misses the actions "summon" and "fall-in-love" in the Aladdin story, which NaRuto finds.

4.4.4 Expert Assessment of Action Models

As discussed above, it is difficult to align predicates between the ground truth and generated domain models; thus, one cannot say that generated action models have "correct" preconditions and effects by a simple comparison with ground truth. Instead, to evaluate the quality of our generated action models, we asked experts to rate the *appropriateness* of each action's preconditions and effects, relative to the predicates that are defined in the respective domain model. We applied this evaluation to all four models of both domains, i.e., the ground truth domain model as well as the domain models generated by StoryFramer, H2020 and NaRuto, as described in Section 4.4.2 above.

We recruited 9 participants, all of whom are experts in AI planning, and familiar with modeling planning domains in PDDL. Each participant was given the four different models of one, or in a few cases both, of the domains, and asked to rate the appropriateness of each precondition and each effect of each action in all four domain versions, using a 5-point Likert scale: 1=not appropriate; 2=probably/maybe not appropriate; 3=undecided; 4=probably/maybe appropriate; 5=appropriate. The models were formatted to appear as similar as possible (e.g., comments were removed from the ground truth domain models, indentation was made uniform, etc). Participants were told only that all four domain models were "automatically learned from narrative text", and the order of presentation of the four models was randomized for each participant. We finally received N=6 responses for each of the two domains (stories).

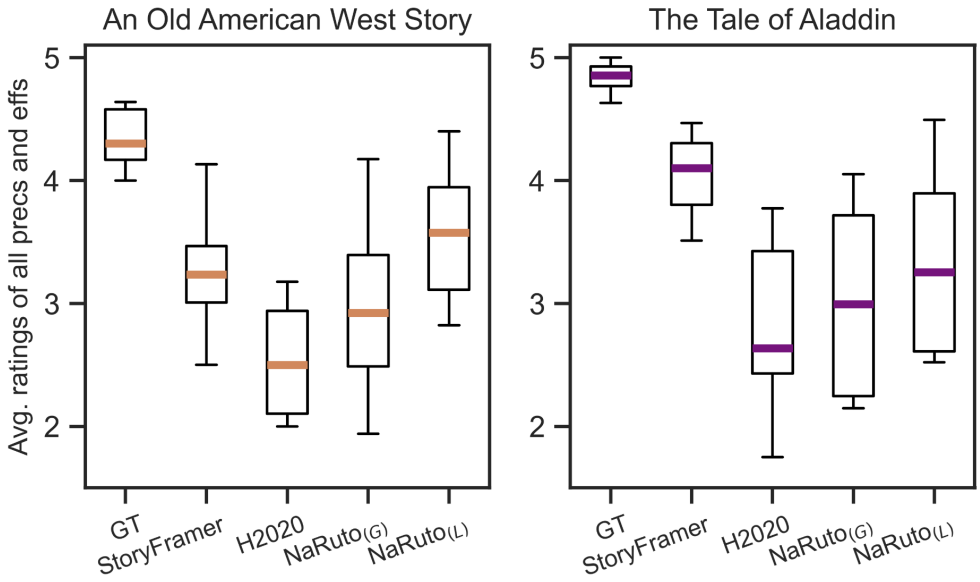


Figure 4.2: Distribution, over respondents, of the average scores for all actions' preconditions and effects within each domain model. The thick line shows the median, box shows the interquartile range. Whiskers extend to the full range of values.

From each response, we compute three summary measures: (1) the average ratings of all preconditions and effects within each domain model; (2) the percentage of ratings that are classified as in agreement (i.e., 4 or 5); and (3) the percentage of ratings that are in disagreement (i.e., 1 or 2), within each domain model. For each model variant, all measures are averaged over the N=6 responses for each domain. The domain model generated by NaRuto in the evaluation is with full (global) negations, called NaRuto$_{(G)}$. We calculate measures for the version with restricted (local) negations, called NaRuto$_{(L)}$, by omitting the negated preconditions and effects that would not be present in this model.

Results are summarized in Table 4.4. Box-and-whiskers plots showing the distribution of average scores across responses for both of the domains are in Figure 4.2.

Unsurprisingly, the hand-written ground truth domain models receive the highest average and percentage-in-agreement scores, as well as the lowest percentage-in-disagreement scores. The StoryFramer domain models also score well on both measures. Again, this is not surprising, since the selection of each action's preconditions and effects in these domain models was done manually (and presumably by a user with knowledge of the story they intend to model). However, using NaRuto$_{(L)}$, our generated model of the Old American West domain is rated better than the StoryFramer model and our model of the Aladdin domain is rated second-best. This indicates that the precondition and effects predictor captures well the commonsense knowledge of actions in these domains. The global negations strategy (NaRuto$_{(G)}$) is intended to capture the ramifications of positive action effects. However, the domain model with restricted negations (NaRuto$_{(L)}$) scores consistently better, indicating that most of the additional negated effects, and preconditions, in the global model are not helpful. The domain models generated by H2020 score lower on both measures, indicating that the somewhat particular structure it encodes,

which captures the sequence of events in the original story, is not perceived by experienced domain modelers as appropriate for a planning domain, which is normally expected to allow for the composition of all valid action sequences.

We also note that in all generated models, the average rating of actions' preconditions is consistently higher than that of their effects, sometimes significantly so (3.61% to 74.80%). This suggests generating appropriate effects is a harder problem.

4.4.5 Alternative Plan Generation

For each generated domain model, the sequence of events, extracted from the story text, that it is based on is a valid plan, provided a suitable initial state and goal. To evaluate the extent to which each domain model supports the generation of other story plans, we created new problem instances with modified initial state and goal, and applied the Fast Downward [Helmert, 2006] system to generate plans.

For each domain model, we extract the minimal set of initial facts that are required for the original action sequence to be executable as the base initial state, and the set of facts made true at the end of its execution, but not initially true, as the candidate goal facts. We exclude from the goal any fact that can be achieved by only one action instance, so as not to over-constrain the problem. We create modified instances by removing one fact from the initial state, with a fixed goal.

We applied this process to the domain models generated by NaRuto and StoryFramer. We did not apply it to the H2020 domain models, because due to the way these are constructed, the modified instances will always either be unsolvable, or admit a trivial variant of the input plan that only substitutes a different object for one parameter of one action.

Table 4.5 shows the number of resulting problems solved, and the number of distinct plans generated. Both models generated by NaRuto show better generalizations than those by StoryFramer. Notably, in the Aladdin domain, both NaRuto models have a 100% solving rate and generate 8 unique plans, while the StoryFramer model becomes unsolvable when any fact is removed from the initial state required by the original story plan.

The planner configuration used in this experiment finds only one, arbitrary, plan per task. Hence, the number of distinct plans found may be lower than what is possible. Using a different configuration, that explicitly searches for diverse solutions, we found there exist at least two distinct plans for each solvable instance in the NaRuto models of the West domain.

4.4.6 Narrative Planning Model Generation

Furthermore, we apply NaRuto to two large sets of narrative texts: a set of movie plot summaries, due to Bamman *et al.* [2013], and news articles from the Goodnews dataset [Biten *et al.*, 2019]. Summary statistics of the two datasets are shown in Table 4.6.

Using a computer with 64 CPUs with 126GB of RAM and one RTX-3090 GPU with 24GB of RAM, applying the entire pipeline to a single document takes an average of 1.6 minutes. A summary of resulting events and action models is shown in Table 4.7. We find over 12.8 million (non-statement) events: 11.9% of them contain argument events, and 1.9% are conditions. Because the input to precondition/event prediction is the event text, not only the action name, we can generate different actions with the same name (i.e., verb). This, to some extent, reflects that verbs often have different usages. Furthermore, we remove generated actions whose preconditions

Table 4.5: For each domain and system, the total number of tasks and how many of them are solvable, and the number of unique solvable plans.

	Method	Total Tasks	Solvable Tasks	Distinct Plans
Aladdin	NaRuto$_{(G)}$	17	17 (**100%**)	**8**
	NaRuto$_{(L)}$	17	17 (**100%**)	**8**
	StoryFramer	30	1 (3.3%)	1
West	NaRuto$_{(G)}$	12	4 (**33.33%**)	1
	NaRuto$_{(L)}$	11	2 (16.66%)	**2**
	StoryFramer	9	2 (22.22%)	1

Table 4.6: Statistics of the two corpora that we use: number of documents, and average number of sentences and words per document.

	Movie-plot summaries	Goodnews
# documents	42,177	251,543
avg # of sentences per doc	15.9	21.4
avg # of words per doc	312.3	452.8

and effects are subsets of those of another action with the same name. Our action models could be further reduced to a smaller number of more general actions by clustering based on action (textual) similarity – this could be our future work. The net result is 10.8 million actions, with over 817K distinct action names, and 1.3 million distinct predicates. On average, each action mentions 10.3 predicates.

Table 4.7: Statistics of the events (left) and actions (right) generated from our pipeline. Details are in the corresponding section.

Event extraction		**Action model generation**	
# of events	12,817,958	# of actions	10,829,651
avg # of events per doc	60.6	# of action names	817,560
% with argument events	11.9%	avg # of preconditions and effects	10.3
% conditions	1.9%	% action linked	72.1%

For action models to be usable for narrative planning, they must be able to generate action sequences that are causally connected, i.e., such that each action's preconditions are achieved, at least in part, by an earlier action, or its effects contribute to enabling some later action. 72.1% of the generated actions are linked in this way. The linkage is somewhat asymmetric: 27.9% of these actions have an effect that matches some other action's precondition.

To estimate the variety of narratives that can be generated from the model, we randomly sample one thousand actions that have at least one following connected action as starting points, and count how many causally connected sequences of different lengths can be generated from them. 334, 299 and 295 of them have at least one action sequence with a length ≥ 3, 4 and 5, respectively. We further obtain a distribution of the number of causally connected sequences for each length,

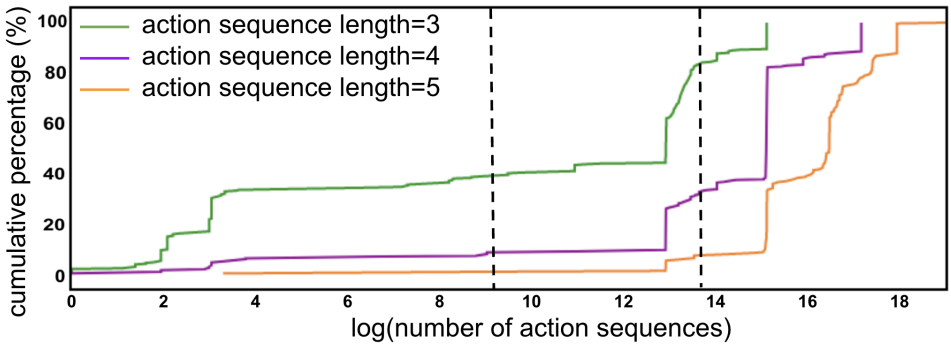


Figure 4.3: Cumulative distribution of the natural logarithm of the number of possible action sequences for different sequence lengths. The two vertical dashed lines mark 10K and 1M.

shown in Figure 4.3. We can observe that the number of possible sequences increases with length. As shown by the vertical dashed lines, over 61% of the 334 actions can start more than 10K distinct sequences of length at least 3. Around 66% of the 299 and 92% of the 295 actions can start more than 1 million sequences with lengths 4 and 5. For example, for the action "find" with two parameters, "ARG0" and "ARG1", there are 911,451 possible action sequences of length 3, starting with this action. Such sequences include "find"–"become"–"kill", "find"–"use"–"do", and so on. These results indicate that our generated action models are adequate for planners to generate variable narrative plans.

4.4.7 Guided Story Generation

Large language models (LLMs) such as GPT given short input prompts (e.g., the first sentences) are a popular means of generating stories. But their stories often lack coherence, and may gradually go off the prompt and fall into repeating the same phrase or sentence indefinitely, until reaching the maximum length [Castricato *et al.*, 2021; Simon and Muise, 2022]. An example of this can be seen in Output II in Table 4.8. We term this degenerative repetition.

Guiding the LLM with a sequence of causally connected narrative actions, instead of only an initial prompt, can make it avoid degenerative repetition and output stories that follow a plot. To test this, we randomly sample nine sequences beginning with an action with no precondition from the generated action sequences (three of each of the lengths 3, 4, and 5) as our storylines, and use the GPT-J-6B text generation model [Wang and Komatsuzaki, 2021] to create corresponding stories. The parameters of the action models are instantiated with made-up names (e.g., in Table 4.8, ?x is "Jack", and ?o is "Daniel"). The prompt given to the GPT model consists of one example mapping the action structures (not using actions that feature in any of the storylines) to simple text, and the action sequence (so-called "one-shot learning"). We set the hyperparameters of the GPT-J-6B as: temperature=0.9 and max_length=512. For comparison, we also generate stories using only the first action of each storyline as the prompt. Examples of the two types of input and the corresponding outputs are shown in Table 4.8.

We generate 100 outputs for each input. To evaluate the quality of the outputs, we first count the incidence of degenerate repetition, as shown in Figure 4.4. Using the first action only as the prompt (type II input), an average of 33.1% of the stories degenerate into repetition, while only

Table 4.8: An example of the LLM generated stories given the action sequence (**Input I**) and the start action only (**Input II**) as inputs. Bold terms in the **Output I** denote actions that correspond to the **Input I**. We observe that **Output II** contains repetition.

Input I {action: Jack meet Daniel. effect: Jack meet new people, Jack feel happy. Daniel get drunk and have fun. Daniel feel happy.} {action: Daniel attack Jack. effect: Jack get attacked. Jack feel scared. Daniel get attacked back and get bitten. Daniel feel scared.} {action: Jack torture Daniel. effect: Jack get arrested. Jack feel angry and bad. Daniel get tortured. Daniel feel sad and betrayed.}
Output I Jack and Daniel had just **met** each other at a party. Jack was excited to meet new people while Daniel **got** drunk and **had** fun. They both felt **happy** in the moment. However, Daniel suddenly **attacked** Jack out of nowhere. Jack was taken by surprise and felt **scared** as he tried to defend himself from the attack. In retaliation, Daniel got **attacked** back and even got **bitten** in the process which made him feel **scared** too. Jack decided to **torture** Daniel instead of calling for help. This act resulted in Jack getting **arrested** while making Daniel feel **sad** and **betrayed** due to all that had happened between them.
Input II Jack meet Daniel. Jack meet new people, Jack feel happy. Daniel get drunk and have fun. Daniel feel happy.
Output II Jack meet Daniel. Jack meet new people, Jack feel happy. Daniel get drunk and have fun. Daniel feel happy. Jack feel sad. Jack feel happy. Jack feel sad. Jack feel happy. Jack feel sad. Jack feel...

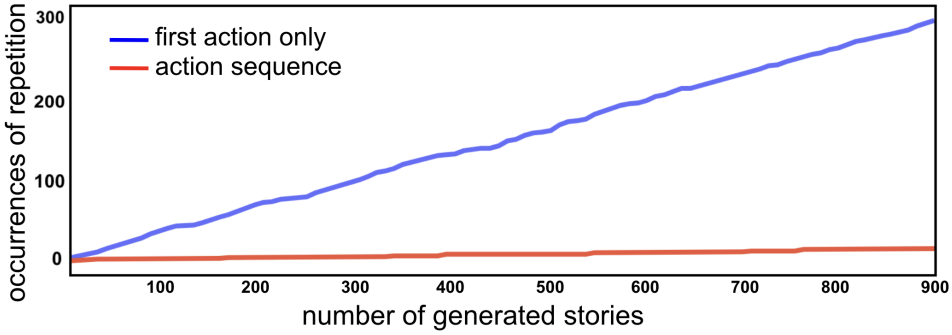


Figure 4.4: Given the two types of inputs, X-axis: number of stories generated, Y-axis: number that contains degenerate repetition.

1.7% of the stories generated from an action sequence (type I input) do. This shows that with the guidance of our action sequences, the story generation process is more controllable.

Second, to assess how well the generated stories adhere to a plot, we pair samples of non-repetitive stories generated from the action sequence and the first action only, respectively, and ask workers via Amazon Mechanical Turk to select which of the two stories better matches a given title, derived from the instantiated first action name (e.g., for the example in Table 4.8, the title is "The story of Jack meet Daniel"). We randomly select one action sequence for each of the lengths 3, 4 and 5 (each sequence has a different first action), and for each of those 10 samples of non-repetitive outputs from the LLM, resulting in 300 paired stories. Each pair is

Table 4.9: Number of the 300 paired stories given each voting ratio (action sequence-guided **:** first action only prompt). For example, **9:0** means all nine workers thought the action sequence-guided story better matched the given title. In 246 (82%) of the 300 pairs, the majority prefer the action-sequence-guided story. In 76.4% of the 246 cases, the majority vote comprises more than two-thirds of the nine answers.

Ratio	**9:0**	**8:1**	**7:2**	**6:3**	**5:4**	**4:5**	**3:6**	**2:7**	**1:8**	**0:9**
Votes	8	36	60	84	58	31	17	4	2	0
Percent	82%					18%				

judged by nine workers. The resulting distribution of votes is shown in Table 4.9, and clearly shows that action sequence-guided generation is much more likely to produce stories that match readers' expectations of a story, given the prompt.

4.5 Conclusion

Narrative texts exhibit many complexities, but it is also a rich source of knowledge about events and actions. We presented the NaRuto system for creating planning-language-style action models from narrative texts. In contrast to previous approaches, our system does not depend on manual input, and creates action models that generalize beyond the event sequence in the source text. Ultimately, this will enable the generation of planning models at a scale to support open-world, creative narrative planning.

In evaluation, our generated action models are rated better than those by comparable fully automated methods, and sometimes even than those by semi-automated systems using manual input or correction. Moreover, using the learned models, we can generate many millions of original narrative action sequences, and use such sequences as inputs to LLMs resulting in stories that are less likely to veer off-topic or degenerate into repetition than using only an initial prompt.

Yet, there are several areas for directing our future work: The commonsense relation predictor considers only two event arguments – subject and object – which currently limits action parameters to two. Our evaluation also revealed that selecting appropriate negative action effects, and taking story context into account when predicting precondition/effects, remain challenging tasks. Specifically, with the rapid development of various LLMs in recent years, these models are being pretrained on increasingly larger datasets and with more parameters. It motivates our future work to explore the possibility of applying advanced LLMs to some components of our pipeline, as other researchers have done with tasks such as semantic role labeling [Das *et al.*, 2023; Cheng *et al.*, 2024] and plan generation [Wang *et al.*, 2024; Lin *et al.*, 2024].

Chapter 5

Learning Event Relations for Action Model Representation

The preceding chapter outlined the various factors of NaRuto, our system for automatic action model generation. When extracting action models, it's crucial to correctly identify and represent argument events to ensure the models faithfully reflect the narrative texts. Broadly speaking, determining the potential interrelationships between events is vital for both information retrieval (IR) and natural language processing (NLP). Such events provide AI agents with a deeper understanding of the world by indicating occurrences and the entities involved. Currently, event relation extraction work focuses on hierarchical, temporal, and causal relations. While these relations treat events as syntactically and semantically independent, they often overlook their inter-relatedness. To address this shortcoming, this chapter introduces a human-annotated Event Dependency Relation dataset (EDeR) based on a selection from the OntoNotes dataset. This selection harmonizes with the existing annotations of OntoNotes. We explore foundational approaches to predict event dependency relations within EDeR and demonstrate their significance in advancing key NLP tasks, such as semantic role labeling and co-reference resolution.

The chapter is organized as follows: We first investigate the methodology behind the dataset's construction and offer a thorough analysis in Section 5.3. This leads to various baseline model formulations for event dependency relation classification, discussed in Section 5.4. We further illustrate the benefit of event dependency relations on crucial NLP tasks in Section 5.5. Finally, we highlight intriguing challenges presented by the EDeR dataset, paving the way for future AI research, in Section 5.6.

5.1 Introduction

Events play a critical role in enabling AI agents to understand and perceive the world, as they provide information on what happened and the entities involved. An event is composed of a predicate (i.e., verb) and arguments, where the predicate indicates the event's action and the arguments represent the subject, object and so on of the predicate [Levin, 1999; Rappaport Hovav *et al.*, 2010].

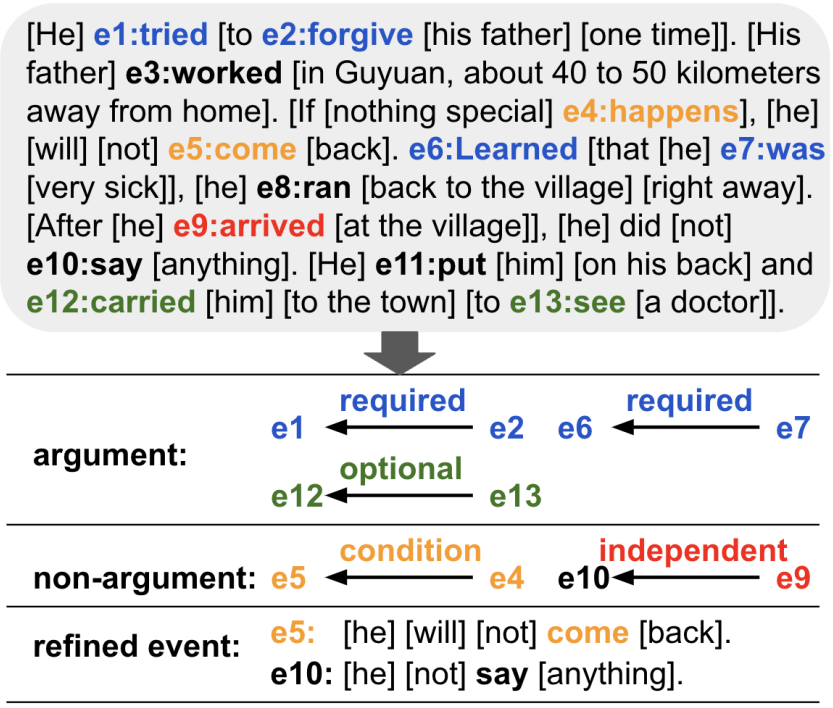


Figure 5.1: Examples of the event dependency relations. Above: source text. (Event predicates are marked in boldface and argument spans are marked with brackets.) Below: event dependency relations and refined events.

Previous work on relation extraction mainly focuses on investigating the relationships between entities [Miwa and Sasaki, 2014; Huguet Cabot and Navigli, 2021; Chen *et al.*, 2020; Ma *et al.*, 2022] rather than events. In the relatively limited research work that studies the relations between events, the types of relations considered are causal [Mariko *et al.*, 2020, 2022; Tan *et al.*, 2022a], temporal [Bethard, 2013; Laokulrat *et al.*, 2013], and hierarchical [Hovy *et al.*, 2013; Glavaš and Šnajder, 2014]. Such relationships consider two events to be independent in terms of syntax and semantics but overlook the interdependency between them. This disregard for interdependence can potentially yield incomplete or erroneous narrative interpretations. Take the sentence "He tried to forgive his father one time." from Figure 5.1, $e2$ with the predicate "forgive" is the action that he "tried" ($e1$) to take. However, there is no temporal or causal or hierarchical relation between $e2$ and $e1$; rather, $e2$ is the object (syntactically depending) and patient (semantically depending) of $e1$. In fact, $e2$ didn't actually happen. Recognizing such relations, that reveal both semantic and syntactic dependencies, is an essential yet challenging task for AI researchers. Correctly identifying these dependency relations can enhance language understanding and provide valuable information for various related NLP tasks.

Motivated by these, we investigate how an event may be an **argument** of another as representing the event dependency relations, rather than being regarded as **independent**. Additionally, we distinguish arguments into **required argument** and **optional argument** events. Required arguments, like $e2$ ("forgive") in $e1$ ("tried") from Figure 5.1, must be present for an event to be complete and meaningful (to "try to", one has to try to do something). Optional arguments,

such as $e13$ ("see") in $e12$ ("carried"), enhance or clarify the event that it is an argument of (for example, indicating the purpose), and their occurrence isn't explicitly stated. Non-argument events, which are conjunctive or conditional, are often mislabeled as arguments in the event extraction task, like $e9$ ("arrived"), which is a temporal modifier for $e10$ ("did not [say] anything") but is independent.

Argument-event dependency relations are frequent in natural language texts, especially in narrative texts such as news articles, conversations, and similar. Hence, we present a human-annotated **E**vent **De**pendency **R**elation dataset (EDeR) that (1) extracts event dependency information based on a sample of 275 documents spanning seven genres from OntoNotes [Pradhan *et al.*, 2013] and integrates with orthogonal annotations of this dataset, and (2) provides refined semantic-role-labeled event representations based on this information. We build the EDeR dataset with 11,852 high-quality annotations, according to the proposed event relation taxonomy. It can serve as the basis for an effective automatic classification process.

We applied both EDeR human-annotated and model-predicted event dependency relations to enhance a state-of-the-art semantic role labeling (SRL) model's [Zhang *et al.*, 2022] event representation generation. Experimental results prove these relations improve event extraction, yielding updated representations from EDeR. These refined event representations were further applied to the co-reference resolution (CR) task. A performance comparison taking the original and our updated event representations as inputs to a CR model [Jiang and Cohn, 2021] affirms the effectiveness and validity of EDeR's refined representations. We release EDeR (e.g., dataset, baseline code, and model) at `https://github.com/RichieLee93/EDeR`.

5.2 Background and Related Work

Relation extraction is a popular branch of information retrieval research. Most recent works, as a downstream task of Named Entity Recognition (NER), concentrate on extracting relations between entities rather than events [Miwa and Sasaki, 2014; Huguet Cabot and Navigli, 2021; Chen *et al.*, 2020; Ma *et al.*, 2022].

As described, events are represented in word span style and such event representations can be regarded as text clips. Recently, there is remarkable progress in text relation classification tasks (e.g., textual entailment detection, sentiment analysis.) using Transformer [Vaswani *et al.*, 2017]-based language models like BERT, RoBERTa, GPT models [Balazs *et al.*, 2017; Ethayarajh, 2019; Laskar *et al.*, 2020; Lee *et al.*, 2022]. Such language models are pre-trained on a tremendous corpus, people further fine-tune it for their specific tasks using their selected datasets (so-called "transfer learning" [Pan and Yang, 2010]). This pretrain-finetune process usually achieves more advanced performance on relation extraction tasks, especially when the data size is small, compared with those solely trained on classical machine learning models or neural networks [Balazs *et al.*, 2017; Papanikolaou *et al.*, 2019; Seganti *et al.*, 2021]. Learning from these, we can also adapt these state-of-the-art language models to classify the dependency relations between event pairs. Furthermore, based on different styles of event representations (i.e., semantic role-based [Carreras and Màrquez, 2005; Hajič *et al.*, 2009; Pradhan *et al.*, 2012, 2013] or predicate-based [Pustejovsky *et al.*, 2003; Cassidy *et al.*, 2014; Ning *et al.*, 2018b; Mirza and Tonelli, 2016]) and other features such as the syntactic dependency relation between predicates and so on, we can combine various inputs with the language models to explore the baseline performance on our EDeR dataset.

5.3 Dataset

We utilize a subset of OntoNotes documents with its human-annotated and predicate-argument formatted events to extract candidate event pairs, where one event's predicate is contained in the span of another. These pairs, denoted as *Event*1 and *Event*2, are labeled by human annotators to indicate whether *Event*2 is a required argument of, an optional argument of, a condition of, or independent from *Event*1. We next detail the way we collect and pre-process the candidate event pairs, the human annotation process and the construction of the dataset.

5.3.1 Data Collection

OntoNotes contains semantic role-formatted event representations, as the OntoNotes example in Figure 5.2 shows. We randomly sampled 275 documents from seven genres: broadcast news (bn), magazine (mz), newswire (nw), pivot corpus (pt), telephone conversation (tc), broadcast conversation (bc), and web data (wb). Data statistics for the 275 raw documents are shown in Table 5.1. The number of sampled documents and the separation of them into training, development and test sets under each genre follow their initial distributions in the OntoNotes dataset.

Table 5.1: Statistics of the documents (under each genre and all) that we sampled from the OntoNotes dataset for annotation: number of documents and the number of documents split into different sets (train, development and test), and average number of sentences, words and events per document.

	bn	**mz**	**nw**	**pt**	**tc**	**bc**	**wb**	**overall**
# documents	104	7	107	31	7	3	16	275
# documents-train	90	5	89	24	5	1	13	227
# documents-dev	7	1	9	4	1	1	2	25
# documents-test	7	1	9	3	1	1	1	23
avg # sentences per doc	6.2	48.9	14.4	40.7	71.7	203.8	59.3	24.9
avg # words per doc	139.7	1599.7	418.7	657.0	1089.4	4045.8	1509.3	543.4
avg # events per doc	23.6	202.1	58.3	148.7	362.1	1058.6	268.5	93.5

Candidate Event Pair Extraction

Because arguments are spans of text, part or all of an extracted event may lie within the argument of another event. If the verb (i.e., predicate) of 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 argument events, but are not necessarily so. For example, in the top part of Figure 5.2, the event in green (**e3** = {ARG0: The man } {R-ARG0: who } {V: works } {ARGM-LOC: here }) is contained in the span of the event in blue (**e1** = {ARGO: The man who works here } {V: tells } {ARG2: me } {ARG1: to get the hell out }), but **e3** is not an argument of **e1**; however, the event in orange (**e2** = {ARG1: me } {V: get } {ARG2: the hell out }) is an argument of **e1**.

We select event pairs (e_i, e_j) where e_j's verb is contained within the span of an argument of e_i as candidate event pairs for annotation.

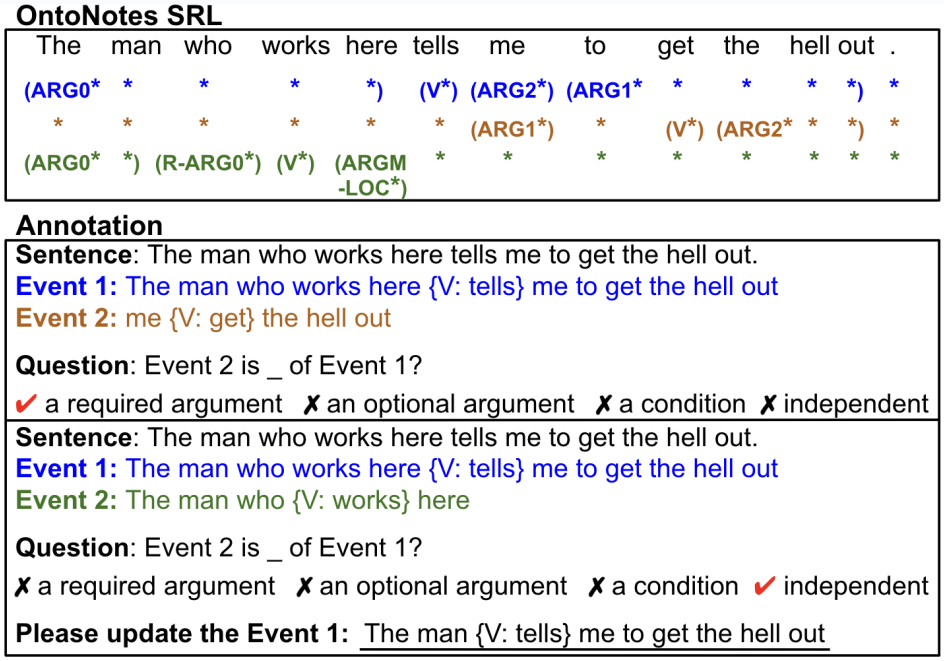


Figure 5.2: Above: a sample sentence with semantic role labels from the OntoNotes dataset. Below: corresponding event pairs presented for human annotation with event dependency relations.

Preprocessing

We apply three filters to the selected candidate event pairs. First, we filter out candidate event pairs (e_i, e_j) in which e_j does not have any arguments. This typically occurs when e_j is a modal verb, indicating the tense of the verb in e_i. Second, events in the OntoNotes dataset sometimes mistake adjectives for verbs (e.g., "reloaded" in "the Matrix reloaded"), so we use the words' POS tags to filter these out. The POS tags of the event predicates are obtained using the Stanford CoreNLP toolkit [Manning *et al.*, 2014]. Third, we remove transitively contained event pairs, i.e., pairs (e_i, e_j) such that there exists another event e_k such that the verb of e_k is contained in e_i and the verb of e_j is contained in e_k. In such cases, (e_i, e_k) and (e_k, e_j) may be candidate pairs, but (e_i, e_j) is not. For example, in "[She] {V: try } [to {V: stop } [[them] from {V: ruining } [...]]]"), "try" and "stop" and "stop" and "ruin" are candidate pairs, but "try" and "ruin" are not.

After the preprocessing, we obtained 11,852 candidate event pairs for the human annotation from the 275 documents.

5.3.2 Annotation

Annotation Instruction

We present each candidate pair with the whole span of both events and highlight the predicate of each event with "{V: }", as shown in the lower part of Figure 5.2. Annotators can choose one of four options to answer the relation between the two events. For the annotation task,

Optional argument: Event 2 is an argument of the verb of Event 1, but Event 2 only provides additional or clarifying information about Event 1 (for example, Event 2 may indicate the purpose or goal of Event 1). Event 1 is still syntactically complete and meaningful, although it may not keep the exact same meaning, if Event 2 is removed.
For example,

Sentence: A power failure with the docking door forces Brodski to go EVA to fix it.

Event 1: Brodski {V: go} EVA to fix it.

Event 2: Brodski {V: fix} it.

Explanation: Event 2 is an optional argument, indicating the purpose of Event 1. Removing it leaves Event 1 "Brodski go(es) EVA", which is still a syntactically complete and meaningful clause.

Figure 5.3: A screenshot of part of the instructions for annotators. For each of the four labels, we provide the definition, examples of the relation, and a detailed explanation of why the example is labeled as it is. The screenshot shows this for the "optional argument" label.

we separate the non-argument case into two: *condition* and *independent* ("not an argument or a condition"). This made the definitions of the labels easier to understand, since a condition is also, like argument events, a hypothetical, rather than actual, event in the text. Identifying conditional statements in the text is itself an interesting and active topic of research [Fischbach *et al.*, 2021; Tan *et al.*, 2022b, Chapter 4], motivated in particular by their relation to causality. However, the events labeled conditions in our dataset may only be a subset of all conditionals in the source texts, because they are restricted to occurring in contained event pairs. For the analysis of the dataset and experiments in the remainder of this chapter, we treat the two non-argument labels as one.

We prepared annotation instructions including the task description, the definitions of the four labels, and 2–3 examples with a corresponding explanation of why the decision is made for each label, as the screenshot of the whole instruction in Figure 5.3 shows. The full annotation instruction is shown in Appendix A.

Annotation Procedure

We adopt a multi-level qualification-based annotation procedure in which annotators' work is sampled and inspected/corrected in several stages, and feedback from the inspections is passed back to the annotators. A schematic of the procedure is shown in Figure 5.4. Compared with the commonly used crowd-sourcing and voting strategies, this procedure makes annotators learn to improve throughout the task. Although, in the end, not all event pairs have been annotated by multiple participants, the strict qualification tests used along the way ensure annotators give their best answers.

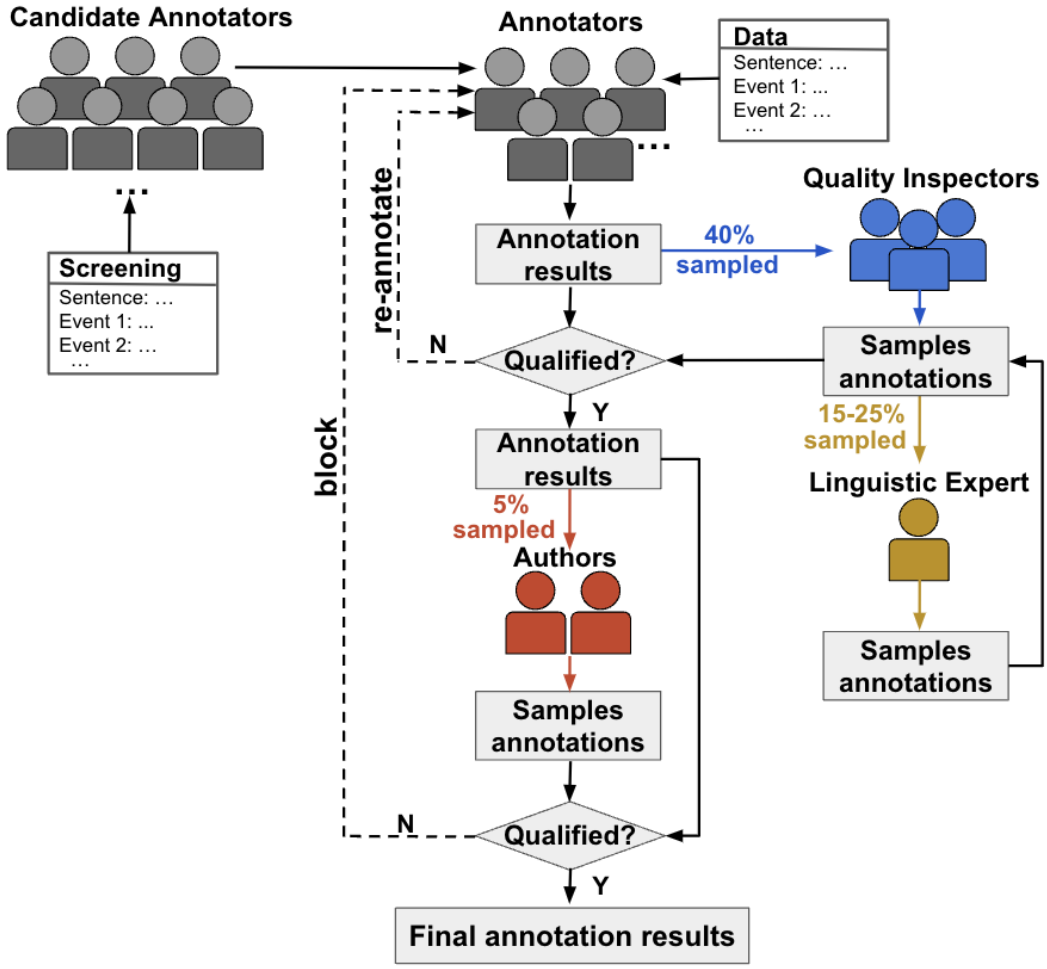


Figure 5.4: Illustration of our multi-level qualification-based annotation procedure.

Over 80 *candidate annotators* were recruited from colleges in China through a data annotation agency. They are English linguistic-major graduate students and fluent in English (passed the TEM-8 test[1]). After reading the instructions, candidates took a screening test, requiring them to annotate 50 examples, also annotated by authors, and cross-checked with the *linguistic expert*. The linguistic expert, who has rich NLP-related annotation experience, was hired via the same data annotation agency. 16 candidates who answered at least 85% of these questions correctly in the test were selected as *annotators*. Among the 16 annotators, the three with the highest accuracy were selected as *quality inspectors* (QIs). After further training, in which the linguistic expert explained the guidelines and their mis-annotated cases from the screening test, the annotators started the annotation.

The event pairs for annotation were evenly split into 3 subsets and sequentially released to the annotators in 3 stages, each one week apart. In each stage, after the annotators submit their

[1]Test for English Majors-band 8, the highest level test in China for English major students that measures the overall English proficiency.

annotations, the QIs randomly select and annotate 40% of the cases. The expert sampled 15-25% of cases inspected by the QIs, and checked the QIs' annotation results. If any incorrect annotations are found, the expert will send them back to the QIs with the expert's annotations and corresponding explanations to make the QIs understand the problems and avoid the same errors in the next stage. This kind of expert supervision controls the quality of the QIs and then avoids untrustable labels from QIs that mislead the annotators. After the linguistic expert reviewed the QIs' annotations, if the agreement between an annotator's answers and the corresponding QI's was lower than 95%, the annotator was asked to re-annotate the inconsistent cases. The loop stops when the agreement ratio reaches 95%.

Myself and a collaborator also monitor the quality of the annotation. In each stage, we sampled 5% of the once-only annotated results from each annotator and labeled these cases to check their accuracy. If the accuracy achieved by an annotator is below 85%, the work of that annotator is discarded, and the annotator is removed from subsequent stages. Only one annotator failed this test and was removed after the first stage. Not counting the author inspection cases, 4,771 (40.2% of all) cases received annotation at least four times.

Event Representation Refinement

As described in the second annotation example in Figure 5.2, one event (predicate) can be contained in another event but is actually not an argument of it. For the event pairs that annotators label as independent, we further ask them to refine the containing event ("Event 1") by removing the span of the contained event ("Event 2") from it. However, the contained event often shares the subject or object with the containing event, so annotators will keep any shared parts in Event 1 and remove the only remaining Event 2 spans. This manual event representation refinement proceeds simultaneously with the annotation of the event dependency relation and is inspected by the QIs and the linguistic expert as well.

Events tagged as conditions typically don't share subjects or objects with their enclosing events. We found automatic revision of the enclosing event feasible, and so didn't request annotators for it. If the conditional event e_j and its containing event e_i appear in the sentence as one of the subsequences (1) $e_i\ s\ e_j$ or (2) $s\ e_j\ e_i$ or (3) $e_j,\ e_i$, where s is one of the signal words/phrases "if", "whenever", "as long as", "on [the] condition that", "unless", or "provided that", then we first remove all words in e_i that are within the span of e_j, and, second, remove any signal word/phrase that is antecedent and adjacent to e_j. In the few cases where no signal word or phrase was detected in the second step, we manually checked and revised the containing event.

5.3.3 Analysis

Table 5.2 shows the distribution of event pairs classified under each label in the annotated dataset, as well as the number of event pairs in the training, development and test document subsets.

It is not surprising that the distribution is biased, with a majority (just over 75%) of event pairs labeled as "argument". The predicate-within-containing-event-span relation between events in the pairs we selected for annotation is a necessary condition for them to be dependent. Of the 2,901 pairs labeled as non-argument (condition or independent), 2,490 distinct events' representations are refined by the annotators and by our automated method.

For human-annotated datasets, there is always a trade-off between the number of instances being annotated and the quality of annotations [Kryscinski *et al.*, 2019; Cui *et al.*, 2020]. The size of our dataset is constrained by the annotation method. But it is comparable with or larger than many

Table 5.2: Distribution of event pairs in the annotated EDeR dataset across labels and across the training, development and test subsets.

	argument		**non-argument**		**overall**	
	required	**optional**	**condition**	**independent**		
train	4,096	2,837	335	1,861	9129	(77%)
dev	635	421	41	355	1,452	(12.3%)
test	594	368	70	239	1271	(10.7%)
overall	5,325 (44.9%)	3,626 (30.6%)	446 (3.8%)	2,455 (20.7%)	11,852	

other human-annotated event relations reasoning datasets, e.g., Glavaš *et al.* [2014]; Cassidy *et al.* [2014]; Mirza and Tonelli [2016]; Ning *et al.* [2018b]; Tan *et al.* [2022a]. We conducted various kinds of inspections to ensure the quality of EDeR. The statistics are as follows: (1) Among the 4,771 samples (40.2% of the total 11,852 cases) from EDeR that have received the three Quality Inspectors' annotations, only 31 (0.65%) of them received three different labels from the three QIs; (2) we randomly sampled 417 cases that received at least four annotations; the accuracy of the annotated labels is 90.65% (378/417); (3) a 5.2% sample of the overall annotations was double-checked by the authors, with larger than 88% accuracy. These results show that the multi-level iterative annotation procedure produces high-quality annotation results.

5.4 Experiment: Event Dependency Relation Prediction

Given two events e_i and e_j from a sentence X, the basic task is to predict whether e_j is an argument of e_i or not. In this section, we evaluate the performance of several baseline methods on this task. The best achieves an accuracy of just over 82%. In Section 5.5, we show that even this binary classification of events into dependent and independent is sufficient to improve performance in two further NLP tasks: event representation extraction and co-reference resolution.

The task can be refined into a three-way classification task, by distinguishing required and optional arguments, and into a four-way classification task by distinguishing the two types of non-argument events: conditions and independent events. The three- and four-way classification tasks are significantly harder. A summary of results is presented in Section 5.6.

5.4.1 Baseline Models

We evaluate several pre-trained language models, and a rule-based heuristic method.

The language models include discriminative models BERT [Devlin *et al.*, 2019], DistilBERT [Sanh *et al.*, 2019], XLNet [Yang *et al.*, 2019], RoBERTa [Zhuang *et al.*, 2021]; as well as two auto-regressive generative models GPT-2 [Radford *et al.*, 2019] and ChatGPT[2] (i.e., GPT-3.5-turbo). Specifically, we feed ChatGPT four examples from the annotation instruction, representing each of the four relations used in the annotation as the prompt, along with the ground truth dependency labels, employing a "few-shot learning" approach.

To recap, we also apply the unsupervised heuristic rules-based method, which is introduced in Section 4.3.1 in Chapter 4. The rules are: If any of them is satisfied, a contained event e_j is an

[2]https://chat.openai.com/chat

Table 5.3: Comparison of the performance on the test set based on varying method and input combinations for the event dependency relation extraction task. The top-3 best results in each column are highlighted. Note: The majority classifier predicts all cases as "argument".

Input	**Model**	**P (%)**	**R (%)**	**F1(%)**	**Acc(%)**
	Majority	75.50	100.00	86.04	75.50
Sentence+predicates	Rule-based	**97.67 (1)**	34.93	51.46	50.08
Event-Event Span	DistilBERT	82.75	90.23	86.33	78.35
	BERT	87.53	85.34	86.42	79.69
	RoBERTa	84.71	87.53	86.10	78.58
	XLNet	83.61	89.60	86.5	78.82
	GPT-2	85.91	86.80	86.35	79.21
	ChatGPT	76.33	85.14	80.49	68.76
Event-Event-SRL	DistilBERT	84.75	87.84	86.27	78.82
	BERT	85.96	87.21	86.58	79.53
	RoBERTa	82.89	**91.16 (3)**	86.83	80.06
	XLNet	82.00	**93.76 (1)**	**87.49 (3)**	79.69
	GPT-2	86.60	85.97	86.28	79.29
	ChatGPT	80.19	89.60	84.63	75.37
Event-Event-SRL-DEP	DistilBERT	85.48	87.53	86.49	79.29
	BERT	85.26	89.60	87.38	80.39
	RoBERTa	83.21	**91.16 (3)**	87.00	80.37
	XLNet	83.11	91.06	86.90	79.21
	GPT-2	85.89	87.94	86.90	79.92
	ChatGPT	82.01	52.60	64.09	55.39
Marked-predicate Sentence	DistilBERT	82.13	**93.14 (2)**	87.29	79.45
	BERT	**91.30 (2)**	85.14	**88.11 (1)**	**82.61 (1)**
	RoBERTa	**89.87 (3)**	85.85	**87.81 (2)**	**81.97 (2)**
	XLNet	88.22	86.38	87.29	**80.94 (3)**
	GPT-2	85.26	89.60	87.38	80.39
	ChatGPT	75.83	80.87	78.27	66.01

argument of the containing event e_i:

(i) The syntactic dependency relation from the predicate of e_i to the predicate of e_j is the clausal complement (*ccomp* or *xcomp*) or clausal subject (*csubj*).

(ii) The syntactic dependency relation from the predicate of e_j to the predicate of e_i is copula (*cop*).

(iii) All of e_j is contained in an argument of e_i that is labeled with either ARGM-PRP ("purpose") or ARGM-PNC ("purpose not cause").

The syntactic dependency relations are obtained by the Stanford CoreNLP dependency parsing [Manning *et al.*, 2014].

5.4.2 Implementation Settings

We employ the `distilbert-base-cased`, `bert-large-cased`, `roberta-large`, `xlnet-large-cased` and `gpt2-large` in HuggingFace's Transformer Library Wolf *et al.* [2020] as representing the DistilBERT, BERT, RoBERTa, XLNet and GPT-2 baseline models in the experiment. For the fairness of comparison and considering the training data size, all the models use the same optimization method as AdamW [Loshchilov and Hutter, 2017] which is adopted with an initial learning rate of 1e−5 and a batch size of 4 and 4 epochs during the fine-tuning process. The model with the highest F1 score on the development set is selected.

5.4.3 Input Variations

Besides various baseline models, we also explore the influence of different types of input.

Event-Event Span We create the two events' span style input. For the mentioned example, the Event-Event Span style input becomes "'The man who works here {V: tells} me to get the hell out. [SEP] me {V: get} the hell out" – using a special token "[SEP]" for separation.

Event-Event-SRL For further checking the influence of the SRL labels, we add the argument label of the argument where the contained event predicate is in the containing event. We also add a special token "[SRL]" for model recognition. E.g., "The man who works here {V: tells} me to get the hell out. [SEP] me {V: get} the hell out [SRL] ARG1."

Event-Event-SRL-DEP Based on the Event-Event-SRL input, we further add the syntactic dependency relation information of the two event predicates. A special token "[DEP]" is added. For instance, "The man who works here {V: tells} me to get the hell out. [SEP] me {V: get} the hell out [SRL] ARG1 [DEP] xcomp."

Marked-Predicate Sentence Inspired by the success of MarkedBERT [Boualili *et al.*, 2020], we add special marks to emphasize the predicate tokens in the sentence. For example, the sentence "The man who works here tells me to get the hell out." with event predicates "tells" and "get" becomes "The man who works here [V1] tells [\V1] me to [V2] get [\V2] the hell out."

5.4.4 Results and Analysis

Table 5.3 shows different models' performance based on different inputs. The heuristic rules-based method achieves the highest precision of 97.67%, yet the recall is considerably low at only 34.93%. We observe that adding the event predicate's syntactic dependency and the (semantic) argument label information improves the performance of most of the models. However, using the

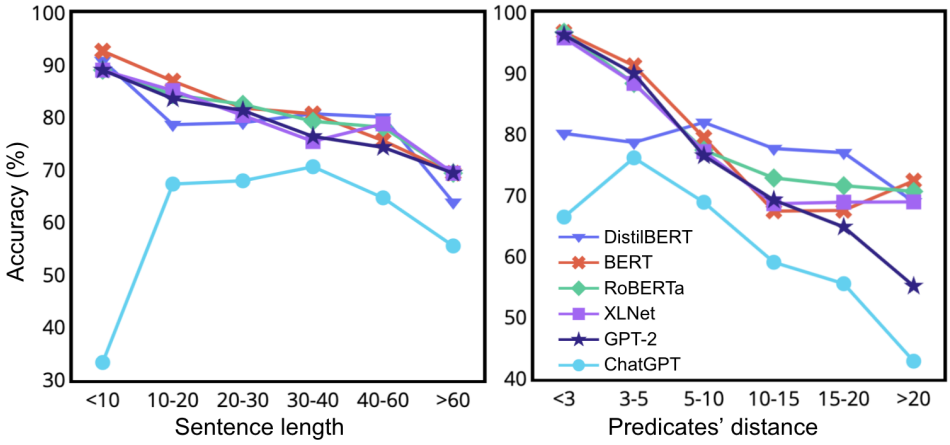


Figure 5.5: Model performances (accuracy) across different ranges of sentence length (Left) and different ranges of predicates' distance (Right).

marked-predicate single sentence as input is more effective than these event-event style inputs. Taking the marked-predicate sentence as input, the BERT and RoBERTa models outperform others.

GPT-2 underperforms compared to discriminative models, indicating its generative architecture struggles with classification tasks such as detecting event dependencies. Similarly, ChatGPT's performance is also unsatisfactory, suggesting that its few-shot learning approach, despite leveraging a powerful pre-trained model, is inadequate for comprehending the dataset. The highest baseline accuracy reaches 82.61%, revealing that EDeR is sufficient for training a good predictor for recognizing event dependency relations based on the language model's pre-train and fine-tune mechanism.

Impact of Sentence Length and Predicate Distance. Figure 5.5 (left) illustrates the performance of the baseline models across sentences with different lengths, given the input as the marked-predicate sentences. Consistent with human intuition, the performance of all the models (except for ChatGPT) decreases when the sentence length increases. Longer sentences may contain more complex structures, which are challenging for models to capture. Similarly, Figure 5.5 (right) shows a decrease in accuracy with an increasing distance (i.e., the number of words) between two predicates, illustrating the models' struggle with distant dependencies. Specifically, few-shot-learning ChatGPT often performs better with moderate sentence length and predicate distance due to challenges with limited information from short sentences and complexity in understanding complex event structures.

5.5 Benefit to Related NLP Tasks

In this section, we show how EDeR data can be used to improve performance on two related NLP tasks.

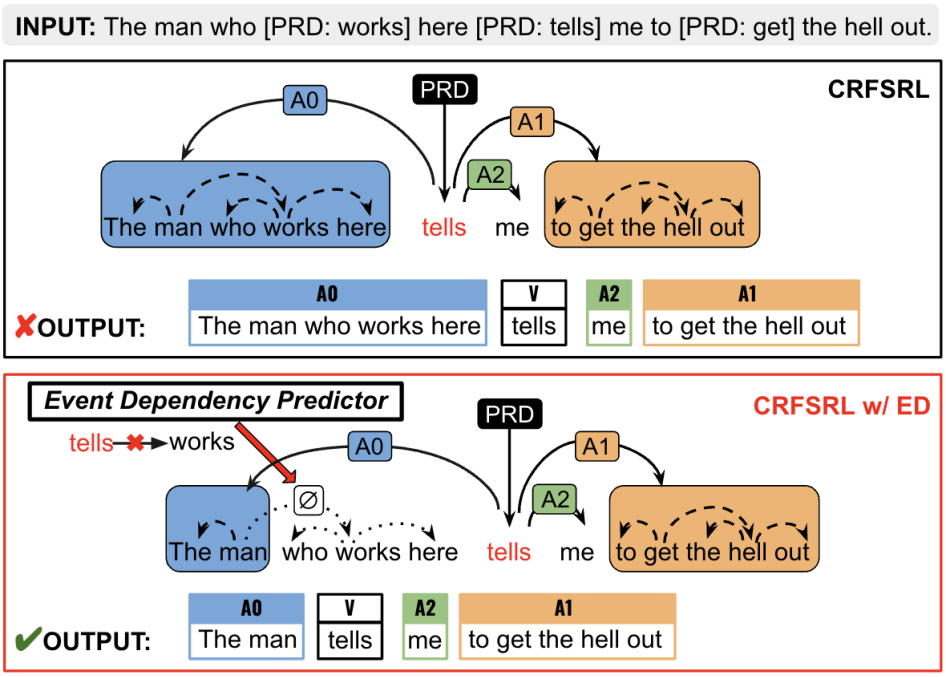


Figure 5.6: An example of the event representation output by the CRFSRL system [Zhang *et al.*, 2022] (above) and this system equipped with our event dependency relation information (below).

5.5.1 Event Representation Extraction

As mentioned in Section 5.3.2, in addition to the classification of event dependencies, in instances where the contained event is not an argument, our data set includes more precise argument spans, which omit parts unique to the contained non-argument event, for containing events. We evaluate the performance of a state-of-the-art SRL system on the resulting refined event extraction task. Our results show that the refined task is harder, and that giving the system access to the dependency classification (annotated or predicted) improves it.

We selected a SOTA SRL system – CRFSRL [Zhang *et al.*, 2022] as the baseline. CRFSRL processes a sentence with a marked event predicate (p) by predicting a labeled tree of syntactic dependencies between words, and extracting sub-trees as argument spans. This is illustrated in the top part of Figure 5.6. Our revised system, CRFSRL with event dependency information (**w/ ED**), prunes sub-trees rooted at non-argument predicates, as shown in the bottom part of Figure 5.6. In the following, **w/ ED (G)** uses the gold-standard event dependency information from the EDeR annotation, while **w/ ED (P)** uses predicted event dependency relations, from the baseline model with the highest accuracy.

We train both the CRFSRL and CRFSRL w/ ED systems on the training portion of the EDeR data set. Thus, both systems are trained to predict the refined argument spans. For comparison, we also trained a version of CRFSRL on the corresponding subset of the unmodified OntoNotes data set. The evaluation was done using the scripts provided for the CoNLL-2012 shared task[3].

[3]https://www.cs.upc.edu/ srlconll/

Table 5.4: Performance of the CRFSRL system [Zhang *et al.*, 2022], and our CRFSRL systems with predicted (**w/ ED (P)**) and annotated (**w/ ED (G)**) event dependency information on the refined event extraction task. (a) For comparison, the performance of the CRFSRL system on the original (unrefined) event extraction task. This system is trained on the subset of OntoNotes corresponding to EDeR-train, i.e., with unrefined argument spans, and tested on the subset corresponding to EDeR-test. (b) This is the subset of the EDeR-test containing only events with at least one refined argument span (i.e., that differ from the original OntoNotes annotation).

System	Precision	Recall	F1-score
Test set: subset of OntoNotes corresponding to EDeR-test$^{(a)}$.			
CRFSRL (OntoNotes-trained)	77.07	81.03	79.00
Test set: EDeR-test.			
CRFSRL (EDeR-trained)	74.19	78.45	76.26
– w/ ED (P)	74.35	78.53	76.38
– w/ ED (G)	**76.09**	**78.81**	**77.43**
Test set: events with refined argument only$^{(b)}$.			
CRFSRL (EDeR-trained)	67.34	65.73	66.53
– w/ ED (P)	81.14	67.97	73.97
– w/ ED (G)	**88.11**	**69.51**	**77.71**

Table 5.4 summarises the result. CRFSRLs performance on the refined extraction task, compared to the original task, suggests that the system finds it more challenging to model the refined event representations in the EDeR dataset. However, results also show that giving the system access to the event dependency information allows some of that drop to be recovered. The performance gap is much more noticeable on the subset of events in the EDeR-test with at least one refined argument span (shown in the bottom part of Table 5.4). In these cases, using either the baseline model-predicted or human-annotated event dependency features, CRFSRL w/ ED (P) and (G) both improve significantly (increases of 13.8% and 20.77% in precision, and 7.44% and 11.18% in F1-score) over the CRFSRL system without event dependency information.

These findings underline the importance of event dependency relation information in enhancing the SRL system for event representation extraction, especially for refined events.

5.5.2 Co-reference Resolution

To further investigate whether the refined event representations from EDeR are more reasonable than OntoNotes' original version, we apply them to a downstream task: co-reference resolution (CR).

Given a text, CR aims to identify all mentions that refer to the same entity. We use coref-HGAT [Jiang and Cohn, 2021], a SOTA CR model, initially trained on OntoNotes CR annotations. This model incorporates both event semantic role and word syntactic dependency information in heterogeneous graphs, creating contextualized embeddings to identify co-reference links. However, the model can misinterpret unrelated entities as co-referents. Figure 5.7 shows an example, where "the girl" is wrongly linked to "she". In this example, there is a sentence, "She went to visit them", prior to the input sentence, and "she" in the input sentence actually

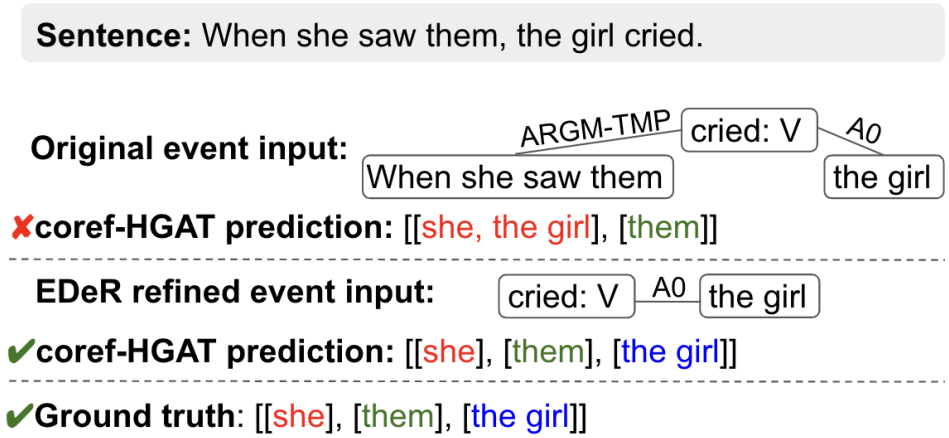


Figure 5.7: An example shows how EDeR-refined event representations help the coref-HGAT model get the correct output, unlike when using OntoNotes' original representations.

refers to "She" in this prior sentence. In this case, EDeR's refined event representation removes the distracting potential referent, leading the model to accurately identify "she" and "the girl" as separate entities. While it is also possible that the reduced event representation prunes information that is important for identifying co-referents, our results show that the cases that benefit from it are at least more frequent.

Table 5.5: Comparison of coref-HGAT model's co-reference resolution performance using annotated (G) and predicted (P) event representations from OntoNotes and EDeR.

Input	Precision	Recall	F1-score
OntoNotes Event Reps. (G)	67.73	65.41	66.50
EDeR Event Reps. (G)	**69.14**	**65.69**	**67.40**
OntoNotes Event Reps. (P)	64.82	63.79	64.28
EDeR Event Reps. (P)	**67.79**	**64.42**	**65.96**

We conduct this CR task using the annotations from the same 275 OntoNotes documents and use the CoNLL-2012 official scripts for evaluation.

As Table 5.5 shows, The EDeR annotated updated event representations (i.e., **EDeR Event Reps. (G)**) boost the coref-HGAT model's precision, recall, and F1-score by 1.41%, 0.28%, and 0.9%, respectively, when compared with the original event representations (i.e., **OntoNotes Event Reps. (G)**). This proves the refined event representations enhance co-reference resolution, demonstrating their validity.

Furthermore, the coref-HGAT model with CRFSRL w/ ED (P) predicted updated events (i.e., **EDeR Event Reps. (P)**) also reveals enhanced precision, recall, and F1-score by 2.97%, 0.63%, and 1.68%, compared with the model with CRFSRL predicted OntoNotes original events (i.e., **OntoNotes Event Reps. (P)**), as the last two lines in Table 5.5) shows. Despite model prediction errors, these results are even comparable to the performance of using OntoNotes Event

Reps. (G). This highlights the feasibility of applying the CR model more widely, by substituting the original annotated event inputs with our predicted refined event representations.

5.6 Fine-grained Event Dependency Relation Extraction

As described, for our EDeR dataset, the argument label can be further categorized into two more fine-grained classes: *required argument* and *optional argument*. Likewise, the non-argument label can be further separated into the two classes *condition* and *independent*. We apply the baseline models and inputs also to these three-way and four-way classification tasks.

The three-way classification results of the baseline language models taking various types of inputs are shown in Table 5.8. First, the Event-Event-SRL-DEP input achieves overall better performance than others in this task. Besides the useful information provided by the syntactic dependencies as discussed in Section 5.4.4, another possible reason is that in the EDeR dataset, the predicates of over 95.79% of the events labeled as required arguments are located within the numbered arguments (ARG0–ARG5) of their containing events. Second, like in the binary classification task, the BERT and RoBERTa models outperform the others. RoBERTa achieves the best overall precision, recall, F1 score and accuracy results, which reveals its better relation reasoning capacity. Third, the overall precision and recall results for the "required argument" label tend to be higher in comparison to other labels, likely as a result of its preponderance in the dataset.

Finally, compared with the binary classification, the overall performance is significantly dropped – the highest accuracy for the three-way and four-way classifications is 70.79% and 69.69%, respectively. It demonstrates that fine-grained classification is more challenging for our current baseline models. Such a significant performance gap will direct our further investigation.

5.6.1 Case Study

We identify two phenomena from the Event-Event-SRL-DEP based RoBERTa model for the fine-grained event dependency relation extraction:

Syntactic Dependency Parsing Errors Some incorrect (three-way) classification results may be because of the misleading syntactic dependency parsing results. As shown in the first case of Figure 5.9, the off-the-shelf dependency parser wrongly assigns syntactic dependency relations between the two event predicates "work" and "have", which could affect the model prediction.

Event Argument Labeling Errors Furthermore, as the second case of Figure 5.9 shows, the semantic labels provided by OntoNotes for the event predicate "had" are incorrect (e.g., "the group" is missing as the ARG0). It may prevent the model from accurately predicting the event dependency relation types.

5.7 Conclusion

In this chapter, we introduced EDeR, a high-quality human-annotated event dependency relation dataset. We presented a thorough description of the dataset construction process, followed by a detailed analysis. We implement various language models and one unsupervised rule-based method, to establish benchmark performance against the dataset. The experimental results of these competitive baselines, along with the evident improvements in some related NLP tasks

Input	Model	Precision (%)				Recall (%)				F1(%)	Accuracy(%)
		required	optional	non-arg.	overall	required	optional	non-arg.	overall		
Event-Event Span	DistilBERT	76.06	52.04	55.06	61.05	81.31	45.11	56.49	60.97	60.90	64.80
	BERT	82.90	58.48	54.62	65.33	85.69	44.02	67.21	65.64	64.92	69.13
	RoBERTa	83.47	54.01	59.41	65.63	83.33	54.89	58.44	65.56	65.59	69.06
	XLNet	80.46	50.71	56.39	62.52	83.16	48.37	55.84	62.46	62.47	66.46
	GPT-2	76.41	52.01	56.68	61.70	82.32	45.65	56.49	61.49	61.49	65.43
	ChatGPT	47.63	31.08	32.08	36.93	52.53	47.55	37.59	35.19	32.32	39.65
Event-Event-SRL	DistilBERT	75.04	51.50	**64.68**	63.74	84.51	55.98	42.21	60.90	61.41	65.98
	BERT	84.59	54.55	57.89	65.68	83.16	53.80	60.71	65.89	65.77	69.21
	RoBERTa	83.75	58.13	56.37	66.08	84.18	50.54	64.61	66.44	66.08	69.69
	XLNet	81.66	59.27	56.20	65.71	84.68	44.29	69.16	66.04	65.28	69.21
	GPT-2	76.12	54.75	58.25	63.04	82.66	47.01	58.44	62.70	62.73	66.46
	ChatGPT	54.55	32.45	26.92	37.97	65.66	46.74	2.27	38.22	34.02	44.77
Event-Event-SRL-DEP	DistilBERT	77.27	53.93	60.23	63.81	81.82	55.98	50.65	62.82	63.15	66.77
	BERT	81.51	56.42	60.70	66.21	85.35	51.36	61.69	66.13	66.12	69.76
	RoBERTa	83.20	**59.38**	59.04	**67.21**	**85.86**	52.45	63.64	**67.31**	**67.15**	**70.79**
	XLNet	**86.27**	53.68	58.07	66.01	82.49	55.43	60.71	66.21	66.08	69.37
	GPT-2	78.58	53.42	56.57	62.86	82.15	46.74	60.06	62.99	62.82	66.54
	ChatGPT	55.07	26.18	27.39	36.21	19.19	37.77	47.25	34.74	31.36	31.39
Marked-predicate Sentence	DistilBERT	73.07	51.91	61.57	62.18	81.31	48.10	53.57	60.99	61.40	64.96
	BERT	85.26	53.80	58.66	65.91	81.82	50.00	68.18	66.67	66.13	69.29
	RoBERTa	84.32	56.93	56.23	65.83	84.18	51.36	62.99	66.17	65.89	69.53
	XLNet	84.75	55.34	54.41	64.83	80.47	46.47	70.13	65.69	64.78	68.11
	GPT-2	77.00	50.00	55.11	60.70	45.38	57.79	**79.46**	60.88	60.74	64.33
	ChatGPT	37.68	28.42	25.00	32.39	17.51	**66.03**	11.33	33.22	27.54	31.86

Figure 5.8: Comparison of the performance for the three-way event dependency relation classification task. The precision and recall results for particular labels are presented as columns under "required", "optional" and "non-argument". The best results in each column are highlighted. The overall precision, recall and F1 score results are macro-averaged.

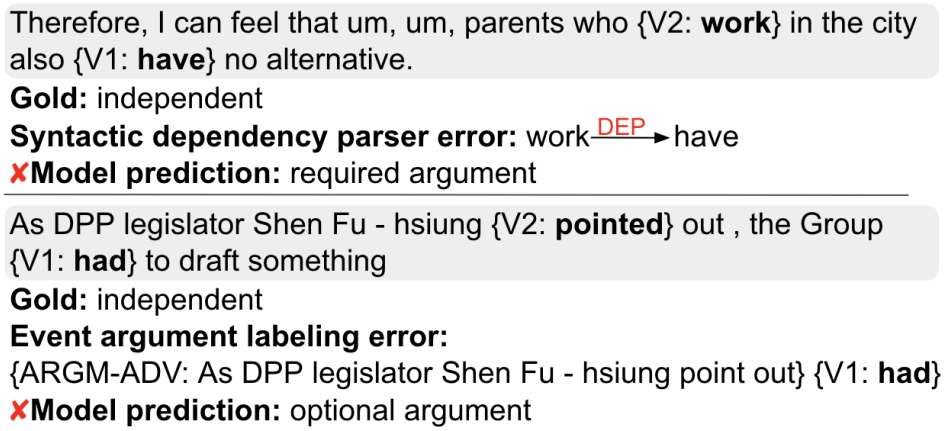


Figure 5.9: Case study for fine-grained event dependency relation extraction.

such as semantic role labeling and cor-reference resolution, demonstrate the utility and benefit of the EDeR dataset. We hope that EDeR facilitates future research in uncovering potential relations among events, thereby enriching our understanding of the linguistic world.

On the other hand, further baseline performance in predicting the three-way and four-way classification results demonstrates the challenge of the dataset: Compared with the binary classification of argument/non-argument, the performance is significantly lower. It demonstrates that fine-grained classification tasks are more challenging for our current baseline models. Such a significant performance gap will direct our further investigation.

Chapter 6

Conclusion

In this thesis, we proposed methods for extracting action models from physical and narrative domains, and further proposed a human-annotated dataset towards the understanding of event-event dependency relationships. In this chapter, we conclude this thesis by summarizing our key contributions and then discussing future work.

6.1 Contributions

First, we performed action description generation and novelty characterization for physical environments. In particular, we creatively applied qualitative spatial relations to describe states in action models, accentuating the physical intersections between objects. We introduced a graph-based hierarchical clustering system, with an emphasis on action extraction from observed changes in spatial relations subsequent to the action. By utilizing this system, we extracted action models from the Angry Bird physical domain, which marked significant progress in the validity, coverage, and size of the models. In addition, we utilize this unsupervised and automated system to identify and characterize novelties in the physical domain, demonstrating its ability to distinguish a wide range of novelty properties presented in actions.

Second, we proposed NaRuto, an innovative automated two-stage system tailored to generate planning action models from narrative texts. The system possesses the capability to generate action models from complex narratives by extracting events and utilizing predicate predictions based on commonsense knowledge. The validation of NaRuto's effectiveness was established by examining its ability to accurately identify action names, generate action preconditions and effects, and facilitate the development of alternative narrative plans. An exemplary application of NaRuto was demonstrated by integrating its generated action models to guide large language models in story generation. This integration produced stories that were considerably more contextual and coherent.

Third, we introduced event dependency relationships that draw attention to situations in which the argument of an event is itself an event. We introduce the EDeR dataset, which is a human-annotated dataset focusing on comprehending event dependency relations. Comprising 275 documents spanning seven diverse genres, it contains 11,852 high-quality annotations that were

annotated using a novel event relation taxonomy. By integrating with the event dependency relations that were either predicted by the model or manually annotated, a state-of-the-art semantic role labeling model's performance was improved. By utilizing the refined event representations curated from EDeR to enhance co-reference resolution tasks, we provide additional evidence of their usefulness.

6.2 Future Work

Despite the significant contributions made by this thesis, there are still some unresolved problems that will direct our future work and inspire the work of other AI researchers as well.

6.2.1 Detecting Latent Correlations Between Actions

The state transitions of objects, represented by their QSRs, are used to describe action models discussed in Chapter 3. The occurrence of the current QSRs may indicate the possibility of QSRs for the upcoming state. If, for instance, the QTC relation between two objects is $(-, 0)$ or $(-, -)$ or $(0, -)$ and their RCC is *ec*, then the next state of their RCC must be *po*. To direct the behavior of an AI agent, it is helpful to predict the prior or next action of a given current action due to such correlations between QSRs. This is particularly important when the AI agent is confronted with novelties.

6.2.2 Action Extraction From Images with Noise

Robustness is essential for action models when planning. Currently, our action models are produced using noise-free images that have been screenshotted during game-playing videos or narrative texts without typos. In the future, we will be able to evaluate the model's capacity to abstract actions on a qualitative level by introducing some noise into the images (for example, by changing the coordinate values of the outlines of the objects) and texts (e.g., by adding grammar errors).

6.2.3 Common-sense Reasoning With Undetermined Entity

As mentioned in Chapter 4, our action precondition and effect predictor takes into account just two event arguments, namely the subject and the object, which at the moment restricts the number of action parameters to two. In contrast, the real world can have multiple factors to consider when taking a single action. To be able to handle such circumstances, the aspect of NaRuto that deals with common-sense reasoning requires further improvement.

6.2.4 Action Effect Negation Creation

The generation of negative action effects consistently poses a challenge for the action model production portion of NaRuto. Global and local negations are constructed for NaRuto by predicting whether a positive effect in a given action contradicts any predicates or predicates that are present in the precondition of that action. According to the results of the expert assessment, neither strategy appears sufficiently effective. It would be worthwhile to investigate more promising approaches that can enhance the action's negated effect creation.

6.2.5 Exploring Connections Between Physical and Narrative Domains

Since action models in the physical domain are described using qualitative spatial reasoning (QSR)-based state transitions, events can be represented as sequences of these transitions (Chapter 3). For example, the event "a red bird flies over a pig" might involve a sequence of QSR changes, including QTC, STAR-4, and QDC. These QSR changes result in a series of state transitions. Translating these state transitions into natural language events and connecting them with events generated from narratives (Chapter 4) will be a challenging but intriguing task. Finding a way to correlate the physical and narrative domains based on such event connections would be highly meaningful.

6.2.6 Improving Fine-grained Event Dependency Relation Predictor

As mentioned in Section 5.6 in Chapter 5, when compared with the results of the binary classification, the results of the three-way and four-way event dependency relation classifications are much lower (around 15% dropped). Doing prompt engineering for the ChatGPT model is one possible approach for future work. Notwithstanding its comparatively lower accuracy compared with other BERT-like models, ChatGPT implements zero-shot learning without utilizing the training set. It displays a significant amount of potential in terms of performance improvement.

6.2.7 Application of Fine-grained Event Dependency Relations

In Chapter 5, we demonstrated that the two-way event dependency relation predictions are enough to contribute to the success of various NLP tasks, such as semantic role labeling and co-reference resolution. The fine-grained classes, particularly required arguments and optional arguments, have the potential to be utilized for a variety of different tasks. One possible application of these is to serve as a reference for human writing. For example, if the predicate of an event is "say", then that event ought to have a "required argument" event nested within it since a person is required to "say" something. It can assist human writers in producing narratives that are more logical and comprehensive in their presentation.

6.3 Summary

In summary, this thesis provides an extensive investigation into advanced techniques and systems that contribute to the automation of action model generation in the domains of physical environments and narrative texts. Moreover, it emphasizes the criticality of comprehending and incorporating event dependency relationships in order to generate more precise event and action model representations. While acknowledging certain constraints and possible avenues for future research, the findings of this thesis make a valuable contribution to the development of more efficient approaches to domain model reasoning and learning for the AI planning community.

Appendix A

Appendix: Annotation Instructions

The annotation instructions for constructing the EDeR dataset (Chapter 5) are shown below:

An event is composed of a verb and arguments, where the verb indicates the event's action and the arguments represent the subject, object and so on of the verb. This task requires you to determine whether, in a pair of events (Event 1, Event 2) from the same sentence, Event 2 is an argument or a condition of Event 1, and if it is an argument, whether it is required or optional.

- **Required argument**: Event 2 is required for the verb of Event 1 to be complete and meaningful.

For example,
Sentence: Jenny tries to stop them from ruining her family.
Event 1: Jenny V: tries to stop them from ruining her family.
Event 2: Jenny V: stop them from ruining her family. ("V: ..." marks the verb of each event.)
Explanation: Event 2 is a required argument. If it is removed, Event 1 becomes just "Jenny tries (to)", which is incomplete and not meaningful (to "try to", one has to try to do something).

Another example of the required argument:
Sentence: Krishna suggests Ramesh to send his liquor to town.
Event 1: Krishna V: suggests Ramesh to send his liquor to town.
Event 2: Ramesh V: send his liquor to town.
Explanation: Event 2 is a required argument. If it is removed, Event 1 becomes "Krishna suggests Ramesh", which is not meaningful (to suggest to someone, one must should suggest something).

The third example of the required argument:
Sentence: "The stock price is unstable." Mr. Jones says.
Event 1: "The stock price is unstable." Mr. Jones V: says.
Event 2: The stock price V: is unstable.
Explanation: Event 2 is a required argument. If it is removed, Event 1 would become "Mr. Jones says", which is incomplete and not meaningful (one must say something, "something" is missing if Event 2 is omitted).

- **Optional argument**: Event 2 is an argument of the verb of Event 1, but Event 2 only provides additional or clarifying information about Event 1 (for example, Event 2 may indicate the purpose or goal of Event 1). Event 1 is still syntactically complete and meaningful, although it may not keep the exact same meaning, if Event 2 is removed.

For example,
Sentence: A power failure with the docking door forces Brodski to go EVA to fix it.
Event 1: Brodski V: go EVA to fix it.
Event 2: Brodski V: fix it.
Explanation: Event 2 is an optional argument, indicating the purpose of Event 1. Removing it leaves Event 1 "Brodski go(es) EVA", which is still a syntactically complete and meaningful clause.

Another example of the optional argument relation:
Sentence: It is a good opportunity, as eventually turned out to be.
Event 1: It V: is a good opportunity, as eventually turned out to be.
Event 2: eventually V: turned out to be.
Explanation: Event 2 is an optional argument, adding information about the timing of Event 1. Removing it leaves Event 1 "It is a good opportunity", which is still a syntactically complete and meaningful clause.

- **Condition**: Event 2 describes a condition for the occurrence of Event 1.

For example,
Sentence: Joe will hate her if she tells the truth.
Event 1: Joe will V: hate her if she tells the truth.
Event 2: she V: tells the truth.
Explanation: Event 2 is a condition of Event 1; it is not an argument.

Another example,
Sentence: Should problems arise, we will seek professional help.
Event 1: Should problems arise, we will V: seek professional help.
Event 2: Should problems V: arise.
Explanation: Event 2 is a condition of Event 1; it is not an argument.

- **Neither an argument nor a condition**: Event 2 is not an optional or required argument, or condition of Event 1. It is an independent/separate event, which happens to be mentioned in the same sentence. Like an optional argument or condition, removing Event 2 still leaves Event 1 complete and meaningful. The difference to Event 2 being an optional argument is that when Event 2 is not an argument it does not add any information/clarification about the verb of Event 1; it may still add some information about one of the arguments of Event 1.

For example,
Sentence: Julia, who has fallen in love with Barnabas, discovers his dalliance with Maggie.
Event 1: Julia, who has fallen in love with Barnabas, V: discovers his dalliance with Maggie.
Event 2: Julia, who has V: fallen in love with Barnabas
Explanation: Event 2 is unrelated to Event 1; it is a separate event that happens to share the same subject. Whether Julia falls in love with Barnabas or not does not affect whether, how, or when, she discovers his dalliance with Maggie.

Another example:

Sentence: It increases the discount rate it offers to known customers.
Event 1: It V: increases the discount rate it offers to known customers.
Event 2: the discount rate it V: offers to known customers.
Explanation: Event 2 is unrelated to Event 1; it is a separate event that only describes the object (the discount rate) of Event 1. Event 2 does not affect the verb of Event 1 “increases”.

The third example of neither an argument nor a condition:
Sentence: After an opening-credits montage of the major players demonstrating their abilities, the story begins.
Event 1: After an opening-credits montage of the major players demonstrating their abilities, the story V: begins.
Event 2: an opening-credits montage of the major players V: demonstrating their abilities.
Explanation: Event 2 is a separate/independent event that happens before Event 1. Event 2 only adds some description of the opening-credits montage; it does not add information about Event 1 (”The story begins”). The ordering between the two events does not make Event 2 an argument of Event 1.

Bibliography

Diego Aineto, Sergio Jimenez, and Eva Onaindia. Learning STRIPS action models with classical planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 399–407, 2018. [Cited on page 12.]

Eyal Amir and Allen Chang. Learning partially observable deterministic action models. *Journal of AI Research*, 33:349–402, 2008. [Cited on page 12.]

Garrett Andersen and George Konidaris. Active exploration for learning symbolic representations. In *Advances in neural information processing systems*, pages 5009–5019, 2017. [Cited on pages 2 and 12.]

Jun Araki, Zhengzhong Liu, Eduard Hovy, and Teruko Mitamura. Detecting subevent structure for event coreference resolution. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4553–4558, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). [Cited on pages 15 and 16.]

Masataro Asai and Alex Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. *arXiv preprint arXiv:1705.00154*, 2017. [Cited on page 12.]

Masataro Asai and Christian Muise. Learning neural-symbolic descriptive planning models via cube-space priors: The voyage home (to strips). *arXiv preprint arXiv:2004.12850*, 2020. [Cited on pages 2, 12, and 30.]

Jorge Balazs, Edison Marrese-Taylor, Pablo Loyola, and Yutaka Matsuo. Refining raw sentence representations for textual entailment recognition via attention. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 51–55, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. [Cited on page 63.]

David Bamman, Brendan O'Connor, and Noah A Smith. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 352–361, 2013. [Cited on pages xviii, 6, 14, 43, 45, and 56.]

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005. [Cited on page 48.]

Cosmin Bejan and Sanda Harabagiu. A linguistic resource for discovering event structures and resolving event coreference. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA). [Cited on page 15.]

Steven Bethard. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Proceedings of the 7th international workshop on semantic evaluation (SemEval 2013)*, pages 10–14, 2013. [Cited on pages 7 and 62.]

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*, 2019. [Cited on page 47.]

Ali Furkan Biten, Lluis Gomez, Marçal Rusinol, and Dimosthenis Karatzas. Good news, everyone! context driven entity-aware captioning for news images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12466–12475, 2019. [Cited on pages 6, 14, and 56.]

Claire Bonial, Jena Hwang, Julia Bonn, Kathryn Conger, Olga Babko-Malaya, and Martha Palmer. English propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*, 48, 2012. [Cited on pages 14 and 42.]

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*, 2019. [Cited on page 47.]

Lila Boualili, Jose G. Moreno, and Mohand Boughanem. Markedbert: Integrating traditional ir cues in pre-trained language models for passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1977–1980, New York, NY, USA, 2020. Association for Computing Machinery. [Cited on page 71.]

TE Boult, PA Grabowicz, DS Prijatelj, R Stern, L Holder, J Alspector, M Jafarzadeh, T Ahmad, AR Dhamija, C Li, et al. A unifying framework for formal theories of novelty: Framework, examples and discussion. *arXiv preprint arXiv:2012.04226*, 2020. [Cited on page 18.]

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP 2015*, 2015. [Cited on page 49.]

Satchuthananthavale RK Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90, 2009. [Cited on page 13.]

SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. Learning high-level planning from text. In *The Association for Computational Linguistics*, 2012. [Cited on page 13.]

Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. [Cited on pages 14 and 63.]

Tommaso Caselli and Piek Vossen. The event StoryLine corpus: A new benchmark for causal and temporal relation extraction. In *Proceedings of the Events and Stories in the News Workshop*, pages 77–86, Vancouver, Canada, August 2017. Association for Computational Linguistics. [Cited on page 15.]

Taylor Cassidy, Bill McDowell, Nathanel Chambers, and Steven Bethard. An annotation framework for dense event ordering. Technical report, Carnegie-Mellon Univ Pittsburgh PA, 2014. [Cited on pages 15, 63, and 69.]

Louis Castricato, Spencer Frazier, Jonathan Balloch, Nitya Tarakad, and Mark Riedl. Automated story generation as question-answering. *arXiv preprint arXiv:2112.03808*, 2021. [Cited on page 58.]

Miao Chen, Ganhui Lan, Fang Du, and Victor Lobanov. Joint learning with pre-trained transformer on named entity recognition and relation extraction tasks for clinical analytics. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 234–242, Online, November 2020. Association for Computational Linguistics. [Cited on pages 7, 62, and 63.]

Ning Cheng, Zhaohui Yan, Ziming Wang, Zhijie Li, Jiaming Yu, Zilong Zheng, Kewei Tu, Jinan Xu, and Wenjuan Han. Potential and limitations of llms in capturing structured semantics: A case study on srl. *arXiv preprint arXiv:2405.06410*, 2024. [Cited on page 60.]

Eliseo Clementini, Paolino Di Felice, and Daniel Hernández. Qualitative representation of positional information. *Artificial intelligence*, 95(2):317–356, 1997. [Cited on page 20.]

Anthony G Cohn and Jochen Renz. Qualitative spatial representation and reasoning. *Foundations of Artificial Intelligence*, 3:551–596, 2008. [Cited on page 12.]

S. N. Cresswell, T. L. McCluskey, and M. M. West. Acquiring planning domain models using LOCM. *Knowledge Engineering Review*, 28(2):195–213, 2013. [Cited on page 12.]

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. MuTual: A dataset for multi-turn dialogue reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1406–1416, Online, July 2020. Association for Computational Linguistics. [Cited on page 68.]

DARPA, Defense Sciences Office. Broad agency announcement: Science of artificial intelligence and learning for open-world novelty (SAIL-ON), March 2019. https://sam.gov/api/prod/opps/v3/opportunities/resources/files/109e04254a2dda31bf362f46fe47bce8/download?api_key=null&status=archived&token=. [Cited on page 34.]

Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, Peng Shi, Wenpeng Yin, and Rui Zhang. Unified low-resource sequence labeling by sample-aware dynamic sparse finetuning. *arXiv preprint arXiv:2311.03748*, 2023. [Cited on page 60.]

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990. [Cited on page 12.]

Matthias Delafontaine, Anthony G Cohn, and Nico Van de Weghe. Implementing a qualitative calculus to analyse moving point objects. *Expert Systems with Applications*, 2011. [Cited on pages 3 and 12.]

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. [Cited on page 69.]

Luciana Olivia Dias, Clécio R Bom, Elisangela L Faria, Manuel Blanco Valentín, Maury Duarte Correia, P Márcio, P Marcelo, and Juliana M Coelho. Automatic detection of fractures and breakouts patterns in acoustic borehole image logs using fast-region convolutional neural networks. *Journal of Petroleum Science and Engineering*, page 107099, 2020. [Cited on page 23.]

Quang Do, Yee Seng Chan, and Dan Roth. Minimally supervised event causality identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 294–303, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. [Cited on page 15.]

Christian Dondrup, Nicola Bellotto, Marc Hanheide, Kerstin Eder, and Ute Leonards. A computational model of human-robot spatial interactions based on a qualitative trajectory calculus. *Robotics*, 4(1):63–102, 2015. [Cited on page 21.]

Krishna SR Dubba, Anthony G Cohn, David C Hogg, Mehul Bhatt, and Frank Dylla. Learning relational event models from video. *Journal of Artificial Intelligence Research*, 53:41–90, 2015. [Cited on page 12.]

Paul Duckworth, Yiannis Gatsoulis, Ferdian Jovan, Nick Hawes, David C Hogg, and Anthony G Cohn. Unsupervised learning of qualitative motion behaviours by a mobile robot. In *AAMAS*, pages 1043–1051. AAMAS, 2016. [Cited on page 12.]

Paul Duckworth, David C Hogg, and Anthony G Cohn. Unsupervised human activity analysis for intelligent mobile robots. *Artificial Intelligence*, 270:67–92, 2019. [Cited on pages 3, 12, and 30.]

Frank Dylla, Jae Hee Lee, Till Mossakowski, Thomas Schneider, André Van Delden, Jasper Van De Ven, and Diedrich Wolter. A survey of qualitative spatial and temporal calculi: algebraic and computational properties. *ACM Computing Surveys (CSUR)*, 50(1):7, 2017. [Cited on pages 3 and 12.]

Kawin Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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)*, pages 55–65, Hong Kong, China, November 2019. Association for Computational Linguistics. [Cited on page 63.]

Wenfeng Feng, Hankz Hankui Zhuo, and Subbarao Kambhampati. Extracting action sequences from texts based on deep reinforcement learning. In *IJCAI 2018*, pages 4064–4070, 2018. [Cited on pages 5, 13, and 40.]

Lucas Ferreira and Claudio Toledo. A search-based approach for generating angry birds levels. In *Proceedings of the 9th IEEE International Conference on Computational Intelligence in Games*, CIG'14, 2014. [Cited on page 28.]

Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971. [Cited on page 2.]

Jannik Fischbach, Benedikt Hauptmann, Lukas Konwitschny, Dominik Spies, and Andreas Vogelsang. Towards causality extraction from requirements. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 388–393. IEEE, 2020. [Cited on page 46.]

Jannik Fischbach, Julian Frattini, Arjen Spaans, Maximilian Kummeth, Andreas Vogelsang, Daniel Mendez, and Michael Unterkalmsteiner. Automatic detection of causality in requirement artifacts: the cira approach. In *Proceedings of the 27th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2021)*, pages 19–36. Springer-Verlag, 2021. [Cited on pages 15, 46, and 66.]

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *CoRR*, 2018. [Cited on page 42.]

Yiannis Gatsoulis, Muhannad Alomari, Chris Burbridge, Christian Dondrup, Paul Duckworth, Peter Lightbody, Marc Hanheide, Nick Hawes, DC Hogg, AG Cohn, et al. Qsrlib: a software library for online acquisition of qualitative spatial relations from video. 2016. [Cited on page 23.]

Daniel Gildea and Daniel Jurafsky. Automatic labelling of semantic roles. *Computational Linguistics*, pages 245–288, 2002. [Cited on pages 6 and 40.]

Goran Glavaš and Jan Šnajder. Constructing coherent event hierarchies from news stories. In *Proceedings of TextGraphs-9: the workshop on Graph-based Methods for Natural Language Processing*, pages 34–38, Doha, Qatar, October 2014. Association for Computational Linguistics. [Cited on pages 7, 16, and 62.]

Goran Glavaš, Jan Šnajder, Marie-Francine Moens, and Parisa Kordjamshidi. HiEve: A corpus for extracting event hierarchies from news stories. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3678–3683, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). [Cited on pages 15, 16, and 69.]

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June 2009. Association for Computational Linguistics. [Cited on pages 14 and 63.]

Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. An introduction to the planning domain definition language. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(2):1–187, 2019. [Cited on page 2.]

Thomas Hayton, Julie Porteous, Joao Ferreira, Alan Lindsay, and Jonathan Read. StoryFramer: From input stories to output planning models. In *Proceedings of the Workshop on KEPS 2017*, pages 1–9, 2017. [Cited on pages xxi, 5, 13, 40, 51, 52, 53, and 54.]

Thomas Hayton, Julie Porteous, João F. Ferreira, and Alan Lindsay. Narrative planning model acquisition from text summaries and descriptions. In *Proceedings of the AAAI 2020*, pages 1709–1716, 2020. [Cited on pages 5, 13, 40, 51, 52, and 54.]

Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, jul 2006. [Cited on pages 6 and 56.]

Eduard Hovy, Teruko Mitamura, Felisa Verdejo, Jun Araki, and Andrew Philpot. Events are not simple: Identity, non-identity, and quasi-identity. In *Workshop on events: Definition, detection, coreference, and representation*, pages 21–28, Atlanta, Georgia, 2013. Association for Computational Linguistics. [Cited on pages 7, 16, and 62.]

Pere-Lluís Huguet Cabot and Roberto Navigli. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. [Cited on pages 7, 62, and 63.]

Yongfeng Huo, Jing Tang, Yinghui Pan, Yifeng Zeng, and Langcai Cao. Learning a planning domain model from natural language process manuals. *IEEE Access*, 8, 2020. [Cited on pages xxi, 5, 13, 40, 51, 52, and 53.]

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. In *Proceedings of AAAI 2021*, 2021. [Cited on pages 42 and 47.]

Fan Jiang and Trevor Cohn. Incorporating syntax and semantics in coreference resolution with heterogeneous graph attention network. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1584–1591, Online, June 2021. Association for Computational Linguistics. [Cited on pages 8, 63, and 74.]

Masayuki Komai, Hiroyuki Shindo, and Yuji Matsumoto. An efficient annotation for phrasal verbs using dependency information. In *Proceedings of the 29th PACLIC*, 2015. [Cited on page 44.]

George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Symbol acquisition for probabilistic high-level planning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015. [Cited on page 11.]

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Neural text summarization: A critical evaluation. 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)*, pages 540–551, Hong Kong, China, November 2019. Association for Computational Linguistics. [Cited on page 68.]

Pat Langley. Open-world learning for radically autonomous agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13539–13543, 2020. [Cited on pages 4 and 17.]

Natsuda Laokulrat, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. Uttime: Temporal relation classification using deep syntactic features. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 88–92, 2013. [Cited on pages 7 and 62.]

Md Tahmid Rahman Laskar, Jimmy Xiangji Huang, and Enamul Hoque. Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5505–5514, Marseille, France, May 2020. European Language Resources Association. [Cited on page 63.]

Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*, 2018. [Cited on page 44.]

Seonghyeon Lee, Dongha Lee, Seongbo Jang, and Hwanjo Yu. Toward interpretable semantic textual similarity via optimal transport-based contrastive sentence learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5969–5979, Dublin, Ireland, May 2022. Association for Computational Linguistics. [Cited on page 63.]

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, pages 167–195, 2015. [Cited on pages 14 and 41.]

D. B. Lenat and R. V. Guha. *Building large knowledge-based systems: representation and inference in the Cyc project*. Addison-Wesley Longman, 1989. [Cited on pages 14 and 41.]

Beth Levin. Objecthood: An event structure perspective. 1999. [Cited on pages 14 and 61.]

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. [Cited on pages 41 and 47.]

Dongxu Li, Enrico Scala, Patrik Haslum, and Sergiy Bogomolov. Effect-abstraction based relaxation for linear numeric planning. In *IJCAI*, pages 4787–4793, 2018. [Cited on pages 4 and 18.]

Fangru Lin, Emanuele La Malfa, Valentin Hofmann, Elle Michelle Yang, Anthony Cohn, and Janet B Pierrehumbert. Graph-enhanced large language models in asynchronous plan reasoning. *arXiv preprint arXiv:2402.02805*, 2024. [Cited on page 60.]

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004. [Cited on page 48.]

Alan Lindsay, Jonathon Read, João Ferreira, Thomas Hayton, Julie Porteous, and Peter Gregory. Framer: Planning models from natural language action descriptions. In *ICAPS 2017*, 2017. [Cited on pages 13 and 40.]

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017. [Cited on pages 51 and 71.]

Youmi Ma, Tatsuya Hiraoka, and Naoaki Okazaki. Joint entity and relation extraction based on table labeling using convolutional neural networks. In *Proceedings of the Sixth Workshop on Structured Prediction for NLP*, pages 11–21, Dublin, Ireland, May 2022. Association for Computational Linguistics. [Cited on pages 7, 62, and 63.]

Lydia Manikonda, Shirin Sohrabi, Kartik Talamadupula, Biplav Srivastava, and Subbarao Kambhampati. Extracting incomplete planning action models from unstructured social media data to support decision making. In *Proceedings of the Workshop on KEPS 2017*, pages 62–68, 2017. [Cited on page 13.]

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of ACL 2014*, pages 55–60, 2014. [Cited on pages 42, 51, 65, and 71.]

Dominique Mariko, Estelle Labidurie, Yagmur Ozturk, Hanna Abi Akl, and Hugues de Mazancourt. Data processing and annotation schemes for fincausal shared task, 2020. [Cited on pages 7 and 62.]

Dominique Mariko, Hanna Abi-Akl, Kim Trottier, and Mahmoud El-Haj. The financial causality extraction shared task (FinCausal 2022). In *Proceedings of the 4th Financial Narrative Processing Workshop @LREC2022*, pages 105–107, Marseille, France, June 2022. European Language Resources Association. [Cited on pages 7 and 62.]

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl–the planning domain definition language–version 1.2. *Yale Center for Computational Vision and Control, Tech. Rep.*, 1998. [Cited on pages 1 and 40.]

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl–the planning domain definition language–version 1.2. *Yale Center for Computational Vision and Control, Tech. Rep. CVC TR-98-003/DCS TR-1165*, 1998. [Cited on pages 19 and 26.]

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI 2016*, 2016. [Cited on pages 5 and 40.]

Shivam Miglani and Neil Yorke-Smith. Nltopddl: One-shot learning of pddl models from natural language process manuals. In *ICAPS'20 Workshop on KEPS'20*, 2020. [Cited on page 13.]

Shivam Miglani. Nltopddl: One-shot learning of pddl models from natural language process manuals. In *Working Notes of the ICAPS'20 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS'20)*. ICAPS, 2020. [Cited on page 12.]

Paramita Mirza and Sara Tonelli. CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In *Proceedings the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 64–75. COLING, 2016. [Cited on pages 15, 63, and 69.]

Paramita Mirza, Rachele Sprugnoli, Sara Tonelli, and Manuela Speranza. Annotating causality in the tempeval-3 corpus. In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 10–19, 2014. [Cited on page 46.]

Makoto Miwa and Yutaka Sasaki. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar, October 2014. Association for Computational Linguistics. [Cited on pages 7, 62, and 63.]

Kira Mourao, Luke S Zettlemoyer, Ronald Petrick, and Mark Steedman. Learning strips operators from noisy and incomplete observations. *arXiv preprint arXiv:1210.4889*, 2012. [Cited on page 2.]

Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. Joint reasoning for temporal and causal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2278–2288, Melbourne, Australia, July 2018. Association for Computational Linguistics. [Cited on page 15.]

Qiang Ning, Hao Wu, and Dan Roth. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*. ACL, 7 2018. [Cited on pages 15, 63, and 69.]

Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Gpt3-to-plan: Extracting plans from text using gpt-3. *arXiv preprint arXiv:2106.07131*, 2021. [Cited on pages 5, 13, and 40.]

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. [Cited on page 63.]

Yannis Papanikolaou, Ian Roberts, and Andrea Pierleoni. Deep bidirectional transformers for relation extraction without supervision. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 67–75, Hong Kong, China, November 2019. Association for Computational Linguistics. [Cited on page 63.]

Hanna M Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007. [Cited on page 11.]

Julie Porteous, Fred Charles, and Marc Cavazza. NetworkING: using character relationships for interactive narrative generation. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pages 595–602, 2013. [Cited on page 40.]

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea, July 2012. Association for Computational Linguistics. [Cited on pages 14 and 63.]

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. [Cited on pages 7 and 63.]

Tao Pu, Tianshui Chen, Hefeng Wu, Yongyi Lu, and Liang Lin. Spatial-temporal knowledge-embedded transformer for video scene graph generation. *IEEE Transactions on Image Processing*, 2023. [Cited on page 2.]

Cristina Puente, Alejandro Sobrino, José A Olivas, and Roberto Merlo. Extraction, analysis and representation of imperfect conditional and causal sentences by means of a semi-automatic process. In *FUZZ-IEEE 2010*, 2010. [Cited on pages 45 and 46.]

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40. Lancaster, UK., 2003. [Cited on pages 15 and 63.]

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. [Cited on pages 41 and 47.]

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. [Cited on pages 48 and 69.]

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. [Cited on page 48.]

David A Randell, Zhan Cui, and Anthony G Cohn. A spatial logic based on regions and connection. *KR*, 92:165–176, 1992. [Cited on pages 3, 12, and 20.]

Malka Rappaport Hovav, Edit Doron, and Ivy Sichel. *Lexical Semantics, Syntax, and Event Structure*. Oxford University Press, UK, 02 2010. [Cited on pages 14 and 61.]

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of EMNLP 2019*, 11 2019. [Cited on page 49.]

Jochen Renz and Debasis Mitra. Qualitative direction calculi with arbitrary granularity. In *PRICAI*, volume 3157, pages 65–74, 2004. [Cited on page 20.]

Jochen Renz and Bernhard Nebel. Qualitative spatial reasoning using constraint calculi. In *Handbook of spatial logics*, pages 161–215. Springer, 2007. [Cited on page 12.]

Jochen Renz, Xiaoyu Ge, Stephen Gould, and Peng Zhang. The Angry Birds AI competition. *AI Magazine*, 2015. [Cited on page 28.]

Mark O Riedl and Robert Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, 2010. [Cited on pages xxi, 51, 52, 53, and 54.]

Christophe Rodrigues, Pierre Gérard, and Céline Rouveirol. Incremental learning of relational action models in noisy environments. In *International Conference on Inductive Logic Programming*, pages 206–213. Springer, 2010. [Cited on page 2.]

Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005. [Cited on page 24.]

Régis Sabbadin, Florent Teichteil-Königsbuch, and Vincent Vidal. Planning in artificial intelligence. In *A Guided Tour of Artificial Intelligence Research*, pages 285–312. Springer, 2020. [Cited on page 2.]

Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3379–3388, 2018. [Cited on page 21.]

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019. [Cited on page 69.]

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of AAAI 2019*, volume 33, pages 3027–3035, 2019. [Cited on pages 41 and 47.]

Karin Kipper Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania, 2005. [Cited on page 13.]

Alessandro Seganti, Klaudia Firlag, Helena Skowronska, Michal Satlawa, and Piotr Andruszkiewicz. Multilingual entity and relation extraction dataset and model. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, online, 2021. Association for Computational Linguistics. [Cited on page 63.]

Peng Shi and Jimmy J. Lin. Simple bert models for relation extraction and semantic role labeling. *CoRR*, 2019. [Cited on page 42.]

Avirup Sil and Alexander Yates. Extracting STRIPS representations of actions and events. In *Recent Advances in Natural Language Processing*, 2011. [Cited on page 13.]

Nisha Simon and Christian Muise. Tattletale: Storytelling with planning and large language models. In *ICAPS Workshop on Scheduling and Planning Applications woRKshop*, 2022. [Cited on page 58.]

P. Singh. The public acquisition of commonsense knowledge. In *Proceedings of AAAI Spring Symposium: Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*, 2002. [Cited on pages 14 and 41.]

Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Proc. AAAI*, 05 2019. [Cited on pages 14, 41, and 47.]

Muralikrishna Sridhar, Anthony G Cohn, and David C Hogg. Benchmarking qualitative spatial calculi for video activity analysis. In *Proceedings ijcai workshop benchmarks and applications of spatial reasoning*, pages 15–20. Leeds, 2011. [Cited on page 12.]

Muralikrishna Sridhar, Anthony G Cohn, and David C Hogg. From video to rcc8: exploiting a distance based semantics to stabilise the interpretation of mereotopological relations. In *COSIT*, pages 110–125. Springer, 2011. [Cited on page 12.]

Fiona Anting Tan, Ali Hürriyetoğlu, Tommaso Caselli, Nelleke Oostdijk, Tadashi Nomoto, Hansi Hettiarachchi, Iqra Ameer, Onur Uca, Farhana Ferdousi Liza, and Tiancheng Hu. The causal news corpus: Annotating causal relations in event sentences from news. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2298–2310, Marseille, France, June 2022. European Language Resources Association. [Cited on pages 7, 15, 62, and 69.]

Fiona Anting Tan, Ali Hürriyetoğlu, Tommaso Caselli, Nelleke Oostdijk, Tadashi Nomoto, Hansi Hettiarachchi, Iqra Ameer, Onur Uca, Farhana Ferdousi Liza, and Tiancheng Hu. The causal news corpus: Annotating causal relations in event sentences from news. *arXiv preprint arXiv:2204.11714*, 2022. [Cited on pages 15, 46, and 66.]

Jawad Tayyub, Aryana Tavanai, Yiannis Gatsoulis, Anthony G Cohn, and David C Hogg. Qualitative and quantitative spatio-temporal relations in daily living activity recognition. In *ACCV*, pages 115–130. Springer, 2014. [Cited on page 12.]

Nico Van de Weghe and Philippe De Maeyer. Conceptual neighbourhood diagrams for representing moving objects. In *International Conference on Conceptual Modeling*, pages 228–238. Springer, 2005. [Cited on page 21.]

Nico Van de Weghe, Bart Kuijpers, Peter Bogaert, and Philippe De Maeyer. A qualitative trajectory calculus and the composition of its relations. In *International Conference on GeoSpatial Sematics*, pages 60–76. Springer, 2005. [Cited on pages 3, 12, and 20.]

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. [Cited on page 63.]

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021. [Cited on page 58.]

Wenqing Wang, Yawei Luo, Zhiqing Chen, Tao Jiang, Yi Yang, and Jun Xiao. Taking a closer look at visual relation: Unbiased video scene graph generation with decoupled label learning. *IEEE Transactions on Multimedia*, 2023. [Cited on page 2.]

Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin Liu. Llmˆ 3: Large language model-based task and motion planning with motion failure reasoning. *arXiv preprint arXiv:2403.11552*, 2024. [Cited on page 60.]

Stephen G Ware. *A plan-based model of conflict for narrative reasoning and generation*. North Carolina State University, 2014. [Cited on pages xxi, 51, 52, 53, and 54.]

Bonnie Webber, Rashmi Prasad, Alan Lee, and Aravind Joshi. The penn discourse treebank 3.0 annotation manual. *Philadelphia, University of Pennsylvania*, 35:108, 2019. [Cited on page 46.]

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL 2018*, 2018. [Cited on page 49.]

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. [Cited on page 71.]

Salisu Wada Yahaya, Ahmad Lotfi, and Mufti Mahmud. A consensus novelty detection ensemble approach for anomaly detection in activities of daily living. *Applied Soft Computing*, 83:105613, 2019. [Cited on page 21.]

Qiang Yang, Kangheng Wu, and Yunfei Jiang. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence*, 171(2-3):107–143, 2007. [Cited on page 2.]

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019. [Cited on page 69.]

Linyi Yang, Zhen Wang, Yuxiang Wu, Jie Yang, and Yue Zhang. Towards fine-grained causal reasoning and qa, 2022. [Cited on page 15.]

Kristina Yordanova. From textual instructions to sensor-based recognition of user behaviour. In *Companion Publication of the 21st International Conference on Intelligent User Interfaces*, pages 67–73, 2016. [Cited on page 13.]

Jay Young and Nick Hawes. Learning by observation using qualitative spatial relations. In *AAMAS*, pages 745–751. AAMAS, 2015. [Cited on page 12.]

Chunhui Yuan and Haitao Yang. Research on k-value selection method of k-means clustering algorithm. *J—Multidisciplinary Scientific Journal*, 2(2):226–235, 2019. [Cited on page 12.]

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019. [Cited on page 48.]

Yu Zhang, Qingrong Xia, Shilin Zhou, Yong Jiang, Guohong Fu, and Min Zhang. Semantic role labeling as dependency parsing: Exploring latent tree structures inside arguments. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4212–4227, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. [Cited on pages xix, xxii, 7, 63, 73, and 74.]

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China, August 2021. Chinese Information Processing Society of China. [Cited on page 69.]