NaRuto: Automatically Acquiring Planning Models from Narrative Texts

Ruiqi Li[1], **Leyang Cui**[2], **Songtuan Lin**[1], **Patrik Haslum**[1]
[1]Australian National University,
[2]Tencent AI Lab
{ruiqi.li,songtuan.lin,patrik.haslum}@anu.edu.au,
leyangcui@tencent.com

Abstract

Domain model acquisition has been identified as a bottleneck in the application of planning technology, especially within narrative planning. Learning action models from narrative texts in an automated way is essential to overcome this barrier, but challenging because of the inherent complexities of such texts. We present an evaluation of planning domain models derived from narrative texts using our fully automated, unsupervised system, NaRuto. Our system combines structured event extraction, predictions of commonsense event relations, and textual contradictions and similarities. Evaluation results show that NaRuto generates domain models of significantly better quality than existing fully automated methods, and even sometimes on par with those created by semi-automated methods, with human assistance.

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

Introduction

AI planning generates action sequences, i.e., plans, from the initial problem state to the goals. To construct such plans, AI planners require action models in a declarative planning language, such as the Planning Domain Definition Language (PDDL) (McDermott et al. 1998). Building action models by hand is labor-intensive and demands domain expertise, driving the need for methods that ease this process (e.g., Lin, Grastien, and Bercher 2023). It is particularly challenging in applications of planning to narrative generation (e.g., Porteous, Charles, and Cavazza 2013), due to the large variety of activities that occur in unstructured narrative texts.

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 effort to complement or correct automatic extraction. Fully automated approaches have been applied to instructional texts such as recipes, manuals and navigational instructions (Mei, Bansal, and Walter 2016; Lindsay et al. 2017; Feng, Zhuo, and Kambhampati 2018; Olmo, Sreedharan, and Kambhampati 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 are more difficult to comprehend, and how to automatically extract action models from them is still an open question.

We present NaRuto[1] (an abbreviation of "narrative" and "automated"), a fully automated system that derives planning action models from narrative texts in two stages: The first stage extracts structured representations of event occurrences from narrative text, and has been described in detail in a previous paper (Li, Haslum, and Cui 2023). Here, we focus mainly on the second stage, which constructs action models by predicting preconditions and effects of extracted events using commonsense relations. We use a GPT (Radford et al. 2018) and BART (Lewis et al. 2019) based model, fine-tuned on commonsense knowledge graphs (Sap et al. 2019), which we call COMET-BM, to predict event preconditions and effects. Given an event description and a relation type (precondition, effect, or reaction) as input, it generates descriptions of corresponding concepts with the highest probabilities. We compare the generated models with the results of previous methods in two classical narrative planning domains. Results indicate that NaRuto consistently generates superior action models, surpassing SOTA fully automated methods and sometimes even methods that rely on human input.

[1]The source code of NaRuto system is at https://github.com/RichieLee93/NaRuto.

Background and Related Work

In narrative texts, events tell us what happened, who or what was involved, and where. Semantic role labelling (SRL) (Gildea and Jurafsky 2002) identifies verbs and their arguments, as spans of text, and labels the arguments with their semantic role (e.g., agent, patient, modal modifier, etc), providing a basis for event extraction. However, current SRL methods fail to distinguish certain semantic relations between events.

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: "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.

Action model We target the classical STRIPS formalism (Geffner and Bonet 2013). A STRIPS action model consists of four components: the action name, parameters, preconditions, and effects. Precondition and effects are conjunctions of literals, formed from domain predicates with arguments drawn from the action's parameters, representing what must hold for the action to be applicable, and what becomes true and false as a result of applying it, respectively.

Action Model Generation From Narratives

Extracting action models from text has recently gained interest in AI planning. Sil and Yates (2011) combined web text correlations and supervised learning to identify pre- and post-conditions of verbs. Branavan et al. (2012) used linguistic cues and reinforcement learning to learn action preconditions, while Manikonda et al. (2017) formed incomplete action models by extracting plan traces from social media texts.

Methods of event extraction vary, from using the dependency parse structure to neural language models and reinforcement learning. However, Branavan et al. (2009), Yordanova (2016), Lindsay et al. (2017), Feng, Zhuo, and Kambhampati (2018), Miglani and Yorke-Smith (2020), and Olmo, Sreedharan, and Kambhampati (2021) focus on instructional texts, such as recipes, game-play instructions and user guides. They avoid much of the complexity of dealing with narrative texts. Hayton et al. (2017) and Huo et al. (2020) explored narrative texts, yet their action model generation largely depends on manual interventions. Hayton et al. (2020) proposed an automated process, but their action models aim only to mirror the input narrative.

Commonsense Knowledge Graphs

Most events in narratives are common, with their preconditions and effects being commonsense knowledge. Thus, we can use commonsense knowledge graphs to infer them.

ConceptNet (Speer, Chin, and Havasi 2019) is a knowledge graph of concepts and their relations, which brings together 3.4M entity-relation tuple information collected from many sources, including crowdsourced (e.g., DBPedia by Lehmann et al. (2015)) and curated (e.g., OpenCyc by Lenat and Guha (1989)). It features relations like *Causes* and *HasPrerequisite* crucial for modeling actions/events. The ATOMIC (Sap et al. 2019) knowledge graph is made up of 880K tuples linking 24K events to statements using 9 relations, including causes, effects, social commonsense and so on. For example, the relation *xNeed* associates an event/action with its prerequisite, such as "X gets X's car repaired" *xNeed* "to have money". ATOMIC-2020 (Hwang et al. 2021) is a similar but more robust dataset, offering 1.33M tuples across 23 relations, encapsulating a broader range of commonsense knowledge.

NaRuto

An overview of the NaRuto process is shown in Figure 2. We extract event occurrences, consisting of verbs and their arguments, using the AllenNLP (Gardner et al. 2018) BERT-based SRL system (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 (cf., Li, Haslum, and Cui 2023). Action models are created from the events in two steps: The first employs a commonsense event relation predictor to generate candidate precondition and effect phrases; second, these are filtered using textual similarity and contradiction, so that both preconditions and effects are distinct and consistent.

Structured Event Representation

Event verb and arguments An event occurrence e consists of a (phrasal) verb $V(e)$ and a set of labelled arguments $\mathbb{A}(e)$. Verbs are lemmatized. Events with verb lemmas "be" or "have" are termed *statements*, describing facts rather than occurrences, and we exclude them from action model generation unless an argument within the statement is an event. The SRL system follows the PropBank annotation schema (Bonial et al. 2012), categorizing argument labels into essential action valency arguments (ARG0-ARG5) like agents or patients, and verb modifiers such as purpose or location. Argument values are spans of text. "Event Occurrences" in Figure 2 shows extracted event arguments with their respective labels as colored chunks.

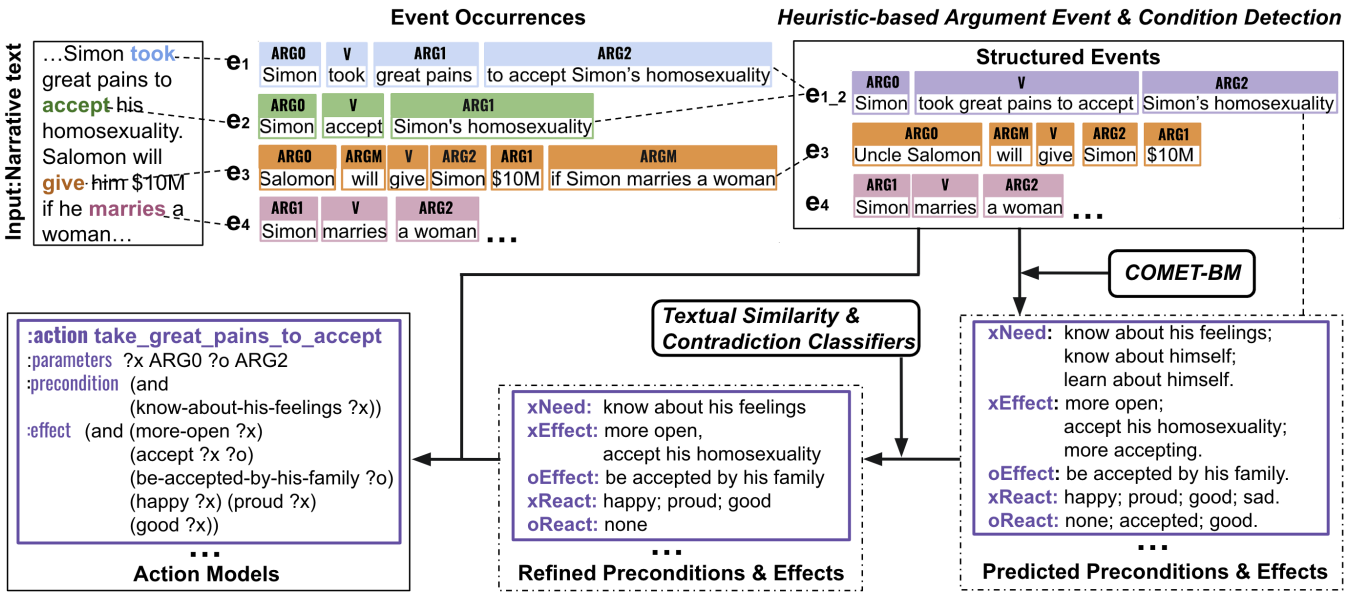

Figure 2: Overview and example of the proposed approach for automatically generating action models from narrative text. The example input is part of a plot summary for the movie "Man Is a Woman", from Bamman et al. (2013).

Entity resolution We apply a document-level inference-based LSTM model (Lee, He, and Zettlemoyer 2018) from AllenNLP to the input narrative text, and substitute the first mention of any resolved entity for later mentions. For example, in Figure 2, "his" in the input text is substituted with the referenced entity "Simon's" in the events $e1$ and $e2$.

Phrasal verb detection Phrasal verbs, prevalent in English, are crucial to identify. Their meaning often diverges from the standalone verb, as "make up" differs from "make". Additionally, a phrasal verb like "make up" can have multiple interpretations. The SRL system, however, extracts only single verbs. We apply the following rule, adapted from (Komai, Shindo, and Matsumoto 2015), to detect phrasal verbs: If a word P either (i) has a *compound part* 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[2].

Argument event detection We use the SRL argument structure, together with a rule-based method that relies on the dependency parse information, to determine which events are arguments of other events. Part or all of an extracted event, e_j, may lie within a span of text that is an argument of another event, e_i. 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. E.g., in Figure 2, the adverbial modifier of e_3 contains $V(e_4)$ ("marries"), but e_4 is not an argument of e_3.

We classify the contained event e_j as an argument of the containing event e_i if any of the following holds: (i) The dependency relation $V(e_i)$ to $V(e_j)$ is clausal complement or clausal subject. (ii) The dependency relation from $V(e_j)$ to $V(e_i)$ is copula or auxiliary. (iii) All of e_j is contained in a purpose modifier of e_i.

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 2, 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".

Condition event detection Conditional promises, threats, etc, are common in narrative text, as e_4 in Figure 2 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 variant of a method introduced by Puente et al. (2010), based on the signal words and phrases "if", "whenever", "as long as", "on [the] condition that", and "provided that". For example, in Figure 2 the signal word "if" is in between the consequence e_3 and the condition e_4. Event e_j is classified as a condition of e_i iff (a) one of the subsequences $V(e_i)\ \ S\ \ V(e_j)$ or $S\ \ V(e_j)\ \ V(e_i)$, where S is one of the signal words/phrases, appear in the sentence, with no other (non-argument) event verb in between; and (b) the tenses of the event verbs, as determined from their POS tags, match one of a list of rules. For example, one rule is that $V(e_i)$ is present simple tense and $V(e_j)$ is future simple. Again, we refer to our earlier paper for a full description (Li, Haslum, and Cui 2023).

The updated event structures after detecting argument and conditional events are shown in the box labeled with "Structured Events" in Figure 2.

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

Action Model Creation

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 the example in Figure 2. Each action has one or two parameters, x and o, representing the subject and (direct) object of the event, selected from the event arguments based on their SRL labels. We trained a commonsense event/concept relation predictor (COMET-BM) to generate candidate preconditions and effects from the event text. Candidate preconditions and effects are filtered to remove semantic duplicate and contradictory phrases, to ensure they are consistent. We also use textual entailment to infer negated effects and preconditions.

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, Chin, and Havasi 2019), introduced in the background section.

We adopt COMET (Bosselut et al. 2019), which is a GPT model (Radford et al. 2018) pre-trained on the ATOMIC and the ConceptNet knowledge graphs, and build a BART-based (Lewis et al. 2019) variation (COMET-BM) by finetuning 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). There are over 472K instances of these relations in the ATOMIC-2020 dataset. In the COMET-BM training procedure, we construct tuples $\langle I, T\rangle$ from the dataset, where I is the concatenation of e, the text describing the event, and r, the relation. T is the textual description denoting the commonsense knowledge inferred from I. Following Bhagavatula et al. (2019), we add special tokens <s> and </s> to mark the beginning and the end of I. The conditional probability of the nth token of T is defined as:

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

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; $H_{T_{[0,n-1]}}$ 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 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 with the *xNeed* relation to the event are candidate preconditions, while others are candidate effects. An action can have multiple preconditions and effects, thus COMET-BM works as a generative model: given the event e and relation r it outputs 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 $K = 6$, $\theta_{xNeed} = 0.7$, $\theta_{xEffect} = \theta_{oEffect} = 0.5$, and $\theta_{xReact} = \theta_{oReact} = 0.2$. If any of the predicted phrases is "none", all lower-probability predictions are omitted. In Figure 2, "Predicted Preconditions & Effects" shows the top predictions for each relation, given e_{1_2} as input.

Precondition and effect selection COMET-BM produces each candidate precondition and effect independently, so they may not always be semantically unique or consistent when combined. In Figure 2, 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 textual similarity predictor[3] is based on a large pre-trained model for natural language understanding and finetuned on over 1B phrase pairs. We judge two phrases to be redundant if their similarity is 0.5 or higher. The second model[4] is finetuned on over 4M sentence pairs. and ranks whether the relation between two phrases is most likely to be entailment, neutral or contradiction. Both models are selected based on 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, selected from the event's arguments based on their SRL labels. Only arguments with numbered labels ($ARG0$–$ARG5$), which represent the event's agent, patient, and so on, are considered; event modifiers can not instantiate parameters. ARG0, if present, becomes the subject x, else ARG1. Given the subject is ARG_s, we then sequentially look for ARG_i, for $i \in [s+1, 5]$, as the object, o.

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

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

[4] https://huggingface.co/cross-encoder/nli-deberta-v3-base

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).

Evaluation

Few attempts have been made to generate planning models from narrative text, and even fewer have done so in a fully automated, unsupervised way. We compare the action domain models generated by NaRuto to those generated by Hayton et al. (2017)'s StoryFramer, their 2020 system (Hayton et al. 2020) (abbreviated "H2020"), and the results reported by Huo et al. (2020). Of those, only H2020 is a fully automated system. The other two rely on a human user to make key modelling decisions. None of the systems are available to use, but the domains produced by StoryFramer for two example stories were provided by its authors, and we have attempted to replicate the results of H2020 using the StoryFramer material and the description in their paper. Huo et al. (2020) provide neither the system nor the domains generated; we compare the aspects of NaRuto that we can with data reported in their paper.

For input, we use two short stories that have appeared in work on narrative planning: the Aladdin story by Riedl and Young (2010), and the Old American West story by Ware (2014). Both are hand-written descriptions of plans generated by narrative planning systems. Hence, there exists a ground truth, in the form of planning domains used by the narrative planners. The same stories were also used for evaluation in previous work on action model learning from narrative text (Hayton et al. 2017; Huo et al. 2020).

We evaluate three different aspects of the domain models: First, we evaluate each model'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. The same comparison was also done by Hayton et al. (2017) and Huo et al. (2020). Second, we conducted a (blind) expert assessment of the internal *consistency* of each domain definition, asking planning experts to rate the appropriateness of each action's preconditions and effects, relative to the predicates defined in the domain. Third, we evaluate the *generality* of each domain model, by applying a planner to problem instances for each domain with varied initial states and recording the number and variety of plans generated.

The next section describes the comparison systems in more detail. The following three sections present the details and results of each of the three evaluations.

Comparison Systems

StoryFramer (Hayton et al. 2017) is a partially automated domain modelling tool. Given a narrative source text, it automatically extracts candidate action verbs, candidate predicates based on properties, and candidate objects (nouns). The remaining modelling 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. A user can also override or edit any system suggestion (e.g., add/remove candidate predicates, objects, etc). Hayton et al. (2017) applied StoryFramer to the Aladdin and Old American West stories. We couldn't replicate their process due to system unavailability and lack of user edit details, but the authors provided their final domain files.

Their recent system (Hayton et al. 2020) ("H2020") is automatic, but also unavailable to us. However, since its event extraction mechanism is very similar to StoryFramer, except for a new 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 the paper. The action models created by H2020 aim to mirror 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 $\langle$ action name, subject, object$\rangle$ triples, representing actions taking place. Their system also involves human users to refine the action model. Because neither the system nor its outputs are available to us, we can only compare with the results included in their paper.

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

Identification of Narrative Actions

Table 1 lists the action names extracted by StoryFramer (Hayton et al. 2017), Huo et al.'s (2020) system and NaRuto. These are compared with actions defined in the hand-written narrative planning domains from Ware's (2014) thesis (West story) and from Riedl and Young (2010) (Aladdin). Results for StoryFramer and Huo et al.'s (Huo et al. 2020) system are from the respective papers. Note that Huo et al. did not try their system on the Aladdin story.

There is no 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 (Ware 2014) Old American West story, the events "use" and "anger" are not actions in the ground truth planning domain, but are 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 "got" from "Hank got bitten by a snake".

GT	StoryFramer	Huo et al.	NaRuto
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 1: 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. Ground truth (GT) lists action names in the narrative planning domain files by Ware (2014) and Riedl & Young (2010). ✗: action is not detected; ✓̸: action partially extracted or incomplete; ∗: action not present in the ground truth domain but occurs in the story.

Moreover, StoryFramer misses the actions "summon" and "fall-in-love" in the Aladdin story, which NaRuto finds.

Expert Assessment of Action Models

A direct comparison of action models' "correctness" across generated domains is difficult, because domain models use very different sets of predictes. Instead, we asked experts to rate the appropriateness of each action's preconditions and effects, relative to the predicates that are defined in that domain model. This holistic assessment of the use of the domain model's predicates across its actions gives us a measure of the internal consistency, or soundness, of our generated domain models. This evaluation covered all models of both domains, including ground truth and models generated by StoryFramer, H2020, and NaRuto.

We recruited 9 AI planning experts familiar with PDDL domain modeling. Each expert 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). Experts knew only that the models were "automatically learned from narrative text". Each expert received the models in random order. We received N=6 responses for each domain (stories).

From each response, we determine three metrics: (1) average ratings of all preconditions and effects within each domain model; (2) percentage of ratings considered in agreement (i.e., 4 or 5); and (3) percentage of ratings in disagreement (i.e., 1 or 2), within each domain model. Metrics are averaged over the $N = 6$ responses per domain. The domain model generated by NaRuto in the evaluation is with full (global) negations, named 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 2. Box-and-whiskers plots showing the distribution of average scores across responses for both of the domains are in Figure 3. Unsurprisingly, the hand-written 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$_{(G)}$) aims to capture the ramifications of positive action effects. However, the domain model with restricted negations (NaRuto$_{(L)}$) scores consistently better, indicating that the positive effects and preconditions play a more important role than those negated ones. The domain models generated by H2020 score lower on both measures, indicating that its encoding of the sequence of events in the original story is not perceived by experienced domain modellers as appropriate for a planning domain.

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.

Alternative Plan Generation

For each generated domain model, the event sequence, extracted from the story text, constitutes a valid plan with a suitable initial state and goal. To evaluate the extent to which each domain model supports the generation of new story plans, we created new problem instances with modified initial states, and used 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

	Method	Type	Score ↑	Agreement ↑	Disagreement ↓
West	GT (Ware 2014)	Manual	4.34	82.7%	10.7%
	StoryFramer (Hayton et al. 2017)	Semi-auto	<u>3.26</u>	42.5%	<u>26.3%</u>
	H2020 (Hayton et al. 2020)	Auto	2.54	17.7%	41.2%
	NaRuto$_{(G)}$	Auto	2.98	<u>43.0%</u>	39.3%
	NaRuto$_{(L)}$	Auto	**3.57**	**60.8%**	**25.3%**
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	<u>3.34</u>	<u>52.0%</u>	<u>29.5%</u>

Table 2: The average scores over all the respondents for all actions' preconditions and effects within each domain model; and the average percentage in agreement and disagreement scores. "Type" indicates if the domain model is generated manually or by a semi-automated or (fully) automated system. Note that both manual and semi-automated systems involve human expertise in action model creation. Bold numbers indicate the best results and underline denotes the second-best results.

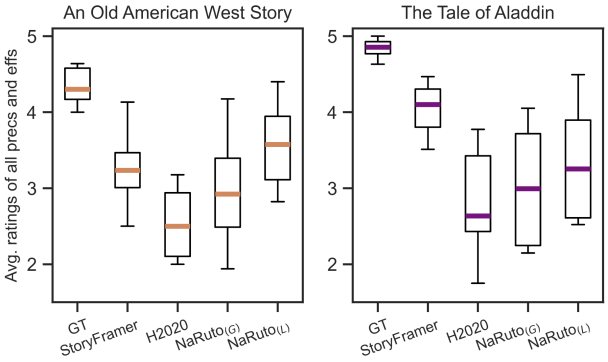

Figure 3: Distribution, over respondents, of the average scores for all actions' preconditions and effects within each domain model. The thick line shows the median, box shows the interquartile range. Whiskers extend to the full range of values.

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, keeping the goal fixed.

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 3 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.

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

Table 3: For each domain and system, the total number of tasks and how many of them are solvable, and the number of unique solvable plans.

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.

Conclusion

Narrative text is complex yet offers deep insights into events and actions. We introduced NaRuto, an autonomous system creating planning-language-style action models from narrative texts. 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 outperform those by comparable fully automated methods, and sometimes even surpass partially-automated ones with manual interventions. There are still some challenges, such as selecting appropriate negative action effects and taking story context into account when predicting preconditions/-effects, which will direct our future work.

References

Bamman, D.; O'Connor, B.; and Smith, N. A. 2013. Learning latent personas of film characters. In *Proceedings of the*

51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 352–361.

Bhagavatula, C.; Bras, R. L.; Malaviya, C.; Sakaguchi, K.; Holtzman, A.; Rashkin, H.; Downey, D.; Yih, S. W.-t.; and Choi, Y. 2019. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*.

Bonial, C.; Hwang, J.; Bonn, J.; Conger, K.; Babko-Malaya, O.; and Palmer, M. 2012. English propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder*, 48.

Bosselut, A.; Rashkin, H.; Sap, M.; Malaviya, C.; Celikyilmaz, A.; and Choi, Y. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*.

Branavan, S.; Kushman, N.; Lei, T.; and Barzilay, R. 2012. Learning high-level planning from text. In *The Association for Computational Linguistics*.

Branavan, S. R.; Chen, H.; Zettlemoyer, L.; and Barzilay, R. 2009. 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*, 82–90.

Feng, W.; Zhuo, H. H.; and Kambhampati, S. 2018. Extracting Action Sequences from Texts Based on Deep Reinforcement Learning. In *Proc. of the 27th International Joint Conference on AI (IJCAI)*, 4064–4070.

Gardner, M.; Grus, J.; Neumann, M.; Tafjord, O.; Dasigi, P.; Liu, N.; Peters, M.; Schmitz, M.; and Zettlemoyer, L. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Geffner, H.; and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool. ISBN: 9781608459698.

Gildea, D.; and Jurafsky, D. 2002. Automatic Labelling of Semantic Roles. *Computational Linguistics*, 245–288.

Hayton, T.; Porteous, J.; Ferreira, J.; Lindsay, A.; and Read, J. 2017. StoryFramer: From Input Stories to Output Planning Models. In *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling*.

Hayton, T.; Porteous, J.; Ferreira, J. F.; and Lindsay, A. 2020. Narrative Planning Model Acquisition from Text Summaries and Descriptions. In *Proc. 34th AAAI Conference on Artificial Intelligence*, 1709–1716. AAAI Press.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Huo, Y.; Tang, J.; Pan, Y.; Zeng, Y.; and Cao, L. 2020. Learning a Planning Domain Model From Natural Language Process Manuals. *IEEE Access*, 8: 143219–143232.

Hwang, J. D.; Bhagavatula, C.; Le Bras, R.; Da, J.; Sakaguchi, K.; Bosselut, A.; and Choi, Y. 2021. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 6384–6392.

Komai, M.; Shindo, H.; and Matsumoto, Y. 2015. An efficient annotation for phrasal verbs using dependency information. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation: Posters*, 125–131.

Lee, K.; He, L.; and Zettlemoyer, L. 2018. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*.

Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; Van Kleef, P.; Auer, S.; et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 167–195.

Lenat, D. B.; and Guha, R. V. 1989. *Building large knowledge-based systems: representation and inference in the Cyc project*. Addison-Wesley Longman.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Li, R.; Haslum, P.; and Cui, L. 2023. Towards Learning Action Models From Narrative Text Through Extraction and Ordering of Structured Events. In *Proc. Australasian Joint Conference on AI (AJCAI)*. https://doi.org/10.1007/978-981-99-8391-9_2.

Lin, S.; Grastien, A.; and Bercher, P. 2023. Towards Automated Modeling Assistance: An Efficient Approach for Repairing Flawed Planning Domains. In *AAAI 2023*. AAAI.

Lindsay, A.; Read, J.; Ferreira, J.; Hayton, T.; Porteous, J.; and Gregory, P. 2017. Framer: Planning Models from Natural Language Action Descriptions. In *Proc. 27th International Conference on Automated Planning and Scheduling (ICAPS)*, 434–442.

Manikonda, L.; Sohrabi, S.; Talamadupula, K.; Srivastava, B.; and Kambhampati, S. 2017. Extracting Incomplete Planning Action Models from Unstructured Social Media Data to Support Decision Making. In *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling*.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J. R.; Bethard, S.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 55–60.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. 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*.

Mei, H.; Bansal, M.; and Walter, M. R. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Miglani, S.; and Yorke-Smith, N. 2020. NLtoPDDL: One-Shot Learning of PDDL Models from Natural Language Process Manuals. In *ICAPS'20 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS'20)*.

Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2021. GPT3-to-plan: Extracting plans from text using GPT-3. *arXiv preprint arXiv:2106.07131*.

Porteous, J.; Charles, F.; and Cavazza, M. 2013. NarrativeING: using character relationships for interactive narrative generation. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 595–602.

Puente, C.; Sobrino, A.; Olivas, J. A.; and Merlo, R. 2010. Extraction, analysis and representation of imperfect conditional and causal sentences by means of a semi-automatic process. In *International conference on fuzzy systems*, 1–8. IEEE.

Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Riedl, M. O.; and Young, R. M. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39: 217–268.

Sap, M.; Le Bras, R.; Allaway, E.; Bhagavatula, C.; Lourie, N.; Rashkin, H.; Roof, B.; Smith, N. A.; and Choi, Y. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 3027–3035.

Shi, P.; and Lin, J. J. 2019. Simple BERT Models for Relation Extraction and Semantic Role Labeling. *ArXiv*, abs/1904.05255.

Sil, A.; and Yates, A. 2011. Extracting STRIPS Representations of Actions and Events. In *Recent Advances in Natural Language Processing*.

Speer, R.; Chin, J.; and Havasi, C. 2019. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Proc. AAAI*.

Ware, S. G. 2014. *A plan-based model of conflict for narrative reasoning and generation*. North Carolina State University.

Yordanova, K. 2016. From textual instructions to sensor-based recognition of user behaviour. In *Companion Publication of the 21st International Conference on Intelligent User Interfaces*, 67–73.