Towards LLM-Driven Synthesis of Narrative Planning Models

Allix Fletcher, Christian Muise

School of Computing, Queen's University
{20ampf, christian.muise}@queensu.ca

Abstract

Narrative planning supports the symbolic modelling of stories, but designing these models is logically complex and manually burdensome. To address this challenge, we introduce an integrated framework for extracting planning models from narrative texts. The pipeline iteratively synthesizes and refines planning models given narrative texts and both planning-specific and narrative-specific validation. Narrative models can provide insights into patterns across stories, and model generation automated by an LLM-powered framework makes this analysis feasible. This work demonstrates a preliminary implementation of our narrative planning model pipeline and a respective analysis as to how best to refine and improve our approach to our implementation as it matures. We found the automated extraction pipeline to have promising characteristics indicative of potential model viability in the future, given significant additional rigour throughout the model extraction and construction processes.

1 Introduction

Narrative planning models (NPMs) are automated planning models that represent a narrative as an environment and associated problem. Manually defining automated planning models is notoriously complex and time-consuming, so researchers have explored how to partially infer candidate models for further use (Miglani and Yorke-Smith 2020). Narrative planning models also serve as a particularly compelling potential use for narrative analysis. The symbolic structure of planning models, represented in Planning Domain Definition Language (PDDL), lends itself to deriving narratological insights. Modes of analysis might include assessments based on the explicit definition of events as actions in PDDL, pertaining to narrative continuity, coherence, or modes of dependence among various narrative elements. Explicit symbolic representation of narratives could also support analysis that considers complex narratological elements as implicit structures. To make this analysis feasible, NPMs would need to be readily obtainable. It follows that an automated model generation solution is desirable for acquiring models without undue manual burden and automated planning expertise.

Methods for extracting PDDL models from natural language are increasingly large language model (LLM)-centric, directly reflecting improving LLM capabilities. Consequently, designs are also becoming more autonomous. These systems typically process short natural language descriptions to produce models. NPM extraction faces a significant additional challenge of interpreting long-form narrative text as its domain description.

Using the Language-to-Plan (L2P) tool suite for planning model generation and validation with LLMs (Tantakoun, Muise, and Zhu 2025), we explore a fully autonomous iterative NPM pipeline design. The preliminary framework we investigate is an L2P implementation of NL2Plan's model extraction architecture (Gestrin, Kuhlmann, and Seipp 2024). We assess the capabilities and shortcomings of this framework as a foundational approach to extracting narrative models and experiment with narrative-driven modifications to improve its extraction capability. We found that the generated narrative models require significantly more narrative-specific nuance than what the generalized extraction approach produces, not only for model soundness but also to enable meaningful analysis. Our primary contribution is to present our preliminary exploration into using L2P-style pipelines configured specifically for converting narrative texts into narrative planning models. We further highlight which aspects work well, and which necessitate richer pipelines and techniques.

2 Related Work

2.1 Extraction of Narrative Structure

There are varied approaches to extracting narrative structure, such as using deep learning for action sequence extraction (Feng, Zhuo, and Kambhampati 2018), transforming unstructured text into story maps (Bartalesi et al. 2023), event-based narrative extraction (Norambuena, Mitra, and North 2023; Yu and Kim 2021), and implementations like text2story (Amorim et al. 2024) which is a Python package of tools for extracting narrative components from real-world narrative domains with natural language processing annotators and representing the components both logically and visually. While these approaches offer insights into computational representations and extraction processes, our approach focuses exclusively on large language models to leverage their inherent suitability for natural language tasks.

2.2 Automated Generation of Planning Models

Several works have investigated AI-based approaches for transforming natural language to planning models (Guan et al. 2023; Hayton et al. 2020; Hernandez, Sreedharan, and Kambhampati 2021; Xie et al. 2025, 2023; Oswald et al. 2024). The most relevant approaches in comparison to our work are fully automated strategies. Gestrin, Kuhlmann, and Seipp designed an LLM-driven pipeline called NL2Plan for iteratively transforming natural language to PDDL. While their system was designed for use with minimal text descriptions as input rather than paragraphs of prose, we utilize their pipeline design as the preliminary infrastructure to consider in our narrative-based setting. To that end, Tantakoun, Muise, and Zhu introduces L2P, an open-source Python library that amalgamates and implements landmark approaches for natural language to PDDL, including NL2Plan (Tantakoun, Muise, and Zhu 2025). L2P facilitates modular pipelines with autonomous capability. Our work takes an L2P-based approach to tightly coupled model generation and verification, and uses the toolkit to produce planning models from narrative texts and to perform planning-specific validation.

To address narrative-specific model generation, one of the most relevant comparisons for our work is NaRuto (Li et al. 2024). NaRuto is a fully automated system for extracting narrative planning domains from narrative texts, outperforming existing fully automated methods and matching the performance of some methods that involve human interaction. Their system works by first extracting structured representations of event occurrences from narrative text using their human-annotated Event Dependency Relation dataset, and then constructing action models by predicting preconditions and effects of extracted events using commonsense relations (Li et al. 2024). You et al. also take a narrative approach, but focus on plan generation (You et al. 2023). While this framework's purpose is to generate plans, its extraction process focuses specifically on long-form narrative coherence, and the use of novels and stories as domains for its problem directly mirrors our setting for model extraction.

3 Background

Automated planning models implemented in PDDL symbolically represent the relevant world as a domain file with boolean predicates and possible actions that both depend on and impact the state of associated predicates. Based on the model defined, an automated planner attempts to find a solution or *plan*. LLMs are a technology within the AI discipline of machine learning, but where automated planning is centred on logical reasoning, language models rely on pattern recognition. The intersection of these disciplines is referred to as neuro-symbolic AI. From the many recent advancements and fast-changing capabilities of LLM technology, more work is being done to investigate its potential uses and applications, including integration with automated planning.

L2P is a library of tools for creating domain and task representations from natural language using LLMs. L2P is domain and LLM-agnostic, allowing diverse implementations and configurations. The library provides a suite of tools for building and validating planning models.

Narratology defines principles of narrative components and structure. Within the context of narrative planning, narratology serves as a guiding set of principles for model design. Within this field, there are varied approaches to defining and assessing fundamental narrative requirements and dynamics. However, the theoretical basis supports and contextualizes the narrative-planning model's quality from a narrative perspective.

4 A Method for NPM Extraction

Our preliminary approach to implementing an NPM pipeline is to use an NL2Plan-style framework as implemented with L2P in Tantakoun, Muise, and Zhu. Gestrin, Kuhlmann, and Seipp outline six steps in their NL2Plan approach as follows: (1) Type Extraction, (2) Hierarchy Construction, (3) Action Extraction, (4) Action Construction, (5) Problem Extraction, and (6) Planning. NL2Plan also defines optional feedback-loop iterations at each step and validation of the generated PDDL. To align with L2P convention, step (5) Problem Extraction is hereafter referred to as Task Extraction.

For our experimental setup, we used `gpt-oss:120b` as the LLM powering the extraction and construction processes. This model showed stronger performance in our setting compared to other models considered, including `gpt-oss:20b` and `gemma4:26b`. The `gpt-oss` model family is noted for completion, reasoning, and thinking capabilities. Similarly, `gemma4` is described as appropriate for thinking, long context, and reasoning tasks. Consequently, these models were both considered for our pipeline. `gpt-oss:20b` was initially promising, given certain pipeline queries, but faced limitations. The most prevalent issue was reaching the configured context limit regardless of reasonable limit increases within the model's capability. `gpt-oss:120b` did not produce the same issue given the same queries and limits, and therefore was selected instead of the `20b` model. `Gemma4:26b` consistently failed to return responses in the format requested and required by L2P infrastructure. In our evaluation, the model parameters were configured as follows: *context length* = 32768, *max completion tokens* = 16384, and *temperature* = 0.0. While the NPM pipeline itself supports a range of LLMs due to L2P's embedded pre-configurations, for testing the pipeline going forward, we will continue to seek emerging models with complementary abilities to our problem space.

To evaluate the preliminary pipeline, we used 15 folktales as our natural language input. The stories were sourced from our larger JSON dataset of approximately 1,000 public-domain stories (Rehal, Muise, and Fletcher 2026). The 15 selected stories range from approximately 1,794 to 23,609 characters, with the majority in the 5k-9k range (approximately 1k-2k words). Our selection is intended to reflect a moderate diversity in topics, styles (e.g., dialogue density), and renown, to indicate potential contributing factors to the results of our generated models.

The preliminary pipeline has some augmentations motivated by the narrative input. Due to the size and ambiguity introduced by using a story as a domain description,

modifications were made to both NL2Plan-based prompting and L2P-embedded parsing to improve the extraction process. The L2P parser uses specific string markers to indicate where the model content appears within the LLM's response. Given our larger input, the response occasionally does not strictly follow these requirements. Since the markers are typically almost as intended in the response, a slight adjustment to the parsing functionality in L2P allows the relevant portions of the LLM response to be successfully extracted. This accommodates the more unruly nature of our queries and subsequent responses with more forgiving processing. We chose this solution since it was more practical to implement than attempting to reaffirm formatting requirements within our prompts.

A notable modification made to our preliminary pipeline is to override validation errors in Task Extraction. For each pipeline step, if verification is not passed after a select number of iterations, the extraction or construction will return an error, terminating the pipeline. However, as a temporary measure, we have chosen to force Task Extraction to output its generation content, regardless of any validation errors, to be able to identify and assess what errors are present. This facilitates our analysis of what challenges the preliminary pipeline faces and motivates design improvements for our approach. Additionally, it allows the pipeline to complete execution and generate the domain.pddl and problem.pddl files for further assessment.

The most significant adjustment made to the prompts provided by Tantakoun, Muise, and Zhu's NL2Plan implementation was to simplify the scope implied in Type Extraction so that fewer redundant elements were considered types. At this stage of the NPM pipeline's development, the focus of the extraction process is to capture overarching characters, items, and events in the story. Where types are concerned, the added ambiguity of focus within the story led to impractical type selection in initial testing. We found that the changes in type extraction better reflected the general requirements of the story models.

Following our preliminary results as a baseline, we further explored narrative-driven modifications to target narrative-specific extraction for select model components. As we intend to modify not only prompting going forward but also the fundamental architectural design necessitated by the narrative complexities we aim to capture, we begin our narrative-specific extraction experimentation with objects and actions. These are fundamental to the design of a narrative planning model.

5 Preliminary Evaluation

5.1 Preliminary NPM Extraction Pipeline Output

`1. Type Extraction`: The type extraction is reasonably successful and comprehensive enough to support the subsequent extraction and construction steps. The types are accurate to the contents of the stories, but beyond this, the relative appropriateness of types included and excluded depends on their implementation in the rest of the generated model.

`2. Type Hierarchy Construction`: The hierarchy construction, while syntactically successful, introduces several concerns for the model. The prompting strongly discourages any introduction of new types in the hierarchy construction unless absolutely necessary. However, there are occasional instances of new types, some of which are ill-advised. For example, the domain for *The Four Skillful Brothers* introduces a redundant type 'ship' as a child of 'vessel', notwithstanding whether or not 'vessel' was an appropriate type to begin with. The addition of 'ship' was likely motivated by the original type description of 'vessel': 'objects used for transport or construction, e.g., ship, plank', demonstrating a gap for additional mechanisms to prevent such redundancies.

Occasionally, some categorization overlaps cause conflicts. For example, in the *Hansel and Grettel* domain, the type 'home' is a child of the type 'location', whereas the witch's 'house' is a child of the type 'structure'. This causes downstream conflicts when the generated actions would have the characters travel between locations, making a plan with travel to the witch's house infeasible. Extending this issue, some conflicts are due to larger inconsistencies with the narrative content. For example, in *The Three Remarks*, aside from the types including unnecessary specificity, an overlap conflict arises from both the 'impostor_king' type and the 'butterman' type representing a single character. This induces problems in action descriptions later in the model.

`3. Action Extraction`: This step in the pipeline faced no syntactic issues, as the output is a dictionary of natural language action names and descriptions with no complex PDDL parsing requirements. However, the actions identified at this step highlight what the currently generated models fail to capture from a narrative perspective. For example, the inciting event in *Chicken Licken* is an acorn falling on Chicken Licken's head, causing him to believe the sky is falling and motivating him to travel out of the woods to inform the king. The generated list of actions includes 'travel', 'meet', and 'eat' to model several birds continuously meeting and joining Chicken Licken's expedition, until they finally meet a fox and are all eaten. Narratively, the acorn falling on Chicken Licken is highly important, but not reflected in the model. Additionally, the model reflects that Chicken Licken and his company travel, but there is no indication of the motivating intent of informing the king of imminent disaster. This form of implicit structure is an avenue we hope to target in the narrative-specific modifications and refinements made to the preliminary NPM pipeline design.

`4. Action Construction`: This step executes iterative action construction, including predicate generation as necessary. This task introduces the most flaws in the models. As part of action formalization, this step defines the parameters of each action. This created two common issues in the PDDL syntax. The first was an illegal space between a few parameter names and their preceding '?'. Secondly, for some actions, not all parameters used in the preconditions and effects had associated declarations, despite a following generated comment outlining the parameter requirements comprehensively. In rare cases, at most one halluci-

nated predicate was found in an action that did not exist in the model elsewhere. The interpretation of the actions extracted in Action Extraction creates a greater divide in narrative cohesion due to ambiguous descriptions translating to some inappropriate preconditions and effects. Furthermore, since predicates are generated as necessary by subsequent action formalization, there were occasional instances of similar predicates designed to capture equivalent concepts, resulting in an inconsistent ability to track their Boolean values. A notable error across several generated domains was the inconsistent use of the 'at' predicate. Often, the actions presumed the arguments for 'at' were an object and a respective location, but the definition would typically incorrectly specify two locations as its arguments. Given how crucial the predicates are to the models' integrity, we intend to address this generation separately in future developments of our NPM extraction pipelines.

`5. Task Extraction:` As discussed in our approach, we override L2P's task validation mechanism that would otherwise cancel further execution after the maximum number of failed attempts. Instead, we enable problem file generation after the final iteration, despite invalid results, to identify what types of issues exist in the problems. We found that the objects tended to miss unnamed supporting characters and other less prominent entities and items in the story. There were also several instances of objects not using the most specific applicable type, which invalidated actions dependent on the existence of objects of those types. In some cases, we believe this occurred due to generated validation feedback encouraging a more general type to match the generality in the provided predicates. The generated initial states tended to lack sufficient declaration of initial positions and symmetrical definition of adjacent locations. In some cases, the initial state included predicates which occur within the story rather than at the onset. The goal state lacked intermediate requirements, which in a better model would be embedded in the causal links in action effects and preconditions, but in this case, oversimplified the problem. Of greater concern is the occasional goal states that directly conflict with the narrative setting. The most pronounced example is *Chicken Licken*, where the goal state is that each bird is alive and the fox is not alive. This is directly opposite to the content of the input story and, consequently, is also infeasible based on the generated actions. This highlights the need for far more robust problem extraction.

`6. Model Generation and Execution:` The generation of domain.pddl and problem.pddl resulted in no additional formatting issues other than those pre-exisiting in Action Construction as described previously. There were too many underlying issues in the models to generate plans. Even if all syntactic issues were resolved, the models currently would either have their goals simplify to false or fail to accurately reflect the narrative requirements of the story, limiting the amount of valuable insights at this step.

5.2 Narrative-Specific Object Prompting

To begin customizing our preliminary pipeline for narrative extraction, we experimented with object-extraction prompting independent of the pipeline. The prompting was designed to capture narrative-specific categories such as characters and items necessary to the story. With this additional specificity and language modification, we found that the characters and story-relevant items were more comprehensively listed. For *The Three Little Pigs*, the original objects omitted two of the three pigs, the minor characters that interact with the pigs throughout the story, and several relevant items. These were all captured with the modified prompt.

5.3 Narrative-Specific Action Extraction

Due to the noted issues in Action Extraction - specifically when the actions fail to capture the full narrative requirements of the given story - we investigated an alternative approach. Instead of prompting for actions generically, we attempted to specify types of narrative actions and separately extract them. Our preliminary investigation includes the three action categories: dialogue, protagonist, and narration. Based on our experiments, this division facilitates comprehensive extraction far more reflective of the narrative content than generic extraction. For example, the original action dictionary for *Jack and the Beanstalk* provided a limited summary of events, omitting actions such as his interactions with the giants, whereas the tri-fold narrative-specific approach yielded a far more comprehensive list. Our initial results demonstrate that the specificity enables more targeted retrieval of narrative-specific action types that, when combined, yield a more robust collection. We suggest that improved Action Extraction will contribute to improvements in Action Construction and enable more nuanced modelling.

6 Conclusion

Our preliminary L2P-style pipeline for NPM extraction exhibits several shortcomings that directly motivate narrative-specific changes to the pipeline design. We demonstrate that initial augmentations to prompting and extraction design yield promising results, indicating improved NPM generation capability with further narrative-driven design modifications. Such designs may include a more prescriptive framework for type extraction emphasizing narrative-relevant categories including characters, items, and locations. Future pipeline iterations may also benefit from task extraction cognizant of narrative paradigms, recognizing, for example, that many folktales end where they begin with some fundamental change, or end in direct contrast to the initial setting. The improvement observed in narrative-specific Action Extraction suggests that greater modularization in the pipeline architecture can improve narrative extraction. This would further support the system's parsing and validation abilities. Future work will highlight specified narratological components to capture in the generated models, and iterative improvements will refine the NPM pipeline to produce robust models that enable rich narratological insights.

7 Acknowledgments

We wish to thank Raksha Rehal for her work in parsing our 'short-stories' dataset. We also wish to thank Marcus

Tantakoun for his contributions in providing L2P and associated support. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) Catalyst Grant.

References

Amorim, E.; Campos, R.; Jorge, A.; Mota, P.; and Almeida, R. 2024. text2story: A Python Toolkit to Extract and Visualize Story Components of Narrative Text. In Calzolari, N.; Kan, M.-Y.; Hoste, V.; Lenci, A.; Sakti, S.; and Xue, N., eds., *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 15761–15772. ELRA and ICCL.

Bartalesi, V.; Coro, G.; Lenzi, E.; Pagano, P.; and Pratelli, N. 2023. From unstructured texts to semantic story maps. *International Journal of Digital Earth*, 16(1): 234–250.

Feng, W.; Zhuo, H. H.; and Kambhampati, S. 2018. Extracting Action Sequences from Texts Based on Deep Reinforcement Learning. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 4064–4070. ijcai.org.

Gestrin, E.; Kuhlmann, M.; and Seipp, J. 2024. NL2Plan: Robust LLM-Driven Planning from Minimal Text Descriptions. *CoRR*, abs/2405.04215.

Guan, L.; Valmeekam, K.; Sreedharan, S.; and Kambhampati, S. 2023. Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Hayton, T.; Porteous, J.; Ferreira, J.; and Lindsay, A. 2020. Narrative Planning Model Acquisition from Text Summaries and Descriptions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(2): 1709–1716.

Hernandez, A. O.; Sreedharan, S.; and Kambhampati, S. 2021. GPT3-to-plan: Extracting plans from text using GPT-3. *CoRR*, abs/2106.07131.

Li, R.; Cui, L.; Lin, S.; and Haslum, P. 2024. NaRuto: Automatically Acquiring Planning Models from Narrative Texts. In Wooldridge, M. J.; Dy, J. G.; and Natarajan, S., eds., *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*, 20194–20202. AAAI Press.

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

Norambuena, B. F. K.; Mitra, T.; and North, C. 2023. A Survey on Event-Based News Narrative Extraction. *ACM Computing Surveys*, 55(14s): 300:1–300:39.

Oswald, J. T.; Srinivas, K.; Kokel, H.; Lee, J.; Katz, M.; and Sohrabi, S. 2024. Large Language Models as Planning Domain Generators. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling*, 423–431. AAAI Press.

Rehal, R.; Muise, C.; and Fletcher, A. 2026. short-stories (Revision 57c3600).

Tantakoun, M.; Muise, C.; and Zhu, X. 2025. L2P: A Python Toolkit for Automated PDDL Model Generation with Large Language Models. In *Bridging Planning and Reasoning in Natural Languages with Foundational Models Workshop*.

Xie, K.; Yang, I.; Gunerli, J.; and Riedl, M. O. 2025. Making Large Language Models into World Models with Precondition and Effect Knowledge. In Rambow, O.; Wanner, L.; Apidianaki, M.; Al-Khalifa, H.; Eugenio, B. D.; and Schockaert, S., eds., *Proceedings of the 31st International Conference on Computational Linguistics*, 7532–7545. Association for Computational Linguistics.

Xie, Y.; Yu, C.; Zhu, T.; Bai, J.; Gong, Z.; and Soh, H. 2023. Translating Natural Language to Planning Goals with Large-Language Models. *CoRR*, abs/2302.05128.

You, W.; Wu, W.; Liang, Y.; Mao, S.; Wu, C.; Cao, M.; Cai, Y.; Guo, Y.; Xia, Y.; Wei, F.; and Duan, N. 2023. EIPE-text: Evaluation-Guided Iterative Plan Extraction for Long-Form Narrative Text Generation. *CoRR*, abs/2310.08185.

Yu, H.; and Kim, M. 2021. Automatic Event Extraction method for Analyzing Text Narrative Structure. In Lee, S.; Choo, H.; and Ismail, R., eds., *15th International Conference on Ubiquitous Information Management and Communication*, 1–4. IEEE.

A Preliminary Pipeline Output vs. Narrative-Specific Prompting Examples

Object Extraction: *The Three Little Pigs*

Objects Generated in Task Extraction:

```
pig3 - pig
wolf - wolf
house1 - house
house2 - house
house3 - house
field1 - field
orchard1 - orchard
hill1 - location
fair1 - fair
butter_churn1 - butter_churn
pot1 - pot
straw1 - material
furze1 - material
bricks1 - material
mortar1 - material
water1 - water
loc_house1 - location
```

Objects Generated with Narrative-Specific Prompting:

```
mother_pig - pig
pig1 - pig
pig2 - pig
pig3 - pig
wolf - wolf
straw_man - man
furze_man - man
brick_man - man
farmer - man
squire - man
straw_bundle - material
furze_bundle - material
bricks - material
mortar - material
trowel - tool
stick - tool
turnip_load - food
apple - food
butter_churn - container
pot_of_water - container
fire - fire
chimney - location
straw_house - house
furze_house - house
brick_house - house
oak_forest - location
turnip_field - location
farmer_house - location
squire_orchard - location
hill - location
fair - location
```

Action Extraction: *Jack and the Beanstalk*

Natural Language Actions Generated in Action Extraction:

```
{
    'move_human': 'Human walks from one location to an adjacent location.',
    'move_animal': 'Animal moves from one location to an adjacent location; ...
    'ride_animal': "Human mounts and rides an animal, moving together to ...
    'trade_cow_for_beans': 'Exchange a cow for magical beans between agents.',
    'plant_beans': 'Human plants beans in a garden, creating a bean seed.',
    'grow_beanstalk': 'Overnight growth of planted beans into a beanstalk ...
    'climb_beanstalk_up': 'Human climbs from ground to the top of the ...
    'climb_beanstalk_down': 'Human climbs from the top of the beanstalk ...
    'hide_in_oven': 'Human hides inside an oven within a building, ...
    'take_treasure': 'Human picks up a treasure item from the current ...
    'cut_beanstalk': 'Human uses an axe or chopper to cut the beanstalk.',
    'enter_building': 'Human moves from outside to the interior of a ...
    'exit_building': 'Human leaves a building to the adjacent outside ...
    'give_food': 'One agent gives a food item to another at the same ...
    'eat_food': 'Human consumes a food item, removing it from inventory ...
    'sell_item': 'Human sells an item at the market, removing it from ...
    'lay_golden_egg': 'Magical hen lays a golden egg, creating a treasure ...
    'sing_harp': 'Golden harp sings, causing the giant to fall asleep.'
}
```

Natural Language Actions Generated by Narrative-Specific Prompting

```
{
    'dialogue_greet': 'A character says a greeting (e.g., "Good morning") ...
    'dialogue_ask_location': "A character asks another character where ...
    'dialogue_offer_exchange': 'A character proposes an exchange of items ...
    'dialogue_accept_exchange': 'A character accepts the proposed exchange. ...
    'dialogue_command_sleep': 'A character orders another to go to bed. ...
    'dialogue_request_breakfast': 'A character politely asks another for ...
    'dialogue_warn_danger': 'A character warns another of imminent danger ...
    'dialogue_inform_secret': "A character reveals hidden information (e.g. ...
    'dialogue_threaten': 'A character threatens another with harm (e.g., ...
    'dialogue_deny_mistake': "A character denies an accusation or corrects ...
    'dialogue_command_hen_lay': 'A character commands a hen to lay an egg. ...
    'dialogue_command_harp_sing': 'A character commands a harp to sing. ...
    'dialogue_call_for_axe': 'A character shouts for a specific tool (e.g., ...
    'dialogue_ask_for_item': 'A character asks another to bring a specific ...
    'dialogue_ask_for_moneybags': 'A character requests money bags from ...
    'dialogue_polite_request': 'A character makes a courteous request ...
    'dialogue_exclaim': 'A character expresses a strong emotion verbally ...
    'dialogue_tell_story': 'A character narrates an event or information to ...
},
{
    'protagonist_ride_cow_to_market': 'Jack mounts the cow Milky White and ...
    'protagonist_trade_cow_for_beans': 'Jack exchanges the cow with the ...
    'protagonist_throw_beans_out_window': 'Jack tosses the beans out of ...
    'protagonist_climb_beanstalk': 'Jack climbs the beanstalk that grew ...
    'protagonist_ask_giant_wife_for_breakfast': "Jack requests a meal from ...
    'protagonist_hide_in_kettle': "Jack slips into the giant's kettle (a ...
    'protagonist_steal_gold_bags': "Jack quietly takes the two bags of gold ...
    'protagonist_escape_down_beanstalk': 'Jack descends the beanstalk ...
    'protagonist_steal_brown_hen': 'Jack grabs the little brown hen that ...
    'protagonist_steal_harp': "Jack seizes the golden harp from the ...
    'protagonist_call_mother_for_axe': 'Jack shouts to his mother to bring ...
    'protagonist_cut_beanstalk': 'Jack uses the axe to chop the beanstalk ...
},
```

```
{
    'narration_cow_gives_milk': 'The cow Milky White provides milk each ...
    'narration_cow_stops_milk': 'One day the cow fails to give any milk, ...
    'narration_butcher_offers_beans': "The butcher pulls out five curious ...
    'narration_widow_throws_beans': 'Angry at Jack, the widow throws the ...
    'narration_beans_grow_beanstalk': 'During the night the beans sprout ...
    'narration_fairy_appears': "A beautiful maiden appears beside Jack, ...
    'narration_fairy_disappears': 'After delivering her message, the fairy ...
    'narration_giant_wife_gives_breakfast': "The giant's wife, despite her ...
    'narration_giant_roars_and_smells_blood': 'The giant bursts into the ...
    'narration_giant_eats_ox': 'Mistaking the scent, the giant devours the ...
    'narration_giant_wife_brings_moneybags': "The giant's wife brings two ...
    'narration_giant_falls_asleep': 'After counting his treasure, the giant ...
    'narration_giant_dog_barks': "The giant's dog barks loudly as Jack ...
    'narration_giant_wife_offers_breakfast_again': "On Jack's second ...
    'narration_giant_eats_roasted_bullock': 'The giant, believing Jack is ...
    'narration_giant_requests_hen': 'The giant commands his wife to bring ...
    'narration_hen_lays_golden_egg': 'The hen immediately lays a golden ...
    'narration_giant_sleeps_and_wakes': 'The giant falls asleep after ...
    'narration_giant_chases_jack': 'The giant grabs his oak-tree club and ...
    'narration_widow_brings_axe': "Jack's mother hurries out with an axe ...
    'narration_giant_falls_and_crashes': 'The giant plummets to the ground ...
    'narration_hen_lays_many_golden_eggs': 'The hen continues to lay ...
}
```