Narrative Fragment Creation: An Approach for Learning Narrative Knowledge

Chris Cervantes CCERVAN2@ILLINOIS.EDU
Wai-Tat Fu WFU@ILLINOIS.EDU
Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA

Abstract

Storytelling is an integral part of the human experience, and understanding how stories - or narratives - are generated can offer insight into their importance. Current research focuses on the introduction of narrative knowledge into the generation process in order to facilitate the creation of qualitatively good narratives. Narrative generation and narrative knowledge, however, are two sides of the same coin; as knowledge about narrative events is necessary for generation, so is the ability to generate narratives indispensable to understanding the events therein. We propose the narrative fragment - a construct intended to capture narrative knowledge - and a method for automatically creating these fragments with narrative generation through partial order planning and analysis through n-gram modeling. The generated plans establish causal and temporal relationships, and by modeling those relationships and creating fragments, our system learns narrative knowledge.

1. Introduction

Storytelling, or narrative, is a prevalent part of the human experience. Though perhaps most easily identified as a device for entertainment, narrative is much more integral to intelligence, serving as a cognitive tool used to engage with and understand the world (Polkinghorne, 1995; Riedl & Young, 2010). Narrative can be studied in myriad ways, but of particular interest is the narrative generation process. Understanding how narratives are created is tantamount to understanding narrative itself. Generation and knowledge of the narrative space are two sides of the same coin; as knowledge about the narrative events is necessary for generation, so is the ability to generate narratives indispensable to understanding the events therein.

Narrative can be defined as a kind of construct that brings together events as a goal-driven process (Polkinghorne, 1995). This offers two key insights: that narratives can be thought of as sequences of events, and that the act of sequencing – the generation task – is driven by goals. This goal-driven process is similar to planning. Narratives are often computationally represented as plans with planners acting as narrative generation mechanisms (Porteous & Cavazza, 2009; Porteous, Cavazza, & Charles, 2010; Riedl 2008; Riedl & León, 2008; Riedl & Young, 2010). In the planning domain, narrative events can be seen as actions: constructs with arguments, preconditions, and effects.

When used in this way, however, planners have limitations. Partial order planners add operators such that the shortest path to a goal state is found that satisfies all preconditions, but

qualitatively good narratives may contain sequences that are not the shortest path (Porteous & Cavazza, 2009). Humans produce this complexity by understanding relationships between events. This understanding about events is narrative knowledge: particularly, which events are related to which others and in what way. Humans gain this knowledge through a lifetime of experiences, but a simple planner has none of this intuitive knowledge about likely event relationships. Were a human and a planner given the same events, the human would draw on specialized knowledge about event relationships that would be unavailable to the planner.

One way this deficiency has been ameliorated is through partial plans, or structures that capture non-obvious relationships (Riedl 2008; Riedl & León, 2008). Since the planner lacks narrative knowledge, the idea behind partial plans is to give the planner sequences that can be used to mimic the kind of narrative knowledge a human has access to. Creating these partial plans, however, is a costly process: each must be hand-authored (Riedl 2008).

To help to reduce the cost of obtaining narrative knowledge, we propose the narrative fragment: a partially ordered set of actions where action selection and ordering are based on a probabilistic model of causal and temporal relationships between actions. We argue these causal and temporal relationships help us understand narrative knowledge. The basic intuition behind creating fragments is to simulate the kind of knowledge humans use in narrative generation. In order to do so, we attempt to create all possible plans from a vocabulary of actions and use statistical methods to approximate knowledge about how those actions relate to one another. We argue that the distributions of causal and temporal links between actions is narrative knowledge. This knowledge – initially latent in the vocabulary – is instantiated by the planner and captured by fragments. By describing the latent relationships between actions, fragments approximate knowledge about the way events relate to one another.

Primarily, narrative fragments highlight which events co-occur and in what ways. Ideally, fragments can be applied to narrative generation in the interactive narrative (IN) domain. Sets of events may be authored by different people with different purposes in mind, and fragments provide the narrative generation mechanism insight into how those disparate sets might intersect.

The goal of this work is to learn narrative knowledge by providing a process for automatically creating narrative fragments from a vocabulary of actions. These fragments will describe action relationships and in so doing approximate latent event relationships, thereby capturing narrative knowledge. We argue that a fragment is successful if it helps to demonstrate at least one of four things: unintended action sequences, action similarity, action hindsight, or action prediction.

2. Related Work

The idea of capturing knowledge is neither new nor limited to a particular area, but of special focus here is where knowledge capturing constructs are used for events, as is the case with both IN – where the narrative generation process suffers from a lack of knowledge – and natural language processing (NLP) – where knowing how events relate to one another helps with rich text understanding. These areas offer different approaches to a fundamentally similar problem: how best to encapsulate knowledge.

A key focus in IN is narrative generation, and a prominent issue involves handling the lack of knowledge possessed by narrative generation mechanisms (planners). One method for dealing with this issue is through partial plans called vignettes. A vignette can be thought of as a story

segment stored as a partial plan (Riedl 2008; Riedl & León, 2008). Vignettes are meant to serve as representations of qualitatively good story situations and they encapsulate the specialized knowledge needed in the narrative generation process because they can be spliced into a story (Riedl & León, 2008). However, vignettes must be hand-authored and this makes the introduction of knowledge into the generation process costly (Riedl, 2008).

In NLP, a similar knowledge encapsulation construct exists at the other end of the cost spectrum. Narrative chains - partially ordered sequences of events – are learned from unlabeled newswire text and have potential uses in inference, question answering, and coherence in generation (Chambers & Jurafsky, 2008). Automatic learning seems implicitly favorable to hand-authoring, but narrative chains and vignettes perform very different functions. Where the former looks at likely event sequences that can be temporally ordered and might share causal relationships (like "joined" occurs before "served" which occurs before "resigned"), the latter attempts to capture prototypical event sequences describing a common scene (like a weaker character wounding a stronger character before being killed) (Chambers & Jurafsky, 2008; Riedl, 2008). Though their purpose and structure is different, the key is that their conceptual underpinning is the same: they encapsulate knowledge about their respective domains.

Our system attempts to perform the narrative chain task – automatic learning of event sequences – in IN space so as to provide another vignette-like mechanism (narrative fragments) in that space. This is not to say that fragments are meant to replicate the qualitatively good sequences represented in vignettes. Where vignettes contain event sequences describing a bank robbery or betrayal, fragments contain sequences describing relationships like when character asks for help, they might be doing so because they've been injured; or when one character flees from another and is pursued, the first character is likely in danger. These kinds of relationships are far less sophisticated than those found in vignettes, and as constructs they are more limited in the knowledge they encapsulate.

Since they are learned automatically, narrative fragments are conceptually close to macros: a group of actions that are combined to function like a single action. Macros are intended to reduce a planner's search space, because while their inclusion into a planner doesn't impact the reachability of a solution, macros ideally reduce the time necessary to find a solution (Newton et al., 2007). In the best case, fragments can serve as macros for narrative generation. Where vignettes capture human knowledge and add it to system, fragments approximate knowledge from the event space: describing but not adding to it.

3. Narrative Fragment Generation

The narrative fragment generation approach relies on the core assumption that while the relationships between events in a single narrative are known, the relationships between events across narratives – narrative knowledge – is difficult to capture. Understanding those relationships has practical implications for IN, where a narrative generation system intended to allow for maximal expression of user agency may require combining disparate action sets in order to produce a large action vocabulary. This approach seeks to highlight those interrelationships through narrative fragments.

Narrative fragments capture narrative knowledge by learning the latent structure between actions as an approximation for the relationships between events. We define narrative as a sequence of causally and temporally related events, but in an actual story those links can be

subtle. As a result, a more natural way to conceptualize a narrative is simply as a sequence of events. When viewed in this way, narratives can be converted to sets of actions, or constructs with arguments, precondition predicates, and effect predicates. This process allows us to make a narrative usable for a planner. Each narrative is thus converted to an action set, and action sets are merged into a single cross-narrative collection of actions, or action vocabulary.

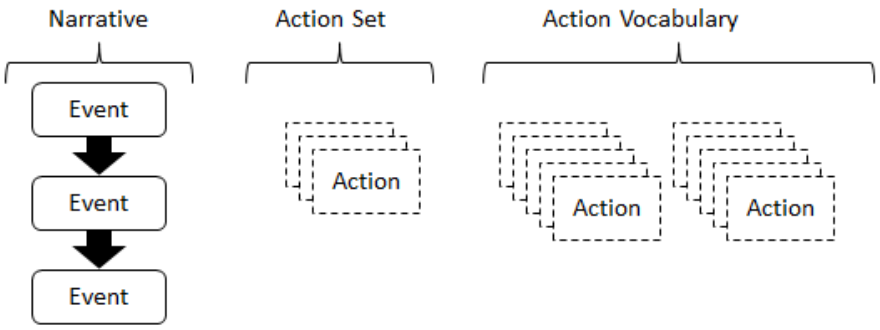

Figure 1. Narrative shown as a sequence of events, an action set shown as a collection of actions, and the action vocabulary shown as a larger collection of actions created by merging action sets.

Merging action sets into a single vocabulary is conceptually rooted in the notion that the predicates in the actions of a set are not mutually exclusive: an action from one set may contain a precondition predicate that satisfies an effect predicate contained by an action from another set.

Narrative events can be difficult to identify and convert to actions because the process requires some insight into the narrative as a whole. In the fable *The Hawk, the Kite, and the Pigeons*, pigeons – afraid of a kite – invite a hawk into their cote to protect them, and the hawk, once invited, kills many more pigeons than the kite could have (Aesop trans. Townsend). In the action set conversation process, the role of the pigeons is reduced to a single character. The action corresponding to the last event in the narrative – the hawk killing the pigeon – is predicated by the pigeon being alive and the hawk being near the pigeon (having been asked to help the pigeon: an earlier action). The effect of the action is that the pigeon is no longer alive.

During the operation of the planner, actions from the vocabulary are instantiated as operators. If an effect from an operator can be used to satisfy a precondition in the plan, the operator is applied to the plan as a node. Plans are sequences of causally and temporally related nodes from a start state to an end state.

In the plan, causal links are directed links that describe a node relationship where an effect predicate of one node satisfies the precondition predicate of another. Temporal links are only added to constrain which nodes must have occurred in what sequence in order to allow for the causal links. Only complete plans are considered to be successful generation attempts, where a complete plan is one where all preconditions are satisfied.

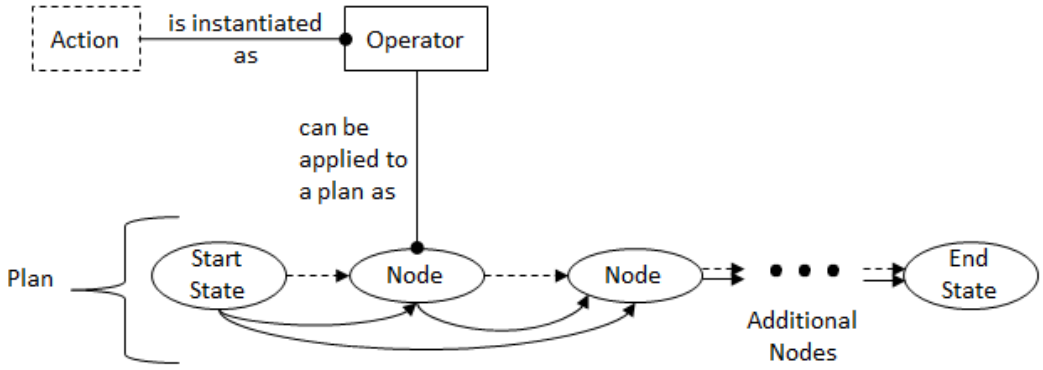

Figure 2. An instantiated action is an operator. If an operator can satisfy a currently unsatisfied precondition in the plan, it can be applied to the plan as a node. The plan shown has a start state, an end state, nodes, causal links (solid) and temporal links (dashed).

By generating plans over the complete action vocabulary, we introduce narrative knowledge that is latent in the cross-narrative space. Given only one action set, the planner will always order the nodes such that the originating narrative's event sequence is recreated. With the complete vocabulary, however, we assume novel sequences will be generated because the actions will have unintended or non-obvious relationships across sets. The role of fragments is to expose these relationships.

The planner instantiates this latent narrative knowledge by creating a collection of plans, or plan library, from the action vocabulary. Some actions tend to relate to certain others, and this becomes apparent through the causal and temporal links created by the planner over the generation of a large number of plans. Simple analysis of how the predicates align – how similar one action is to another – does not provide information about how the actions are actually related in the narrative generation context, and this is what motivates plan library creation. By creating plans from the action vocabulary, we see action relationships as determined by their use in the goal-driven narrative generation process.

N-gram modeling is then used to analyze the relationships in the plan library. N-gram models are traditionally used in NLP to model language, and the task here is analogically similar. Nodes can be seen as words, plans as sentences, and the plan library as a document. One apparent distinction between the two is that while sentences in a document are coherently ordered, plans in the library are not. In the simplest of NLP n-gram modeling techniques, though, documents are treated as unordered collections of sentences; only intra-sentence word order informs the model. Thus the real distinction between the two approaches is that while words have one sequence within a sentence, nodes have two sequences within a plan: causal and temporal.

In practice, these two orderings must be handled separately. A bigram model is created for causal relationships and another for their temporal relationships.

$$p_c(x_i|x_{i-1}) = \frac{\mathrm{C}_c(x_{i-1}, x_i)}{\mathrm{C}(x_{i-1})} \qquad p_t(x_i|x_{i-1}) = \frac{\mathrm{C}_t(x_{i-1}, x_i)}{\mathrm{C}(x_{i-1})}$$

Equation 1. The probability of x_i (the i^{th} node in the plan), given x_{i-1} (the i-1^{th} node in the plan), where $\mathrm{C}(x_{i-1})$ is the number of instances x_{i-1} has appeared in the plan library and $\mathrm{C}(x_{i-1}, x_i)$ is the number of instances in the plan library where x_{i-1} has had a link to x_i. The equation can be used to compute the causal model p_c using the count of causal links $\mathrm{C}_c(x_{i-1}, x_i)$, or the temporal model p_t by using the count of temporal links $\mathrm{C}_t(x_{i-1}, x_i)$.

The causal and temporal bigram models p_c and p_t are computed as shown in Equation 1. The two models are then combined using product of experts, shown in Equation 2.

$$p(x_i|x_{i-1}) = \frac{p_c(x_i|x_{i-1}) * p_t(x_i|x_{i-1})}{\sum_{x'}(p_c(x'|x_{i-1}) * p_t(x'|x_{i-1}))}$$

Equation 2. The combined bigram model, where x_i is the i^{th} node in the plan, p_c is the causal model, and p_t is the temporal model.

Product of experts allows us to combine the two separate models of the same data – causal and temporal – into a single framework by multiplying the models together and normalizing (Hinton 1999). The idea here is to strongly penalize the likelihood of actions appearing together if they are not directly joined by both causal and temporal links. This scenario can occur because the plans are minimally constrained. A causal link infers a temporal relationship, but because temporal links are only added when necessary for the plan to remain coherent, the relationship may be indirect (the plan in Figure 2 shows an example of this scenario). Product of experts, by harshly penalizing relationships deemed unlikely by either model, allows us to focus on those actions with direct relationships.

With a single bigram model, narrative fragments of variable length are generated such that the likelihood of the fragment is above a threshold. Fragment likelihood is calculated using the equation in Equation 3.

$$L = \prod_i p(x_i|x_{i-1})$$

Equation 3. The fragment likelihood (L), where x_i is the i^{th} action in the fragment.

Likelihood calculation begins at the second action. This is an artifact of our approach; start actions are selected randomly to serve as start states during plan creation, so while initial probabilities can be calculated, they would only describe an ideally even distribution.

Narrative fragments are intended to occupy a similar conceptual space as partial plans, but discussing them in this way has limitations. Where the nodes in a plan are ordered by directed causal and temporal links, fragments have only one type of link which conceptually captures both

causal and temporal relationships. Further, fragments learn relationships from plans. During training, the relationships are learned between nodes, but by treating all nodes from a given action identically, the model generalizes the relationships as being between actions. Fragments show the likely inter-action relationships.

Figure 3. A simple fragment composed of a sequence of three actions.

As Figure 1 and Figure 3 show, there is a high-level similarity between narratives and fragments, and this is intended. Human storytellers generate narratives not from the possible ordering of events, but by using likely orderings based on their narrative knowledge. Fragments highlight these likely action sequences by analysis of all possible sequences and in so doing approximate narrative knowledge.

4. Procedure

4.1 Data

The action vocabulary is created by hand using *Aesop's Fables* (Aesop trans. Townsend), where each action set was converted from a single fable. Fables as a type of narrative were chosen because they are short, have a limited number of characters, and can easily be described as a sequence of events. These qualities make them ideal for conversion to sets, but doing so flattens the narrative to only contain those events that can be clearly expressed as actions with preconditions and effects. Information about character communication, motivation, and intent is either incorporated into predicate creation or omitted. In this way we hope to minimize the human influence in the action creation process.

Actions are only created from those fables that are suitable for the planning task. Suitability is not entirely straightforward, but always centers around the fable's ability to be recreated as a sequence of events. In some cases, the sequence of events present in a fable was deemed too complex to allow for straightforward action creation, and these fables were not included in our data set. While it could be argued that this is a limitation in our approach, given arbitrary additional actions – even those collected from complex event sequences – the model would behave identically so long as those actions shared some predicates with other actions in the vocabulary.

While fable events easily correlate with actions conceptually, inferring preconditions and effects is a more complex task. Most often, actions with straightforward preconditions and effects are populated first and the creation process works backward. The action shown in Figure 4, for example, requires the char1 to start alive and end up not alive. Additionally, char3 must be near char1. Depending on the context of the originating fable, this precondition might require another action to have taken place with its own preconditions and effects (of which *isNear(char3, char1)=true* might be one of the effect predicates).

action(kill(char3, char1)
preconditions:
isNear(char3,char1)=true
isAlive(char1)=true
effects:
isAlive(char1)=false **)**

Figure 4. A sample action created from a fable event. In it, char3 kills char1, where both characters are action arguments. The precondition predicates specifying that char3 is near char1 and that char1 is alive must be satisfied in order for the kill action to occur and establish the effect.

In practice, predicate creation is a process similar to planning. It is common to consider, if an action has a particular precondition, what might be necessary for the action to transpire and chain this reasoning backward. Action creation occurred in this manner for the complete vocabulary.

4.2 Planner

The planner used to create the plan library follows the usual partial order planning algorithm, with minor variations.

1. Randomly select a start action (a_s) and an end action (a_e) to serve as the start and end states such that a_s does not equal a_e, and a_e has preconditions that cannot be satisfied by a_s alone. Remove the preconditions from a_s. Instantiate a_s and a_e as nodes. They form the current plan.
2. Determine if *G* is empty, where *G* contains all currently unsatisfied preconditions in the plan. If so, the current plan is complete. If not, continue to step 3.
3. Randomly select subgoal, *g*, where $g \in G$
4. Select the next operator *o*, where $o \in O$ and *O* is the operator queue (explained below)
5. Attempt to solve *g* with *o*. If *g* can be solved with *o*, add *o* to the plan and recurse to step 2. If not, return to step 4.
6. If each element in *G* has not been tried, return to step 3.
7. If no subgoals could be solved, no plan could be found using a_s and a_e at this level. Backtrack.

The operator queue, *O*, is composed first of all nodes currently in the plan, then those operators in the action vocabulary that are not currently in the plan. After these, all operators are considered where argument order is ignored. For example, a plan might have the open precondition *fears(char1, char2)=true*. *O* may contain an operator with effect predicate *fears(char2, char1)=true*. Under normal circumstances, this effect predicate could not satisfy the precondition predicate because of argument order. The design of the operator queue, however, allows for an operator with this effect predicate to be selected, so long as all normally-constrained operators have already been tried.

A plan length threshold is used to discourage arbitrarily long plans. When a plan has grown beyond the threshold, the plan attempt is deemed a failure. This is intended to prevent the case where adding operators to the plan does not bring the plan closer to completion, as would be the

case when solving one precondition only opens another that cannot be satisfied without opening yet another.

4.3 Narrative Fragment Generation

Action relationships are modeled with bigrams, using the methods described in section 3. During bigram counting, Laplacian (add-one) smoothing is used because while product of experts is used to penalize those actions that have direct causal but indirect temporal relationships this penalty is too strict in practice; many related actions are often indirectly temporally related, given the minimal temporal constraints imposed by the planning mechanism. As a result, smoothing is needed to allow for the combination of models.

Once a single bigram model is created, fragments are generated according to the likelihood calculation in section 3 and the following algorithm.

1. Place all actions in the vocabulary as single-action fragments. Each of these new fragments is considered to be incomplete.
2. Choose an incomplete fragment, *f*, where *f* is of length *i*. Remove *f* from the list of incomplete fragments.
3. Create *f* ' by adding the most likely next action (based on the action at position *i*) to a copy of *f*.
4. If the likelihood of *f* ' is still above the fragment likelihood threshold (*L*) store it as a new incomplete fragment.
5. Repeat step 3 until all actions have been tried.
6. If no new incomplete fragments (*f* ' instances) were added, *f* is marked as complete.
7. Repeat step 2 until there are no longer incomplete fragments in the list.

Because of step 1, above, actions without likely relationships with any other actions will technically be stored as complete singleton fragments. Such fragments are pruned after fragment generation is complete.

5. Results

The goal of this study was to automatically learn narrative knowledge as defined by the events present across fables. To this end, the data used for this experiment consisted of 114 actions spread over 18 sets, with about 3 (precondition and effect) predicates per action.

In order to make the strongest claim about the validity of the extracted relationships, the intent was to analyze all possible plans that could be created from the action vocabulary. This is possible using a smaller vocabulary. A saturation point can be reached where no more plans will be created regardless of how many more attempts are made. Finding this point using the full vocabulary, however, proved to be untenable given a reasonable run time.

From this vocabulary, $7x10^5$ plans were generated, about half of which were unique. This was determined to be an acceptable approximation of an exhaustive plan library, given no saturation point could be found. During the bigram model construction, plans that were generated more than once had their relationships weighed appropriately to the number of times the plan appeared in the plan library. The intuition behind doing so is that while a planner produces plans

systematically, some plans as units may be more likely than others, given an action vocabulary and a stochastic planner. By including these into the counts, we account for this plan likelihood.

5.1 Plan Library Composition

Each plan in the plan library contained on average 5.65 actions originating from, on average, 4.09 action sets.

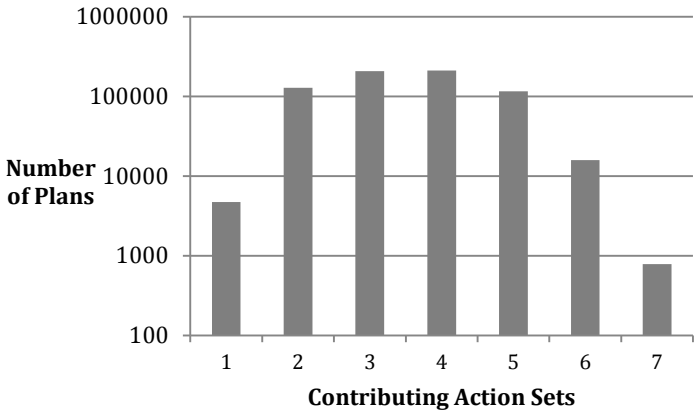

Figure 5. The number of action sets contributing actions to plans in the library. Plans of any length with actions all drawn from the same set have 1 contributing set. All available data is shown.

It's worth reiterating that the plans in the plan library are complete, suggesting that rather than generating a transition matrix - where action relationships are determined based solely on how well their predicates align - the relationships are learned in a larger context. Plans don't just connect actions, but do so in order to solve a goal.

5.2 Fragment Library Composition

The fragment library is composed of 66 fragments, generated using the approach outlined in section 4.3. In response to the way the threshold discourages long fragments, fragments are more likely to have fewer actions and fragments with a length greater than 1 are always at least 3 actions long.

More often than not, fragments contain actions drawn from multiple sets. The average number of contributing sets (2.1) is smaller than the average fragment length (3.7), suggesting that in the usual case fragments contain more than one action from a given set, though actions from more than one set, actions transition to actions from the same set 57% of the time.

In order to create the fragment library, a likelihood threshold of -2 was used. This threshold was found experimentally, and was intended to balance between generating a few strongly related fragments (12 fragments with a threshold of -1) and many weakly related fragments (176 with a threshold of -3).

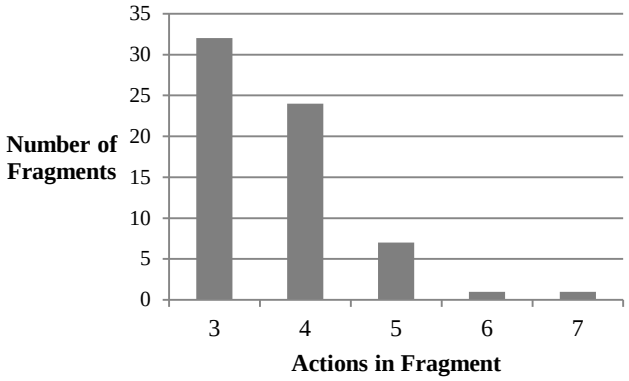

Figure 6. Fragment length. All available data is shown.

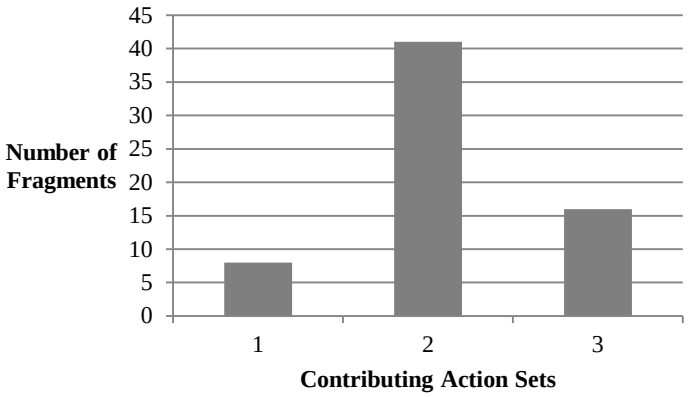

Figure 7. The number of action sets contributing actions to fragments. Fragments of any length with actions all drawn from the same set have 1 contributing set. All available data is shown.

Figure 8 shows both the fragments actually considered in this study – those that had likelihoods above -2 – and those that would be generated had the threshold been lowered to -3. As the data suggests, the overall likelihood tends to decrease as the number of actions that share an action set. This is why those fragments with likelihoods greater than -1 have, on average, 2.5 average actions per contributing set, while the fragments with likelihoods greater than -3 have, on average, 1.8 actions per contributing set. A fragment with actions from only one set tends to have a higher likelihood than a fragment with actions from different sets.

This trend is problematic, though, because fragments with actions from a single set tend to be partial reconstructions of their originating fable and are thus uninteresting. The best fragments are those that have actions from multiple sets yet remain plausible. Likelihood gives us a mechanism to filter these types of fragments out.

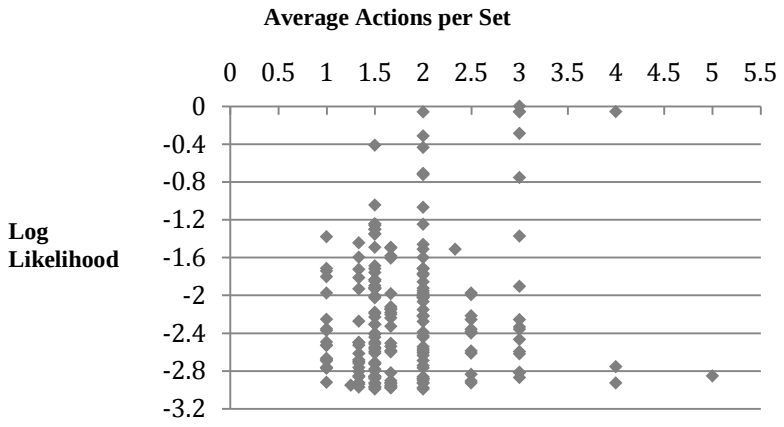

Figure 8. Fragment log likelihood, given the average actions per contributing set. Fragments with likelihoods below -2 are shown for comparison.

By taking a fragment's likelihood into account, we can sort out both uninteresting and unlikely fragments. To use the language metaphor in which actions are words, plans are sentences, and the plan library is the document, the goal is to produce fragments that are neither reproductions of the training text nor word salad, where actions are arbitrarily ordered. We aim to produce fragments that are new compositions which nonetheless adhere to the latent structure present in the narrative space. The -2 likelihood threshold was found to be close to this ideal by capturing those fragments at the beginning of this trend, thereby separating out fragments that are both likely and interesting.

5.3 Narrative Fragment Examples

Narrative fragments are intended to be used as approximate descriptors for the event space. By seeing which fragments have been generated and which actions they contain, we can identify likely and interesting action sequences. This is particularly important in the situation we simulate, where several distinct sets were combined into a single vocabulary. This adds extensibility to this approach. Narrative fragment creation functions exactly as described with an arbitrarily large vocabulary, allowing changes (adding, removing, or changing actions) to be tracked by through the fragments created from that vocabulary.

We propose four ways these fragments can be evaluated to have successfully encapsulated narrative knowledge: by generating likely but unintended action sequences, by uncovering action similarity, and through action hindsight and prediction. If our system has succeeded, there should exist fragments with unintended but related action sequences, fragments that uncover similarity between before-unrelated actions, and fragments that show likely action sequence origins or outcomes.

5.3.1 Unintended Action Sequences

The most straightforward way to use narrative fragments is to highlight unintended action relationships, or show how actions not created with one another in mind – actions from different sets – can nonetheless function together. The fragment in Table 1 is one such example.

Table 1. A fragment in which char1 disguises itself in order to approach char2. A third character (char3) then convinces char1 to leave char2, which it does.

Action	**Set Index**
disguise(char1)	11
approach(char1, char2)	11
convince(char3, char1)	25
leave(char1, char2)	25

This fragment contains transitions between actions of the same set (11 to 11, 25 to 25), but the 11 to 25 transition makes this fragment interesting. This sequence, where an entity disguises itself to get close to another and then is talked out of its plan, may require a human to fill in the expectations and motivations of the entities involved, but the key for this fragment is that it's logically consistent. What matters here is not why char1 is convinced to leave, only that it is convinced.

This fragment typifies the best case scenario from fragment creation, where a fragment is both likely (its likelihood is -1.4627, putting it well in the top half of likely fragments) and interesting (because the event sequence does not exist in any of the originating fables). In fact, both of the originating fables (11 and 25) have very different roles for the characters therein. In fable 11, there is a char3 character, but they ultimately kill char1. In fable 25, char3 persuades char1 to leave char2 in order to kill char1 and char2 after they've been separated. Fragment creation allows us to find all likely action relationships, even if the respective source content did not intend for similar relationships.

Through our simulation and analysis of the narrative generation task, these related actions cluster together and can therefore produce these unintended sequences. The power of narrative fragments comes from generation. The relationships represented in a fragment could not be learned simply by determining predicate alignment, and the fragment in Table 1 demonstrates this. The action *approach(char1, char2)* and *convince(char3, char1)* share an effect and precondition predicate: *isNear(char1, char2)=true*. However, *approach(char1, char2)* also shares this predicate with a dozen other actions in the vocabulary. Without the generation task these actions would be equally related, making this particular fragment impossible to create using predicate alignment alone. Plans establish context, and it's this context that allows fragments to highlight latent narrative knowledge.

5.3.2 Action Similarity

In some cases fragments will be the same in all but one action. Such sets of superficially identical fragments demonstrate when actions are functionally similar – that is, when they can be used for similar purposes. Our method produced one such set, where five fragments shared all but one action in common. For each fragment, the different action was *approach(char1, char2)*, sourced

from different sets (7, 8, 11, 22, and 24, respectively). Table 2 shows a sample fragment from one such set.

Table 2. A sample fragment from a superficially identical quintet. In each fragment in the quintet, char1 approaches char2. A third character (char3) convinces char1 to leave char2, and once separated, char3 kills char1.

Action	Set Index
approach(char1, char2)	7
convince(char3, char1)	25
leave(char1, char2)	25
kill(char3, char1)	25

The fragments in this quintet are superficially identical, and as such it may seem trivial to suggest that *approach(char1, char2)* from one set behaves similarly to *approach(char1, char2)* from another. The fact that the five start actions have the same name, however, does not factor into their appearance in fragments. Instead, their use in fragments is a direct result of how they behave in plans. In the plan library, *convince(char3, char1)* is frequently causally and temporally preceded by *approach(char1, char2)*, though which action used is different. This therefore suggests that though the actions differ, they are functionally similar: when used in a plan, the actions have similar causal and temporal relationships with the same nodes.

Table 3. The number of shared precondition predicates of the approach(char1, char2) action, by originating action set, where the diagonal shows the number of precondition predicates the action actually has. For example, the action from set 11 has two precondition predicates, and both can be found in the action from 24.

Set Index	7	8	11	22	24
7	2	1	1	1	1
8		2	1	1	1
11			2	1	2
22				3	1
24					4

This measure of functional similarity is another way fragments provide insight unavailable with predicate analysis alone. The *approach(char1, char2)* action from set 7, 8, and 11 all have two precondition predicates, but they only share one. From a predicate alignment perspective, these actions are not strongly related; sharing one predicate is common for actions in the vocabulary. The narrative fragment creation process, though, allows for those latent, functional relationships – that one *approach(char1, char2)* will behave like another whether their predicates neatly align or not – to become manifest. These actions are tied together not because of their names (which are not considered in the generation process) nor because of their predicates (which don't align particularly well) but instead because they serve similar purposes when in a goal-driven process, and it's this that can be extracted using fragments.

5.3.3 Action Hindsight

Sometimes fragments will end with the same subsequence, but begin with different actions. Table 4 shows one such example.

Table 4. A fragment pair in which char2 frightens or injures char1. Afterward, char1 requests help from a char3, whereupon char3 gets close to and ultimately kills char1.

Action	**Set Index**	**Action**	**Set Index**
frighten(char2, char1)	7	injure(char2, char1)	7
requestHelp(char1, char3)	16	requestHelp(char1, char3)	16
approach(char3, char1)	16	approach(char3, char1)	16
kill(char3, char1)	16	kill(char3, char1)	16

In this fragment pair, *frighten(char2, char1*) and *injure(char2, char1)* are functionally similar as described in section 5.3.2: given their common position in an otherwise identical fragment, these actions serve similar purposes: to give some rationale for char1 to ask char3 for help. The rationale, however, is not at the character level. Indeed, the rationale element – that char1 is motivated by the frighten or injure actions – is a human reader's imposition onto the fragment. What the fragment captures is the causal and temporal relatedness of the actions. Thus, this fragment is not describing something about the state of char1, but is instead describing something about the interrelationships between *frighten(char2,char1), injurte(char2,char1)* and *requestHelp(char1,char3).*

These fragments provide hindsight about the *requestHelp(char1, char3)* action. If char1 is requesting help from char3, these fragments suggest that two probable causes were char1 being injured or frightened. The particular actions are defined by the vocabulary. If another action behaved similarly in the plan library as the *frighten(char2,char1)* or *injurte(char2,char1)* actions such that a fragment featured it in this way, that action would be another probable cause for the *requestHelp(char1,char3)* action. These fragments therefore capture narrative knowledge about what might cause a later subsequence.

5.3.4 Action Prediction

Fragments can also capture narrative knowledge by providing actions or action subsequences that are likely to follow a common starting point. The fragments in Table 5 begin the same way (use the same actions in the same order): *flee(char1, char3)* is followed by *approach(char3, char1).* This in itself is an interesting sequence – as these actions come from different sets – but their inclusion highlights a relationship between them: if one character flees from another, the latter is then likely to approach the former.

This pair demonstrates action prediction. The beginning of the sequence is followed either by *kill(char3, char1)* or *capture(char3, char1)* and then *kill(char3, char1*). Through these two fragments, the system can be said to have learned that probable outcomes to a scenario in which one character flees and another follows include the following character killing the fleeing one, or the following character capturing and then killing the fleeing one.

Table 5. A fragment pair in which char1 flees from char3. Thereafter, char3 approaches and kills char1, or approaches, captures, and then kills char1.

Action	Set Index	Action	Set Index
flee(char1, char3)	24	flee(char1, char3)	24
approach(char3, char1)	22	approach(char3, char1)	22
capture(char3, char1)	11	kill(char3, char1)	23
kill(char3, char1)	11		

This relationship also hints at a more subtle understanding of the sequence. If char1 flees and char3 follows (*approach(char3, char1)* – functionally: brings char3 to char1, so "follows" is a more appropriate colloquial term), these fragments suggest that the outcome for char1 is poor. This makes an intuitive sense. The specialized knowledge a human might have about this situation includes an understanding that char1 fled for a reason – most likely fear for their life – and so the pursuit of char1 by char3 creates an expectation that char1 will suffer as an outcome.

This kind of subtlety is in part conferred by the low action per set ratio present in this fragment (1.3 and 1, respectively). By generating a fragment that draws from multiple sets without much originating fable recreation, we capture truly unintended relationships that, in this case, allow us to predict the probable outcome of the starting subsequence.

6. Discussion and Conclusion

The goal of this study was to automatically learn narrative knowledge through the creation of narrative fragments that described relationships between actions as an approximation for describing relationships between events. We proposed that fragments would have successfully encapsulated narrative knowledge if they described unintended action sequences, if they helped to identify similar actions, or if they were able to aid in action hindsight or prediction.

The core assumption behind the narrative fragment creation approach is that statistical methods can be used to learn narrative knowledge. Through simulation and analysis of a vocabulary of actions, a better understanding of how those actions relate to one another can be learned, and this understanding approximates the higher level intuition a human might have about narrative events. The fragments generated using this approach accomplished the four tasks we proposed would successfully demonstrate learning narrative knowledge, but did so in a narrow way. The kind of knowledge learned is strictly limited to the way actions relate to one another. Thus, the system can be said to have learned that *injure(char2,char1)* and *frighten(char2,char1)* can both precede *requestHelp(char1,char3)*, but not why. Nor does the system understand anything about the intentions or motivations of the characters involved. As a result, while it can be argued this approach learns some narrative knowledge, there is still a wealth of additional information that a human has access to that this approach doesn't capture.

Since humans acquire knowledge through a lifetime of experiences, it was hoped that the narrative generation process – creating an exhaustive plan library over the action vocabulary – would simulate these experiences. This assumption held but was limited by the data. A human has access to information about both the events that happen and why they happen, therefore they can draw inferences that this system – which only has access to the actions themselves – cannot.

Our data set may have been limiting so as to only highlight simple action relationships, showing which actions are related but not why. The strength of this approach, though, is that it is not tied to a particular data set. Any action vocabulary from which a plan library can be created can be analyzed using narrative fragments. Indeed, this may be the best avenue for future work. Using larger action vocabularies authored with more complex relationships in mind may provide interesting insight and serve to demonstrate whether fragments can capture subtleties not seen in this data.

The other main avenue of future work would be to create a narrative generation mechanism that takes fragments into account. A narrative generation mechanism that incorporates fragments and the knowledge they encapsulate would not be able to reach otherwise unreachable goals, but by giving preferential treatment to likely action sequences, would be able to reach goals in more likely ways, ideally improving the quality of the generated narrative.

References

Aesop. Aesop's Fables. Trans. George Fyler Townsend. Project Gutenberg. www.gutenberg.org

Chambers, N. & Jurafsky, D. (2008). Unsupervised Learning of Narrative Event Chains. *ACL/HLT*, .

Hinton, G. (1999). Products of Experts. *Proceedings of the Ninth International Conference on Artificial Neural Networks*, 1, 1-6.

Newton, M. A. H., Levine, J., Fox, M. & Long, D. (2007). Learning Macro-Actions for Arbitrary Planners and Domains. *ICAPS*, 256-263.

Polkinghorne, D. E. (1995). Narrative configuration in qualitative analysis. *International Journal of Qualitative Studies in Education*, 8, 5-23.

Porteous, J. & Cavazza, M. (2009). Controlling Narrative Generation with Planning Trajectories: The Role of Constraints. *ICIDS*, 234-245.

Porteous, J., Cavazza, M. & Charles, F. (2010). Applying Planning to Interactive Storytelling: Narrative Control Using State Constraints. *ACM TIST*, 1, 10.

Riedl, M. O. (2008). Vignette-Based Story Planning: Creativity Through Exploration and Retrieval. *Proceedings of the 5th International Joint Workshop on Computational Creativity*.

Riedl, M. O., & León, C. (2008). Toward Vignette-Based Story Generation for Drama Management Systems. *2nd International Conference on Intelligent Technologies for Interactive Entertainment, Workshop on Integrating Technologies for Interactive Story*.

Riedl, M. O. & Young, R. M. (2010). Narrative planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research* 39, 1 (September 2010), 217-268.