Select the Unexpected: A Statistical Heuristic for Story Sifting

Max Kreminski[1,3], Melanie Dickinson[2], Noah Wardrip-Fruin[3], and Michael Mateas[3]

[1] Santa Clara University
mkreminski@scu.edu
[2] Independent
meldckn@gmail.com
[3] University of California, Santa Cruz
{nwardrip,mmateas}@ucsc.edu

Abstract. Story sifting techniques, which aim to excavate potentially compelling microstories from vast chronicles of storyworld events, present a promising solution to the challenges of interactive emergent narrative. However, current sifting techniques (which rely on large numbers of hand-specified *story sifting patterns* to identify compelling microstories) are limited by their inability to determine which of many sifting pattern matches are likely to be the most interesting to a human interactor. We present a higher-level *story sifting heuristic* that addresses this problem by identifying sifting pattern matches that are especially *unlikely* from a statistical perspective, and illustrate how this heuristic leads to the surfacing of more interesting microstories.

Keywords: Story sifting · Sifting heuristics · Statistical methods.

1 Introduction

Story sifting techniques [27, 19] are a family of computational methods (generally employed as part of a larger generative or interactive storytelling pipeline) that aim to automatically identify and extract potentially storyful information from a vast chronicle of narrative events, often generated by a *storyworld simulation* containing a large number of autonomous characters. Sifting techniques have been employed in several contexts, including in podcast generation [27, ch. 12], computationally engaged improvisational theater [31], social simulation [11, 35], and mixed-initiative co-creative [21] storytelling [30, 12, 14]. Moreover, sifting techniques are sometimes used by players of interactive emergent narrative games in the process of constructing *retellings* [6] of their play experiences [16], and sifting has been pitched as a potential solution [28] to several key challenges of emergent narrative [22, 15, 36] broadly construed.

Most current approaches to story sifting make use of many hand-authored *story sifting patterns* to identify potentially compelling nuggets of narrative content. A sifter that has more of these patterns at its disposal is able to recognize

a wider variety of potentially interesting emergent story patterns, and consequently is better able to respond to the unexpected outcomes of storyworld simulation, so past research in story sifting has aimed to make these patterns easy to author in large numbers.

However, once a sifter is equipped with a wide range of sifting patterns, a new problem emerges: the difficulty of determining, from among many matches against many sifting patterns, which matches are the most worthwhile to surface to the player. Since even a single pattern may yield hundreds or thousands of matches when executed against a chronicle of thousands of simulation events, the mere knowledge that a sifting pattern found a match isn't necessarily enough to determine whether this match is likely to be particularly interesting from a human's perspective.

Beyond low-level sifting patterns, past work on story sifting [27, p. 237] has also raised the idea of higher-level *sifting heuristics* that encode a sense of what makes for narratively interesting material in a more generic way. If we had such a heuristic, it might prove useful in selecting the most interesting sifting pattern matches—but the specifics of how to implement sifting heuristics have not yet been seriously investigated.

In this paper, we present a candidate sifting heuristic that uses statistical methods to identify which of many sifting pattern matches are most likely to be interesting from a human's perspective. We call this heuristic the SELECT THE UNEXPECTED heuristic, or simply STU. Intuitively, this heuristic is based on the narrative quality of *unexpectedness*: sequences of events are more likely to be perceived as storyful, and thus to be considered *tellable* as stories [1], if they somehow deviate from expectations. We operationalize unexpectedness by searching for matches against a sifting pattern that are particularly *unusual* relative to other matches against the same sifting pattern. We evaluate our approach by comparing our heuristic's judgments of microstory interestingness to those of human raters (with favorable results); by highlighting several cherry-picked examples of microstories that our heuristic surfaced within the output of a bare-bones test simulation; and by illustrating how our heuristic can obtain results similar to a canonically successful example of story sifting (Ryan's arson-revenge example) with more generic sifting patterns than those employed in Ryan's work. Both data and code for this paper is publicly available online.[4]

2 Background

2.1 Story Sifting

Story sifting as an approach was originally proposed by Ryan under the name "story recognition" [28] and later defined further under the updated name "story sifting" [27]. Ryan's dissertation work also introduced Sheldon [27, ch. 12], a simple autonomous story sifter that makes use of sifting patterns specified as

[4] https://github.com/mkremins/statistical-sifting

blocks of code in the Python programming language. Prior to Ryan, sifting-like techniques were employed in a few different contexts: Cardona-Rivera and Young's work on symbolic plan recognition in narrative domains [3] bears some substantial similarities to pattern-based story sifting, as does Osborn et al.'s work on applying regular expressions to sequences of game states in a social simulation game [25] and Elson's work on using logical patterns to conduct analogy search between plot structures [7]. "Story trees" in *The Sims 2* were also used in a sifting-like fashion [2, 23]. However, it is Ryan's work that gave the "curationist emergent narrative" tradition a distinct name and identity for the first time.

Since Ryan's early work, a number of other approaches to authoring story sifting patterns have been introduced [19], including declarative specification of sifting patterns using domain-specific logic programming languages like Felt [13] and Winnow [10], as well as example-based [18] and visual [8] tools to assist users in authoring these declarative sifting patterns.

Relatively little work on story sifting has been done outside the pattern-based paradigm to date. One recent exception to this rule—Arc Sift [20]—performs story sifting via dynamic time warping, aiming to find stories that match specific character fortune arcs drawn by a user without the use of sifting patterns. However, the dominant approach to story sifting still relies heavily on sifting patterns to locate sites of potential narrative interest.

2.2 Toward Sifting Heuristics

The sifting patterns that are used in existing story sifters tend to be fairly low-level, concrete specifications of emergent story patterns that make for good narrative material. Patterns at this level, however, do not necessarily capture more generic notions of what makes for a good story, for instance those that have been set out in cognitive narratology research. This raises the question of how a more generic sense of narrativity [29] or tellability [1] could be encoded into the machine, such that sifters can leverage this information to better understand the player-perceived story—for instance by using tellability to gauge which of many viable sifting pattern matches are most likely to be important to the player-perceived narrative. In the story sifting literature, encodings of abstract narrativity or tellability are called *sifting heuristics* [27, p. 237].

Sifting heuristics may attempt to operationalize constructs from cognitive narratology, including story *interestingness* as defined by Schank [33] and *event salience* [9] (a proxy for story *memorability*) as operationalized in Indexter [4]. An operationalization of *surprise*—which is often treated as a key component of interestingness, and which may be detectable via statistical approaches such as anomaly detection—could also prove useful in sifting heuristics.

Sifting heuristics might also be learned from data on how users interact with existing interactive story sifters, for instance the *Bad News* "wizard console" or the Legends Viewer interface for exploring *Dwarf Fortress* worlds. Samuel et al. have recently conducted an analysis of interaction trace data with the *Bad News* wizard console [32], revealing that certain sets of wizard console commands are often executed together. Recurring patterns of interaction with these lower-level

sifting interfaces could potentially be abstracted into high-level sifting heuristics, since a human user's sense of what information is needed to identify a compelling narrative throughline for a whole *Bad News* play session (for instance) could be expected to serve as a good proxy for the information that a computational system would need to make similar determinations.

The sifting heuristic we present in this paper is based on (a relatively naïve interpretation of) a cognitive-narratological construct—namely that of surprise. However, we hope that future work will explore other approaches to sifting heuristic development as well.

3 Pattern-Based Story Sifting

Our story sifting heuristic (which we discuss further in Section 4) extends the pattern-based approach to story sifting, particularly the flavor of this approach in which patterns are specified as logic programs (as in the Felt and Winnow story sifting languages). Before discussing the details of our heuristic, we will first briefly recap how pattern-based sifting works in Felt.

Here is an example Felt sifting pattern:

```
(eventSequence ?harm ?scheme ?arson)
[?harm tag harm] [?harm actor ?revengeTarget] [?harm target ?arsonist]
[?scheme eventType hatchRevengeScheme] [?scheme actor ?arsonist]
[?scheme target ?revengeTarget]
[?arson eventType setFire] [?arson actor ?arsonist]
[?arson target ?revengeTarget] [?arson victim ?victim]
```

This pattern, based on the `arson-revenge` pattern discussed by Ryan [27, p. 671–674] and Kreminski et al. [10, 19], aims to match an ordered sequence of three events—`?harm`, `?scheme`, and `?arson`—with arbitrarily many other events interspersed between. The first of these events, `?harm`, represents a character (the eventual `?revengeTarget`) doing something that harms another character (the eventual `?arsonist`). The second event, `?scheme`, represents the `?arsonist` resolving to avenge the harm in some way. Finally, the third event, `?arson`, represents the `?arsonist` getting their revenge by burning down a building owned by the `?revengeTarget`—potentially harming a third character, the `?victim`, in the process.

Identifiers in the pattern that begin with the question mark character (such as `?scheme` and `?arsonist`) represent *logic variables*: "roles" that are bound to concrete values, such as character or event IDs, in the process of pattern execution. A match against this sifting pattern consists of a set of bindings for all of these logic variables that satisfies the pattern's constraints.

Constraints, meanwhile, are specified in terms of `[square-bracketed]` and `(parenthesized)` expressions. An expression of the form `[e a v]` represents an assertion that the entity with ID `e` has an attribute named `a` whose value is `v`; generally speaking, `e` is always a logic variable, while `a` is usually a literal

string (the name of a particular attribute) and `v` can be a literal string or another logic variable depending on context. An expression of the form `(ruleName args...)`, on the other hand, represents an assertion that the inference rule named `ruleName` holds true for the given `args`. Inference rules are Datalog rules [5], written in the DataScript [26] dialect of Datalog. The `eventSequence` rule in our example `arson-revenge` sifting pattern is one such rule; it holds true if each of its arguments is the ID of an event entity and the events these IDs point to are chronologically ordered from left to right.

Executing a sifting pattern returns a list of all valid matches against this pattern that are possible based on the events that have transpired in the storyworld so far. Depending on the pattern and on how long the storyworld has been allowed to evolve, a single Felt sifting pattern may return anywhere between a handful of matches and tens of thousands. (Even larger numbers of matches against a single pattern are theoretically possible, but because Felt runs in browser JavaScript, it generally bogs down when the number of candidate matches is any larger than this.) It is at this point that the need to select the most interesting matches becomes evident—and it is at this point that our sifting heuristic is deployed.

4 Our Heuristic: Prefer Matches with Unusual Properties

Once we have identified many matches against a single sifting pattern, how do we determine which matches are most unusual? Our approach is based on the generation and comparison of a *property signature* [24, 18] for each match: a set of simple statements about the plot events captured in this match and how they are related to one another.

In evaluating a single match against a particular sifting pattern, we first generate a list of properties that hold true for this match. For each of the match's properties, we then determine how frequently this property appears in all matches against the same sifting pattern—i.e., the property's *likelihood* of appearing in a match against this sifting pattern. We then average together the likelihood scores of each of a match's properties to determine an overall likelihood score for the match as a whole. A pseudocode definition of our heuristic is given in Algorithm 1.

We employ several distinct *property generation strategies* to discover potentially interesting details about each sifting pattern match. These property generation strategies are based on, but not limited to, those used in Synthesifter [18]. Specifically, for each match, we generate the following properties:

- One **event type property** indicating the event type of each event in the matched event sequence. An event type is a string that uniquely identifies a particular type of action that a character in the simulation can perform.
- Zero or more **event tag properties** indicating the tags of each event in the matched event sequence. A tag is a string that indicates an event has a particular characteristic; for instance, events that relate to an (actual or potential) romantic relationship between two characters may be tagged `romantic`,

```
Input: pattern: a story sifting pattern
Output: matches: a list of pattern matches with associated likelihood scores
matches ← GetAllMatches(pattern);
propertyCounts ← {};
foreach match ∈ matches do
    match.properties ← GenerateProperties(match);
    foreach property ∈ match.properties do
        prevCount ← propertyCounts[property] or 0;
        propertyCounts[property] ← prevCount + 1;
    end
end
propertyLikelihoods ← {};
foreach property ∈ Keys(propertyCounts) do
    propertyLikelihoods[property] ← propertyCounts[property] / |matches|;
end
foreach match ∈ matches do
    match.likelihood ← Average(Map(λ.prop → propertyLikelihoods[prop],
      match.properties));
end
return matches
```

Algorithm 1: The StU story sifting heuristic. The likelihood score of a match against a given sifting pattern is the average likelihood score of that match's properties in the context of that pattern. Matches with lower likelihood scores are more surprising and therefore preferable to surface.

while events that represent a character attempting to take an action but failing may be tagged `failure`.

- One **character trait property** for each matched character that has a notable trait. In the simple test simulation that we used for our evaluation, these traits include `friendly`, `unfriendly`, `romantic`, and `secretlyFamous` (with the last of these being especially uncommon).
- One **character relationship property** for each pair of matched characters that are connected by a particular kind of dyadic relationship. In our test simulation, these relationships include `viewsAsFriend`, `viewsAsEnemy`, `onesidedFriendship`, `onesidedEnmity`, `mutualFriendship`, `mutualEnmity`, `attractedTo`, `onesidedAttraction`, `mutualAttraction`, and `childhoodFriends`. (As with character traits, the last of these is especially uncommon.) Each of these relationships is defined by a DataScript inference rule, so new forms of potentially interesting dyadic character relationships can be added to the property generation process relatively easily: for instance, the `mutual` and `onesided` relationships are defined in terms of other, more basic unidirectional relationships.
- One **same-character property** for each pair of matched characters that are actually the same character. For instance, if a sifting pattern contains two distinct character roles for the `?arsonist` and `?victim` characters in an `arson-revenge` sequence, these roles are unlikely to be played by the

same character—but the possibility of both roles being played by the same character is not completely excluded (for instance, if the arsonist accidentally dies in the fire that they set). Consequently, if the same character ends up cast in both of these roles within a particular sifting pattern match, we generate a same-character property to mark this occurrence.

For instance, suppose we are generating properties for the following sequence of events—a match against a `romanticFailureThenSuccess` sifting pattern, in which a single protagonist character experiences two romantic rejections followed by a romantic success. Here's the sifting pattern:

```
(eventSequence ?e1 ?e2 ?e3)
[?e1 actor ?protag] [?e1 tag failure] [?e1 tag romantic]
[?e2 actor ?protag] [?e2 tag failure] [?e2 tag romantic]
[?e3 actor ?protag] [?e3 tag success] [?e3 tag romantic]
[?e1 target ?c1] [?e2 target ?c2] [?e3 target ?c3]
```

And here's an example match, in which the character Alex is bound to the `?protag` logic variable, while Brian and Cara take on the roles of the other characters (`?c1` through `?c3`):

1. Alex flirts with Brian and is rejected
2. Alex flirts with Cara and is rejected
3. Alex successfully asks Brian out on a date

As part of property generation, beyond examining the events themselves, we also query for the traits and relationships of the characters involved. This query tells us that the character Cara has the `friendly` trait, and the character Brian has the `viewsAsFriend` relationship toward Cara. This event sequence therefore yields the following set of properties:

- `eventHasType_e1_flirtWith_rejected`
- `eventHasType_e2_flirtWith_rejected`
- `eventHasType_e3_askOut_accepted`
- `eventHasTag_e1_romantic`
- `eventHasTag_e1_failure`
- `eventHasTag_e2_romantic`
- `eventHasTag_e2_failure`
- `eventHasTag_e3_romantic`
- `eventHasTag_e3_success`
- `eventHasTag_e3_major`
- `charHasTrait_c2_friendly`
- `charsAreRelated_viewsAsFriend_c1_c2`
- `charsAreRelated_viewsAsFriend_c3_c2`
- `sameChar_c1_c3`
- `sameChar_c3_c1`

This list of properties aims to capture the potentially interesting features of this pattern match—a set of assertions that are true about this match, and that might or might not be true for other matches against the same pattern. Some properties might hold for all or almost all matches against a particular pattern: for instance, by definition, every event included in a `romanticFailureThenSuccess` pattern match will always have the `romantic` tag, so the `eventHasTag_e1_romantic` property will hold for every match against this pattern. However, some properties are likely to be less common: for instance, since the `romanticFailureThenSuccess` sifting pattern doesn't specify that the first and last targets of attempted romantic interaction (`?c1` and `?c3` respectively) need to be the same character, the `sameChar_c1_c3` property is likely to occur fairly infrequently. Thus we return to the key idea behind our heuristic: **we can define the likelihood of a particular sifting pattern match in terms of the context-specific likelihoods of its various properties.**

5 Evaluation

We evaluate our approach in three ways. First, to determine whether our heuristic's judgments of story interestingness agree with those of human raters, we apply our heuristic to a small test simulation and five Felt sifting patterns designed to sift the output of this simulation. Using data drawn from a single run of this simulation and sifting process, raters are asked to perform blind comparisons between heuristic-preferred and randomly selected matches against these sifting patterns. Second, we highlight several cherry-picked examples of microstories surfaced by our heuristic (as applied to our test simulation and test sifting patterns) that we find particularly compelling. Third, we briefly illustrate how our heuristic is capable of obtaining results similar to Ryan's arson-revenge example—a canonical example of story sifting success—even with more generic sifting patterns than those Ryan employed.

5.1 Comparison with Random Baseline

In order to compare heuristic-preferred with randomly selected sifting pattern matches, we first ran our test simulation once to generate a chronicle of 1000 events. Our test simulation contains 24 possible action types for characters to perform and a cast of 20 characters. At the start of a simulation run, characters are randomly initialized with traits and dyadic relationships. Then, on each timestep, we randomly select a single actor character, a single target character, and a single type of action to perform between them. Actions are chosen probabilistically depending on the traits and relationships of the characters involved; for instance, a character who is `attractedTo` another character is more likely to perform an action that is tagged `romantic` toward them.

After running the simulation, we then executed five Felt sifting patterns against the resulting chronicle to collect all possible matches for each pattern.

Once the matches were collected, we applied our heuristic to the complete set of matches for each pattern and calculated a likelihood score for each match.

The five sifting patterns with which we tested include:

- `romanticFailureThenSuccess` (previously introduced in Section 4): A character experiences two romantic rejections followed by a romantic success.
- `establishFriendship`: A character performs a friendly gesture toward another character, and this friendliness is subsequently reciprocated.
- `revengeAlliance`: A character performs actions that harm two other characters; these two harmed characters have a friendly interaction; and one of the harmed characters subsequently takes revenge on the character who harmed them.
- `statusReversal`: A character performs two low-status actions toward a second character, followed by a high-status action toward that same character.
- `cantCatchABreak`: A character is harmed three times in succession by other characters' actions, with the last of these harms being major.

Based on a single run of the simulation and sifter, we collected 15 pairs of microstories: three pairs of matches for each of our five sifting patterns. Each pair of matches contained one of the three top-scoring matches for a particular sifting pattern according to our heuristic (i.e., one of the three *least likely* matches for the pattern in question) and one randomly selected match for the same pattern. We chose to compare heuristic-selected matches against random matches because random selection of matches to surface is part of current practice in story sifting (and thus a realistic baseline for comparison), and because random selection is essentially a "zero knowledge" heuristic for match selection (under which the system makes no attempt to discern which matches are most interesting).

We then presented these pairs of matches to three human raters, with each rater reviewing all 15 pairs. For each pair of matches, raters were asked to indicate which match was more interesting, resulting in a total of 45 pairwise interestingness judgments across all raters. To minimize ordering effects, pairs were presented in a randomized order from one rater to the next, and the order of matches within the pair (i.e., whether the heuristic-preferred or randomly selected match was presented first) was also randomized. Raters therefore had no way of knowing which matches were selected by the heuristic and which were selected randomly.

Altogether, across our three raters, we found that raters agreed with our heuristic's judgments of story interestingness in 38 of 45 cases. In other words, raters selected the heuristic-preferred match as the more interesting microstory 84.44% of the time. A breakdown of heuristic/rater agreement across each of the five sifting patterns is given in Table 1.

It is worth noting that this evaluation context represents something like a worst-case scenario for the presentation of sifted microstories. Unlike in interactive emergent narrative gameplay, raters were not given access to the underlying test simulation or any broader context for the microstories we presented, so their knowledge of what kinds of events are particularly common or uncommon in the

Sifting pattern	# Agreements	% Agreement
`establishFriendship`	7/9	77.78%
`romanticFailureThenSuccess`	7/9	77.78%
`revengeAlliance`	7/9	77.78%
`statusReversal`	8/9	88.89%
`cantCatchABreak`	9/9	100.00%
Overall	38/45	84.44%

Table 1. Heuristic/rater agreement on microstory interestingness, both per sifting pattern and overall. The "# Agreements" column shows the total number of cases in which a human rater agreed with the heuristic's pairwise interestingness judgment between two microstories, while the "% Agreement" column shows how often a rater's assessment agreed with the heuristic's as a percentage of total cases.

simulation was limited. Additionally, microstories were presented to the raters in a bare-bones text format that required significant mental work to interpret as a story—see Figure 1 for an example. Finally, two of our sifting patterns (`establishFriendship` and `revengeAlliance`) returned very small numbers of matches in our test run of the simulation (13 and 12 matches respectively). This limited the amount of training data to which our heuristic had access in attempting to determine per-property likelihoods for matches against these patterns, and limited the extent to which we might expect noticeable differentiation between heuristic-preferred and randomly selected matches for these patterns (since the heuristic only had around a dozen matches to select from in each case). Nevertheless, agreement between human-rated and heuristic-based judgments of story interestingness remained high. Consequently, we expect that agreement between our heuristic's judgment of story interestingness and that of human interactors will at worst remain the same and at best substantially increase in contexts more similar to typical interactive emergent narrative gameplay.

Which romanticFailureThenSuccess story is more interesting?

Story 1: Isla flirtWith_rejected Quinn. Isla flirtWith_rejected Sarah. Isla flirtWith_accepted Mira. *(Isla is romantic. Quinn is unfriendly. Mira is romantic. Mira viewsAsEnemy Sarah. Mira attractedTo Sarah.)*

Story 2: Mira askOut_rejected Peng. Mira askOut_rejected Peng. Mira flirtWith_accepted Riva. *(Mira is romantic. Peng is unfriendly. Riva is unfriendly. Riva viewsAsFriend Peng. Riva attractedTo Peng.)*

Fig. 1. Example comparison between a pair of microstories, as presented to raters.

5.2 Cherry-Picked Successes

During testing, we found several of the emergent microstories surfaced by our heuristic to be particularly compelling. One of these was a match against the `romanticFailureThenSuccess` pattern, discovered during early testing when only this pattern was active. The character Oswald asked the character Lexi out on a date and was turned down. Oswald then flirted with another character, Victor, and was turned down again. But when Oswald asked Lexi out a second time, Lexi accepted. Intriguingly, though neither Lexi nor Victor felt much of anything toward Oswald, they both viewed one another as enemies and were attracted to one another—suggesting that Lexi's acceptance of Oswald's second invitation was motivated more by complex feelings of jealousy toward Victor than by any actual interest in Oswald.

Another exemplary microstory was a match against the `statusReversal` pattern, surfaced as the top heuristic-preferred match on the very first run of the simulation with this sifting pattern enabled. The character Quinn first felt condescended to by the character Isla, but shortly thereafter deferred to Isla's expertise on a career-related topic. Much later, presumably after Quinn had risen to a much greater level of prominence within Isla's social circle, Quinn ended up shunning Isla from this circle. Tragically, the initially higher-status Isla had viewed the initially lower-status Quinn as both a friend and a potential love interest—but Quinn did not reciprocate these feelings, and Isla's attempts to support Quinn ultimately brought Isla nothing but suffering.

More generally, we found examples of all of the following unlikely occurrences in matches that were surfaced by our heuristic:

- **Globally uncommon event types.** For instance, characters in our test simulation rarely physically attack one another even when there exists conflict between them, so microstories involving physical violence stand out as unusual. (In simulations where physical violence is the predominant means of conflict resolution, microstories involving peaceful resolutions to conflict might be highlighted instead.)
- **Pattern-specific uncommon event types.** For instance, even globally common event types (such as visiting a graveyard) may occur very rarely in particular narrative contexts (such as a romance microstory), so a romance microstory in which the initial meet-cute took place in a graveyard stands out as unusual.
- **Characters more related than necessary.** For instance, romantic rivals do not necessarily have to be related in any other way, so a microstory involving two romantic rivals who were also childhood friends stands out as unusual.
- **Characters related in unexpected or apparently contradictory ways.** For instance, romantic affection is generally correlated with platonic affection, so a microstory in which two characters who view one another as enemies become involved in a romantic relationship stands out as unusual.
- **Characters acting against type.** For instance, characters with the `friendly` trait in our test simulation tend to behave in consistently friendly ways, so

a microstory in which a friendly character performs several mean actions stands out as unusual.

Unexpected occurrences like these are often featured in player-constructed retellings of their gameplay experiences in interactive emergent narrative games, and they may serve as especially fertile jumping-off points for *extrapolative narrativization* [17]: the process by which players who are writing retellings of their gameplay experiences invent additional details (often involving character motives) that are consistent with simulation events, but go beyond the level of detail that the simulation actually models. For instance, in the cherry-picked story of Oswald, Lexi and Victor, the character motivations *implied* by the characters' actions are much more complicated than the motivations actually modeled in the simulation engine—but the story still "works" on an intuitive level because of the reader's capacity for extrapolation.

5.3 Generalizing Arson-Revenge

Perhaps the strongest argument for our heuristic is that it could successfully identify the canonical arson-revenge microstory from Ryan's dissertation [27, p. 637–638] as especially notable, even if it was only given a more generic sifting pattern for revenge stories that do not necessarily involve arson. Ryan's arson-revenge microstory involves a farmhand (Roy Champ) who is bullied by the farm's owner (Julius Eckert) and decides to enact a revenge scheme against him, specifically by burning down his farmhouse. However, in the process, Champ himself is trapped in the burning building and ends up dying, while Eckert (who was away for the weekend) is physically unharmed. Ryan discovered this instance using a sifting pattern that was specifically tailored to search for stories of arson being used as a means of revenge.

Using our heuristic, this microstory would stand out as unusual among all kinds of sifted revenge stories for two key reasons. First, because arson is a rare event both globally (since characters rarely commit arson in general) and contextually (since most revenge schemes are enacted by means other than arson), the mere presence of a `set-fire` action in a match against a generic revenge story sifting pattern would tend to make this match less likely from a statistical perspective. Second, since the avenger and victim roles in this microstory are played by the same character (even though a revenge sequence does not necessarily *have* any victims, and victims would usually be assumed to be characters other than the avenger), this microstory would stand out as an exemplary case of the involved characters being *more related than necessary*. In summary, our heuristic would generate two statistically unusual properties for this match—`eventHasType_e3_set-fire` and `sameChar_avenger_victim`—then detect these properties (and particularly their simultaneous presence in a single match) as statistically unusual. Consequently, our heuristic would lead to the discovery and surfacing of this match—even in the absence of a sifting pattern specific to arson-revenge stories—as long as a more generic sifting pattern for revenge stories of any kind was available.

6 Discussion

6.1 Considerations for Sifting Heuristic Design

Our use of pattern-based sifting as a foundation atop which to implement higher-level sifting heuristics is essentially a solution to a narrative-specific form of the *frame problem* [34]: the difficulty of determining what context is relevant when trying to determine whether a given event is worth mentioning in a story. Interestingly, though past work on story sifting has generally emphasized the importance of giving sifters access to many highly concrete story sifting patterns [19], we found that the presence of our heuristic substantially improves the leverage of more abstract sifting patterns, since more generic patterns tend to match more microstories within the same chronicle of events, giving statistical heuristics more training data on which to base their judgments of microstory interestingness and more options to choose between when selecting microstories to surface. This result suggests that, in two-layer approaches to sifting like ours (which use both patterns and heuristics), the role of patterns is different than in single-layer approaches to sifting (which make use of patterns alone): instead of attempting to capture only tellable sequences of events, patterns should be written to capture event sequences that have narrative structure without regard to tellability, and tellability judgments should be left to heuristics.

The heuristic we describe here offers a form of built-in explainability as to why a particular sifting pattern match has been surfaced over others. Since not just matches themselves but also the individual *properties* of each match are given their own likelihood scores, it is possible to sort all of a given match's properties by their likelihood and directly present the specific properties that are judged as especially unlikely in order to explain why this match was judged as unusual or worth surfacing. Depending on the context in which story sifting is deployed, this form of explanation may help to mitigate the *Tale-Spin effect* [37] (in which human interactors fail to appreciate the complexity of an AI system due to the opacity of its outputs) in interactive emergent narrative games that make use of story sifting. Alternatively, if sifting is deployed in the context of mixed-initiative co-creative storytelling (as it sometimes has been in the past), explanation generation may help to enrich co-creative interaction in the ways envisioned by Zhu et al. [38] in their exploration of explainable AI for designers.

6.2 Limitations and Future Work

One limitation of our heuristic is its poor handling of *correlated unlikelihoods*. For instance, if there exists a rare `kickPuppy` action that most characters never perform, and a character with a rare `villain` trait that causes them to perform the `kickPuppy` action whenever possible, the overall prevalence of sifting pattern matches containing `kickPuppy` actions will be very low overall—as will the overall prevalence of matches involving a `villain` character. But when a match contains a `villain` character who performs a `kickPuppy` action, both the presence of the individually unlikely character trait and the individually

unlikely event type will cause the heuristic to rate this match as very unlikely overall—even though this particular rare action is performed very frequently by characters with this particular rare trait. Since our heuristic only considers whether a particular property of a sifting pattern match is common or uncommon among all other matches against the same pattern, it has no way of determining that puppy-kicking is to be expected from villains, and consequently to treat a match that contains both of these correlated unlikelihoods as a relatively likely microstory overall. This remains an issue for future work.

In the future, we also expect that it will be valuable to explore additional property generation strategies beyond those presented here. One set of potentially valuable property generation strategies, which we made no attempt to implement in the initial version of our heuristic, involves pairwise relationships between events themselves. For instance, in a simulation that performs what Ryan terms "causal bookkeeping" [27, p. 162–163], causal relationships between events captured in the same sifting pattern match could be surfaced as properties. Similarly, the amount of time elapsed between events in a matched event sequence could be used for property generation as well: narrative interest can be derived from the fact that a pair of matched events occurred very close together, or very far apart, in time.

Finally, we note that because the comparison-against-random-baseline portion of our evaluation made use of only three human raters, the results of this comparison remain somewhat tentative: although all raters agreed with our heuristic's interestingness judgments in a clear majority of cases, it remains possible that the raters were all outliers in this regard. Future studies could provide stronger evidence for the success of our heuristic by applying a similar study design to a substantially larger number of raters.

7 Conclusion

We present the StU sifting heuristic, which is (to the best of our knowledge) the first high-level story sifting heuristic that can be used to automatically evaluate microstory interestingness in a story sifting context. Agreement between our heuristic's judgments of interestingness and those of human raters is high, and our approach can easily be generalized to other simulation engines besides the simple one with which we conducted our initial testing. We believe that the two-layer approach to story sifting presented here (in which sifting patterns are used in conjunction with sifting heuristics) represents a substantial advance over single-layer approaches to sifting that make use of patterns alone, and we look forward to the development of additional sifting heuristics that can further improve on this approach.

Acknowledgements Max Kreminski conducted part of this research while in residence at Stochastic Labs.

References

1. Baroni, R.: Tellability. In: Hühn, P., Meister, J.C., Pier, J., Schmid, W. (eds.) The Living Handbook of Narratology. Hamburg University, April 18, 2014 edn. (2014), https://www-archiv.fdm.uni-hamburg.de/lhn/node/30.html
2. Brown, M.: The power of projection and mass hallucination: Practical AI in The Sims 2 and beyond. Invited talk at AIIDE 2006 (2006)
3. Cardona-Rivera, R., Young, R.: Symbolic plan recognition in interactive narrative environments. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 11 (2015)
4. Cardona-Rivera, R.E., Cassell, K.B., Ware, S.G., Young, R.M.: Indexter: A computational model of the event-indexing situation model for characterizing narratives. In: Proceedings of the 3rd Workshop on Computational Models of Narrative. pp. 34–43 (2012)
5. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about Datalog (and never dared to ask). IEEE Transactions on Knowledge and Data Engineering **1**(1), 146–166 (1989)
6. Eladhari, M.P.: Re-tellings: the fourth layer of narrative as an instrument for critique. In: International Conference on Interactive Digital Storytelling. pp. 65–78. Springer (2018)
7. Elson, D.K.: Detecting story analogies from annotations of time, action and agency. In: Proceedings of the LREC 2012 Workshop on Computational Models of Narrative. pp. 91–99 (2012)
8. Johnson-Bey, S., Mateas, M.: Centrifuge: A visual tool for authoring sifting patterns for character-based simulationist story worlds. In: Joint Proceedings of the AIIDE 2021 Workshops (in press) (2021)
9. Kives, C., Ware, S., Baker, L.: Evaluating the pairwise event salience hypothesis in Indexter. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 11, pp. 30–36 (2015)
10. Kreminski, M., Dickinson, M., Mateas, M.: Winnow: a domain-specific language for incremental story sifting. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 17, pp. 156–163 (2021)
11. Kreminski, M., Dickinson, M., Mateas, M., Wardrip-Fruin, N.: Why Are We Like This?: Exploring writing mechanics for an AI-augmented storytelling game. In: Proceedings of the 2020 Conference of the Electronic Literature Organization (2020)
12. Kreminski, M., Dickinson, M., Mateas, M., Wardrip-Fruin, N.: Why Are We Like This?: The AI architecture of a co-creative storytelling game. In: International Conference on the Foundations of Digital Games (2020)
13. Kreminski, M., Dickinson, M., Wardrip-Fruin, N.: Felt: a simple story sifter. In: International Conference on Interactive Digital Storytelling. pp. 267–281. Springer (2019)
14. Kreminski, M., Dickinson, M., Wardrip-Fruin, N., Mateas, M.: Loose Ends: A mixed-initiative creative interface for playful storytelling. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 18 (2022)
15. Kreminski, M., Mateas, M.: A coauthorship-centric history of interactive emergent narrative. In: International Conference on Interactive Digital Storytelling. pp. 222–235. Springer (2021)

16. Kreminski, M., Mateas, M.: Toward narrative instruments. In: International Conference on Interactive Digital Storytelling. pp. 499–508. Springer (2021)
17. Kreminski, M., Samuel, B., Melcer, E., Wardrip-Fruin, N.: Evaluating AI-based games through retellings. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 15, pp. 45–51 (2019)
18. Kreminski, M., Wardrip-Fruin, N., Mateas, M.: Toward example-driven program synthesis of story sifting patterns. In: Joint Proceedings of the AIIDE 2020 Workshops (2020)
19. Kreminski, M., Wardrip-Fruin, N., Mateas, M.: Authoring for story sifters. In: The Authoring Problem: Challenges in Supporting Authoring for Interactive Digital Narratives. Springer (2022)
20. Leong, W., Porteous, J., Thangarajah, J.: Automated sifting of stories from simulated storyworlds. In: Raedt, L.D. (ed.) Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22. pp. 4950–4956. International Joint Conferences on Artificial Intelligence Organization (July 2022), https://doi.org/10.24963/ijcai.2022/686
21. Liapis, A., Yannakakis, G.N., Alexopoulos, C., Lopes, P.: Can computers foster human users' creativity? Theory and praxis of mixed-initiative co-creativity. Digital Culture & Education **8**(2), 136–153 (2016)
22. Louchart, S., Truesdale, J., Suttie, N., Aylett, R.: Emergent narrative: past, present and future of an interactive storytelling approach. In: Interactive Digital Narrative, pp. 185–199. Routledge (2015)
23. Nelson, M.J.: Emergent narrative in The Sims 2 (2006), https://www.kmjn.org/notes/sims2_ai.html
24. Odena, A., Sutton, C.: Learning to represent programs with property signatures. In: International Conference on Learning Representations (ICLR) (2020)
25. Osborn, J., Samuel, B., Mateas, M., Wardrip-Fruin, N.: Playspecs: Regular expressions for game play traces. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 11 (2015)
26. Prokopov, N.: DataScript (2014), https://github.com/tonsky/datascript
27. Ryan, J.: Curating Simulated Storyworlds. Ph.D. thesis, University of California, Santa Cruz (2018)
28. Ryan, J.O., Mateas, M., Wardrip-Fruin, N.: Open design challenges for interactive emergent narrative. In: International Conference on Interactive Digital Storytelling. pp. 14–26. Springer (2015)
29. Ryan, M.L.: The modes of narrativity and their visual metaphors. Style pp. 368–387 (1992)
30. Samuel, B., Mateas, M., Wardrip-Fruin, N.: The design of Writing Buddy: a mixed-initiative approach towards computational story collaboration. In: International Conference on Interactive Digital Storytelling. pp. 388–396. Springer (2016)
31. Samuel, B., Ryan, J., Summerville, A.J., Mateas, M., Wardrip-Fruin, N.: Bad News: An experiment in computationally assisted performance. In: International Conference on Interactive Digital Storytelling. pp. 108–120. Springer (2016)
32. Samuel, B., Summerville, A., Ryan, J., England, L.: A quantified analysis of Bad News for story sifting interfaces. In: International Conference on Interactive Digital Storytelling. pp. 142–156. Springer (2021)
33. Schank, R.C.: Interestingness: controlling inferences. Artificial Intelligence **12**(3), 273–297 (1979)
34. Shanahan, M.: The frame problem. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Spring 2016 edn. (2016), https://plato.stanford.edu/archives/spr2016/entries/frame-problem/

35. Summerville, A., Samuel, B.: Kismet: A small social simulation language. In: Joint Workshops of the International Conference on Computational Creativity (2020)
36. Walsh, R.: Emergent narrative in interactive media. Narrative **19**(1), 72–85 (2011)
37. Wardrip-Fruin, N.: Expressive Processing: Digital Fictions, Computer Games, and Software Studies, chap. The Tale-Spin Effect. MIT Press (2009)
38. Zhu, J., Liapis, A., Risi, S., Bidarra, R., Youngblood, G.M.: Explainable AI for designers: A human-centered perspective on mixed-initiative co-creation. In: 2018 IEEE Conference on Computational Intelligence and Games (CIG). IEEE (2018)