Computational Recognition and Comprehension of Humor in the Context of a General Error Investigation System

by

Ada Taylor

B.S., Massachusetts Institute of Technology (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2018

Author .....
Department of Electrical Engineering and Computer Science
Feb 02, 2018

Certified by .....
Patrick H. Winston
Ford Professor of Artificial Intelligence and Computer Science
Thesis Supervisor

Accepted by .....
Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Computational Recognition and Comprehension of Humor in the Context of a General Error Investigation System

by

Ada Taylor

Submitted to the Department of Electrical Engineering and Computer Science
on Feb 02, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Humor is a creative, ubiquitous, and powerful communication strategy, yet it is currently challenging for computers to correctly identify instances of humor, let alone understand it. In this thesis I develop a computational model of humor based on error identification and resolution, as well as methods for understanding the mental trajectory required for successful humor appreciation. An infrastructure for constructing humor detectors based on this theory is implemented in the context of a general error handling and investigation system for the Genesis story-understanding system.

The computational model consists of a series of `Experts` that quantify important story elements such as allyship, harm to characters, character traits, karma, morbidity, contradiction, and unexpected events. Due to the homogeneous structure of their interactions, `Experts` using different methodologies such as simulation, Bayesian reasoning, neural nets, or symbolic reasoning can all interact, share findings of interest, and suggest reasons for each other's issues through this system.

This system of `Experts` can identify the resolvable narrative flaws that drive humor, therefore they are also able to discover unintentional problems within narratives. I have additionally demonstrated successful quantification of indicators of effective human engagement with narrative such as suspense, attention span length, attention density, and moments of insight. Variations in `Expert` parameters account for different senses of humor in individuals. This new scope of understanding allows Genesis to help authors search their narratives to determine if higher level narrative mechanics are well executed or not, a crucial role usually reserved for a human editor. By successfully demonstrating a framework for computational recognition and comprehension of humor, I have begun to show that computers are capable of sharing an ability previously considered an exclusively human quality.

Thesis Supervisor: Patrick H. Winston
Title: Ford Professor of Artificial Intelligence and Computer Science

Acknowledgments

My wholehearted appreciation goes to the many people who contributed to the Genesis, evolution, and completion of this thesis.

Thank you to my family for their love, encouragement, and support at every step of this journey.

To Genesis' own society of experts, I am deeply grateful to have been able to work with you. I always left conversations fueled by each of your unique perspectives and your boundless enthusiasm. I would like to give particular thanks to lab members Caroline Aronoff, Suri Bandler, Jake Barnwell, and Jessica Noss for great advice, thought provoking discussions, and for the contribution of so many excellent samples of humor.

Thank you to Alex Konradi for his meticulous editing.

I am also particularly grateful to Dylan Holmes, and would like to thank him for his insight, time and support, and his obvious passion for improving the field of artificial intelligence and the work of our group as a whole.

My deepest gratitude goes to Patrick Winston for the wisdom you shared that continue to inspire me to grow as a student, researcher and person, as well as for always humoring my attempts at humor. Your guidance, encouragement, and support for your students are without equal, and I am incredibly glad to have been able to learn from you. Thank you.

Contents

1 Introduction **15**

1.1 Vision 15

1.1.1 Why is Humor Important? 17

1.1.2 The Perils of False Understanding: No Soap, Radio 20

1.2 Exploring Humor Understanding through the Example of the Roadrunner 22

1.2.1 Background and Expectations 23

1.2.2 Underlying Meaning 24

1.2.3 Sufficient Reasons for Surprises 25

1.2.4 Avoidance of Killjoys 26

1.3 What Is Humor? 28

1.3.1 Expectation Repair Hypothesis 28

1.4 Implementation 29

1.4.1 Genesis Story Understanding System 30

1.4.2 Experts 31

1.4.3 Consulting with Experts 32

1.4.4 Humor Identification 33

2 Experts: Agents for Story Analysis **37**

3 Expert Implementations **41**

3.1 Contradiction Expert 41

3.1.1 Flags 42

3.1.2 Repairs 42

3.2 Unexpected Expert 42
3.2.1 Flags 44
3.2.2 Repairs 44
3.3 Ally Expert 44
3.3.1 Flags 45
3.3.2 Repairs 47
3.4 Harm Expert 47
3.4.1 Flags 47
3.4.2 Repairs 48
3.5 Karma Expert 48
3.5.1 Flags 50
3.5.2 Repairs 50
3.6 Morbidity Expert 50
3.6.1 Flags 51
3.6.2 Repairs 52
3.7 Trait Expert 52
3.7.1 Flags 52
3.7.2 Repairs 53

4 Experts in Action: Flagging Surprises 55
4.1 Flag Examples 55
4.1.1 Contradiction Expert 55
4.1.2 Unexpected Expert 56
4.1.3 Ally Expert 57
4.1.4 Harm Expert 58
4.1.5 Karma Expert 60
4.1.6 Morbidity Expert 61
4.1.7 Trait Expert 62

5 Experts in Collaboration: Resolving Confusion 65
5.1 Expert Resolution Relationships: A Hierarchy of Abstraction 65

5.2 Non-Humorous Resolution Examples 67
5.2.1 Contradiction Expert Resolution 67
5.2.2 Unexpected Expert 68
5.2.3 Ally Expert 69
5.2.4 Harm Expert 70
5.2.5 Karma Expert 71
5.2.6 Morbidity Expert 73
5.2.7 Trait Expert 74

6 Feature Metapatterns and Humor Identification 75
6.1 Analyzing `Expert Features` 75
6.1.1 "Bug or a Feature?": Feature Resolution Status 76
6.1.2 Explanation or Realization: Order of Feature Components . . 76
6.1.3 Separation between sub-Features 80
6.1.4 Combining Feature Information 81
6.1.5 Genre: Which Experts Participated 81
6.2 How to Identify an Instance of Humor 81
6.2.1 Narrative Histograms provide Visual Signatures 82
6.2.2 Extracting the Communication within an Instance of Humor . 88
6.2.3 Genre Classifications 88
6.2.4 Personal Preference and Modeling the Mind of the Individual 88
6.3 Example Joke Applications 90
6.3.1 Roadrunner Joke 90
6.3.2 Baby Rhino Joke 96
6.4 Future Directions 103

7 Contributions 107

List of Figures

1-1 Expectation Repair as a Model for Detecting Humor 16

1-2 A sign by a child intended to read "I love Santa" that instead reads "I love Satan" [19] 26

1-3 Expert Society Consultation Process 33

1-4 The process of finding and fixing `Expert Features` 34

1-5 The anatomy of the `Expert Features` and their component Flags and Fixes returned after an `Expert Society` examines a story 35

2-1 Overview of a code scanning workflow with Lint, as described by Android Studio [9]. Note the similarities to the `Expert Society` structure, input, and output. 40

3-1 Morbidity Humor in Batman #321. [24] 51

4-1 An example story that the `Contradiction Expert` would issue a flag on. 56

4-2 An example story that the `Unexpected Expert` would issue a flag on. 57

4-3 An example story that the `Ally Expert` would issue a flag on. 58

4-4 An example story that the `Harm Expert` would issue a flag on. 59

4-5 An example story that the `Karma Expert` would issue a flag on. . . . 60

4-6 An example story that the `Morbidity Expert` would issue a flag on. . 61

4-7 An example story that the `Trait Expert` could issue a flag on. . . . 62

5-1 The relative abstraction of `Experts` within the story. Arrows go from flagging `Expert` to repairing `Expert` 67

5-2 Contradiction flag being resolved. 68
5-3 Ally flag being resolved. 69
5-4 Harm flag being resolved. 71
5-5 Karma flags being resolved. 72
5-6 Morbidity flags being resolved 73

6-1 `Callback Feature`: Webcomic XKCD Strip 475 [1] and 939 [2] . . . 79
6-2 Method for Narrative Histogram Creation 83
6-3 Histogram of a successful joke with a clear punch line moment 84
6-4 Histogram of a less successful joke with a slight punch line moment followed by explanations that stagger and diffuse the punch line. . . . 84
6-5 Histogram of a normal story, with a more randomized blend of feature repairs and completions. 85
6-6 Histogram of the general plot of Jurassic Park. The more unanswered questions we have open, the higher more units of suspense at a given point in time. Jurassic Park builds to a crescendo, then resolves, with a slight uptick of a cliffhanger at the end. 86
6-7 Histogram of a general murder mystery. Clues are given before the climax, a climax raises a lot of flags, but the clues let us resolve those flags immediately. Some falling action. Not a joke due to unresolved Karma flag; an innocent is not okay because they were murdered. . . 87
6-8 Looney Tunes Roadrunner and Coyote Joke Example 94
6-9 Humor Histogram of the Roadrunner Scenario 96
6-10 Suspense Histogram of the Roadrunner Scenario 96
6-11 Genesis Diagram of the Baby Rhino Video 100
6-12 Humor Histogram of the Baby Rhino Scenario 102
6-13 Suspense Histogram of the Baby Rhino Scenario 103

List of Tables

1.1 Anatomy of an `Expert Feature` 33

3.1 Contradiction Flag 1: 42
3.2 "Alice is likely happy." 43
3.3 "Alice is likely happy. Alice is happy. Bob runs to the store." 43
3.4 "Alice is likely happy. Alice is happy. Bob runs to the store." 44
3.5 "Batman fights the Joker. Robin is friends with Batman. Gordon does not arrest Batman." 45
3.6 Ally Flag 1: 46
3.7 Ally Flag 2: 46
3.8 Ally Flag 3 (Betrayal): 46
3.9 Harm Statuses 1: 48
3.10 Harm Statuses 1: 48
3.11 Karma Statuses 1: 49
3.12 Morbidity Flag: 52
3.13 `Expert Feature` of "The roadrunner is fast. The roadrunner is clever. The roadrunner likely does not escape. The roadrunner escapes." . . 53

4.1 `Expert Feature` of Contradiction Investigation 56
4.2 `Expert Feature` of Unexpected Investigation 57
4.3 `Expert Feature` of Ally Investigation 58
4.4 `Expert Feature` of Harm Investigation 59
4.5 `Expert Feature` of Karma Investigation 61
4.6 `Expert Feature` of Morbidity Investigation 62

4.7 `Expert Feature` of Trait Investigation 63

5.1 `Expert Feature` Flags to Possible Resolutions 66
5.2 `Expert Feature` of Contradiction Resolution 68
5.3 `Expert Feature` of Unexpected Resolution 69
5.4 `Expert Feature` of Ally Resolution 70
5.5 `Expert Feature` of Harm Resolution 70
5.6 `Expert Feature` of Karma Resolution 72
5.7 `Expert Feature` of Morbidity Resolution 74

6.1 `Unexpected Expert Features` of Looney Tunes Example 93
6.2 `Ally Expert Features` of Looney Tunes Example 95
6.3 `Harm Expert Features` of Looney Tunes Example 95
6.4 `Karma Expert Features` of Looney Tunes Example 95
6.5 `Morbidity Expert Features` of Looney Tunes Example 95
6.6 `Unexpected Expert Features` of Baby Rhino Example 101
6.7 `Ally Expert Features` of Baby Rhino Example 101
6.8 `Karma Expert Features` of Baby Rhino Example 102

Chapter 1

Introduction

1.1 Vision

The vision for this project is to put forward and successfully demonstrate a computational model of humor. This model is designed to be used to evaluate non-textual comedic moments as well as written narrative. It enlists feature-detecting experts that flag broken expectations and collaborate to repair these expectations as a method of humor recognition. (See Figure 1-1.)

The addition of this new model of humor comprehension to the Genesis story understanding system gives the system a greater understanding of how humans branch through different levels of abstraction in their interpretations of text and how potential errors in understanding are handled. Overall, these mechanisms enable Genesis to interact more deeply with humans to avoid critical misunderstandings and to understand a key human capability that has a huge impact on our learning, memory, and engagement.

Figure 1-1: Expectation Repair as a Model for Detecting Humor

To achieve the goals of this project, I have:

- Identified the need for a theory of humor that defines the implicit information and mental trajectory required to appreciate a humorous scenario.
- Developed a theory of how humor works from a computational perspective focusing on the process of repairing broken expectations.
- Implemented and tested aspects of that theory by building and testing a foundational collection of feature-detecting experts that recognize broken expectations within a story across seven different domains of narrative understanding.
- Demonstrated programmatic methods for resolving broken expectations discovered by individual experts through their collaborations with other experts.
- Created a humor detection algorithm using these interactions between experts.
- Created a methodology for tracing the mental trajectories represented by interactions between experts.
- Created the idea of Narrative Histograms to efficiently communicate the findings of my humor detection algorithm and other metrics of audience engagement with a story.
- Simulated these proposed algorithms at work on an example of humor in a cartoon script, as well as a humorous video script taken from the real world.

1.1.1 Why is Humor Important?

Understanding jokes and their subtext is critical for machines to undertake tasks such as police work or counseling, and can greatly facilitate skillful emotional support and companionship for humans by machines. However, it is difficult to codify humor due to its underlying complexity and the fact that novelty plays such a large role in its effectiveness. While it is natural for humans to mine joking statements for crucial implicit context such as self-deprecation, cynicism, or empathy, machines have not yet made much progress at inferring these important by-products of humor within

communication.

Of the capabilities that make us human, one of the skills often ascribed uniquely to humanity is humor. However, current definitions of humor are inconclusive and rarely quantified, leaving no computational road map for approaching the important communication goals of identifying or analyzing comedic moments. Even in technologically optimistic fictions such as *Star Trek*, *Terminator*, and *Lost in Space*, humor is seen as one of the few attributes remaining unique to humanity and inaccessible to machines.

Therefore, the understanding of humor provides a valuable target for artificial intelligence and potential insight into a universal human capability. Laughter develops in infants as young as five weeks, and seems linked to their early learning processes and curiosity while investigating novel stimuli [21]. Given relevant background information, humans seem to recognize when a joke has been told, even if it was not a joke they particularly liked or would have told themselves [15]. Jokes are common in human communication, and individuals laugh louder and longer when in a group setting [10], indicating that humor is not just for individuals, but is an important aspect of interpersonal communication. For these reasons, I posit that humor is a core capability of our human experience.

Humor also presents a rich domain for computers to learn from [5]. A simple joke is compact and self-contained, which can reduce the computational load required for analysis. Humor also presents a clear example of Chomsky's merge operator in action: a joke uses pieces of previously known information to build a new and unexpected effect. In fact, jokes are almost always more funny if the person has not heard the joke before or does not see the punch line coming. Genesis' use of common sense and rules to build more complex interpretations of narrative seems a natural complement to these goals.

The main hindrance to computational approaches to understand humor is the lack of a quantitative or consistent definition. The first two prominent theories focus on psychological effects for the amused individual, with Relief Theory characterizing humor as a mechanism of relief of tension from fears [10] or learned rules [18],

and Superiority Theory focusing on humor as a method for highlighting the relative misfortune of others [16]. However, other definitions do a better job on examining outer structure, not just emotional reaction. Benign Violation Theory is defined by the juxtaposition of a rule about the world being broken in a threatening manner, and this incongruous "threat" being overall harmless. However, these categories are relatively broad and imprecise. Of most interest are the theories of Incongruity and Incongruity-Resolution [20] which focus on finding incongruities with known rules, and in the case of Incongruity-Resolution, finding a rule that allows this initial incongruity. However, I make the case that prediction and expectation are crucial aspects of humor understanding, and they are key to concepts such as comedic timing, ambiguous resolutions of incongruities, and dual-meaning jokes.

Current engineering efforts in the study of humor are primarily focused on humor generation, a task I argue is both smaller in scope and more prone to incomplete, mechanistic solutions than humor recognition. Several of the most effective of these methods still work by generating randomized jokes en masse for a human to choose from [22] [17], or tightly applying human-created templates [14].

In the area of humor identification, several promising approaches are provided by neural nets, yet none can explain why a scenario is humorous. One graphically focused system attempts to increase or decrease the humor in images created from a set of paper-doll-like templates [6]. However, while this algorithm could discover that animate objects tended to be more funny than inanimate objects, it was not able to interpret why a scene was funny, nor was the model highly accurate. Another approach focused on the identification of sexual innuendo improved on existing techniques, but was so domain-specific as to be non-extensible, and still did not match human performance [13]. Most importantly, neither of these engineering models put into effect any falsifiable, cohesive definition of humor. Additionally, all the non-neural net models were focused around solely text-based input, and supply no mechanism to quantitatively account for visual or temporally based humor.

A robust understanding of humor would be of immense value to any computational system that interacts with humans. Understanding sarcasm, for example, would be a

crucial component of artificially intelligent policing in the future. Understanding the underlying frustrations expressed by a self-deprecating comment could aid robotic caretakers in better aiding their charges, and a computational political analyst could identify which components of a platform are potentially vulnerable to satire. With a solid and proven theory of humor, developers can then turn to ways to actively create humor while communicating with humans. Video games [7], teaching [26], and advertisement [23] are all fields where leveraging humor has been shown to significantly increase satisfaction, and engagement.

1.1.2 The Perils of False Understanding: No Soap, Radio

> *Man, that guy is the Redgrin Grumble of pretending he knows what's going on. Oh, you agree, huh? It's funny. You like that Redgrin Grumble reference? Yeah. Well, guess what? I made him up. You really are your father's children. Think for yourselves. Don't be sheep.*
>
> Rick from *Rick and Morty*

Notably, this model seeks to produce understanding of jokes through the decomposition and identification of the elements that produced an instance of humor. This is an important concept that treats humor understanding as a strategy for communication, rather than simply an additional layer of human-like camouflage for machines. In short, knowing when to laugh is insufficient without also being able to articulate why something is funny.

Given the importance of the Turing test or the Chinese room argument to the field of computer science, this may seem like an overly idealistic or insufficiently pragmatic approach to humor identification. However, I argue that understanding of the component pieces in a joke is required for fully cataloging the intended subtexts. This, then, allows a system to correctly communicate in response.

To better understand how shallow understanding is possible, we can examine the following setup:

Q: "What is pink and has fleeb?"

A: "A plumbus."

More generally:

Q: "What is [predicate] and [other predicate]?"

A: "Noun."

Even if we do not comprehend the actual joke directly, we can tell that this template means a joke was likely intended. This can be productive, perhaps, for future information gathering attempts, or future quips that directly echo this joke. However, if a listener using a superficial method of determining when to signal laughter was asked fairly basic questions about the joke, they would have difficulty using this technique to separate the deceptive or misleading elements common to riddles of this format from the true content. And it remains possible that this joke identification is a false positive.

The response to joke identification can be broken into several steps: noting the location of a joke, signaling joke detection, and incorporating joke information into future interactions. A template-based or superficial approach to joke response as shown in the riddle example above can be helpful for putting a communication partner at ease in the short term, but can be insufficient in the long term. Only knowing when to laugh is not enough.

So while being able to recognize this format can help us to learn new information, it can be actually counterproductive to fake understanding when none exists. False laughter means the speaker can no longer effectively use the joke as a checksum for audience understanding, and that the conversation can accelerate too rapidly before crucial keystone information is taught. Even if the goal were to simply evoke positive emotions from the joke teller by laughing, when the lack of understanding is revealed, emotions like disappointment or betrayal can easily arise.

You may have run into the famous "No Soap, Radio!" meta-joke that illustrates there principles:

"This prank usually requires a teller and two listeners, one of whom is a confederate who already knows the joke and secretly plays along with the teller. The joke teller says something like, "The elephant and the hippopotamus were taking a bath. And the elephant said to the hippo, 'Please pass the soap.' The hippo replied, 'No soap, radio.' " The confederate laughs at the punch line, while the second listener is left puzzled. In some cases, the second listener will pretend to understand the joke and laugh along with the others to avoid appearing foolish."

The two end states of this kind of prank demonstrate the major perils of poor humor understanding:

Negative understanding When the victim admits not understanding, and the pranksters mock them for not understanding. An inability to understand humor is alienating.

False Understanding: The victim of the joke pretends they understand, though they do not understand, and are revealed by the pranksters. This exposes the victim in a lie.

From a communication standpoint, false understanding can be dangerous, irresponsible, or at a minimum insensitive. Dark humor or self-deprecating humor is particularly prone to this fallacy, as false understanding or laughter can come off as particularly insensitive and cruel.

1.2 Exploring Humor Understanding through the Example of the Roadrunner

To develop a better understanding of how humans identify instances of humor, I will turn to an iconic example of humor: the Roadrunner series of *Looney Tunes* animated shorts, created by legendary director Chuck Jones [12]. These segments are short, wordless, and intended for a broad age range. These follow some general patterns, despite each having a unique twist and comedic effect.

The general flow of a story is as follows:

> "A coyote is hungry, and therefore sets his sights on eating a passing roadrunner. The coyote initially tries to chase the roadrunner, but he cannot catch it. Therefore, he decides to turn to guile. The coyote orders special equipment from the Acme company to set increasingly zany traps for the Roadrunner. While it is clear how the coyote intends for these traps to work, through speed, luck, or cleverness the roadrunner escapes each one. The escapes and taunts of the roadrunner increasingly frustrate the unlucky coyote. He becomes so fixated on capturing the roadrunner that he triggers a trap he himself set for the roadrunner, and is hoisted by his own petard. The coyote survives, chagrined."

This story will provide a framework for understanding the components of successful humor. The rules that Chuck Jones and his team used to create these shorts are also discussed in greater depth in section 6.3.1.

1.2.1 Background and Expectations

Notably, despite jokes often being associated with surprise or absurdity, all of the "surprises" outlined in the Roadrunner story pattern are in some manner telegraphed. As this meta-script conveys, the audience is given some expectation of how elements of the story will go overall, though they are not sure exactly how they will play into the story.

In a story without rules of any kind, anything is equally possible and therefore nothing is remarkable. An audience can never be surprised without some kind of preconception of what will happen, therefore effective comedy actually cares deeply about the knowledge and predictive rules understood by the audience. Lack of knowledge of the rule systems the world or characters operate by can easily destroy humor comprehension and appreciation.

The need for background knowledge and expectations can be observed in the following joke:

> "Q: What kind of dog does a *shtriga* have?"

"A: A bloodhound."

Unless one is familiar with Albanian folklore, it is not obvious that a *shtriga* is a global variant on the "vampire" myth. With this knowledge, we can see that while a dog still may not seem obviously useful for a vampire, the association between the word "blood" and "bloodhound" give a reason for this unexpected link.

In order to detect humor, a program must model the listener's mental trajectory. This requires a method of expressing the knowledge that a reader contributes to their understanding of a tale through background information, common sense rules, and methods of describing expectations of story behavior.

1.2.2 Underlying Meaning

Humor is a method of communication, and therefore it often has a message or intention. A deliberate act of humor is a skillful communicative act drawing upon shared storyteller and listener knowledge to misdirect. The audience is led down an initial avenue of thought that proves to be incomplete, and then realizes the existence of an equally or even more effective course correction. While some humorous incidents are found rather than created, they still represent a consistent and enjoyable roller coaster of thought that can be mapped. Humans then often recount or share enjoyable comedic moments, serving to transform even naturally occurring humor into a vessel for communication.

In the Roadrunner story, we can extract a consistent message about the perils of unprovoked aggression and fanaticism, as well as the value of having roadrunner-like traits of being clever or quick. This is because these traits are key for resolving the unexpected components of the comedy. This indicates that being able to expose the origins of components of a joke can help reveal the intention of a joke. Therefore, the information and methods we use to resolve instances of comedy can also provide useful information about what was communicated between two people laughing at that humorous moment. Reactions to naturally occurring humor also provide information on an audience's expectations and mental processes.

It is notable that an audience often finds a joke actively negative if it requires an explanation to be understood, perhaps because part of the point of a joke is a pleasant verification that both participants share relevant background information and thought processes. An explanation being required emphasizes the differences between joke creator and consumer rather than their similarities, and acts to alienate the two rather than bring them together.

When the communication act of a piece of humor is unclear, the joke also often falls flat. This can be easily observed in ambiguously sarcastic written statements, as in the case of complimenting a group after a merely average team performance in a game. Similarly, many "random" and thus unexpected events happen to us every day, yet many do not trigger sufficient mental architecture to trigger a humorous response from us.

1.2.3 Sufficient Reasons for Surprises

Surprise alone is not enough to characterize an instance of humor. In the "Roadrunner" story, if the roadrunner were to suddenly disappear or the Coyote were to become vegetarian without reason, the audience would likely be more confused than amused. Without an underlying reason understood by the audience for instances of unexpectedness, they will likely be more annoyed than pleased.

This can be exemplified by a small defective riddle:

"Q: What is green and has wheels?"

"A: Grass."

A listener with sufficient knowledge of grass, green, and wheels is definitely surprised, but likely does not find this joke very funny. However the completed joke is likely more funny:

"Q: What is green and has wheels?"

"A: Grass. I lied about the wheels."

There is now a reason for the confusion, and one that plavs on our assumptions of truthfulness in social interactions. Similarly, the misspelling of "plays" as "plavs" at the beginning of this paragraph was likely not particularly humorous; there was no obvious reason for it. On the other hand, the errors of children learning to write are often humorous, particularly when a misspelling overlaps with another correct word option. The mistaken interpretation, as well as the reason for its generation are both made clear. This can be seen in Figure 1-2.

Figure 1-2: A sign by a child intended to read "I love Santa" that instead reads "I love Satan" [19]

1.2.4 Avoidance of Killjoys

The intentional misspelling error I made at the end of the previous section exposes another issue: sometimes insufficiently resolved elements of a potentially humorous scenario can interfere with the audience finding it funny. In the case of spelling error, it is possible that the audience does not have tolerance for broken rules of this kind and will not accept any rationale as sufficient. Similarly, there are certain problems the audience may find irreparable. Annoyance at confusing delivery or broken patterns such as spelling and grammar can overwhelm a potential joke if not incorporated into the moment of humor.

The Roadrunner story can also be easily spoiled by a quick change to the final line:

"The coyote dies."

In fact, it would also be spoiled by the following:

"The coyote kills and eats the roadrunner."

In both of these jokes, mortality spoils the mood. However, this does not seem to be a constraint on all jokes, as evidenced by the following:

> "When I die, I want to die like my grandfather who died peacefully in his sleep. Not screaming like all the passengers in his car."

I argue that the audience is by default sympathetic to the coyote, because he is the viewpoint character and protagonist. This means that the audience would likely find his death too negative to be trivially resolved and enable resulting humor. Similarly, while the roadrunner is an enemy of the coyote, he does not take any aggressive actions towards the coyote. He is actively an innocent in the story, therefore we would also be disturbed by his death.

In the case of the joke about the grandfather's death, we have an expectation that grandparents are closer to death, so the obstacle to resolving this issue is not as large, particularly because this death is presented as a positive ideal. While the passengers in the car do die, we do not have the same attachment to them as we do the narrator, so the well-resolved unexpected element of the grandfather having also been in a car when he died is still humorous.

The roadrunner example also shows us that harm can be acceptable in a story if there is a resolvable reason for it. The coyote is harmed quite often within a single episode, yet we find this harm to our protagonist funny. I believe this is because the audience feels the coyote deserves these actions, due to the fact that he is the one who inflicts them on himself, and attempts to inflict them on the roadrunner.

Profanity can also interfere with humor, or alternatively present a resolvable broken expectation in the same manner as character harm can. In both cases it depends on whether the current audience has a tolerance for this kind of "harm" occurring and distracting from enjoyment of the humor, as well as how compelling the craftsmanship of the joke is.

1.3 What Is Humor?

Using techniques inspired by the mechanics of the "Roadrunner and Coyote" story, I propose a computational definition and corresponding model for recognizing and interpreting instances of humor never before encountered by the system. This definition allowed me to construct an infrastructure for humor detectors.

To aid in executing this vision, I put forward the `Expectation Repair Hypothesis` in section 1.3.1 that defines and explains humor, and three corollaries that explain our understanding of the purpose of humor, how to search for humor-dense moments, and how genres of humor are defined.

1.3.1 Expectation Repair Hypothesis

Humor requires:

Expectation Break There is a sharp shift in the initial estimation and final evaluation of the behavior in story events in one of the layers of our interpretation of a narrative. For example, we might at first interpret a word using a most-common meaning, but the final analysis would lead us to a less-common interpretation, as in the case of many dual-meaning puns. This process can also end with an ambiguity, where it's uncertain which meaning of multiple possible meanings was meant by the statement.

Different Interpretation Repair This broken expectation is then repaired by our knowledge at another layer of abstraction in our understanding. For example, in the joke "Q: Why were the raindrops so heavy? A: It was raining cats and dogs", our expectation of receiving a reason directly related to heaviness is broken, but it is repaired by an idiomatic understanding of the phrase.

Meta-patterns Intact Finally, other layers must continue acting as normal for the humor to make cohesive sense. For example, expectations of sentence structures must remain intact, or the rules of physics should continue to act consistently.

With the following corollaries:

Humor Purpose Corollary The pleasure and purpose in humor is to indirectly verify that we share a specific mental trajectory triggered by an instance of comedy with other humans, as well as all the expectations, priors, and required information for that specific path to be taken.

Humor Punch Line Corollary The punch line of a joke can be found by looking for a particularly dense concentration of expectations being rapidly broken and then fixed, so long as all broken expectations each also have a valid repair. The distribution and relative positions of these pairs of breaks and repairs can be used to extract instances of humor as a whole.

Humor Category Corollary Different kinds of humor can be characterized in terms of the pair of pattern-understanding-agents that perform the break and the repair functions.

It is notable that quantifying the exact levels of humor and surprise are supported by this model, but not the focus of this project. This model is also able to account for differences in humor recognition by unique individuals.

1.4 Implementation

As our exploration of the Roadrunner story indicated, understanding instances of humor that humans find amusing requires modeling the background information, commonsense reasoning, and expectations that humans themselves use. Using this information to implement the Expectation-Repair Hypothesis of Humor then requires:

1. Creating `Experts` that examine stories for subversions of reader expectations of a given narrative.

2. Using these `Experts` to flag surprising inflection points within a story for additional investigation and possible explanation by other `Experts`.

3. Consulting with other `Experts` to resolve these potential anomalies.

4. Analyzing these flags and any resolutions found for patterns.

1.4.1 Genesis Story Understanding System

The Genesis story understanding system is a computational architecture developed by the Genesis group at the MIT Computer Science and Artificial Intelligence Laboratory to provide a robust and versatile framework for modeling human understanding of narrative [25]. The group believes that story understanding capabilities are a keystone of human intelligence, and seeks to model the mechanisms that enable narrative comprehension in humans to better understand the workings of the human mind.

The Genesis system reads short story summaries in English, and translates these sentences into its own internal representation of a story using Boris Katz' START parser. Entities expressed in this "innerese" representation are semantically unambiguous, and provides a useful structure for story analysis. This symbolic representation of a story can be combined with similar representations of low-level common sense rules, higher level concept patterns, casual connections, and mechanisms for story understanding to uncover deeper understanding of a story and model human reasoning.

To date, the Genesis system has demonstrated story understanding capacities such as story summarization, answering questions about stories, presenting stories in a flattering or unflattering light to specific characters, reasoning hypothetically about future narrative events, applying rules specific to character personalities, detecting recurring conceptual patterns, and reflection on its own thought processes. The strong capabilities of the Genesis system in modeling human commonsense reasoning in relation to stories, as well as its emphasis on modeling accurately modeling methods of human thought provide a natural fit for the approach I have outlined for computational understanding of humor.

This project also adds useful capabilities to the Genesis story understanding ecosystem. My work enables Genesis to discover and handle surprising events ro-

bustly, and to add this capability to any new Genesis module. Potentially problematic features are highlighted by a series of `Experts` that each comment on story elements ranging from character traits to logical errors to genre shifts. The system then pinpoints elements comprising the surprising features, traces their sources, and presents reasons why these features might have occurred. I also demonstrate how these skills allow Genesis to perform three useful tasks: answering questions within the domain of each `Expert`'s knowledge, labeling potential problems and successful narrative techniques within prose, and identifying humor through patterns found in the groupings of error-solution pairs within a given story.

1.4.2 Experts

The flags for story comprehension are generated by individual `Experts` with different areas of knowledge, mediated by their membership in an `Expert Society`. The `Experts` that I have created are as follows:

Contradiction Expert Detects contradictions within the story. This `Expert` is primarily used for finding potential errors rather than repairing them.

Unexpected Expert Detects surprises in the form of unlikely events happening over the course of a story. This `Expert` also primarily finds potential problems rather than repairing them.

Ally Expert Tracks character allegiances over the course of the story, as well as who the protagonist is. It is used for investigating the status of characters the reader cares about, as well as dismissing concerns about less relevant entities.

Harm Expert This `Expert` determines whether entities within a story have been hurt over the course of the story, as well as their final condition. This expert often interacts with the `Ally Expert`, as we care about the safety of protagonists and sympathetic group members within a story.

Karma Expert This expert tracks the positive and negative actions of characters within the story. This allows us to check whether characters get their "just

rewards". It also provides initial assumptions of karma for those who are "innocent", like children or animals. Interplay between the assumptions of the Ally Expert and the Karma Expert can also allow us to find satisfaction in anti-heroes or the partial successes of sympathetic villains. This expert often works with the `Harm Expert` to investigate the status of characters with strongly positive or negative karma.

Morbidity Expert Tracks the danger level of the story. For example, a reader would be surprised to find a story for children suddenly having deadly conditions such as murder or war. Similarly, a war story is unlikely to swerve into playful or non-deadly stakes.

Trait Expert Tracks the traits that various characters have. This is primarily used for resolving investigations by other experts, and can account for non-standard behaviors by characters. For example, a character may have pulled off an unlikely escape because they are "lucky", or have behaved in a contradictory manner because they are "stupid".

1.4.3 Consulting with Experts

Each of these `Experts`, as well as any new ones created, are managed by an `Expert Society`. This entity knows of all the different kinds of `Experts`, as well as the story that the `Experts` will be called to analyze. The `Expert Society` then mediates the process of requesting any `Expert Features` that these `Experts` discover within a story, and then inquiring of other `Experts` for further information that might explain these `Features`.

The `Expert Society` tracks all `Expert Features` found for a given story, and then submits them to `Experts` to try to find additional resolutions. This flow is depicted in figure 1-3.

The `Expert Feature` object contains information on the context of a feature, the type of feature flagged, and a minimal pair of entities that represent the error discovered. In the case of a contradiction, for example, this would consist of the two

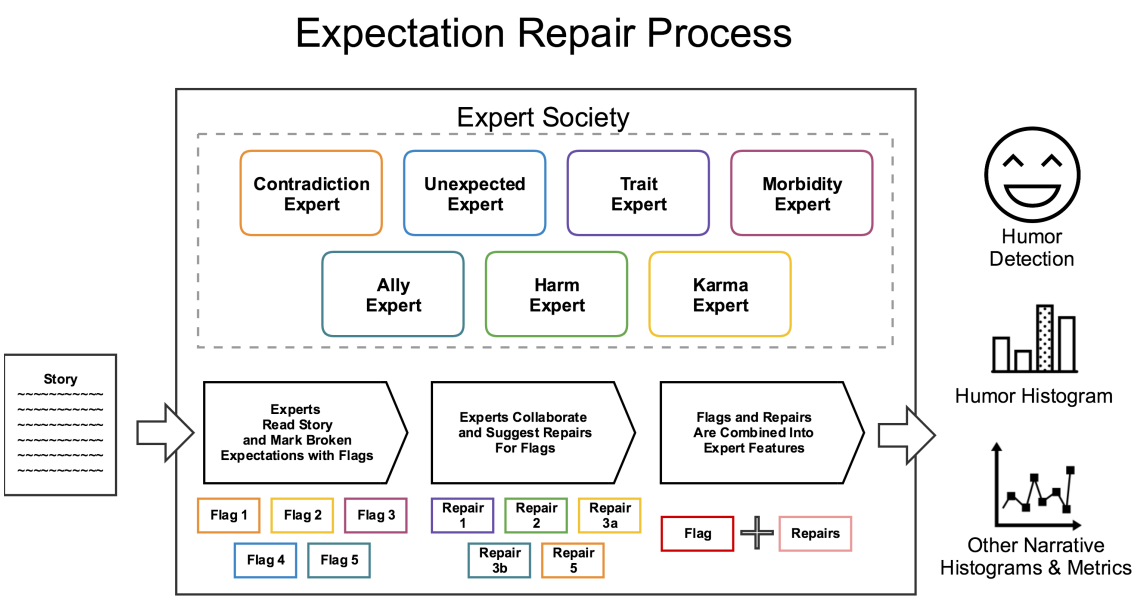

Figure 1-3: Expert Society Consultation Process

conflicting lines, as well as the flag label corresponding to a general contradiction.

Table 1.1: Anatomy of an `Expert Feature`

Field	Use
Issuer	The Expert that discovered this feature
Story	A reference to the story currently being analyzed
Flag ID	An ID indicating the kind of feature that has been found
Background Entity	The `Entity` that established our expectations
Break Entity	The `Entity` that led to something unexpected relative to `Background Entity`
Fix Entities	A list of all potential resolutions found by each other Expert.

Figure 1-5 displays an intuitive view of some potential `Expert Features` generated by a story.

1.4.4 Humor Identification

The approach to detecting humor modeled here aligns closely with the Expectation-Repair Hypothesis of Humor. We can translate its tenets to use my `Expert Feature` flagging system like so:

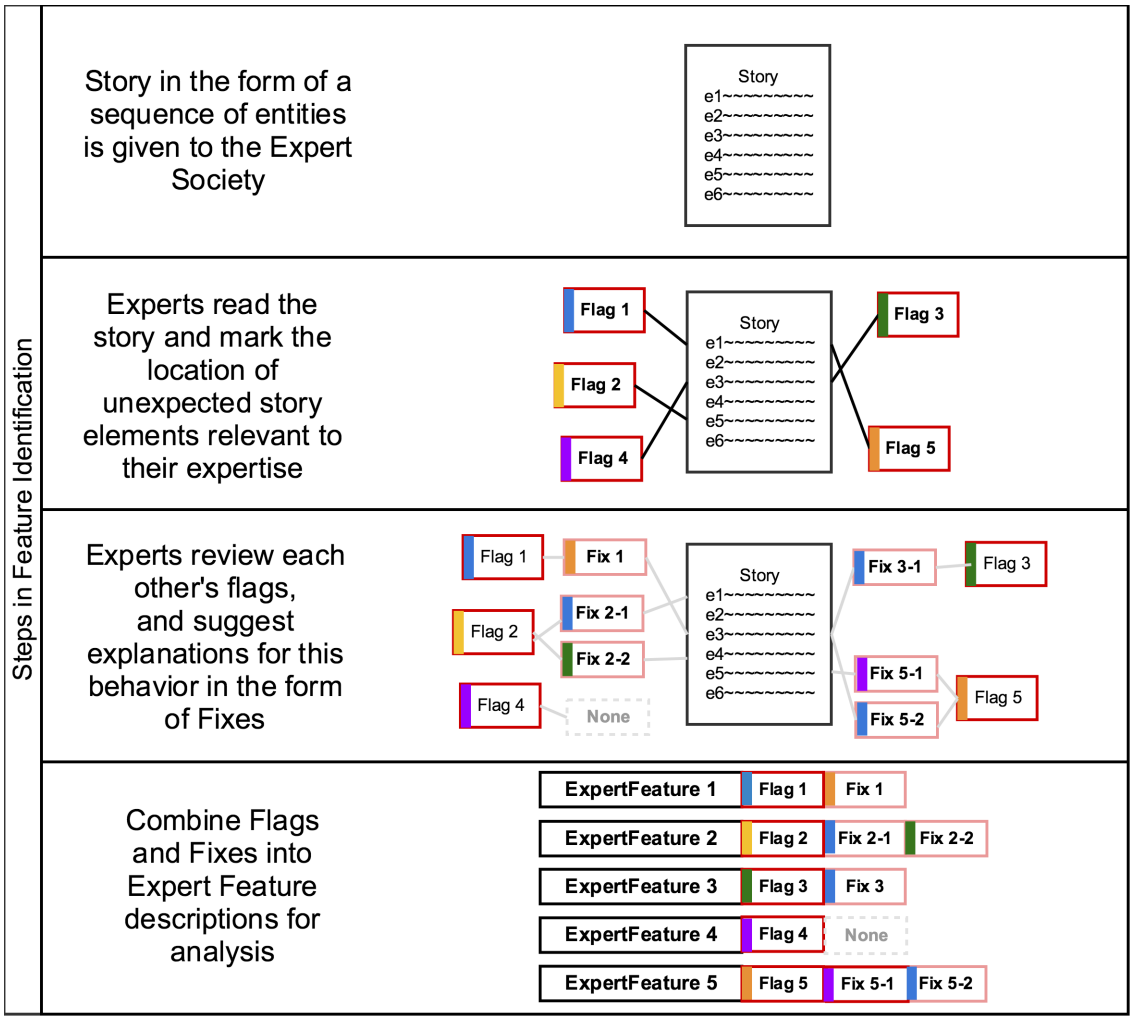

Figure 1-4: The process of finding and fixing `Expert Features`

Figure 1-5: The anatomy of the `Expert Features` and their component Flags and Fixes returned after an `Expert Society` examines a story

Expectation Break Locate and flag `Expert Features` within a story.

Different Interpretation Repair Find resolutions to each of the flagged `Expert Features` found within the story.

Meta-patterns Intact Make sure that all `Expert Features` have some kind of resolution.

As well as its corollaries:

Humor Purpose Corollary We can determine the elements shared between those who laugh at the same piece of humor by examining the commonsense rules and story units that led to `Expert Features` and their resolutions.

Humor Punch Line Corollary The punch line of a joke can be found by looking for a particularly dense concentration of `Expert Features`, and in particular the location of their Break Entities.

Humor Category Corollary Different kinds of humor can be characterized in terms of the unique pair of Experts that discovered the break and repair components of their features.

Chapter 2

Experts: Agents for Story Analysis

In my system, humor understanding is performed by a society of error detecting `Experts`. Each `Expert` detects particular kinds of features to report, analyze, and resolve bugs within its expertise. A successful joke depends on every potential problem that has been detected by an `Expert` being repaired by another story understanding Expert with knowledge of a different domain. The system can also use unresolved bugs to explain errors in narratives more generally.

My approach to humor and error handling is inspired by Minsky's *The Society of Mind*, which describes an ecosystem of simple agents that collaborate to understand more complex behaviors than they could individually. In my system, story analysis is conducted using a series of `Experts`, specialized entities that can comment on a given story. These Experts correspond to Minsky's description of *agents*, and within the context of Genesis act as specialized story-readers. An `Expert Society` moderates their interactions, acting as a kind of "chairperson" for this society by soliciting opinions and facilitating communication between Experts.

`Experts` highlight features for further investigation through the creation of an `Expert Feature`. An `Expert Society` includes a flag label describing the type of investigation being marked, the minimum information from the story required to induce this kind of feature, pointer to the issuer of the `Expert Feature`, and a free field for any further commentary. The `Expert Society` then solicits other `Experts` for resolution information for inclusion within the `Expert Feature`.

An `Expert` can either digest a completed story, or analyze the story line by line as it is recounted to them. Line by line analysis exposes moment-to-moment story understanding and how `Expert` state is updated over time. Notably, an `Expert` can make assessments using any information gleaned from a story such as words, syllables, sentence content, or additional story markup such as images. As long as the `Expert` responds in the homogeneous format that other `Experts` and the `Expert Society` can understand, its assessments can use any method. This flexibility enables agents to work together even if they use heterogeneous methods such as symbolic reasoning, neural nets, or Bayesian reasoning to come to their conclusions.

Any Genesis story understanding module can be incorporated as an `Expert` by implementing the `Expert` interface and registering with the `Expert Society`. This requires that an `Expert` be able to generate `Expert Features`, which consist of a flag labeling the feature found, and a minimal set of story fragments needed to induce that flag. These `Experts` must be able to examine the `Expert Features` generated by fellow `Experts`, to see if they can resolve them or provide additional clarity. In addition to contributing concerns in a codified flag format, these `Experts` can optionally be called upon to answer questions relevant to their expertise, generate additional markup or information, or describe their current state of understanding.

Many of the `Experts` I have implemented computationally correlate with generally unspoken but consistent narrative expectations of "good" storytelling. While the real world is unconstrained by genre conventions or any expectation of being satisfying and reasonable, narrative often embraces these constraints and their interplay. Though these expectations can be broken, as the audience we often seek a reason why.

Each `Expert` can be understood as a quantification of audience knowledge of common narrative promises. For this purpose I have built the following `Experts`, implemented as described in section 1.4.2 and representing these corresponding storytelling tropes:

Contradiction Expert A story does not contradict itself capriciously.

Unexpected Expert A story follows the expectations it foreshadows.

Ally Expert We care most about the protagonist and their allies. Characters within a story have relationships, and do not break those relationships without a reason.

Harm Expert The emotional impact of harm depends on our attachment to the victim.

Karma Expert Good things happen to good people, bad things happen to bad people.

Morbidity Expert The danger level of a narrative world stays relatively constant.

Trait Expert Characters act according to their previously established character traits.

Therefore, my society of `Experts` supports general purpose story debugging for authors composing prose. In software development, programmers often use "linter" tools to ensure code quality [8]. This kind of tool automatedly scans source code for constructs that could be problematic, and suggests potential fixes for them. The scope and severity of these issues can range widely, and users can add new "lint rules" for use by the system. Example warnings might be issued for code that diverges from stylistic conventions, does not use correct syntax, or references undeclared variables. The use of this kind of tool is particularly valuable for maintaining standards across large and complex codebases and reducing the workload of code reviewers [11].

The system I have implemented for error flagging and resolution could serve as a "linter" tool for authors composing prose. This would help authors find potential errors, areas of ambiguity, or conceptual issues in their writing with less reliance on human editing. Much like programmers implement new "linter rules" to handle new classes of problems, users of a prose "linting" `Expert Society` could create and share `Experts` that address recurring concerns. This kind of automated analysis would decrease the workload for human editors and accelerate the process of prose review.

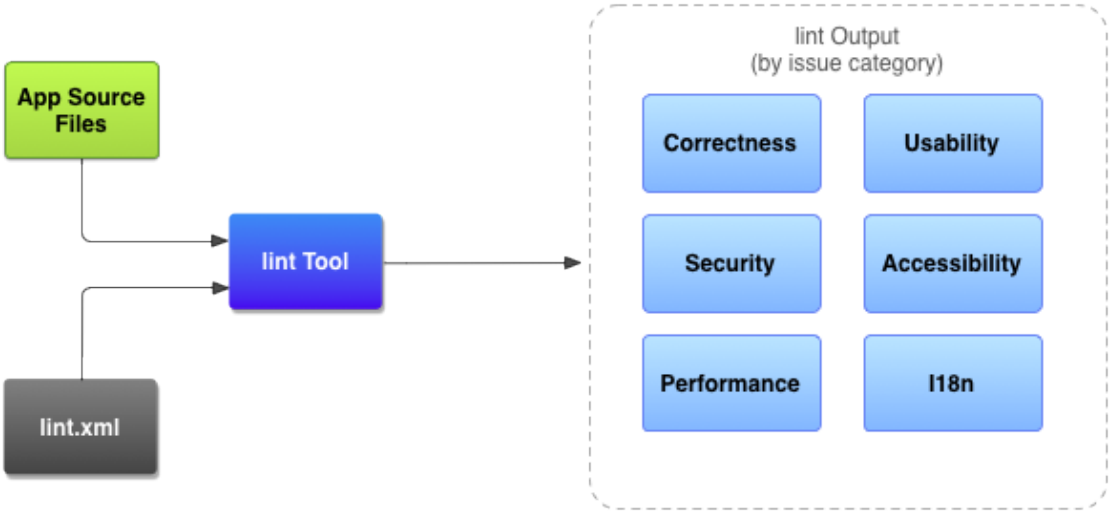

Figure 2-1: Overview of a code scanning workflow with Lint, as described by Android Studio [9]. Note the similarities to the `Expert Society` structure, input, and output.

Chapter 3

Expert Implementations

Given the narrative goals of each `Expert's` domain, the following examples aim to outline the methodology and outputs of each `Expert`.

3.1 Contradiction Expert

> Fry: *So, Leela, how about a romantic ride in one of those swan boats? They're kinda dangerous, but I finally mastered them.*
> Leela: *Those aren't swan boats, they're swans.*
> Fry: *Oh. That explains these boat eggs.*
>
> ---
>
> Futurama

The `Contradiction Expert` checks the story for contradictory events occurring. This can be created by faulty assumptions higher upstream causing two incompatible events to be added to the story, or simpler conditions such as a change of state or opinion.

This expert can help answer questions such as:

- Does the story so far contain any contradictions?
- If so, how many?
- Would adding a given additional statement cause a contradiction?
- What future statements could cause a contradiction?

3.1.1 Flags

Contradictions are flagged whenever two statements occur within the story which conflict. The minimal set of statements that constitute a contradiction are relatively straightforward: a statement and its opposite, denoted in Genesis by the feature "NOT".

Example: Alice is a dog. Alice is not a dog.

Table 3.1: Contradiction Flag 1:

Field	**Content**
Flag	CONTRADICTION-EXPERT-GENERAL
Minimal Set	["Alice is a dog.", "Alice is not a dog."]

3.1.2 Repairs

While this expert might be able provide supporting evidence for other experts to use, it does not directly provide fixes for any other Experts.

3.2 Unexpected Expert

> *Just remember every time you look up at the moon, I too will be looking at a moon. Not the same moon, obviously, that's impossible.*
>
> Andy from *Parks and Rec*

The `Unexpected Expert` checks the story for events that at some point had a low probability of happening, yet occurred anyways. To detect this, every time a line within a story contains a descriptor of how common the statement is, that statement is logged along with an estimate of its likeliness. Examples of such words would be "likely", "usually", or "rarely". For completeness, the inverse of any probable state is also logged, with the inverse probability of the original statement.

This `Expert` can therefore answer questions such as:

- Did the story ever contain any surprising turns of events?
- If so, how many?
- What future statements would be considered surprising?

Unexpected Example 1: Alice is likely happy.

Table 3.2: "Alice is likely happy."

Statement	Value	Example Estimate
"Alice is happy."	PROBABILITY-LIKELY	0.8
"Alice is (not happy)."	(1 - PROBABILITY-LIKELY)	0.2
"Alice (is not) happy."	(1 - PROBABILITY-LIKELY)	0.2

When any event actually occurs within a story, the probability of an event taking place is also logged or updated, with a probability of 100% because the event did in fact occur.

Unexpected Example 2: Alice is likely happy. Alice is happy. Bob runs to the store.

Table 3.3: "Alice is likely happy. Alice is happy. Bob runs to the store."

Statement	Value	Example Estimate
"Alice is happy."	PROBABILITY-OCCURRED	1
"Alice is (not happy)."	PROBABILITY-NONE	0
"Alice (is not) happy."	PROBABILITY-NONE	0
"Bob runs to the store."	PROBABILITY-OCCURRED	1
"Bob does not run to the store."	PROBABILITY-NONE	0

Notably, the `Unexpected Expert` tracks changes over time, so one can also request a list of past states:

Unexpected Example 3: Alice is likely happy. Alice is happy. Bob runs to the store.

Table 3.4: "Alice is likely happy. Alice is happy. Bob runs to the store."

Statement	Value Log
"Alice is happy."	[0.8, 1]
"Alice is (not happy)."	[0.2, 0]
"Alice (is not) happy."	[0.2, 0]
"Bob runs to the store."	[1]
"Bob does not run to the store."	[0]

3.2.1 Flags

This Expert flags any surprising events. This can be described using two kinds of flags, one for events that have transitioned rapidly from a high probability to a low probability, and another for events transitioning from a low probability to a high probability. The threshold that determines a sufficient shift and interpretations of probability descriptors, are left at the discretion of a specific instance of this Expert and easily configurable.

3.2.2 Repairs

This expert is not used for any repairs.

3.3 Ally Expert

> *Tragedy is when I cut my finger. Comedy is when you fall into an open sewer and die.*
>
> ---
>
> *Mel Brooks*

The `Ally Expert` specializes in tracking group allegiances over the course of a story. After a first reading of a story, the `Ally Expert` seeks an explicit declaration of a protagonist. If none is found, then it picks the first animate sentence subject within the story to be the protagonist. From there, actions and relationships between characters are characterized as beneficial or harmful and each is logged. The valence of these interactions allows us to sort characters into the broad categories of protagonists and antagonists.

This `Expert` can answer questions such as:

- Does this sentence contain a relationship between characters, and if so, is it positive or negative?
- Who is the protagonist of the story?
- Who is an ally of the protagonist?
- Who is an enemy of the protagonist?
- What factions exist within the story?
- Who changed allegiances over the course of the story?

Example: Batman fights the Joker. Robin helps Batman. Gordon does not arrest Batman.

Note that although no explicit connection was listed between the Joker and Robin, nor between Gordon and Robin, the `Ally Expert` can still intuit their relationships using the principle that "the enemy of my enemy is my friend". The `Ally Expert` can also understand double negatives, so Gordon is understood to be in a positive relationship with Batman. Gordon is therefore also in the same ally group as Robin.

Table 3.5: "Batman fights the Joker. Robin is friends with Batman. Gordon does not arrest Batman."

Group	**Members**
Protagonist	[Batman]
Protagonist Group	[Batman, Robin, Gordon]
Antagonist Group	[Joker]

3.3.1 Flags

A flag is raised for each character within the story that fits within the "protagonist" category. This correlates with the literary rule of thumb that we root for the protag-

onist, and want them to be okay at the end of the story. This kind of flag is usually resolved by a consultation by the `Harm Expert`.

The minimal statements required to describe this kind of flag are the statement that indicates a character is the protagonist, and the statement that demonstrates the relationship between the two.

Flagging Example: Cinderella is the protagonist. The fairy helps Cinderella.

Table 3.6: Ally Flag 1:

Field	**Content**
Flag	ALLY-EXPERT-PROTAGONIST
Minimal Set	[“Cinderella is the protagonist.”, “Cinderella is the protagonist.”]

Table 3.7: Ally Flag 2:

Field	**Content**
Flag	ALLY-EXPERT-PROTAGONIST-GROUP
Minimal Set	[“Cinderella is the protagonist.”, “The fairy helps Cinderella.”]

This Expert can also flag shifts of allegiances within the story in order to seek reasons for this kind of shift. If in the previous example the fairy were to betray Cinderella's trust, then:

Ally Flagging Example: Cinderella is the protagonist. The fairy helps Cinderella. The fairy tricks Cinderella.

Table 3.8: Ally Flag 3 (Betrayal):

Field	**Content**
Flag	ALLY-EXPERT-PROTAG-TO-ANTAG-GROUP
Minimal Set	[“The fairy helps Cinderella.”, “The fairy tricks Cinderella.”]

3.3.2 Repairs

This Expert usually works most closely with the Harm Expert, and can resolve harm flags by declaring a character an antagonist. In such a case, the reader might find acceptable levels of harm to that character satisfying rather than transgressive.

3.4 Harm Expert

> Black Knight: *'Tis but a scratch.*
> King Arthur: *A scratch? Your arm's off!*
>
> ---
>
> *Monty Python and the Holy Grail*

The `Harm Expert` tracks the injury, death, and recovery of characters within a story. This `Expert` can answer questions such as:

- Which characters within the story have been harmed?
- What is the status of a particular character?
- What is the history of a character's status over the course of the story?

3.4.1 Flags

While the interest of the `Harm Expert` very often intersects with the Ally Expert, because we care about the status of protagonists and protagonist-aligned characters, generally any harmed character is of interest to us. Experts that often would be able to address these kinds of issues are the `Morbidity Expert` and Ally Expert, who can testify that these levels of harm are usual for the story so far or that the characters were deserving enemies, respectively.

Harm Flag Example: Alice dies. Bob lives. Cal has fun.

Adding additional elements shows characters leaving previous Harm statuses.

Table 3.9: Harm Statuses 1:

Character	Status
Alice	HARMED
Bob	SAFE
Cal	NEUTRAL

Harm Flag Example: Alice dies. Bob lives. Cal has fun. Alice heals. Bob is reborn.

Table 3.10: Harm Statuses 1:

Character	Status
Alice	`SAFE`
Bob	`SAFE`
Cal	`NEUTRAL`

3.4.2 Repairs

This Expert can address flags of harm to individuals in the story by giving an estimate of their status within the story as `SAFE`, `NEUTRAL`, or `HARMED`. It often works with the `Ally Expert` to verify the survival of the protagonist, or check in on the status of classes of interest such as children or the innocent.

3.5 Karma Expert

You're trying to kidnap what I've rightfully stolen!

Vizzini from *The Princess Bride*

The `Karma Expert` tracks the total valence of actions by characters within a story. This `Expert` can answer questions such as:

- Which characters within the story have ever taken harmful actions?
- Which characters are perfectly innocent within the context of the story?

- What is the history of a character's status over the course of the story?

This class recognizes some entities as initially innocent (babies and animals) and therefore having large positive karma, while others start at a neutral zero karma. Harmful or aggressive actions against anyone increase negative karma, while helpful or positive actions increase positive karma.

Note that while this concept of karma is highly intuitive, the implementer has a lot of discretion about how to weight various bad deeds. These can also be categorized in a more granular fashion.

In the following example, "innocence" gives entities a default state of positive infinity karma. Harmful actions subtract one karma, and positive actions add one. Murder has a much higher penalty at negative 100 karma.

Karma Flag Example: Alice is innocent. Bob heals the sick. Cal is a rabbit. Duncan is a rabbit. Duncan harms puppies. Eve murders Bob.

Table 3.11: Karma Statuses 1:

Character	**KARMA**
Alice	infinity
Bob	+1
Cal	infinity
Duncan	-1
Eve	-100

Note that while Duncan is a rabbit and therefore starts with very good karma, once he commits a misdeed he loses the perfect karma that being an innocent rabbit grants him.

In the case of ambiguous reasoning for `FLAG-INNOCENT` or `FLAG-VERY-BAD`, the first reference to the entity in question is labeled the `Break Entity`, and the last reference to that character is considered the `Repair Entity`.

3.5.1 Flags

Flags of the Karma class operate with a sense of good things happening to good people, and bad things happening to bad people. This allows us to take satisfaction from negative statuses happening to an antihero protagonist, or allow us to enjoy the successes of an incredibly virtuous stranger or even enemy. Therefore we raise flags to investigate the end status of:

- Innocent creatures remaining alive and unharmed.
- Sufficiently evil creatures being harmed.

3.5.2 Repairs

This Expert can address flags from the `Harm Expert` by noting that a character has "bad karma", therefore harm to them is satisfying rather than upsetting.

3.6 Morbidity Expert

What has four legs and flies? A dead horse.

Anonymous

The `Morbidity Expert` tracks the level of violence of a story. This expert essentially answers the question of what the "movie rating" of a story might be. If a story is grim and gritty, it is unexpected that is would rapidly shift to realms we do not associate with violence such as children, toys, animals, or the elderly. Similarly, the reader's expectation is usually that these "harmless" topics or characters will turn truly deadly.

A classic example of this in comedy is when a seemingly deadly gun does not shoot bullets, but instead displays a harmless flag which says "BANG". In the following comic, the Joker subverts the audience's morbidity expectations not once but twice.

This `Expert` uses a state machine to determine morbidity expectations and levels of surprise. At the start of a story, the expert is in a neutral mode. As soon as

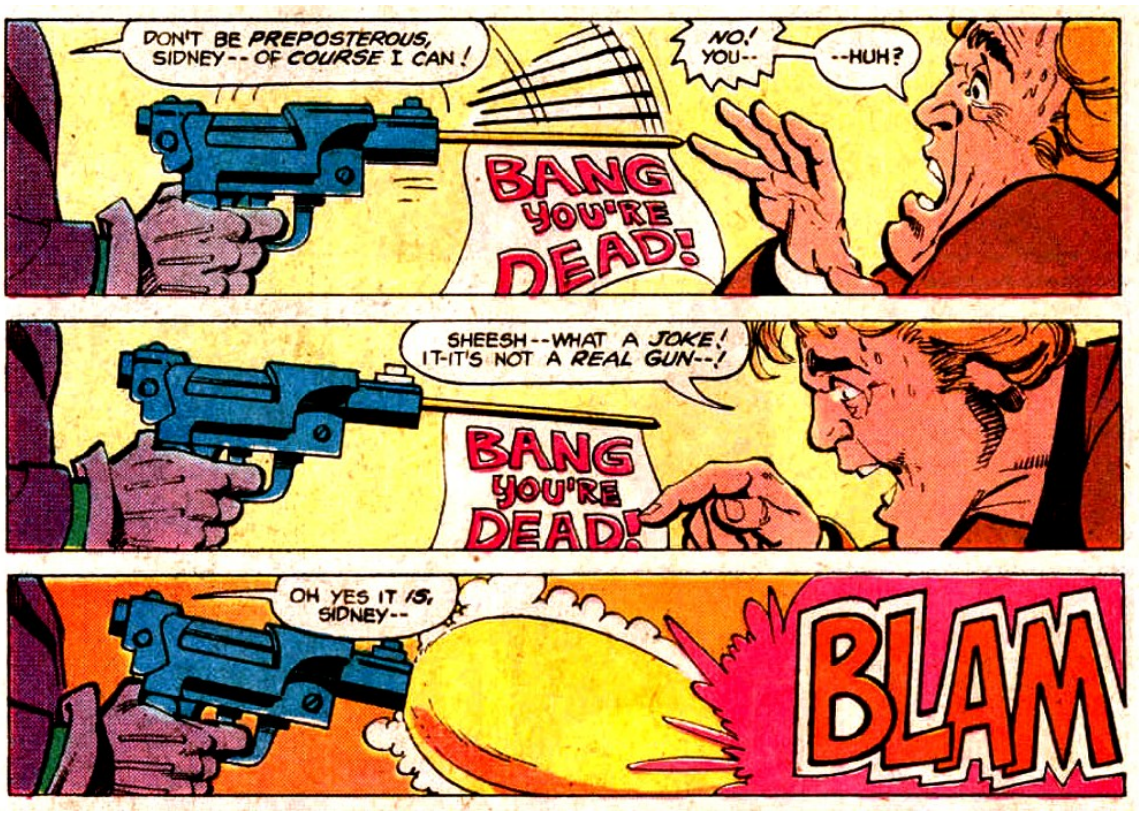

Figure 3-1: Morbidity Humor in Batman #321. [24]

signs of morbidity level are found among the words in a sentence, the state machine shifts to expect more content of that category. These categories include "harmful", "deadly", or "safe". If a dramatic shift in morbidity occurs, then a flag is raised. This currently only happens when a large shift occurs from "deadly" to "safe", or "safe" to "deadly".

The `Morbidity Expert` can answer questions such as:

- Is a given word "harmful", "deadly", or "safe"?
- What is the expected level of danger of this story?

3.6.1 Flags

Flags are found by examining individual words within the story for morbidity levels. This means that a single sentence can trigger a flag for deeper investigation.

Morbidity Flagging Example: Alice is a girl. Grandma is friends with Alice. The wolf kills Grandma.

Table 3.12: Morbidity Flag:

Field	Content
Flag	MORBIDITY-EXPERT-SAFE-TO-DEADLY
Minimal Set	["Alice is a girl.", "The wolf kills Grandma."]

3.6.2 Repairs

This expert does not perform any repairs. In the future it could be used to confirm genre expectations for categories such as "war documentary" or "kindergarten picture book".

3.7 Trait Expert

I wasn't a failed DJ. I was pre-successful.

Jason from *The Good Place*

The `Trait Expert` tracks the traits and characteristics of entities within the story. This `Expert` can therefore answer questions such as:

- What are the traits of a given character?
- Which characters in the story have a certain trait?
- What traits are currently active in the story?
- What future traits might make sense for the story so far?

3.7.1 Flags

The `Trait Expert` does not flag any issues for investigation, because it is fairly common for characters to have traits they do not apply in all stories.

A more stringent check might require that all explicitly mentioned character traits are used within the story. This could be considered a Chekhov's gun of character traits, the dramatic principle where any trait directly mentioned should eventually be either used or trimmed from the story altogether by the editor.

3.7.2 Repairs

The `Trait Expert` is a versatile tool for suggesting reasons for strange behaviors flagged by other experts. After all, a trait is by definition an abnormal feature in addition to our generic expectations of an entity. In general, the `Trait Expert` examines pairs of Entities generated by other experts, and looks for classifications that it knows can mitigate features those two entities have in common.

For example, given the story: "The roadrunner is fast. The roadrunner is clever. The roadrunner likely does not escape. The roadrunner escapes."

The `Trait Expert` is presented with a Feature highlighted by the `Unexpected Expert`.

Table 3.13: `Expert Feature` of "The roadrunner is fast. The roadrunner is clever. The roadrunner likely does not escape. The roadrunner escapes."

Field	**Value**
Issuer	Unexpected Expert
Story	*see above*
Flag ID	`FLAG-UNEXPECTED`
Background Entity	"The roadrunner likely does not escape."
Problem Entity	"The roadrunner escapes."
Fix Entities	*none*

The `Trait Expert` examines the minimal pair of `Entities` contained in this `Expert Feature` for similarities and differences. In this case, both share the subject "roadrunner" and the verb "escape". Over the course of the feature, escape becomes more likely. The `Trait Expert` therefore examines the traits it has noted as belonging to the subject ("fast" and "clever"), and looks to see if either might be able to explain higher likeliness of escape.

This story would therefore be repaired by the `Entity` of:
"The roadrunner is clever."

The `Trait Expert` currently uses a variety of lookup tables to determine which scenarios traits can affect. In the future, `ConceptNet` or other tools could be used to identify relevant traits within the story or suggest additional relevant traits.

Chapter 4

Experts in Action: Flagging Surprises

This section displays a few minimal examples for each `Expert`. These short stories cause each `Expert` to flag corresponding `Features` for investigation by creating new `Expert Features`.

4.1 Flag Examples

4.1.1 Contradiction Expert

"Alice is red. Alice is not red."

Figure 4-1: An example story that the `Contradiction Expert` would issue a flag on.

Table 4.1: `Expert Feature` of Contradiction Investigation

Field	Use
Issuer	Contradiction Expert
Story	*Figure 4-1*
Flag ID	`FLAG-CONTRADICTION`
Background Entity	"Alice is red"
Problem Entity	"Alice is not red"
Fix Entities	*none*

4.1.2 Unexpected Expert

"Alice is red likely. Alice is not red."

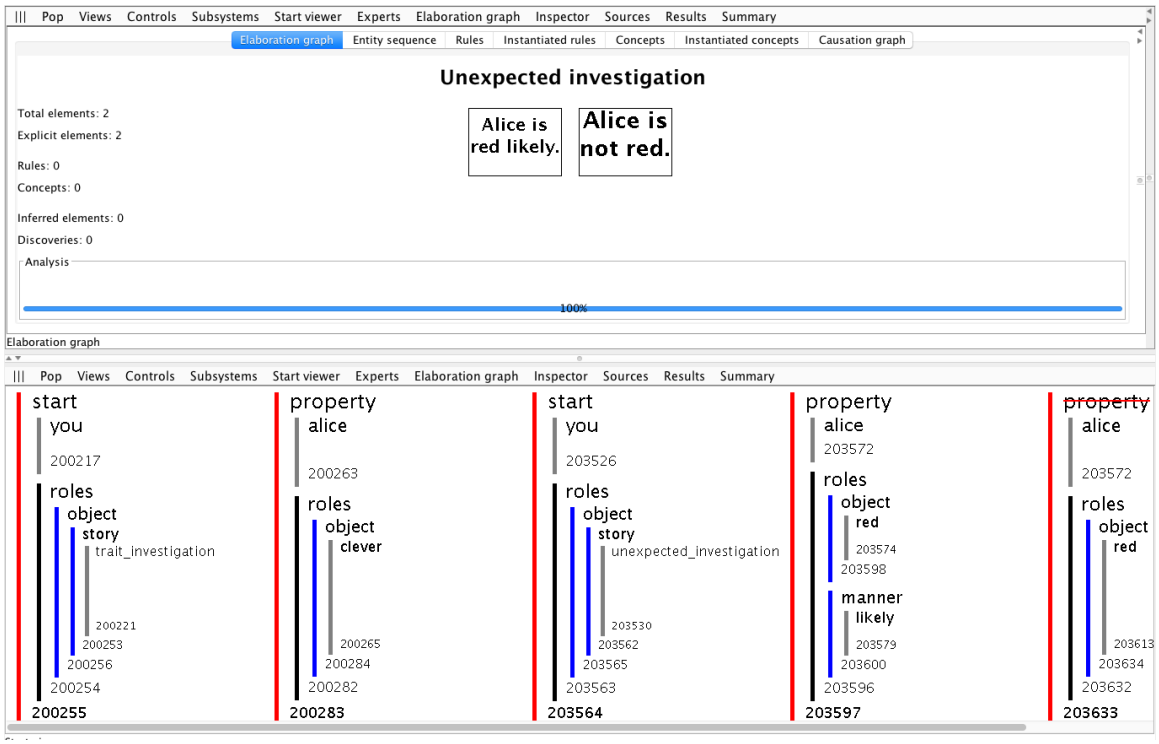

Figure 4-2: An example story that the `Unexpected Expert` would issue a flag on.

Table 4.2: `Expert Feature` of Unexpected Investigation

Field	Use
Issuer	Unexpected Expert
Story	*Figure 4-2*
Flag ID	`FLAG-UNEXPECTED`
Background Entity	"Alice is likely red"
Problem Entity	"Alice is not red"
Fix Entities	*none*

4.1.3 Ally Expert

"Alice is out shopping. Brenda attacks Alice. Christina defends Alice."

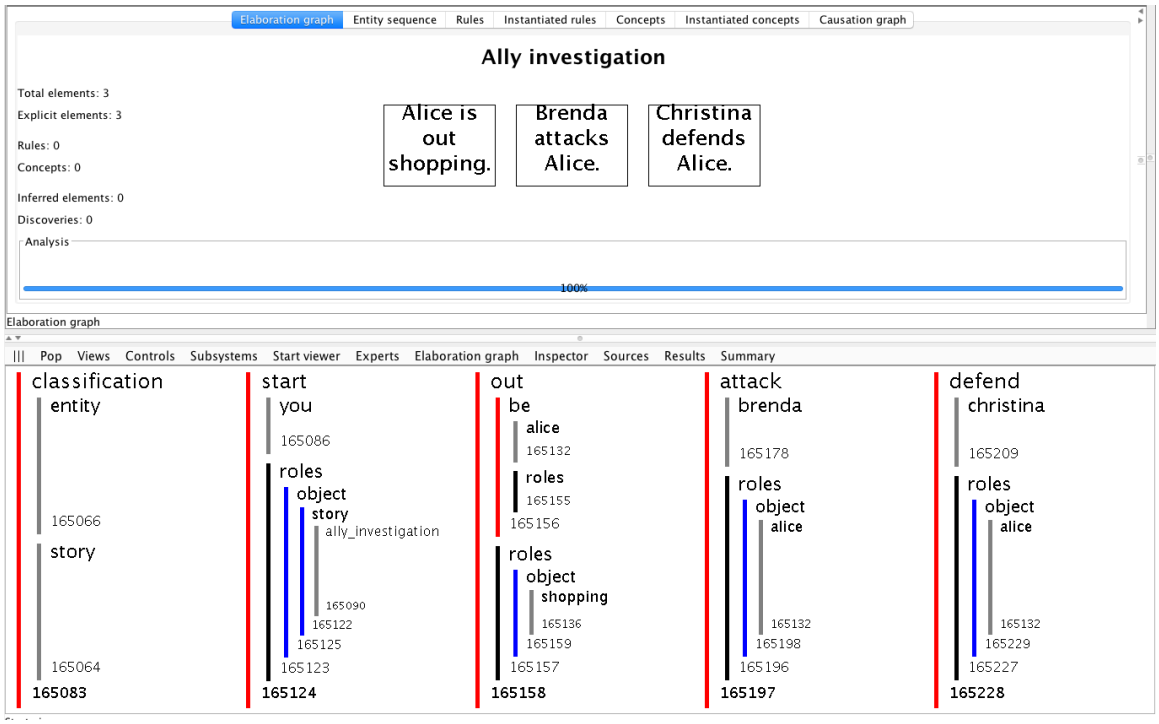

Figure 4-3: An example story that the `Ally Expert` would issue a flag on.

Table 4.3: `Expert Feature` of Ally Investigation

Field	Use
Issuer	Ally Expert
Story	*Figure 4-3*
Flag ID	`FLAG-ALLY-PROTAGONIST`
Background Entity	"Alice is out shopping"
Problem Entity	"Alice"
Fix Entities	*none*

4.1.4 Harm Expert

"Alice dies. Bob is hurt. Cal murders a person."

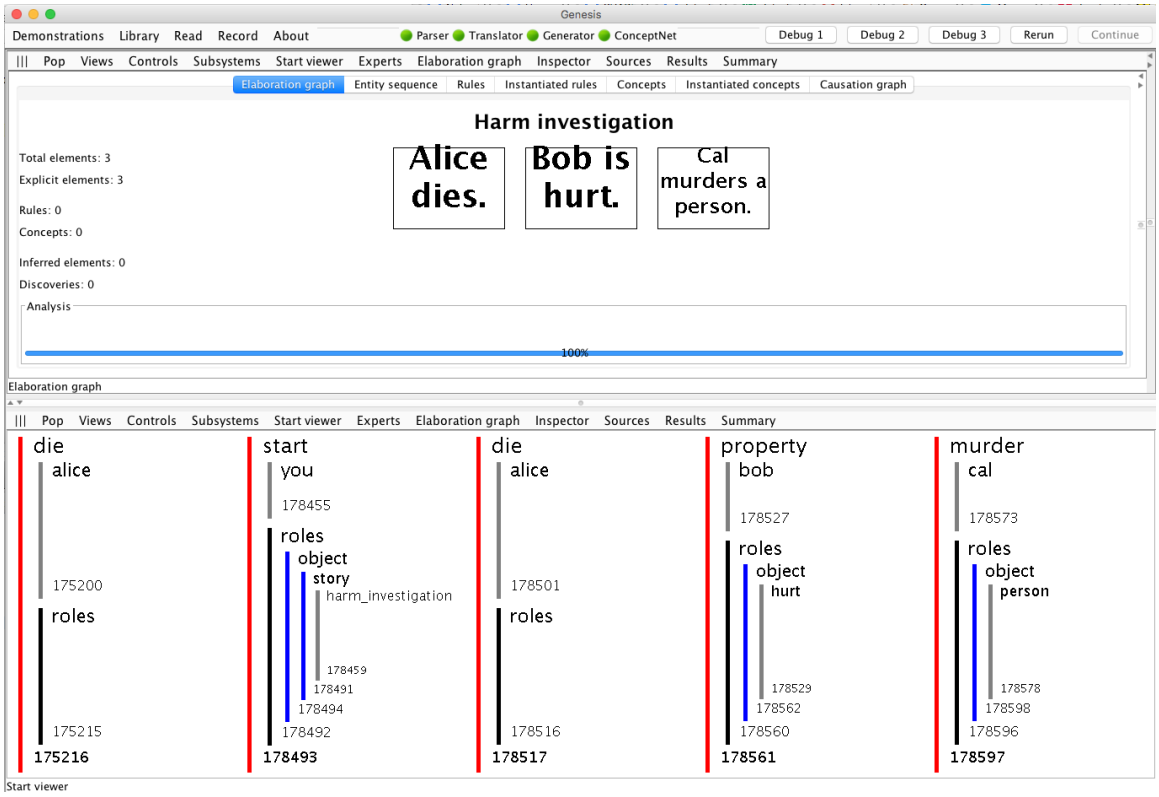

Figure 4-4: An example story that the `Harm Expert` would issue a flag on.

Table 4.4: `Expert Feature` of Harm Investigation

Field	Use
Issuer	Harm Expert
Story	*Figure 4-4*
Flag ID	`FLAG-HARM-KILLED`
Background Entity	"Alice dies."
Problem Entity	"Alice dies."
Fix Entities	*none*
Issuer	Harm Expert
Story	*Figure 4-4*
Flag ID	`FLAG-HARM-HARMED`
Background Entity	"Bob is hurt."
Problem Entity	"Bob is hurt."
Fix Entities	*none*

4.1.5 Karma Expert

"Alice is a baby. Bob shot the sheriff."

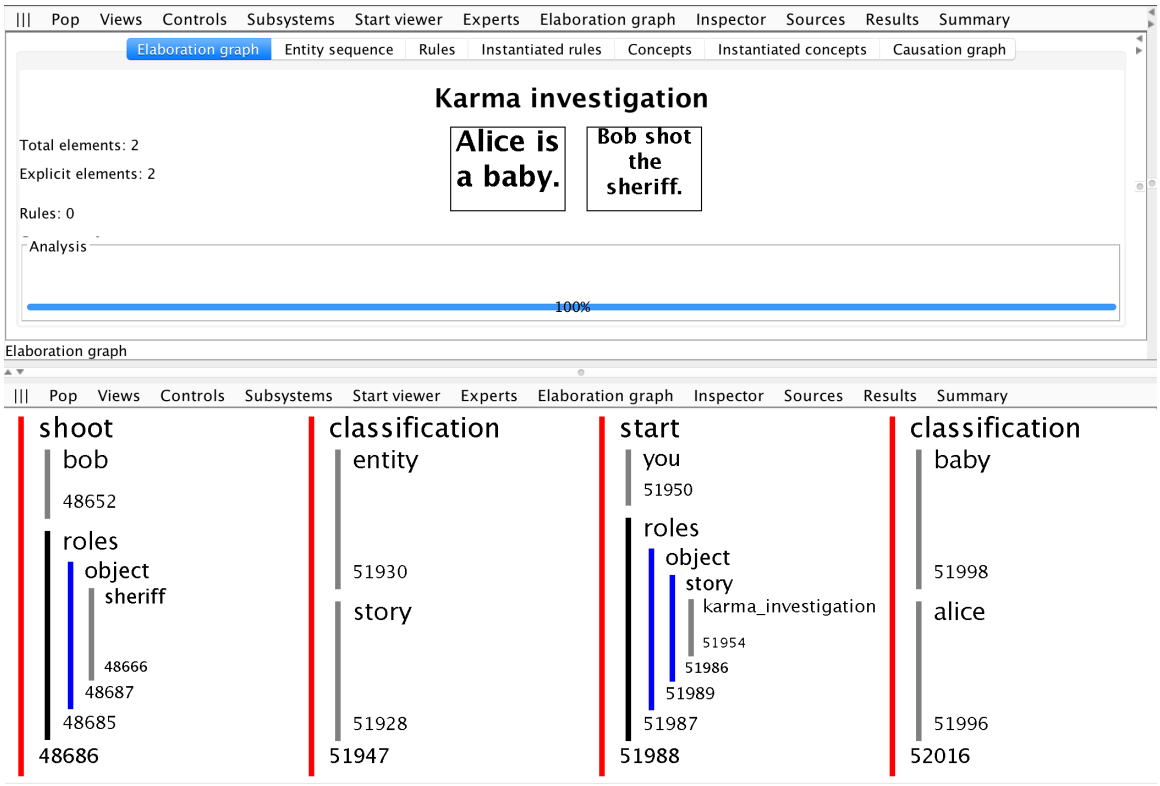

Figure 4-5: An example story that the `Karma Expert` would issue a flag on.

Table 4.5: `Expert Feature` of Karma Investigation

Field	Use
Issuer	Karma Expert
Story	*Figure 4-5*
Flag ID	`FLAG-KARMA-INNOCENT`
Background Entity	"Alice is a baby."
Problem Entity	"Alice is a baby."
Fix Entities	*none*
Issuer	Karma Expert
Story	*Figure 4-5*
Flag ID	`FLAG-KARMA-VERY-BAD`
Background Entity	"Bob shot the sheriff."
Problem Entity	"Bob shot the sheriff."
Fix Entities	*none*

4.1.6 Morbidity Expert

"Alice is a cute baby bunny. Alice is murdered in war."

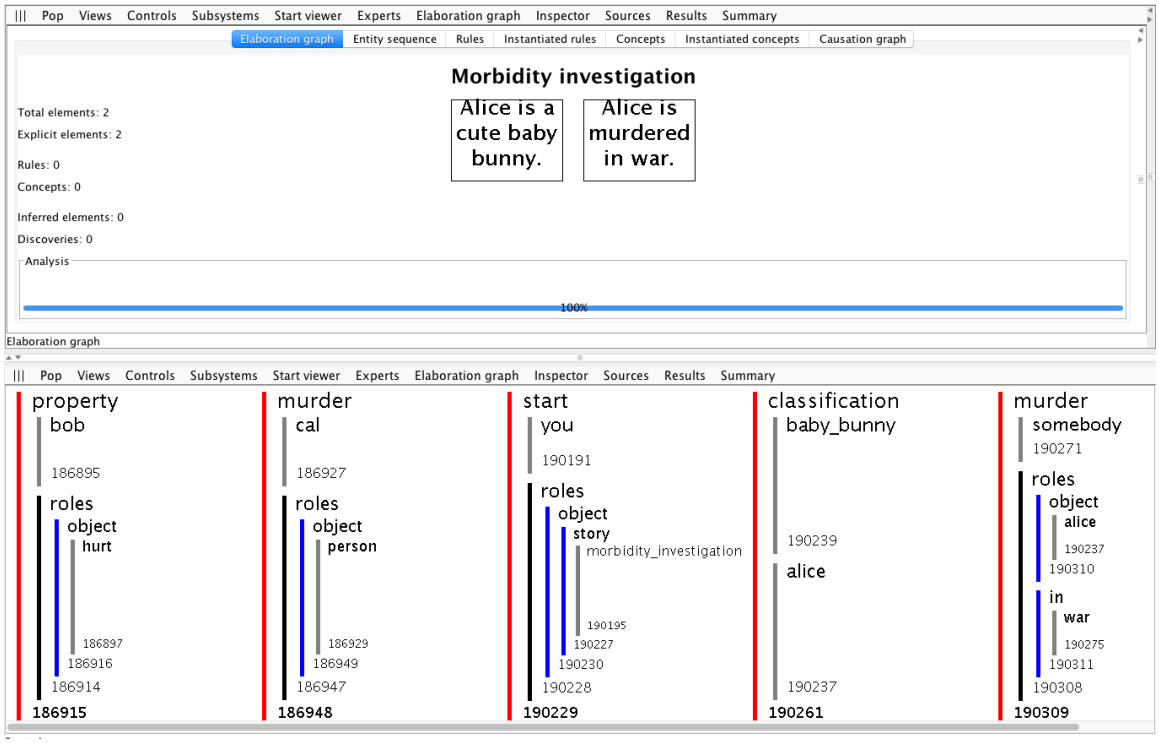

Figure 4-6: An example story that the `Morbidity Expert` would issue a flag on.

Table 4.6: `Expert Feature` of Morbidity Investigation

Field	Use
Issuer	Morbidity Expert
Story	*Figure 4-6*
Flag ID	`FLAG-MORBIDITY-HARMLESS-TO-DEADLY`
Background Entity	"Alice is a cute baby bunny."
Problem Entity	"Alice is murdered in war."
Fix Entities	*none*

4.1.7 Trait Expert

"Alice is clever."

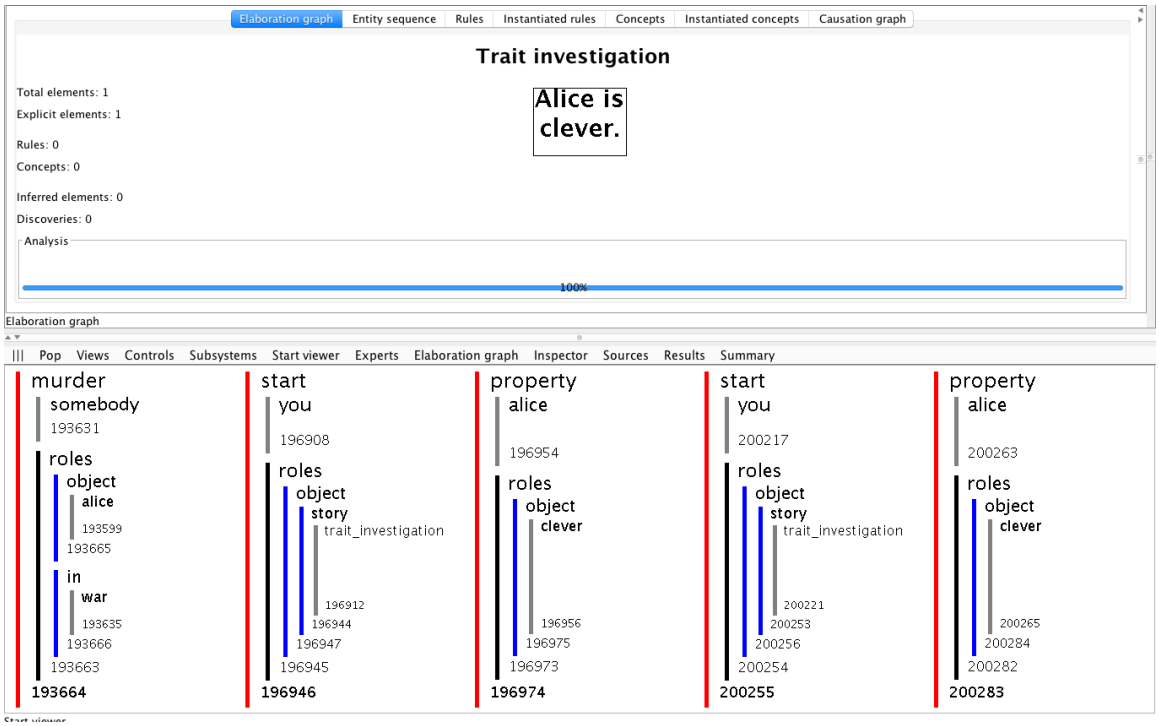

Figure 4-7: An example story that the `Trait Expert` could issue a flag on.

Table 4.7: `Expert Feature` of Trait Investigation

Field	**Use**
Issuer	Trait Expert
Story	*Figure 4-7*
Flag ID	`FLAG-TRAIT-USED`
Background Entity	"Alice is clever."
Problem Entity	"Alice is clever"
Fix Entities	*none*

Chapter 5

Experts in Collaboration: Resolving Confusion

5.1 Expert Resolution Relationships: A Hierarchy of Abstraction

Table 5.1 lists the `Expert` repair relationships I have modeled using the Genesis Story Understanding System and my `ExpertSociety`.

Adding additional `Experts` to the system in the future can add more options for possible resolutions by any existing experts, and can also gain benefit from examining the trace of a `Feature`.

Interestingly, looking at these `Experts` and their repair relationships, it seems that they can be arranged in a hierarchy of abstraction. At the lowest level are `Experts` that deal with the individual entities within a story and are focused on the literal elements of a story, `Contradiction Expert` and `Unexpected Expert`. Above those in the hierarchy are `Experts` that track more sophisticated concepts such as character actions, interactions, and statuses. At the top of the order are `Experts` assessing patterns in entities and patterns in entity relationships, the `Morbidity Expert` and `Trait Expert`. A diagram of these relationships can be found in Figure 5-1.

I suspect that this underlying hierarchy provides a reason why puns are considered

Table 5.1: `Expert Feature` Flags to Possible Resolutions

Issuer	**Flag**	**Resolver**
Contradiction Expert	`FLAG-CONTRADICTION-GENERAL`	Trait Expert
Unexpected Expert	`FLAG-LIKELY-TO-UNLIKELY`	Trait Expert
Unexpected Expert	`FLAG-UNLIKELY-TO-LIKELY`	Trait Expert
Ally Expert	`FLAG-ALLY-PROTAGONIST`	Harm Expert
	`FLAG-ALLY-PROTAGONIST-GROUP`	Harm Expert
	`FLAG-ALLY-BETRAYAL`	Trait Expert
Harm Expert	`FLAG-HARM-HARMED`	Trait Expert
		Morbidity Expert
		Ally Expert
		Karma Expert
Karma Expert	`FLAG-KARMA-INNOCENT`	Harm Expert
		Morbidity Expert
		Trait Expert
Karma Expert	`FLAG-KARMA-VERY-BAD`	Harm Expert
		Morbidity Expert
		Trait Expert
Morbidity Expert	`FLAG-MORBIDITY-DEADLY-TO-SAFE`	Trait Expert
	`FLAG-MORBIDITY-SAFE-TO-DEADLY`	Trait Expert
Trait Expert	optional: *`FLAG-TRAIT-USE`*	—

the "lowest form of wit". Homophonic puns require repairing breaks in word meaning with phonetic sub-components, one of the lowliest substructures of language, and this repair moves down this hierarchy rather than up it.

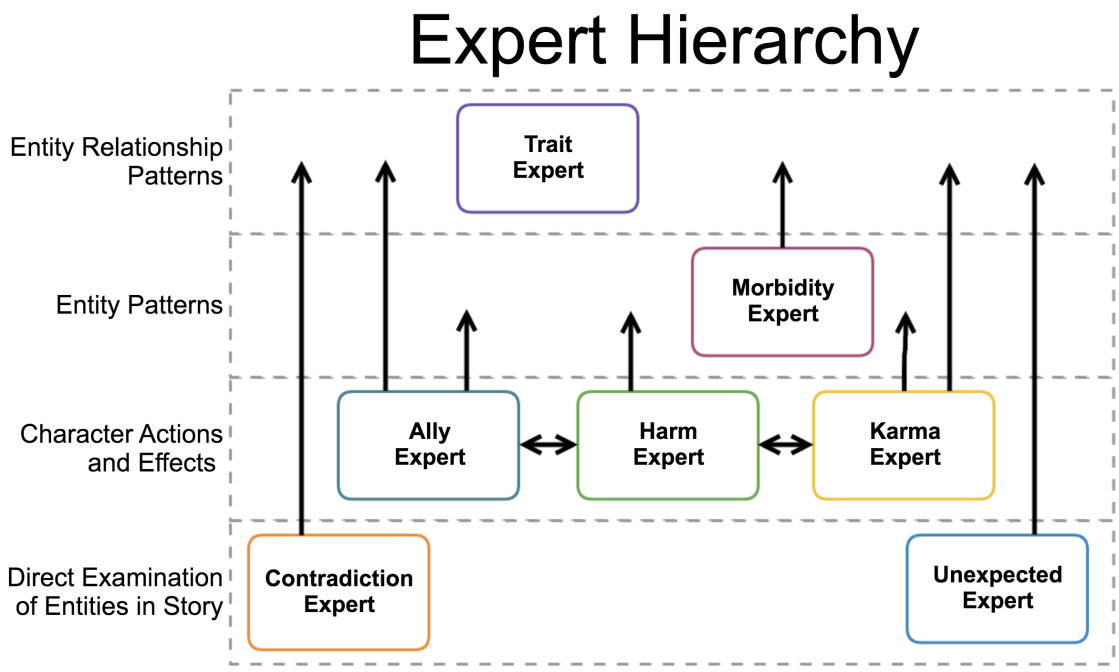

Figure 5-1: The relative abstraction of `Experts` within the story. Arrows go from flagging `Expert` to repairing `Expert`

5.2 Non-Humorous Resolution Examples

For each of the following, the story is shown in Genesis as well as in plain English text. Not all `Features` are listed, just those relevant to the `Expert` being demonstrated. This is to avoid very common `Experts` such as `Ally Expert` from dominating the examples.

5.2.1 Contradiction Expert Resolution

"Alice is red. Alice is not red. Alice is multicolored."

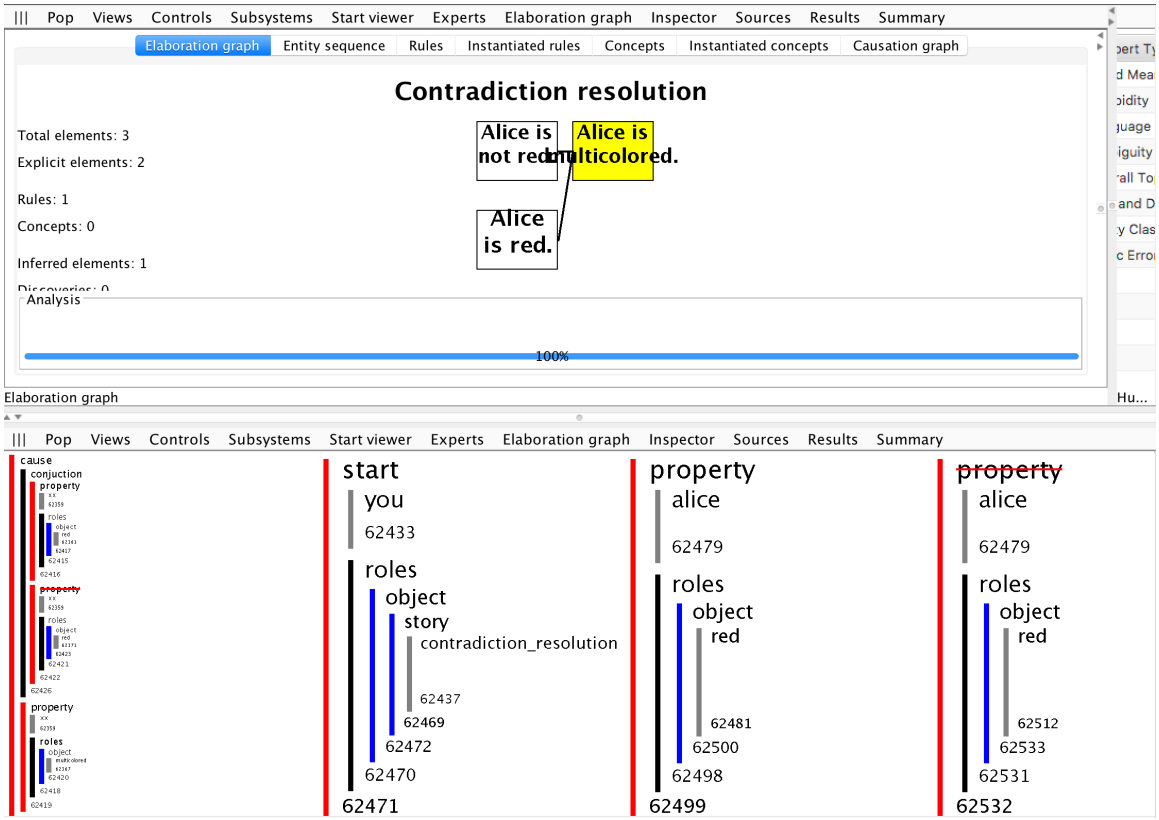

Figure 5-2: Contradiction flag being resolved.

Table 5.2: `Expert Feature` of Contradiction Resolution

Field	Use
Issuer	Contradiction Expert
Story	*Section 4.2.1*
Flag ID	`FLAG-CONTRADICTION-GENERAL`
Background Entity	"Alice is red."
Problem Entity	"Alice is not red."
Fix Entities	Trait: "Alice is multicolored."

5.2.2 Unexpected Expert

"Alice is clever. It is unlikely Alice escapes. Alice escapes."

Table 5.3: `Expert Feature` of Unexpected Resolution

Field	Use
Issuer	Unexpected Expert
Story	*Section 4.2.2*
Flag ID	`FLAG-UNEXPECTED-GENERAL`
Background Entity	"It is unlikely Alice escapes."
Problem Entity	"Alice escapes."
Fix Entities	Trait: "Alice is clever."

5.2.3 Ally Expert

"Bob helps Alice and Bob kills Alice because Bob is untrustworthy."

Figure 5-3: Ally flag being resolved.

Table 5.4: `Expert Feature` of Ally Resolution

Field	Use
Issuer	Unexpected Expert
Story	*Section 4.2.3*
Flag ID	`FLAG-ALLY-PROTAGONIST`
Background Entity	"Bob helps Alice."
Problem Entity	"Bob helps Alice."
Fix Entities	Harm: "Bob is okay."
Issuer	Unexpected Expert
Story	*Section 4.2.3*
Flag ID	`FLAG-ALLY-BETRAYAL`
Background Entity	"Bob helps Alice."
Problem Entity	"Bob betrays Alice."
Fix Entities	Trait: "Bob is untrustworthy."

5.2.4 Harm Expert

"Alice goes on a quest. A random peasant dies."

Table 5.5: `Expert Feature` of Harm Resolution

Field	Use
Issuer	Harm Expert
Story	*Section 4.2.4*
Flag ID	`FLAG-HARM`
Background Entity	"A random peasant dies."
Problem Entity	"A random peasant dies."
Fix Entities	Ally: "A random peasant dies."

Note in this case that the peasant is not attached to any ally groupings, so the `Ally Expert` knows that harm to them can be safely ignored.

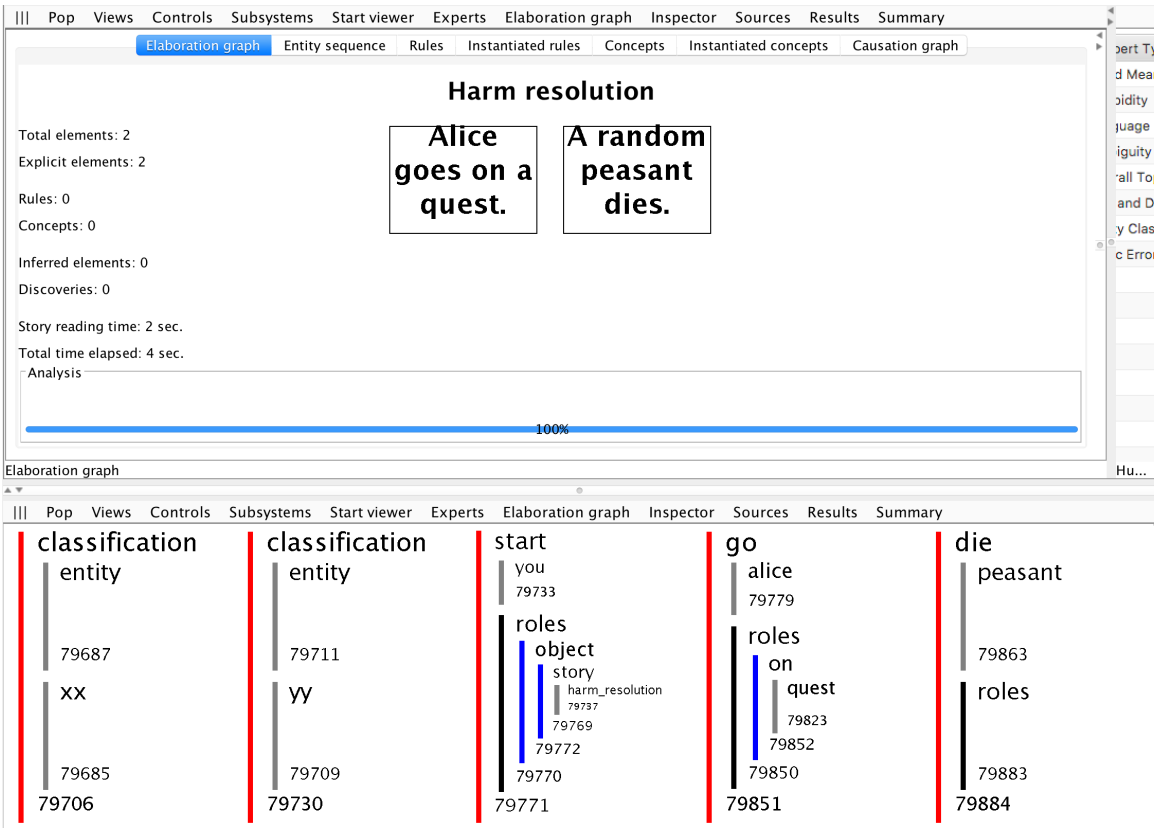

Figure 5-4: Harm flag being resolved.

5.2.5 Karma Expert

"Alice kills a man. Alice is punched. Bob is kind. Bob wins a prize."

Figure 5-5: Karma flags being resolved.

Table 5.6: `Expert Feature` of Karma Resolution

Field	Use
Issuer	Karma Expert
Story	*Section 4.2.5*
Flag ID	`FLAG-INNOCENT`
Background Entity	"Bob is kind."
Problem Entity	"Bob is kind."
Fix Entities	Ally: "Bob wins a prize."
Issuer	Karma Expert
Story	*Section 4.2.5*
Flag ID	`FLAG-VERY-BAD`
Background Entity	"Alice kills a man."
Problem Entity	"Alice kills a man."
Fix Entities	Ally: "Alice is punched."

5.2.6 Morbidity Expert

Safe to Deadly

"Alice is a baby. Alice becomes disillusioned. Alice fights in a war."

Deadly to Safe

"Bob is a soldier. Bob becomes old. Bob adopts several animals."

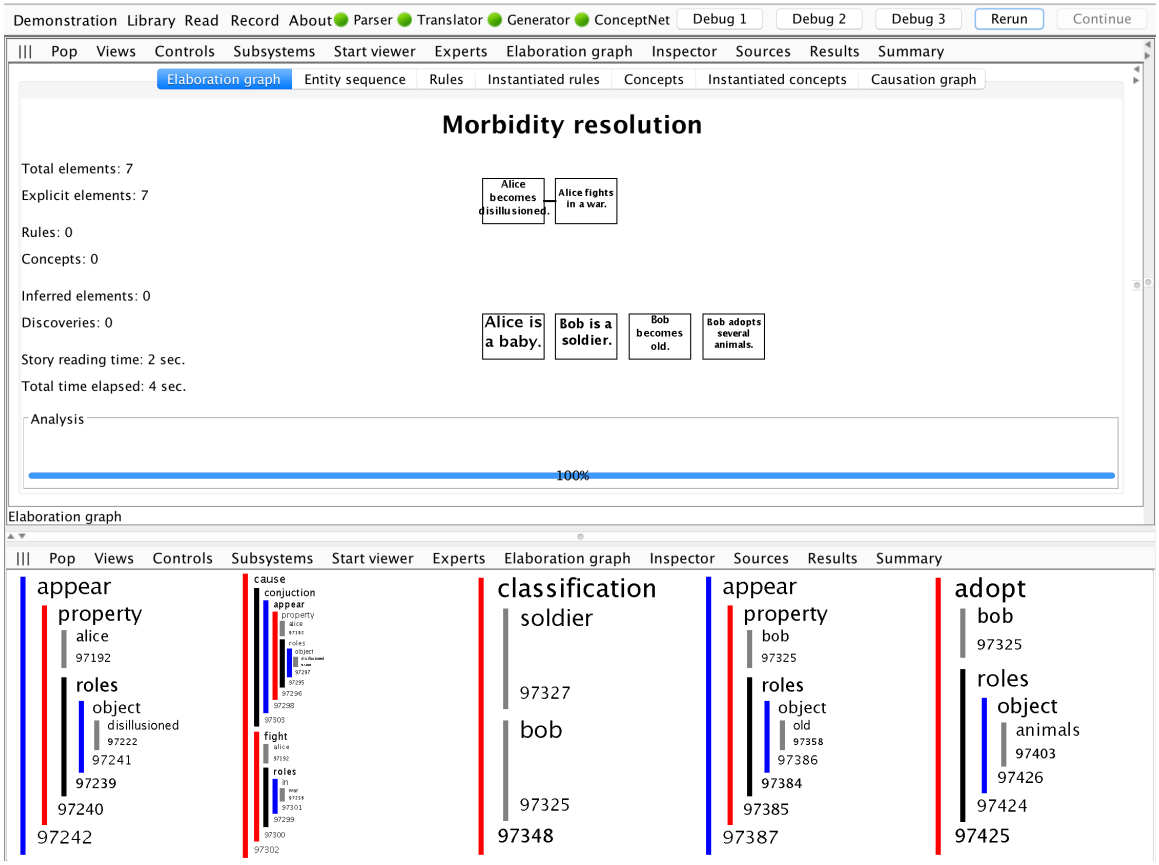

Figure 5-6: Morbidity flags being resolved

Table 5.7: `Expert Feature` of Morbidity Resolution

Field	Use
Issuer	Karma Expert
Story	*Section 4.2.6*
Flag ID	`FLAG-DEADLY-TO-SAFE`
Background Entity	"Bob is a soldier."
Problem Entity	"Bob adopts several animals."
Fix Entities	Trait: "Bob becomes old."
Issuer	Karma Expert
Story	*Section 4.2.6*
Flag ID	`FLAG-SAFE-TO-DEADLY`
Background Entity	"Alice is a baby."
Problem Entity	"Alice fights in a war."
Fix Entities	Trait: "Alice becomes disillusioned."

5.2.7 Trait Expert

None. The `Trait Expert` does not currently have an `Expert` that can resolve its flags. This is possibly because it is at the highest level of abstraction and dominates all the other current `Experts`.

As previously mentioned, intuitively a `Trait` resolution could occur if the `Trait Expert` was more strict, and raised a flag to make sure every `Trait` was evidenced in the text. This could be executed by adding an `Expert` that tracked multiple instances of behavior consistent with a `Trait` across a story. Doing this could enforce the literary rule of thumb of "Show Not Tell" for character development. This rule encourages the storyteller to convince the audience of world building properties or character traits by demonstrating them multiple times throughout a story rather than simply declaring them. As I investigate in the following chapter, opening this kind of `Feature` can be a strong strategy for keeping the audience engaged with the story.

Chapter 6

Feature Metapatterns and Humor Identification

6.1 Analyzing `Expert Features`

As we begin to look for patterns in the resolution of `Features`, it becomes clear that each individual `Feature` has several interesting metrics that we can use to describe it further. These include:

- Whether a `Feature` is resolved or not
- The relative order of the three entities within the `Expert Feature`: `Background Entity`, `Break Entity`, and `Repair Entity`
- The temporal separation between these entities
- Which `Experts` participated

Each of these components can individually contribute to our understanding of a story, as well as contributing to trends in the overall distribution of `Expert Features` and their sub-features across a story.

6.1.1 "Bug or a Feature?": Feature Resolution Status

The simplest application of `Features` is to examine a story for errors. Any Feature that pinpoints an issue that is not resolved by any other `Expert` can be considered an error.

Though unresolved `Features` are classified as errors, there are several places an individual problem could come from, and it is at the discretion of the user to decide how to address it. Errors can indicate areas for improvement by the storyteller, or areas where additional understanding or missing background information is required by the reader. Alternatively, they can be indicators that a new `Expert` might be needed, particularly when the author feels multiple unresolved errors share a common and generalizable resolution method.

6.1.2 Explanation or Realization: Order of Feature Components

While the graph of human understanding of a story can be incredibly convoluted, the story itself is delivered in a serialized manner. This means that the components of a completed Feature must have an order, and we can extract meaning from this order.

The three Entities that comprise a `Feature` can be ordered several different ways, and each has a different effect on the reader. While the `Background Entity` always comes before the `Break Entity` by definition, the `Repair Entity` can be in several places in the story relative to them both.

Most notable is whether the `Repair Entity` is after the `Break Entity` or not. Interestingly, all three of these categorizations represent methods of keeping the reader engaged with the text.

`Explanation Feature`: **Background-Break-Repair**

> "You know a conjurer gets no credit when once he has explained his trick; and if I show you too much of my method of working, you will come to the conclusion that I am a very ordinary individual after all."
>
> Sherlock Holmes, *A Study in Scarlet*

This kind of `Feature` provides a setup, breaks those expectations, and only afterwards explains why that break in expectations was allowable. This flow can be thought of as an Explanation, because only after the problem was found by the reader were those actions "explained." It is also notable that there is a unit of suspense added by this kind of feature, because the reader does not yet know the solution and cannot "close out" the `Feature` until a resolution is found.

This kind of `Feature` is frequent in pedagogy, as well as in mysteries. However, these do not help add to humor, other than to resolve lingering problems. Using the `Experts` I have created, the most frequent use of these in a humorous instance is resolving `Ally Flags` and `Karma Flags`. In these cases, reader interest is established in a character when they are introduced, and resolution usually only occurs at the end of a story. This does not add to the humor of a story, but it remains helpful.

It may at first appear that riddles are an example of an `Explanation Feature` as humor: after all, they have a question, surprise, and then resolution with an answer. However, this is not the case. While an Explanation does provide the core suspense that binds together a riddle, the humorous `Break Entities` actually occur concurrently with the `Repair Entities` in the answer part of the joke. After all, up until that point we could also be given a standard, non-surprising answer to the question that avoids the creation of any other flags at all.

`Eureka Feature`: **Background-Repair-Break**

> *Q: What is brown and sticky?*
>
> *A: A stick.*

This kind of `Feature` represents a scenario where a moment of surprise is introduced, but it is resolved with previous knowledge. Notably, that previous `Repair Entity` must have been insufficient to avoid the surprising moment, or this kind of ordering could not occur.

This ordering is one that I use as a metric of humor. It also represents an "Ah Ha!" or "Eureka" moment in story understanding. A scenario was introduced, then a narrative tool or piece of information that will allow the reader to understand future uncertainties, and finally that tool was successfully applied. For this reason, this `Feature` can also be useful in assessing pedagogy.

Stories where the `Break Entity` and the `Repair Entity` are triggered simultaneously also fall into this category, because there is no period of suspense waiting for a `Repair`. Instead, the reader has a single concentrated moment of understanding.

`Callback Feature`: Repair-Background-Break

> *When I was a little kid, I thought that [our babysitter] was like... 30 years old. I was just talking to my mom the other week, I found out that when I was ten Veronica was thirteen. So why was she in charge? ...that's just like hiring a slightly bigger child!* ***That would be like if you're going out of town for the week and you paid a horse to watch your dog."***
>
> Comedian John Mulaney, "New in Town", Callback Joke Part 1

> *"I walked into this [high school] party... people were drinking like it was the civil war and a doctor was coming to saw our legs off. It was totally unsupervised;* ***we were like dogs without horses..."***
>
> Comedian John Mulaney, "New in Town", Callback Joke Part 2

This kind of `Feature` resolution requires slightly more sophistication: the reader has background knowledge that they recall when faced with a problem that needs solving. It can be very compelling when executed correctly, and serves to highlight

the initial `Repair Entity` information for the audience. In fact, it is so useful that this technique is commonly advised in effective public speaking.

This kind of `Feature` is also used in this project as a metric for humor. Intuitively, it can be observed in several classic humor scenarios. The use of "callbacks" in stand-up comedy is common, and a signature of shows such as *Arrested Development* and *Archer*. The "Brick Joke" is another example, where a listener is told a joke that initially seems non-humorous (and therefore also can present the opportunity for a No-Soap-Radio false flag joke). Later, sometimes even after several other jokes have been told in between, the joke teller introduces a new `Background` and `Break Entity` to set up a joke, then resolves it using the old poor quality joke as a `Repair Entity`. An example of such a joke told in a webcomic over a gap of years can be found in Figure 6-1.

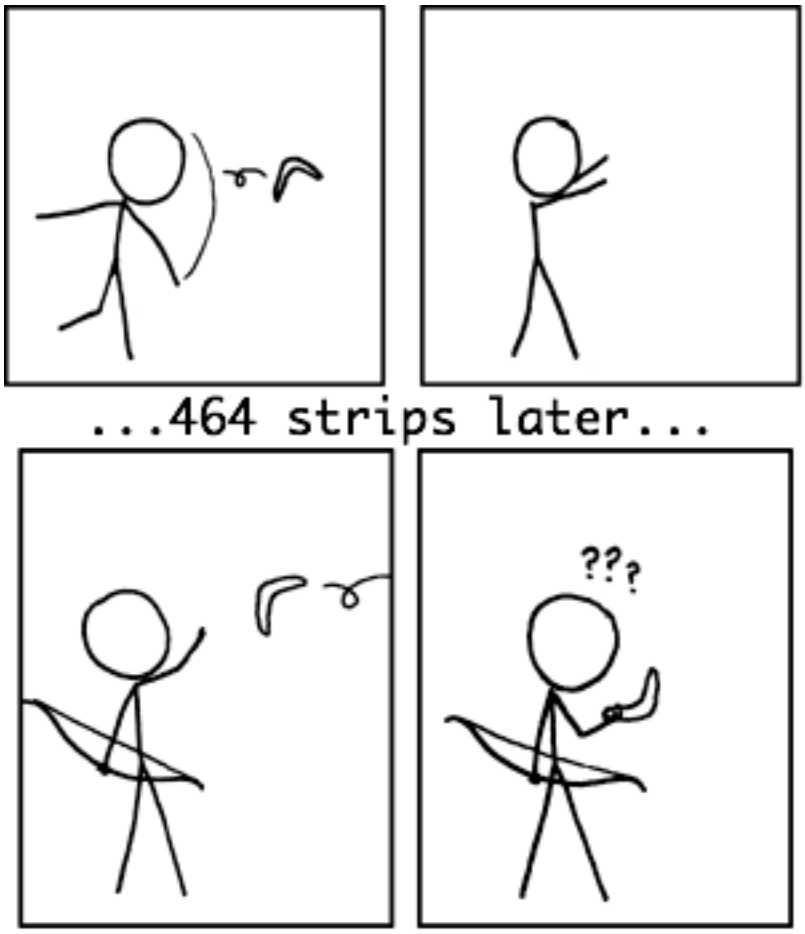

Figure 6-1: `Callback Feature`: Webcomic XKCD Strip 475 [1] and 939 [2]

The inherent risk in telling this kind of joke is that the audience may have forgotten or dismissed the earlier information as irrelevant, or simply may not have the background information required if it is not directly introduced by the storyteller. If the reader does not realizing you are *playing* this kind of game, then the joke may be lost on them (see section 1.2.3).

6.1.3 Separation between sub-Features

In addition to having an ordering, these components of `Expert Features` have relative distances between each other that can provide information. Several of these distances make intuitive sense:

Suspense Period: Distance after Break before Repair

This distance represents how long a problem was opened before a reason was introduced that allowed it to be closed.

It can be useful to track how many overlapping suspense periods there are in a story and assess the success of particularly suspenseful genres such as mysteries.

Callback Period: Distance from Repair to End of `Feature`

This metric describes how long ago we were introduced to the background material required to resolve a `Feature`.

Total Length: Distance from Beginning to End of `Feature`

This metric describes how long the joke is, and can be useful in assessing how jokes perform given constraints on attention span.

It is at the discretion of the user how to define a timescale or interpret this kind of information. In video or real-world interactions, timestamps may be sufficient. In prose, the Genesis System currently examines on the basis of `Entities` or lines, though these could also easily be combined with a timescale. In order to sufficiently analyze puns, a granularity of words or syllables may be required.

6.1.4 Combining Feature Information

Combining the information provided by multiple `Expert Features` can also reveal patterns in storytelling.

Attention Density: Controlling the Number of Open Questions

Particularly when examining narrative `Features` through the lens of teaching, it becomes important to examine the number of open questions that occur at any one moment within a story. A large number of open inquiries can potentially overwhelm the audience, or make realizations less clear in a teaching context.

Capping the number of active `Features` allowed open at any point in a story may help encourage storytellers to avoid several classic bad habits that authors sometimes overuse when trying to increase audience engagement. An overgrowth of characters to track, "mysterious" plot threads that dangle forever tauntingly unresolved, or inconsistent characterization can often leave narrative consumers with a dauntingly large pile of questions that make it harder to take satisfaction from future `Feature` resolutions. This is a frequent criticism of long running or large-scope soap operas such as *Game of Thrones*.

6.1.5 Genre: Which Experts Participated

As I will cover in greater depth in the humor section, we can also link humorous incidents to genres of humor by examining the `Experts` involved in them.

6.2 How to Identify an Instance of Humor

Given that a story has been analyzed, and `Expert Features` fully discovered and then resolved, I have formulated this question as a graph analysis question:

- Verify that no `Expert Features` are without valid repairs. This enforces that our story has meta-patterns still intact.

- Examine the distribution of `Expert Feature` sub-components across the entire story.
- Remove `Features` where the `Repair Entity` follows the `Break Entity` (these are `Explanation Features`)
- Trace each `Break Entity` of each `Feature` to the last explicit entity in the story that triggered the `Break Entity`.
- Trace each final sub-component of each `Expert Feature` to the last explicit entity in the story that triggered the closure of the `Feature`.
- Count the number of each of these concentrated at various points in the story, and look for story entities with a disproportionate number of `Feature Break Entities` and final resolutions being triggered on the same story `Entity`. If the number of these occurring at once is above a threshold, report an instance of humor.

It may be non-intuitive that all features require repairs; after all, some jokes seem purely nonsensical on the face without a valid "logical" parse. These jokes actually do have a "logical" parse, and it is that the story contains a trait such as "sarcasm", "whimsy" or "ridiculousness" that intentionally leads to the incongruity. If the listener does not know that one of these explanations is available to them, "illogical" conditions remain an error and not a joke. Intuitively, this kind of reader correlates with the case of a child that does not understand sarcasm, or perhaps a person at their doctor's appointment who is not expecting humorous elements. These individuals may identify a set of features that could be resolved using a trait of "whimsy", for example, but if that is impossible in their opinion then the communication will be seen as simply error-filled instead of successfully humorous.

6.2.1 Narrative Histograms provide Visual Signatures

Summarizing the distribution of these `Expert Features` allows the reader to quickly gain intuition for stories and their flaws. I have developed a simple diagramming

method for scanning stories for successful punch lines, displayed in Figure 6-2. This histogram method provides a visual signature for humor, as well as other genres of story.

Humor Histograms

In the case of a Humor Histogram, for each `Expert Feature` that is not an Explanation, the component `Background Entities`, `Break Entities`, and `Repair Entities` are each traced to their parent entities within the original explicitly stated lines of the story. A graph is formed by making a timeline of each of the explicit statements from the story, and stacking markers for the subcomponents they triggered above them. Finally, the location of the last component in each `Feature` is marked with a tick mark below the timeline. This gives us a sense of overall information-dense sections of the story in the top of the figure, and the punch line density in the lower portion of the figure.

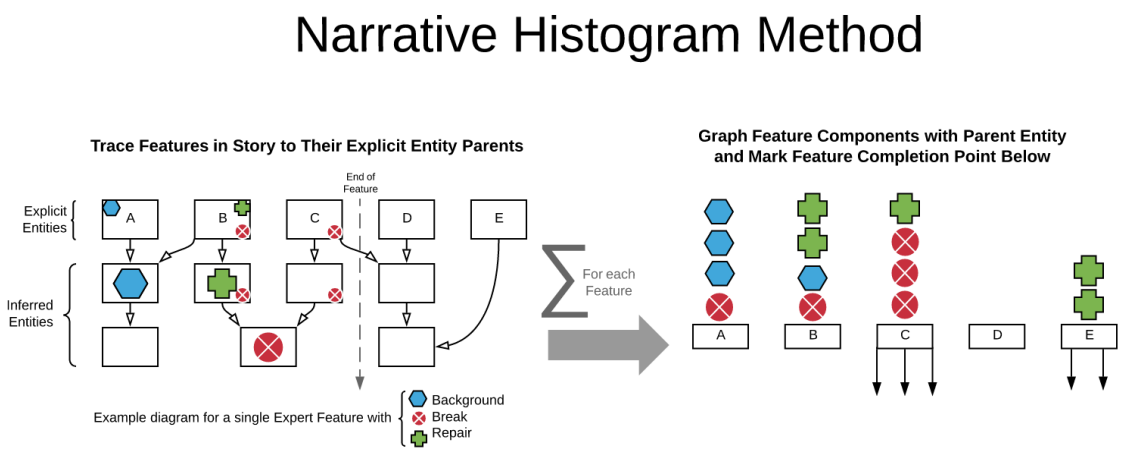

Figure 6-2: Method for Narrative Histogram Creation

This technique can be used to diagram jokes that have varying levels of successful humor, as seen in Figures 6-3, 6-4, and 6-5.

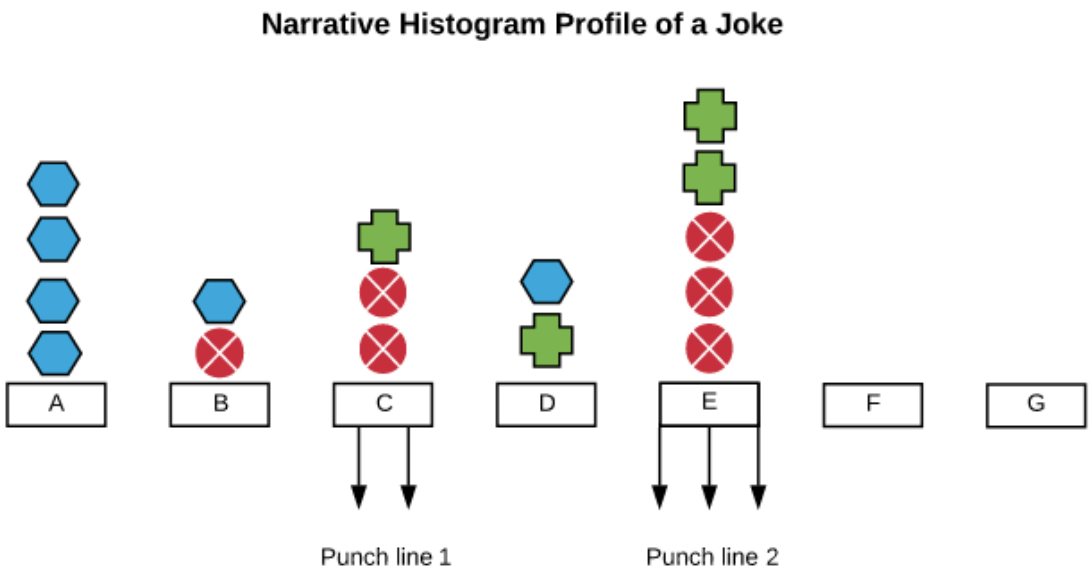

Figure 6-3: Histogram of a successful joke with a clear punch line moment

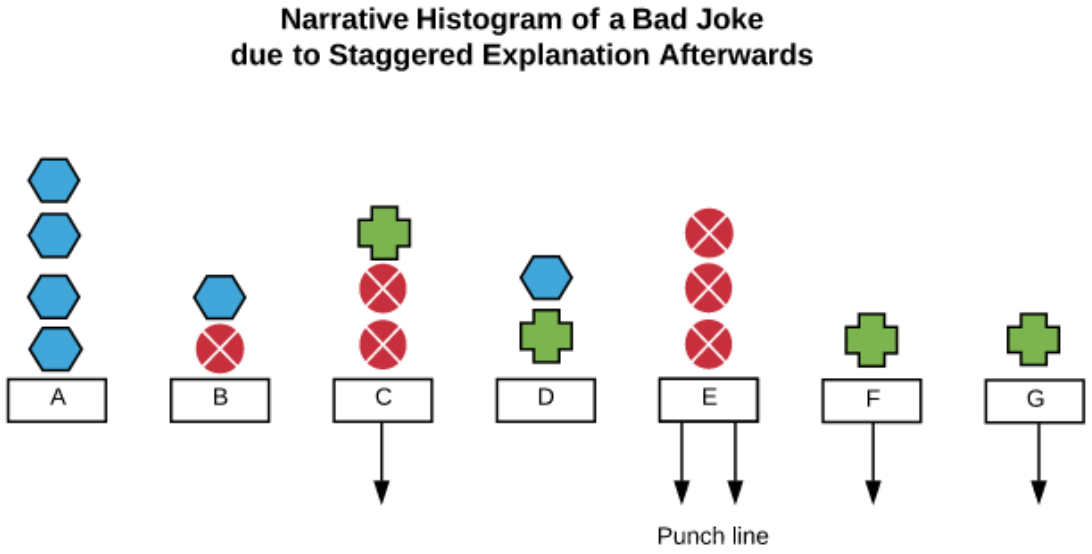

Figure 6-4: Histogram of a less successful joke with a slight punch line moment followed by explanations that stagger and diffuse the punch line.

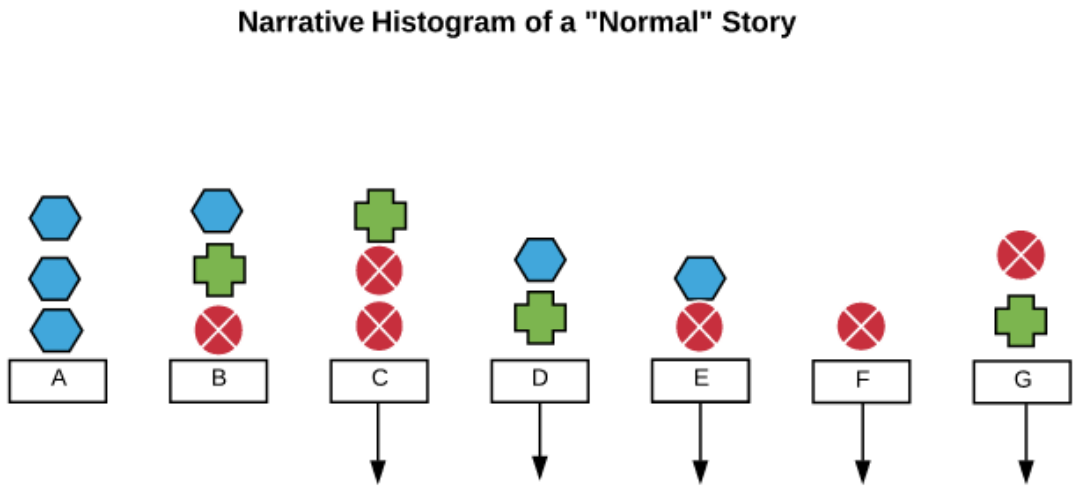

Figure 6-5: Histogram of a normal story, with a more randomized blend of feature repairs and completions.

Suspense Histograms

The reader can similarly assemble a histogram of the suspense over the course of a story by tracing the Suspense Periods of all the `Expert Features` found within a story, and for each entity in the story graphing the number of `Expert Features` that are open but not yet unresolved at that moment.

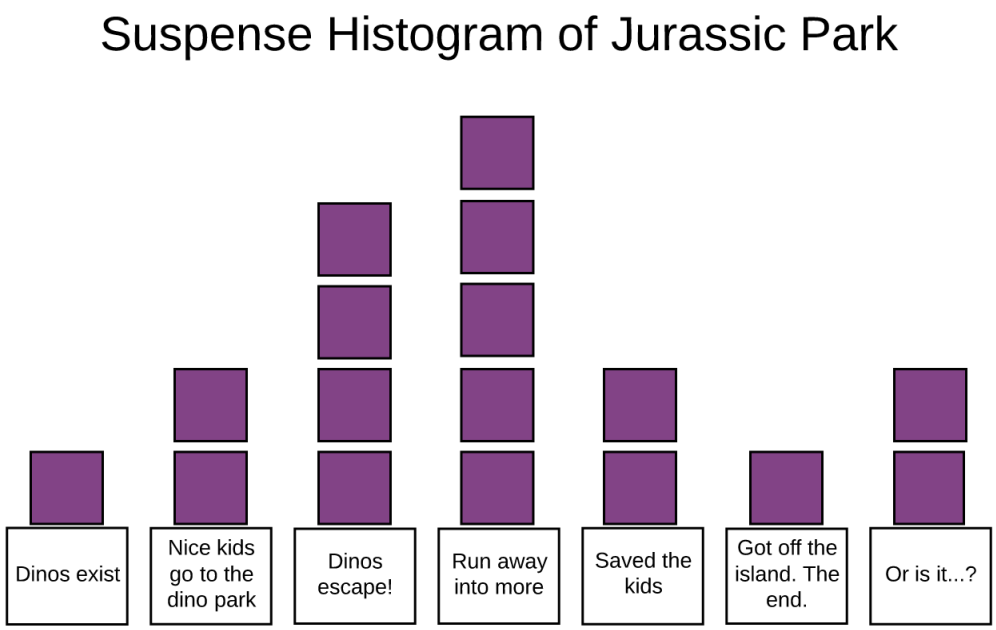

Figure 6-6: Histogram of the general plot of Jurassic Park. The more unanswered questions we have open, the higher more units of suspense at a given point in time. Jurassic Park builds to a crescendo, then resolves, with a slight uptick of a cliffhanger at the end.

Application to Other Genres

This method can be used to examine stories of different genres that require specific patterns in suspense, mystery resolution, and satisfaction. Figure 6-7 displays a high level view of a murder mystery, with all kinds of `Features` displayed.

One could imagine that character-focused genres such as a romance novel would also have a distinctive profile. In romance, a common pattern is for the two love interest characters to have an inner self (essence) and outer self (mask). This outer

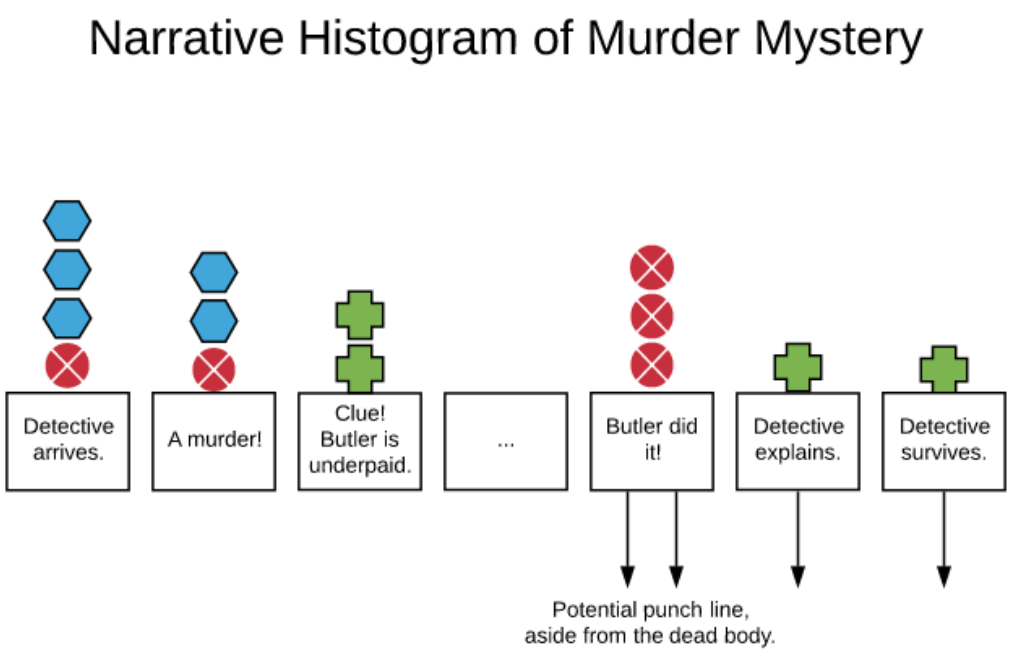

Figure 6-7: Histogram of a general murder mystery. Clues are given before the climax, a climax raises a lot of flags, but the clues let us resolve those flags immediately. Some falling action. Not a joke due to unresolved Karma flag; an innocent is not okay because they were murdered.

self frequently generates obstacles and misunderstandings that drive characters apart, then the reveal of their essences repairs these issues and creates a happy ending [4] [3].

In Pride and Prejudice, for example, a number of expectations are set by early negative interactions between characters: "pride" and "prejudice", specifically! However, at the climax events force Elizabeth to rapidly reassess her position given new information about Darcy, and new information reveals that Darcy has performed many good deeds (increase in karma). Both then finally achieve a happy ending. The distribution is different from a joke, but can nonetheless be distinctive and useful for analyzing storytelling effectiveness.

6.2.2 Extracting the Communication within an Instance of Humor

In order to understand a joke, we trace all the entities required in finding and fixing our features. All of these are assumed to be shared between speaker and listener in the case of a successful joke, and they include background information, commonsense rules, and explicit story elements.

6.2.3 Genre Classifications

By comparing the Experts involved in classifying an instance of humor, one can describe the genre of the joke found.

For example, we can describe many of the Features addressed by the Trait Expert as character humor. This is when a person in a story acts in a surprising manner, but their longstanding character traits provide an explanation. Examples would be Fry from *Futurama* or Andy from *Parks and Rec* having the trait of "stupid" that can explain them taking actions based on contradictory beliefs. Another common pattern is when a character is underestimated or put in a predicament that seems nearly impossible to escape, and then they use an existing expertise in the form of a character trait to make the event much more likely. Interestingly, both of the characters mentioned above with the trait of "stupid" also have the trait of "lucky" which helps them to stay out of harm and resolve problems brought about by their more problematic character trait.

Another example would be "Dark Humor" being discovered by the macabre expertise of the Morbidity Expert.

6.2.4 Personal Preference and Modeling the Mind of the Individual

The personal preference of individuals can be modeled using different implementations of each of the `Experts` that I created, adding more `Experts` to the `Expert Society`,

or changing the allowed links between Experts for resolutions by adding or subtracting some. Some examples of differences easily added based on the current system are:

Low Tolerance for Gore Any kind of harm no matter how minor registers as harm and the Morbidity Expert is easily shifted into the "DEADLY" state. This makes it harder for jokes to pass `Ally Expert` and `Karma Expert` checks.

Not Easily Impressed Unexpected events require a more shocking swing to be considered surprising and trigger a feature.

Naive or Child Listener Reader is familiar with fewer Traits.

Ain't No Place for an (Anti)-Hero Karmic penalties are much higher than normal, especially for non-lethal acts.

All Guns Fired Every Trait found by the `Trait Expert` raises a flag, to strictly require that all trait elements are used for some purpose within the story, to avoid unfired "Chekhov's Gun" scenarios.

Vulgarity Expert Much like the `Morbidity Expert`, this expert would estimate the vulgarity level of the story by searching for keywords. Stories that rapidly shift from clean to dirty or vice-versa would be inspected for a compelling reason.

Vigilante Justice The `Karma Expert` can allocate karma based on both deeds and the karma of those affected by them. This means that punching a robber could have a lower penalty than punching a baby bunny, or might even contribute a positive score.

Relative Tragedy The `Karma Expert` could return a normalized range of values, such that karma scores are relative to the population of a story. This would allow our judgments of characters to shift with genre. On a show for children, stealing cookies may constitute villainy. A war drama might leave both warring factions looking equally morally grey, even though the number of murders is quite high. Similarly, proportionate karmic responses could also scale.

6.3 Example Joke Applications

6.3.1 Roadrunner Joke

The following reviews the classic Looney Tunes cartoon example discussed in section 1.2, examining conflict between the Roadrunner and the Coyote.

While canonically the Roadrunner is actually named "BeepBeep" after his iconic taunting sound effect, for the sake of succinctness and clarity I have named him "Steve." The Coyote's full name is "Wile E. Coyote", here abbreviated as "Wiley."

Chuck Jones' Rules of the Road(runner) [12]

Interestingly, one of the iconic creators of the Roadrunner cartoon, Chuck Jones, created a list of rules for creating this series of cartoons, and many of them overlap with the `Experts` I have demonstrated here, or specific constraints of `Trait` choice. This is no coincidence!

Rule 1: The roadrunner cannot harm the coyote except by going "BEEP-BEEP!"

Rule 2: No outside force can harm the coyote only his own ineptitude or the failure of the Acme products.

Rule 3: The coyote could stop anytime if he were not a fanatic. (Repeat: "A fanatic is one who redoubles his effort when he has forgotten his aim." George Santayana)

Rule 4: No dialogue ever, except "BEEP-BEEP!"

Rule 5: The roadrunner must stay on the road otherwise, logically, he would not be called roadrunner.

Rule 6: All action must be confined to the natural environment of the two characters the southwest American desert.

Rule 7: All materials, tools, weapons, or mechanical conveniences must be obtained from the Acme corporation.

Rule 8: Whenever possible, make gravity the coyotes greatest enemy.

Rule 9: The coyote is always more humiliated than harmed by his failures.

Within the `Expert Society` model, these correlate to:

Rule 1: The roadrunner remains "INNOCENT."

Rule 2: The Karma Expert is responsible for all harm to the coyote.

Rule 3: The coyote has trait "fanatic."

Rule 4: No need to implement a `Dialogue Expert`.

Rule 5: The roadrunner does not leave the road. This could be considered a trait of the environment, or setup for future contradiction constraints.

Rule 6: The environment has traits "southwest American desert."

Rule 7: Acme belongs to the ally group of the Coyote, and no other entities aid him in attacking the roadrunner.

Rule 8: This can be considered a meta-trait of the story itself that readers grow familiar with, or gravity can be listed as an antagonist of the coyote.

Rule 9: The `Harm Expert` can attest the Coyote is okay at the end.

Genesis Representation

Rules
```
If e is a roadrunner, e is clever.
If e is a cartoon, then e is immortal.
If e explodes f, then f likely dies.
If e touches dynamite, then e likely dies.
If e does not explode then e is okay.
If e does not explode then e does not die.
If e buys dynamite to set a trap for Steve, Steve likely explodes.
If e is a coyote then e is unlucky.
```

If e is a roadrunner, then e is clever.

If e explodes and e is immortal then e survives.

If e survives then e is okay.

If e survives then e is not dead.

If e is a coyote and f is a roadrunner, then e wants to destroy f.

Genesis Representation Wiley is a cartoon and a coyote. Steve is a cartoon and a roadrunner. Wiley buys dynamite to set a trap for Steve. Wiley sets a trap with dynamite to likely destroy Steve. Steve touches dynamite and does not explode. Because Steve does not explode, Wiley may touch dynamite. Wiley touches dynamite and explodes. Because Steve does not explode, Wiley does not destroy Steve. The end.

Fully Expanded Story Wiley is a cartoon. Wiley is immortal because Wiley is a cartoon. Wiley is immortal. Wiley is a coyote. Wiley is unlucky because Wiley is a coyote. Wiley is unlucky. Steve is a cartoon. Steve is immortal because Steve is a cartoon. Steve is immortal. Steve is a roadrunner. Steve is clever because Steve is a roadrunner. Steve is clever. Wiley wants to destroy Steve because Steve is a roadrunner, and Wiley is a coyote. Wiley wants to destroy Steve. In order to set a trap for Steve, wiley buys dynamite. Wiley buys dynamite. Wiley sets a trap for Steve. In order to destroy Steve likely, wiley sets a trap with dynamite. Wiley sets a trap with dynamite. Wiley destroys Steve likely. Steve touches dynamite. Steve dies likely because Steve touches dynamite. Steve dies likely. Steve does not explode. Steve is okay because Steve does not explode. Steve is okay. Steve does not die because Steve does not explode. Steve does not die. Wiley touches dynamite because Steve does not explode. Wiley touches dynamite. Wiley dies likely because Wiley touches dynamite. Wiley dies likely. Wiley explodes. Wiley survives because Wiley explodes, and Wiley is immortal. Wiley survives. Wiley is okay because Wiley survives. Wiley

`is okay. Wiley is a not dead because Wiley survives. Wiley is a not dead. Wiley does not destroy Steve because Steve does not explode. Wiley does not destroy Steve. The end.`

Genesis Elaboration Graph

`Expert Features`

Table 6.1: `Unexpected Expert Features` of Looney Tunes Example

Field	**Use**
Issuer	Unexpected Expert
Story	*Figure 4-5*
Flag ID	FLAG-UNEXPECTED
Background Entity	"Wiley is likely dead."
Problem Entity	"Wiley is not dead."
Fix Entities	TRAIT: "Wiley is immortal"
Issuer	Unexpected Expert
Story	*Figure 4-5*
Flag ID	FLAG-UNEXPECTED
Background Entity	"Steve dies likely."
Problem Entity	"Steve is not dead."
Fix Entities	TRAIT: "Steve is immortal"
Issuer	Unexpected Expert
Story	*Figure 4-5*
Flag ID	FLAG-UNEXPECTED
Background Entity	"Steve is likely exploded."
Problem Entity	"Steve is not exploded."
Fix Entities	TRAIT: "Steve is clever"

Examining these histograms, from the `Suspense Histogram` we can see fairly even suspense throughout the joke, with a slight increase near the end as our interest is further piqued. From the humor histogram, two punch lines leap out: when the roadrunner is not blown up, and then when the coyote is.

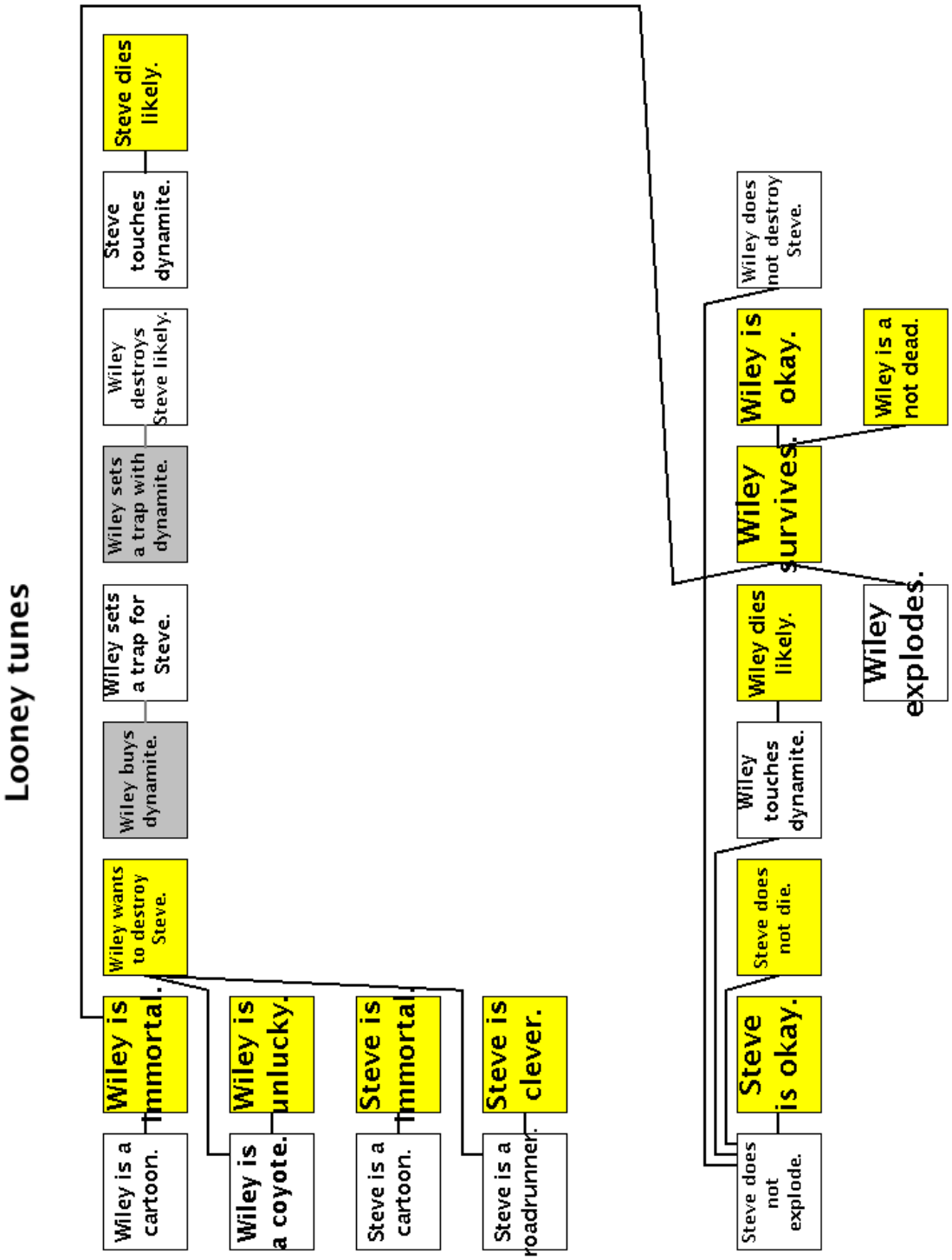

Figure 6-8: Looney Tunes Roadrunner and Coyote Joke Example

Table 6.2: `Ally Expert Features` of Looney Tunes Example

Field	Use
Issuer	Ally Expert
Story	*Figure 4-5*
Flag ID	FLAG-PROTAGONIST
Background Entity	"Wiley is a cartoon."
Problem Entity	"Wiley is a cartoon."
Fix Entities	TRAIT: "Wiley is okay."

Table 6.3: `Harm Expert Features` of Looney Tunes Example

Field	Use
Issuer	Harm Expert
Story	*Figure 4-5*
Flag ID	FLAG-HARMED
Background Entity	"Wiley explodes."
Problem Entity	"Wiley explodes."
Fix Entities	HARM: "Wiley is okay."
	KARMA: "Wiley wants to destroy Steve."

Table 6.4: `Karma Expert Features` of Looney Tunes Example

Field	Use
Issuer	Karma Expert
Story	*Figure 4-5*
Flag ID	FLAG-KARMA-INNOCENT
Background Entity	"Steve is a cartoon."
Problem Entity	"Steve is a cartoon."
Fix Entities	HARM: "Steve is okay."

Table 6.5: `Morbidity Expert Features` of Looney Tunes Example

Field	Use
Issuer	Morbidity Expert
Story	*Figure 4-5*
Flag ID	FLAG-MORBIDITY-SAFE-TO-DEADLY
Background Entity	"Wiley is a cartoon and a coyote."
Problem Entity	"Wiley buys dynamite."
Fix Entities	TRAIT: "Wiley is okay."

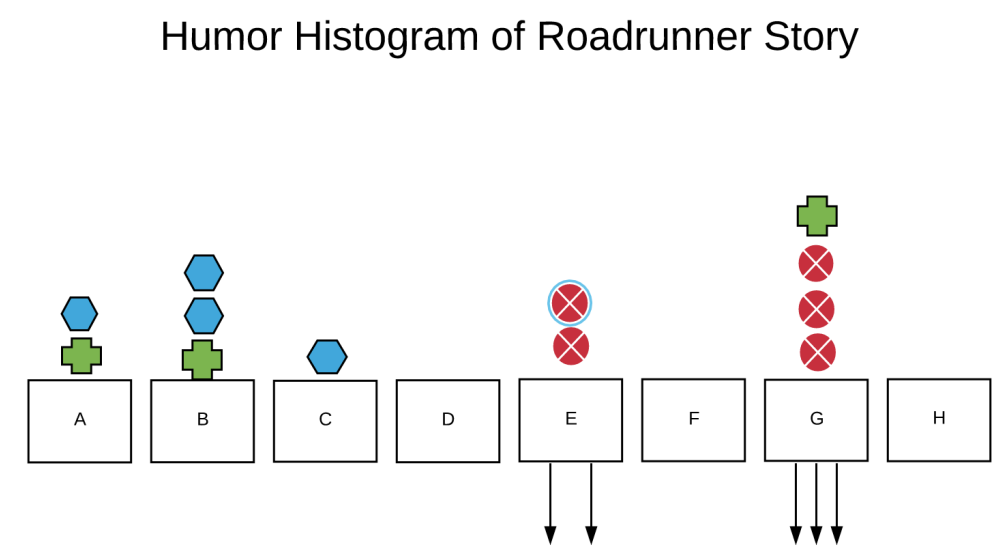

Figure 6-9: Humor Histogram of the Roadrunner Scenario

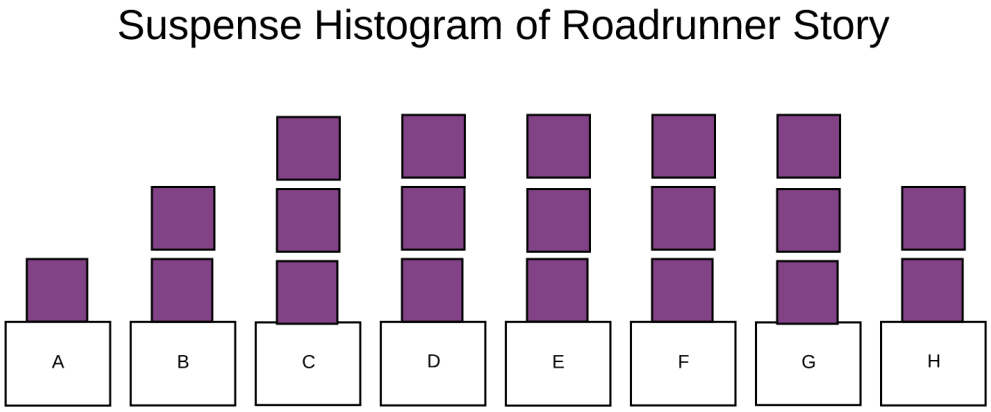

Figure 6-10: Suspense Histogram of the Roadrunner Scenario

6.3.2 Baby Rhino Joke

This joke is a "YouTube Haiku", a modern format of humor constrained to a single 14 second video clip. In this clip, a man details for the viewer his intent to contain a dangerous rhino. This rhino is revealed to be an adorable and harmless baby. Furthermore, the man is so focused on describing his efforts to keep the rhino in its cage that he does not notice as the rhino walks out of the cage behind his back in a very obvious manner.

https://www.youtube.com/watch?v=b-nwRDNoJR4

Genesis Representation

Rules: If ee is a rhino then ee is probably dangerous.

If ee is a rhino then ee is probably large.

If ee is a baby then ee is tiny.

If ee is tiny then ee is not large.

If ee is harmless then ee is not dangerous.

If ee is probably zz and ff is a human, then ff believes that ee is zz.

If ee believes that ff is dangerous, then ee wants ff to be secure.

If ee is focused on s, then ee usually notices not ss.

If ee is small and there is a gap, ee can escape.

If ee can escape ee will escape.

ee being tiny enables ee to escape.

If ee wants ff to be secure, then ee puts ff in a large cage.

ee may think ff is dangerous because ff is a rhino.

ee may put ff in a cage to keep ff secure.

If ff escapes then ff is not secure.

If ff is a cage and ff is large, then ff has small gaps.

If ff is in a cage then ff probably is secure.

If ff is in a cafe then ff likely does not escape.

If ff escapes then ff is clever.

If ff is a rhino then ff is likely not clever.

Explicit Story: Ivan is a human.

Betty is a rhino.

Ivan built a large cage to try to secure Betty.

Betty is a baby.

Betty escapes because the large cage has small gaps.

Ivan does not notice Betty because he is distracted.

Because Betty is not dangerous, Ivan is okay.

Fully Expanded Story: Ivan is human.

Betty is a rhino.

Betty is dangerous probably because Betty is a rhino.

Betty is dangerous probably.

Betty is large probably because Betty is a rhino.

Betty is large probably.

Betty is not clever likely because Betty is a rhino.

Betty is not clever likely.

In order to try securing Betty, Ivan built a large cage.

Ivan built a large cage.

Ivan tries to secure Betty.

Ivan is afraid of Betty because Ivan tries to secure Betty, and Betty is dangerous probably.

Ivan is afraid of Betty.

Ivan notices Betty likely because Ivan is afraid of Betty.

Ivan notices Betty likely.

Betty is a baby.

Betty is tiny because Betty is a baby.

Betty is tiny.

Betty is not large because Betty is tiny.

Betty is not large.

Betty is in a large cage because Betty is not large.

Betty is in a large cage.

Betty is secure probably because Betty is in a large cage.

Betty is secure probably.

A large cage has the cage's small gaps because Betty is not large.

A large cage has the cage's small gaps.

Betty escapes because a large cage has the cage's small gaps.

Betty escapes.

Betty is not secure because Betty escapes.

Betty is not secure.

Betty is clever because Betty escapes.

Betty is clever.

Ivan is distracted.

Ivan does not notice Betty because Ivan is distracted.

Ivan does not notice Betty.

Betty is not dangerous.

Ivan is okay because Betty is not dangerous.

Ivan is okay.

Genesis Elaboration Graph

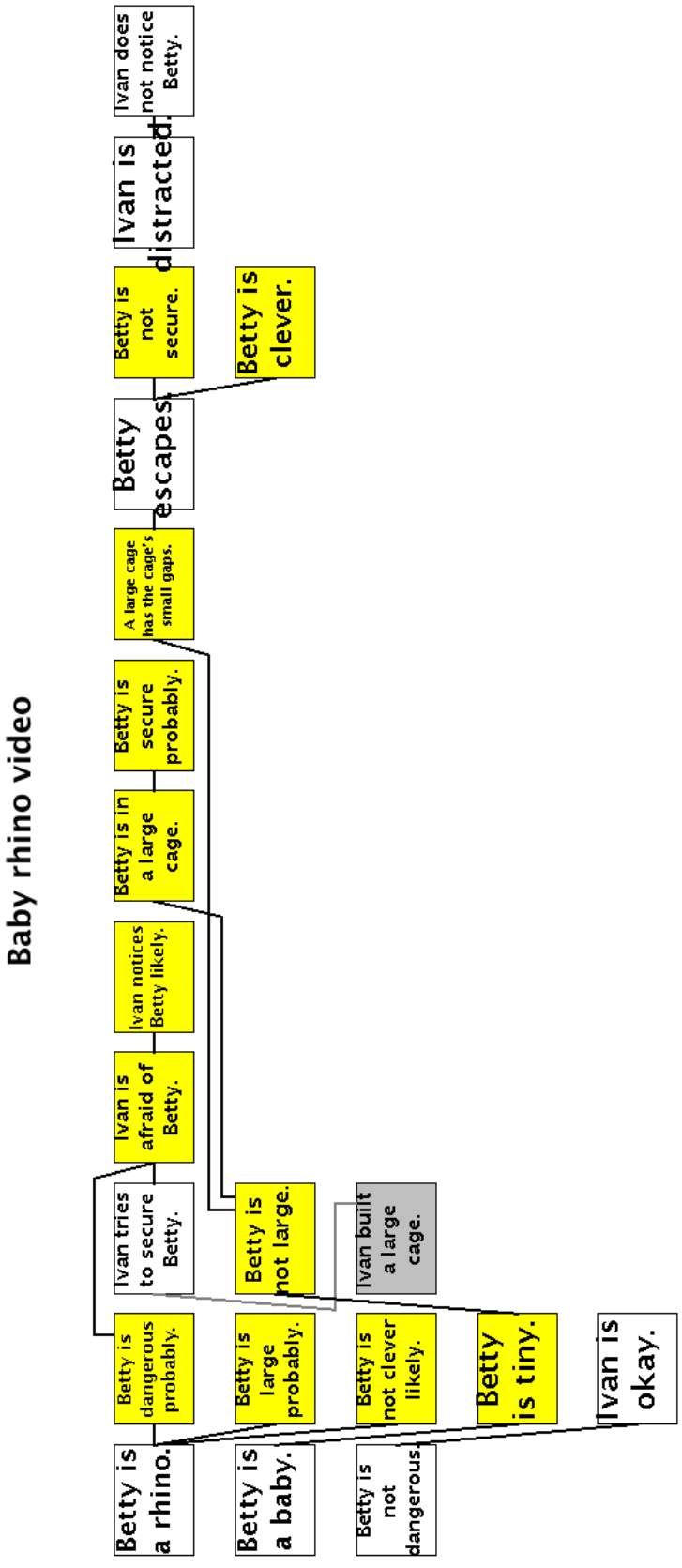

Figure 6-11: Genesis Diagram of the Baby Rhino Video

Expert Features

Table 6.6: `Unexpected Expert Features` of Baby Rhino Example

Field	Use
Issuer	Unexpected Expert
Story	*Figure 6.3.2*
Flag ID	FLAG-UNEXPECTED
Background Entity	"Betty is dangerous probably."
Problem Entity	"Betty is not dangerous."
Fix Entities	TRAIT: "Betty is a baby."
Issuer	Unexpected Expert
Story	*Figure 6.3.2*
Flag ID	FLAG-UNEXPECTED
Background Entity	"Betty is not clever likely."
Problem Entity	"Betty is clever."
Fix Entities	TRAIT: "Betty escapes."
Issuer	Unexpected Expert
Story	*Figure 6.3.2*
Flag ID	FLAG-UNEXPECTED
Background Entity	"Betty is large probably."
Problem Entity	"Betty is not large."
Fix Entities	TRAIT: "Betty is a baby."
Issuer	Unexpected Expert
Story	*Figure 6.3.2*
Flag ID	FLAG-UNEXPECTED
Background Entity	"Ivan notices Betty likely."
Problem Entity	"Ivan does not notice Betty."
Fix Entities	TRAIT: "Ivan is distracted."

Table 6.7: `Ally Expert Features` of Baby Rhino Example

Field	Use
Issuer	Ally Expert
Story	*Figure 6.3.2*
Flag ID	FLAG-PROTAGONIST
Background Entity	"Ivan is human."
Problem Entity	"Ivan is human."
Fix Entities	TRAIT: "Ivan is okay."

Table 6.8: `Karma Expert` `Features` of Baby Rhino Example

Field	**Use**
Issuer	Karma Expert
Story	*Figure 6.3.2*
Flag ID	FLAG-KARMA-INNOCENT
Background Entity	"Ivan is human."
Problem Entity	"Ivan is human."
Fix Entities	HARM: "Ivan is okay."
Issuer	Karma Expert
Story	*Figure 6.3.2*
Flag ID	FLAG-KARMA-INNOCENT
Background Entity	"Betty is a baby."
Problem Entity	"Betty is a baby."
Fix Entities	HARM: "Betty is okay."

Histograms

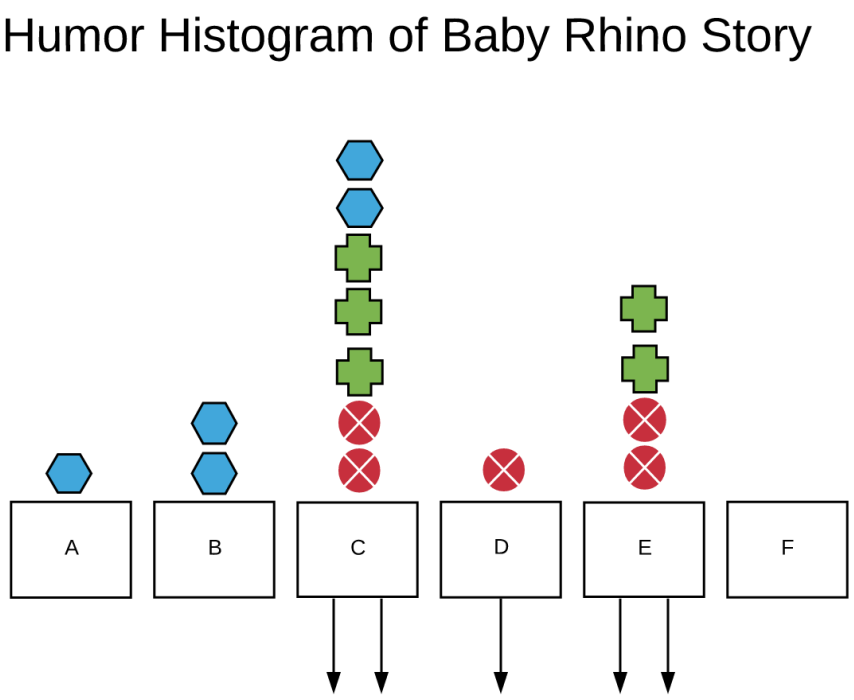

Figure 6-12: Humor Histogram of the Baby Rhino Scenario

Examining these histograms, from the `Suspense Histogram` we can see fairly

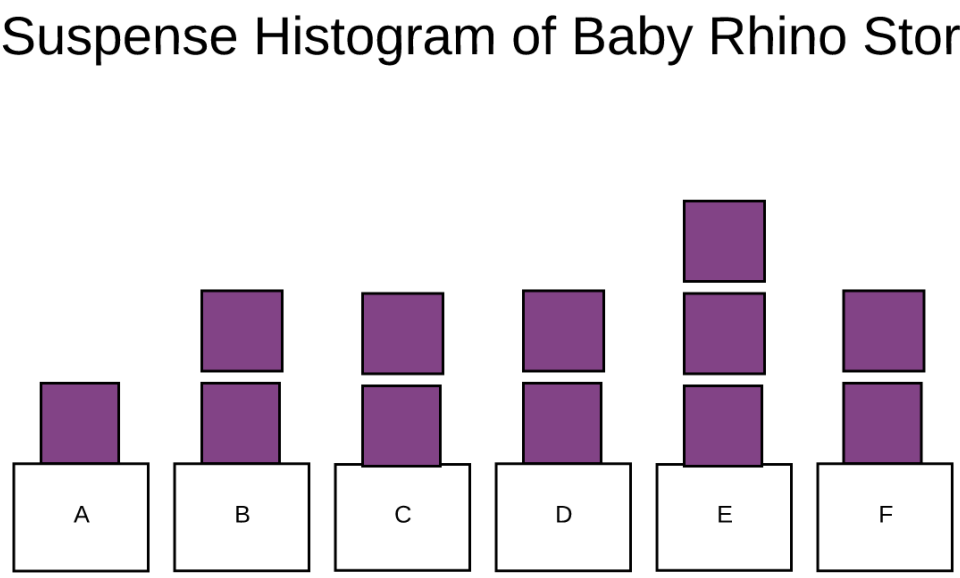

Figure 6-13: Suspense Histogram of the Baby Rhino Scenario

even suspense throughout the joke, particularly because both characters are innocent. From the humor histogram, we again have a double punch line: when the rhino is revealed to be a baby and thus non-threatening, and when the rhino uses these traits to escape without being noticed by Ivan. Interestingly, this set of punch lines build on each other. This is because the information that provided the `Break Entity` in the first humorous incident acts as a `Repair Entity` in the next humorous peak of the joke.

6.4 Future Directions

With these demonstrations, we can see the potential of the `Society of Experts'` work with the Genesis System for quantitatively assessing both humor and audience engagement. While each of these `Experts` does an excellent job of demonstrating the power of this approach, additions can always be made.

Weight for It The relative strength of `Features` when summed can be weighted to model an individual's sense of humor. For example, the system could give

the `Karma Expert` and the `Harm Expert` to have half the weight of the `Ally Expert`, so the lower weighted `Experts` need to combine with each other or other signals to have equal weight.

`Trait Expert` **Suggestions** The `Trait Expert` currently relies on mappings of verbs to adjectives to determine useful traits for repairs. The `Trait Expert` could trace flagged features and return all adjectives that led to the break feature, then look those up in a broader external large-scale data source.

Trait Evidence `Expert` As demonstrated, the `Trait Expert` can raise a flag for each trait to make sure it is used within the story. In order to resolve these flags, the system would need another `Expert`. This one would look for evidence of each trait affecting the flow of the story.

"Solving for Unknowns" `Expert` Every time we are given only partial information about an object, the reader makes a mental placeholder for it and begins to imagine what that object might be. This is particularly true of the common question-answer riddle format.

To this end, it might be useful to create an `Expert` that flags whenever a question word is used to ask about an object, and verifies when a correct answer is found. This would clearly have additional use outside of humor questions, as well.

Dynamic Background Material The Genesis System provides powerful tools for describing commonsense rules and using them to build story understanding. Particularly when humor is extremely sensitive to background knowledge, it could greatly expand the practical use of this system to have stories consult with external data sources to use more commonsense information.

Mental Model Expert Each character within the story has different knowledge depending on what other story events they have observed, or how the story has affected their feelings. This is a key component of literary irony in particular. This kind of understanding can be expressed in terms of `Unexpected` or `Contradiction` features by adding additional commonsense rules to a story, but

adding an `Expert` that focuses on this kind of modeling could make increasingly complex analysis easier.

Belief and Speech Expert Currently, differences between actions, beliefs, and spoken statements by characters require additional commonsense rules to translate those conditions into ones that the `Contradiction Expert` or `Unexpected Expert` can handle. It could be useful to add an `Expert` that specializes in understanding these conditions.

Deadline Expert Some story conditions have a time limit or time-variable expectation, and therefore open the question for the reader as to when they will be satisfied. Once the dynamite is introduced in the story above, the reader has some increasing expectation that it will be set off before the end of the story. Similarly, if a character says that an event must take place before the winter solstice, the audience will open a `Feature` to track this event until it successfully takes place, or expects a reason if it fails to occur.

Chapter 7

Contributions

Through this project, I developed the Expectation Repair Hypothesis, an error-correcting focused computational theory of humor recognition and interpretation. I also established the need for models of humor that account for the mental state information that is communicated by successful or unsuccessful reception of humor.

To implement this system, I identified and implemented seven key `Experts` with unique domains of expertise applied to stories read by the Genesis story understanding system. Each `Expert` can pinpoint specific categories of potential errors within a story, as well as answer unique questions relevant to their domain of expertise.

I demonstrated how `Experts` that operate with different hidden states, methods of story understanding, and levels of abstraction within a story can interact in a productive manner and collectively reveal more complex narrative features than each could alone. This highlighted the importance of collaboration among heterogeneous agents with different methodologies and areas of expertise. I constructed standardized methods for these `Experts` to collaborate, and this ability was used to synergistically resolve errors between `Experts`. These errors range in scope from clerical errors leading to a contradiction, to high level concerns such as verifying that protagonists survive a story or that good things happen to good people. I also demonstrated how these `Expert` interactions can provide general purpose error identification and resolution for authors editing prose.

I outlined a formula for using patterns in `Expert` interactions to identify the punch

lines of successful moments of humor, as well as new metrics for extracting other story characteristics from `Expert` error handling interactions such as audience engagement in terms of suspense, attention span length, attention density, and moments of insight. I also introduced Narrative Histograms as a visual signature for narrative engagement, and showed how this representation supports humor identification and story genre analysis.

I simulated successful computational recognition of humor on real world humorous narratives by examining their Narrative Histograms and tracing the punch line moments that initiated rapid clusters of breaks and repairs in `Expert` story understanding, as well as verifying `Expert Feature` resolution. This approach also traced and revealed the background knowledge and commonsense rules that are implicitly verified and communicated by shared appreciation of an instance of humor.

Bibliography

[1] https://xkcd.com/475/.

[2] https://xkcd.com/939/.

[3] Michael hauge's workshop: An antidote to "love at first sight". http://jamigold.com/2012/08/michael-hauges-workshop-an-antidote-to-love-at-first-sight/, Jun 2017.

[4] Michael hauge's workshop: Are these characters the perfect match? http://jamigold.com/2012/08/michael-hauges-workshop-are-these-characters-the-perfect-match/, Jun 2017.

[5] Debra Aarons. *Jokes and the linguistic mind*. Routledge, 2012.

[6] Arjun Chandrasekaran, Ashwin K. Vijayakumar, Stanislaw Antol, Mohit Bansal, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. We are humor beings: Understanding and predicting visual humor. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] Dormann Claire. A battle of wit: Applying computational humour to game design. *Entertainment Computing - ICEC 2015 Lecture Notes in Computer Science*, page 7285, 2015.

[8] Ian F. Darwin. *Checking C programs with lint*. O'Reilly, 1996.

[9] Google. Improve your code with lint. https://developer.android.com/studio/write/lint.html, Jan 2018.

[10] Matthew M. Hurley. *Inside jokes: using humor to reverse-engineer the mind*. MIT Press, 2013.

[11] Stephen C. Johnson. *Lint: a C propram checker*. 1978.

[12] Chuck Jones. *Chuck Amuck*. Farrar Straus Giroux, 1999.

[13] Chloé Kiddon and Yuriy Brun. That's what she said: Double entendre identification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 89–94, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[14] Igor Labutov and Hod Lipson. Humor as circuits in semantic networks. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 150–155, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[15] Steven LaCorte. *An Examination of Personal Humor Style and Humor Appreciation in Others*. PhD thesis, John Carroll University, 2015.

[16] Amogh Mahapatra and Jaideep Srivastava. Incongruity versus incongruity resolution. *2013 International Conference on Social Computing*, 2013.

[17] Rada Mihalcea and Carlo Strapparava. Making computers laugh. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT 05*, 2005.

[18] Marvin Minsky. Jokes and the logic of the cognitive unconscious. *Cognitive Constraints on Communication*, page 175200, 1980.

[19] Duncan Riley. Kids and anagrams: When santa goes wrong. https://www.inquisitr.com/11396/when-santa-goes-wrong/, Dec 2008.

[20] Graeme Ritchie. Developing the incongruity-resolution theory. Technical report, 1999.

[21] Mary K Rothbart. Laughter in young children. *Psychological bulletin*, 80(3):247, 1973.

[22] Oliviero Stock and Carlo Strapparava. Hahacronym. *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions - ACL 05*, 2005.

[23] Madelijn Strick, Rick B. Van Baaren, Rob W. Holland, and Ad Van Knippenberg. Humor in advertisements enhances product liking by mere association. *Psychology of Popular Media Culture*, 1(S):1631, 2011.

[24] Len Wein. Batman #321. *DC Comics*.

[25] Patrick Henry Winston. The genesis story understanding and story telling system a 21st century step toward artificial intelligence. Technical report, Center for Brains, Minds and Machines (CBMM), 2014.

[26] Avner Ziv. Teaching and learning with humor. *The Journal of Experimental Education*, 57(1):415, 1988.