A Quantified Analysis of *Bad News* for Story Sifting Interfaces

Ben Samuel[1], Adam Summerville[2], James Ryan[3], and Liz England[4]

[1] University of New Orleans bsamuel@cs.uno.edu
[2] Cal Poly Pomona asummerville@cpp.edu
[3] Carleton College jryan@carleton.edu
[4] lizengland07@gmail.com

Abstract. *Bad News* is an award winning game, simulation, and performance art piece which depends on story sifting and live acting performed simultaneously and in real time by two human performers taking on two roles: a *Wizard* and an *Actor*. This paper offers a quantified analysis of eighty-one playthroughs of *Bad News*, identifying common patterns that were frequently used by the story sifting *Wizard* to unearth narratively salient content buried in the simulation to reveal to the player. This analysis informs the design of an in-development *WizActor* interface; a tool intended to reduce the cognitive overhead of story sifting such that future pieces of computationally assisted performance might only depend on one person to fulfill both roles.

Keywords: Bad News · Computationally assisted performance · Story sifting · Kismet · social simulation · WizActor.

1 Introduction

Social simulation has demonstrated itself to be a powerful tool for creating works of interactive digital storytelling [22, 18, 20]. Social relationships and dynamics are often a core part of resonant stories, and therefore finding ways to computationally represent them remains an important area of research for the interactive narrative community. Thus far, many narratives driven by social simulation have relied on the properties of emergent narrative [32], in which no formal representation of narrative structure (e.g., an Aristotelian dramatic arc [2]) is encoded, and instead narrative patterns naturally emerge through the actions of the simulated agents. These patterns are then surfaced to readers/players through mechanisms appropriate to the experience in question, in a process that has come to be known as story sifting [29, 15]: somehow searching the simulated space to find narratively salient gold amidst the dross of other simulated output.

Story sifting technology is still nascent, however, with ample room for exploration. A notable example of this can be seen in the award-winning game / theatrical performance piece *Bad News* [34]. Powered by the social simulation system *Talk of the Town* [30], *Bad News* asks players to explore a unique procedurally generated small American town with one hundred and forty years of

simulated history. The player is tasked with three objectives: discovering the identity of a resident that has recently passed away, searching for the deceased's next of kin, and delivering the eponymous bad news of their passing. Interaction in this piece is real-time spoken communication between the player and a live human performing as the game's *Actor*, who may be asked to assume the role of any of the hundreds of the town's residents. Behind the scenes, another human deemed the *Wizard* (so termed from Wizard-of-Oz interaction techniques [31]) is performing the role of story sifter; scouring the simulated history for material that is narratively salient. In truth, the *Wizard* bears the responsibility of both story sifter and drama manager [23], as the material they find must not only be narratively salient, but also in service of the ultimate quality of the player's experience, whose needs are dynamically changing based on a number of factors. A simple example of one such factor is the experience duration; the designers of *Bad News* determined that the ideal length for a playthrogh is between forty-five and sixty minutes. Thus, the *Wizard* had the task of searching the simulated history for interesting narrative patterns (e.g., starcrossed lovers, rival businesses, disgruntled family dynamics, and other stressors of smalltown life), but revealing them in such a way as to not make the game too easy or too hard (e.g., laying breadcrumbs that would bring the player to the next of kin immediately upon the start of play—even if narratively salient—would likely produce a short and ultimately narratively unsatisfying experience). This is all made more complex by virtue of the fact that the Wizard does not present their findings directly, but rather conveys this information to the *Actor*, who bears ultimately responsibility over how and when any of this information is revealed to the player. Moreover, the simulated history has no GUI interface; it can only be explored via Python code. This led to the *Bad News* experience resulting in the *Wizard* livecoding in Python, the *Actor* performing with the player, and the *Wizard* and *Actor* simultaneously communicating with each other via text messages while performing their respective roles.

Though perhaps the spectacle of this is unique to *Bad News*—and indeed likely contributed to its positive reception—it renders the game frustratingly difficult to distribute. The work presented in this paper is an attempt to enable all of the above in a manner which consolidates the responsibilities of *Wizard* and *Actor* into a single role. At present, the cognitive load of simultaneously role playing and story sifting are too much for a single person to perform. Thus, we present the initial development of an interface intended to enable the consolidation of the *Actor* and *Wizard* roles, tentatively titled the ***WizActor*** interface. Its design draws directly from quantitative and qualitative data drawn from dozens of *Bad News* playthroughs. However, though *Bad News* informs the design of this interface, it is intended to be generalizable enough to be applicable in any situation in which an interactive narrative driven by social simulation and storysifting leverages a human storyteller. This has a wide range of application areas, from entertainment based experiences such as tabletop role-playing games to simulated role-playing exercises for training and education.

The contributions of this paper are the following: We present for the first time a quantitative analysis of logs of previous *Bad News* playthroughs. These logs are complete records of every Python command typed by the *Wizard*, but do not include any communication between the *Wizard* and *Actor*, nor do they include any input from the Player. Though the complete theatrical experience can't be recreated from a log alone, each log still offers an invaluable lens into the overall flow of the experience. Exploring them provides a concrete look at the types of stories and relationships that were most prevalent in *Bad News*, and serves as the first quantified evaluation of this award winning game. Additionally, this paper will present an in-development prototype of a computational story-sifting assistant: the aforementioned *WizActor* interface, whose design is informed by this analysis of *Bad News*. The data used in this preliminary presentation of the *WizActor* interface was generated via the small social simulation system *Kismet* [38], a system designed to enable authors to create simulation-rich *Bad News*-like playable theatrical experiences of their own. The *Kismet* simulation itself depicts a regency-era ball, authored by a professional developer with a specialty in world simulation and procedurally generated content.

2 Related Work

This work finds itself at a cross section of many different branches of research and practice (akin to other cross-disciplinary techniques such as Research Creation [33]). One such branch is the theatre, both traditional and experimental. *Bad News* is a piece of interactive theatre such as *Coffee! A Misunderstanding*, *Sleep No More*, and the performances of Improbotics [13, 41, 19]. Of these three it is most closely aligned with Improbotics, as they are both fully improvised. However, *Bad News* dialogue is informed by the simulation but is generated by the human *Actor*, whereas improbotics uses natural language generation techniques to produce dialogue for both human and robotic performers. *Bad News* is also a piece of procedural content generation. Though there are many approaches to procedural content generation [40, 39], *Bad News* takes much inspiration from the world generation and simulation techniques of *Dwarf Fortress* [1]. It is also a piece of table top role-playing akin to *Dungeons and Dragons* [9], a domain which has seen its own share of of PCG research ranging from the generated music of *Bardo* [25], the characters and relationships of *Fiascomatic* [11], and the settings and story beats of *Dear Leader* [12].

Gamalyzer [24] analyzed playthroughs of *Prom Week* in a quantitative manner, but was concerned with similarity/dissimilarity between playthroughs as opposed to any specific patterns found in the playthroughs. Rameshkumar and Bailey [27] performed high-level textual analysis of role playing sessions from the *Critical Role* series, but as with Gamalyzer, there was no assessment of patterns in the playthroughs. Leece and Jhala [16] used sequence mining similar to our approach to find patterns of actions in *StarCraft*, but to our knowledge no one has investigated logs of actions of simulation sifting/role playing before.

Kismet is a tool for social simulation. While other social simulation approaches, such as *PsychSim* and *Comme il Faut*, are large systems that attempt to operationalize specific theories from the psychology literature [17, 21], *Kismet* is intentionally designed to be smaller and more accessible. Authoring for *Kismet* is intended to not depend on the use of a specialized authoring tool [35] like other social simulation systems [6]. Its design was informed by the easy-to-use philosophy of Casual Creators [5], akin to other creativity enabling tools such as *Tracery* [4] and *Imaginarium* [10].

The *WizActor* interface is intended to be a computational assistant for performing many of the roles currently borne by the *Wizard*. These roles include story-sifting through generated content to find salient emergent narrative as backstory [15, 14], as well as performing drama management [28] to find relevant dramatic moves to further the ongoing narrative, and even aspects of player modeling [36] to shape content around the player's predilections. Though intended to be used as a computational assistant during live play, the *WizActor* interface can also be used as a lens into the expressive range [37] of any *Kismet*-powered system, assisting authors in their design of the simulation [8].

3 Quantitative Analysis of *Bad News* Wizard Logs

As mentioned above, *Bad News* is a "game about death, death notification, and everyday life, combining deep social simulation and live performance." [34] The two most critical non-player roles are those of the *Wizard* – the one who interprets commands from the player and interfaces with the simulation – and the *Actor* – the one who takes on the role of characters in the simulation. One of the key interactions between *Wizard* and *Actor* is where the *Wizard* queries the simulation and then communicates the salient information to the *Actor*. In looking to support the fusion of *Wizard* and *Actor*, we analyzed records of the commands used by the *Wizard* during playthroughs of *Bad News*. The end goal being an understanding of what types of information the *Wizard* found salient and useful, so as to streamline the information gathering process for the *WizActor*.

In total, we analyzed the logs from 81 playthroughs. The *Wizard* interface for a *Bad News* playthrough is an interactive Python terminal that allows the *Wizard* to execute arbitrary code – in this analysis we wanted to find commonalities between these commands, especially in how they relate to querying, saving, and storing information from the simulation. As such, there were two major forms of command:

- **Print** command – A command that took the form of `print(SUBJECT.INFORMATION)` or `print(INFORMATION)`, where the `SUBJECT` is one of the default named entities found in *Bad News* (e.g., pc = 'Player Character', d = 'Deceased', etc.) or a previously assigned entity from an **Assignment** command, and `INFORMATION` is the specific piece of information that is being requested (e.g., current location, family structure, residence address, etc.).

– **Assignment** command – A command that took the form of `LABEL = VALUE`, where `LABEL` is the *Wizard* assigned label for the queried information (e.g., mom, dad, bar, susan) , and `VALUE` is the specific piece of information or entity being assigned to that label.

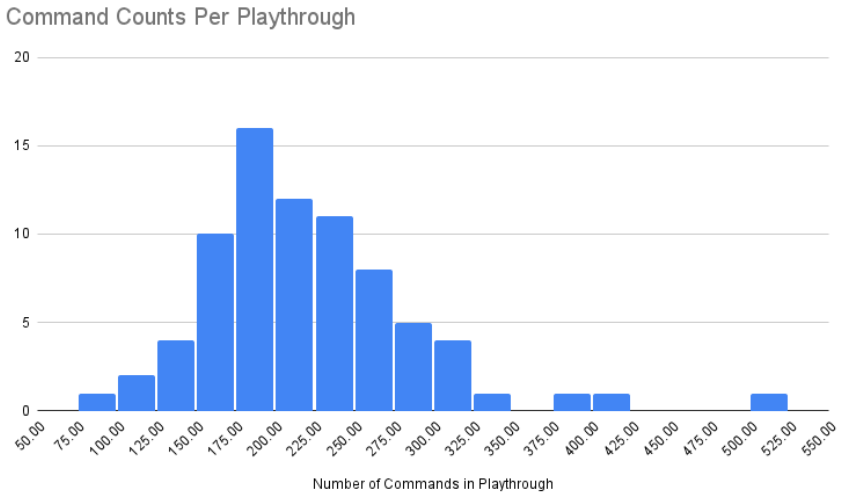
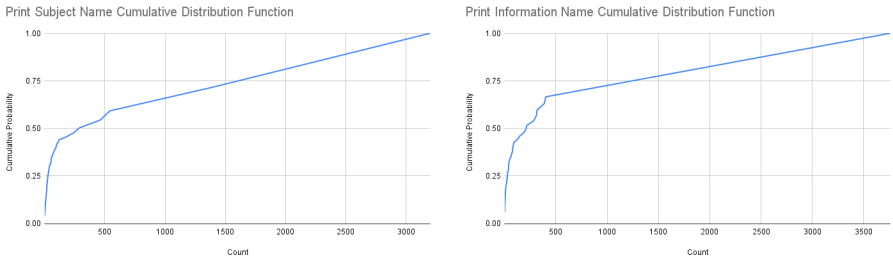


Fig. 1: A histogram of the number of commands per playthrough across 81 runs of *Bad News*.

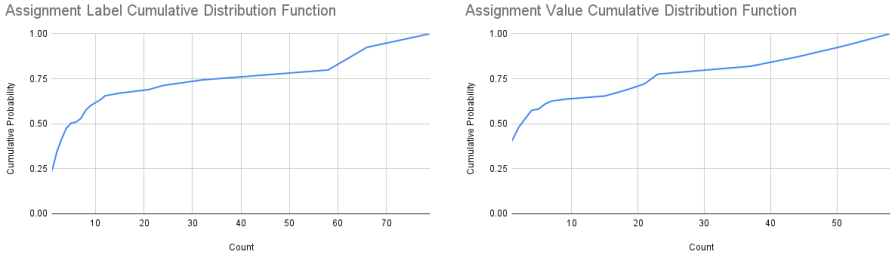
The 81 playthroughs comprise 12,084 commands, with median length of 209 commands in a playthrough (see figure 1 for a histogram of number of commands per playthrough). Of these 12,084 commands 16,866 (71.6%) were either a **Print** or **Assignment** command.

Looking to the **Print** commands, we found that the single most common subject (the deceased) showed up on 28.4% of all print commands (3,199 commands in total), while subjects that showed up only once only accounted for 3.8% of print commands (432 commands) – the 5 most common subjects were (1) deceased at 3,199 commands, (2) player at 1,392, (3) mom at[542, (4) dad at 463, and (5) next-of-kin at 293. Figure 2a shows the full Cumulative Distribution Function (CDF) of **Print** command subject counts. The most common information was in fact, no information – which in Python would display the entire object – occurring 33.4% of the time (3,757 commands), and again the information that was requested only once occurred 6.2% of the time (694 commands) – the 5 most common pieces of information were (1) ∅ at 3,757 commands, (2) location at 404, (3) occupation at 404, (4) occupations at 387, and (5) home at 311. Figure 2b shows the full Cumulative Distribution Function (CDF) of **Print** command information counts.



(a) (b)

Fig. 2: Cumulative Distributions for Print related commands (2a for Subject, 2b for Information)



(a) (b)

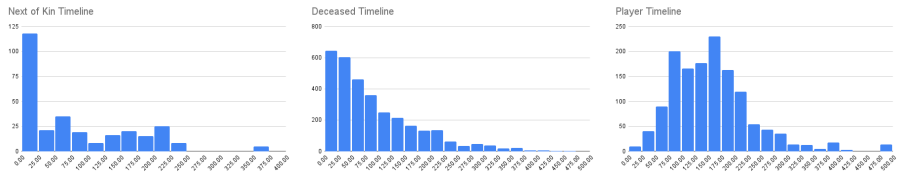
Fig. 3: Cumulative Distributions for Assignment related commands (3a for Label, 3b for Value)

Turning to the **Assignment** commands, we first note that they were much less common than the **Print** commands – 1,048 **Assignment** vs 11,260 **Print** commands. Another note is that the **Assignment** commands are not as extreme as **Print** commands – the most common label ('bars') only accounts for 7.5% and the most common value (`d.mother, d.father`) only accounts for 6.8% of the **Assignment** commands respectively. We note that the discrepency in that the most common label does not match the most common information – this comes from the fact that 'bars' was often assigned to multiple times in the face of missing information (e.g., First `g.city.businesses_of_type('Bar')`, then `g.city.businesses_of_type('Hotel')`, and finally

`g.city.former_businesses_of_type('Bar')`). The five most common labels were (1) bars at 79 commands, (2 & 3) mom & dad at 66, (4) 'sibs' (siblings) at 58, and (5) hotel at 32. Figure 3a shows the CDF for assignment labels. The five most common values were (1) `d.mother, d.father` at 66 commands, (2) `d.siblings` at 58 , (3) `g.city.businesses_of_type('Bar')` at 44,

(4) `d.love_interest` at 37, and (5) `nok[0]` at 24. Figure 3b shows the CDF for assignment information.

Both **Print** and **Assignment** commands take the form of a Pareto distribution [26] – sometimes known as the 80-20 distribution (i.e., 80% of outcomes are due to 20% of causes and 20% of outcomes are due to 80% of causes). In the case of Bad News **Print** commands – the top 5 most common subjects account for over 50% of all use-cases, and the top 20 account for two-thirds. This means that being able to prioritize those 20 subjects will cover the bulk of interactions, easing the requirements for a *WizActor*.



(a) (b) (c)

Fig. 4: Command usage as a function of turn number in *Bad News* playthroughs.

We also analyzed the time distribution of commands – knowing when they are more likely occur can help orient the design of the *WizActor* interface. If they show up early in a playthrough, then it would perhaps be best to include them in an initialization step, while if they show up throughout the playthrough then they should be easily accessible on the fly. Figure 4 shows the histogram of when a subject appears in the course of the playthroughs. We see that there are a range of patterns that appear in *Bad News*. The next of kin is mostly referenced at the beginning, and would be well suited to mostly being handled in an initialization step. The deceased is referenced most at the beginning, but is also referenced throughout the playthrough. Finally, the player character is not referenced much at the beginning, but is referenced throughout the playthrough, mostly for the sake of being moved around the world. The differences in these usage patterns speaks to the need to design different ways of accessing and manipulating the simulation depending on the type of entity.

Finally, we wanted to assess the kinds of patterns found in the commands. If certain patterns of commands show up commonly together, then it stands to reason that the creation of *macros* to handle those patterns might be useful in easing the burden of the *WizActor*. To assess this, we used Byte Pair Encoding (BPE) [7] – a technique where the most common pair of tokens is replaced with a new token, a process that is repeated until a vocabulary of token sequences is filled up to a certain size. In this case, we first anonymized the variable names – e.g. if there was a command sequence of:

```
dad.occupations[-1]
dad.occupations[-1].terminus.date
```

We would want it to be considered the same as:

```
patricia.occupations[-1]
patricia.occupations[-1].terminus.date
```

I.e., we only care about the macro pattern of "Find the last job this person had, and its end date". So we turn those into:

```
ENTITY.occupations[-1]
ENTITY.occupations[-1].terminus.date
```

After running a character level BPE to fill up a vocabulary of size 5,000, we sort the vocabulary by size, to find long, common sequences. We found some interesting patterns emerge:

"Get general info and job information of the person you are speaking to."

```
print ENTITY.all_interlocutors[INT]
print ENTITY.all_interlocutors[INT]
print ENTITY.all_interlocutors[INT].occupations
print ENTITY.all_interlocutors[INT].occupations[INT]
print ENTITY.all_interlocutors[INT].occupations[INT]
```

"Get the family tree of a person"

```
print ENTITY.father
print ENTITY.father.father
print ENTITY.mother
print ENTITY.mother.father
print ENTITY.mother.father.father
print ENTITY.mother.father.father.father
print ENTITY.mother.father.father.occupations
```

"Get the personality traits of a person"

```
print ENTITY.personality.e
print ENTITY.personality.a
print ENTITY.personality.o
print ENTITY.personality.c
print ENTITY.personality.n
```

All of these are sequences of commands that could be turned into macros that get all of the information at once. Furthermore, the different types of information could (and should) be presented to the *WizActor* in different ways. A family tree is different than a history is different than the presentation of 5 numbers – each should be presented in ways to assist the *WizActor* in understanding the most salient details quickly and easily.

4 The *WizActor* Interface

The *WizActor* tool is currently a prototype and in active development. As discussed above, its primary purpose is to provide an interface that makes exploring the underlying simulation for dramatic backstory elements and inspiring

future moves simple and accessible. Though ultimately intended for use in many simulation-driven performance experiences, in the parlance of *Bad News*, it is intended to facilitate the *Wizard*'s most common needs (outlined in section 3), such that the roles of *Actor* and *Wizard* could conceivably be collapsed and performed by a single individual with minimal cognitive overhead.

The prototype is being developed as a web application using *Data Driven Documents* (the *D3.js* library) [3]. Figure 7 depicts an early version of the tool, demonstrating its initial functionality. Though the inspiration for the design of this tool comes in part from the aforementioned analysis of *Bad News* playthroughs, it is intended to be usable for a variety of domains and game experiences that depend on human performers (e.g., the Bad News "actor", a Tabletop Role Playing Game "game master", etc.) parsing through, discovering, and delivering simulation-generated content. To showcase a different narrative domain, the underlying data used in the following examples was generated via a *Kismet* simulation that represents a collection of thirty regency era socialites attending a ball. Each character in the simulation is composed of individual character traits (such as "snoopy", "extrovert", and "drunkard"), statuses (such as "sad", "drunk", and "embarrassed") and is depicted as a node in a *D3* force directed graph. The edges of the graph are directed and represent the source character's affinity towards the target; unbounded integers that could be negative (indicating disdain) or positive (indicating affection). In the following figures the solid green edges represent positive affinities and the dashed red edges represent negative ones.

Figure 5 presents a view of the entire interface. The top portion depicts the complete cast of characters, who can be added or removed from the graph individually via checkboxes. Beneath that there is a slider that adjusts the "relationship threshold" value which impacts how many edges between nodes are drawn. Beneath that is trait and status information about a selected character (here Penelope Leosfel), and to the right of that are buttons that highlight narratively salient patterns (such as "Find a Rival" or "Find all Friends"). Beneath all that is, finally, the graph itself.

Being able to add characters to the graph individually is helpful; in *Bad News*, for example, players were often most concerned with learning more information about characters they had already met. This allows for the *WizActor* to gradually add characters as the player encounters them, focusing the information offered by the *WizActor* interface to be most pertinent to the current game. However, as discussed in section 3, the *Wizard* also commonly needs to query the simulation for information regarding characters that the player has yet to encounter – either at the start or in media res. However, these queries nearly always were in relationship to characters the player had already encountered or learned about – e.g, finding the family tree of an encountered character. The *Kismet* world used for this didn't have familial relations, but finding friends, love interests, and rivals are spiritually similar tasks. There are a set of buttons (e.g., *Find a Friend* and *Find a Rival*) that adds a new character to the graph who fulfills the desired pattern when pressed. This automated search for patterns is

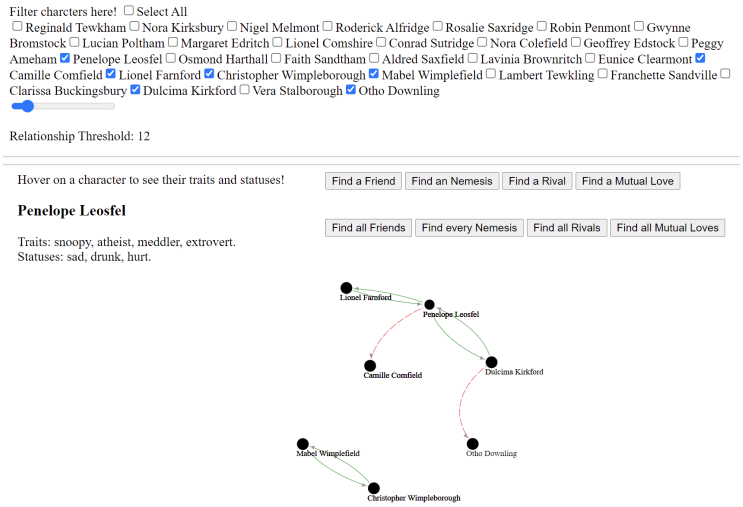


Fig. 5: An early prototype of the *WizActor* interface using generated *Kismet* data simulating a regency era ball. Solid green lines signify positive affinities, dashed red lines represent signify negative affinities.

a simple form of story-sifting, and enables the *WizActor* to quickly discover new characters to weave into the story. The patterns presented here ("friend", "rival", etc.) were authoring decisions specific to this particular simulated world. *Kismet* permits authors to create their own patterns of arbitrary complexity, and the *WizActor* interface dynamically populates its search functionality based on the current world.

Though its primary intended use case is to facilitate live story-sifting while in-game, the *WizActor* interface is also a helpful mechanism for better understanding the generative space of a *Kismet* simulation, and can serve as a tool to verify that generated content is desirable. Figures 6, 7a, and 7b highlight some of this sanity checking.

Figure 6 depicts the results of clicking the "Find all Friends" button; it introduces into the graph every character who appears in the "Friends" pattern. Note how the resulting graph is clearly partitioned; the simulation has naturally created a group of people who all love each other (through the mutually connected solid green lines towards the top of the graph) and another group that all despise each other (connected through the dashed red lines towards the bottom). This is an interesting phenomenon in and of itself—it enables the author of the simulation to reflect on whether this result is desirable and to modify the *Kismet* file accordingly—but it also reveals potentially interesting characters for narrative moves. Here, there are a few characters, such as Peggy Ameham, Geoffrey Edstock, and Dulcima Kirkford who are generally liked but who also have connections to the "angry" group. Those characters serve as narratively

interesting candidates if the *WizActor* needs to find a way to ferry the player from one group to another.

Figures 7a and 7b highlight the significance of the aforementioned "relationship threshold" value. In this *Kismet* world, every character has an affinity value for every other character; depicting them all would result in a graph that is difficult to decipher. The threshold value determines the minimum magnitude a relationship must have in order to be drawn (e.g., if the value is twelve, then dashed red edges will only be drawn at negative twelve or less, and solid green edges will only be drawn at twelve and above). Figure 7a has the threshold at twenty, and consequently shows significantly fewer links, while figure 7b relaxes the threshold to eight (and begins veering dangerously close to losing decipherability). Higher numbers can be used to discover particularly strong connections between characters, which can form the basis of narrative content. For example, Figure 7a tells us that Alfred and Lionel have a stronger affinity for one another than any other pair in the simulation, and Camille and Lambert are linked via a very interesting chain of people who strongly hate each other. Both of these phenomena were naturally occurring—but buried—in the simulation, but the *WizActor* tool unearthed them easily, offering potential narrative fuel to in turn to be surfaced to the player. Similarly, lowering the threshold consequently shows more links, further confirming the two 'factions' here and revealing additional characters that could be considered "on the border" between the two groups.

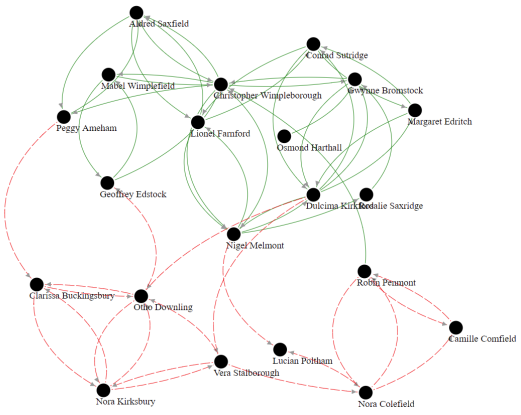
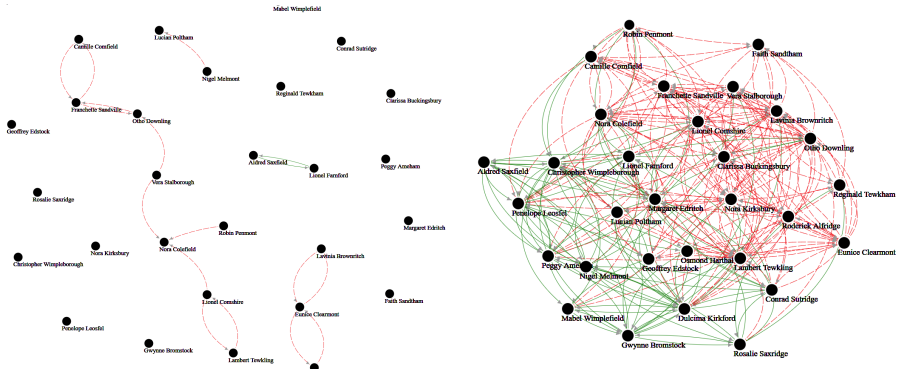


Fig. 6: A depiction of the interface after the "Find all Friends" button is pressed

5 Conclusion and Future Work

This paper presents initial findings from a quantified analysis of *Wizard* logs from the game *Bad News*. These findings revealed the commands most often



(a) (b)

Fig. 7: Two *WizActor* graphs depicting character affinities. Figure 7a includes all thirty characters with the "relationship threshold" value set to twenty. 7b is the same but with the value at eight.

entered by the *Wizard* while playing *Bad News*, which in turn speaks to the simulated information and emergent narrative most crucial to be surfaced for a successful play-through. These findings have informed the development of a *WizActor* interface, intended to be used in any piece of computationally assisted table-top role-playing powered by the social simulation system *Kismet*. Though the interface is still in its early stages, it shows promise as both a mechanism for story-sifting, as well as for visualizing the generative space of *Kismet* simulations. Additionally, the authors hope this work highlights why these types of evaluations are important, as the insights garnered from them have the potential to enable entirely new sets of work. We encourage other researchers and developers to analyze their own playthrough data (or make it publicly available) to enable these valuable lines of research.

There is ample future work remaining on this project. Further analysis of *Bad News*, taking player actions and *Wizard-Actor* communication into account, remains to be performed. There is still additional wisdom learned from the existing *Bad News* analysis to integrate into the *WizActor* interface. As this work continues, the authors hope to leverage *Kismet* and the *WizActor* interface to develop a new "*Bad News*-like" game to serve simultaneously as evaluation of these systems, and a standalone piece of creative work. Once completed, the ultimate plan is to provide these tools to all, to nurture a blossoming genre of computationally assisted performance games. Additionally, a professional developer has been working with *Kismet* and the *WizActor* interface, offering their insight and feedback on using these tools. Codifying this feedback could be valuable future work of interest to interactive storytelling researchers that aspire to produce narrative content and tools that are relevant in both academic and industry circles.

References

1. Adams, T., Adams, Z.: Dwarf fortress. Game [Windows, Mac, Linux], Bay **12** (2006)
2. Aristotle: Poetics (Penguin Classics). Penguin Classics (1997), http://www.amazon.com/Poetics-Penguin-Classics-Aristotle/dp/0140446362
3. Bostock, M., Ogievetsky, V., Heer, J.: D^3 data-driven documents. IEEE transactions on visualization and computer graphics **17**(12), 2301–2309 (2011)
4. Compton, K., Kybartas, B., Mateas, M.: Tracery: an author-focused generative text tool. In: International Conference on Interactive Digital Storytelling. pp. 154–161. Springer (2015)
5. Compton, K., Mateas, M.: Casual creators. In: Proceedings of the Sixth International Conference on Computational Creativity. p. 228 (2015)
6. DeKerlegand, D., Samuel, B., Leichman, J.: Encoding socio-historical exegesis as social physics predicates. In: International Conference on the Foundations of Digital Games. pp. 1–9 (2020)
7. Gage, P.: A new algorithm for data compression. C Users Journal **12**(2), 23–38 (1994)
8. Garbe, J., Reed, A.A., Dickinson, M., Wardrip-Fruin, N., Mateas, M.: Author assistance visualizations for ice-bound, a combinatorial narrative. In: FDG (2014)
9. Gygax, G., Cook, D.: The Dungeon Master Guide, No. 2100, 2nd Edition (Advanced Dungeons and Dragons). TSR, Inc (1989)
10. Horswill, I.: Imaginarium: A tool for casual constraint-based pcg. In: Proceedings of the AIIDE Workshop on Experimental AI and Games (EXAG) (2019)
11. Horswill, I.D.: Fiascomatic: A framework for automated fiasco playsets. In: Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference (2015)
12. Horswill, I.: Dear leader's happy story time: A party game based on automated story generation. In: WS-16-21. pp. 39–45. AAAI Workshop - Technical Report, AI Access Foundation (Jan 2016), 12th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2016 ; Conference date: 08-10-2016 Through 09-10-2016
13. Kiai, D.: Coffee! a misunderstanding (2014)
14. Kreminski, M., Dickinson, M., Mateas, M.: Winnow: A domain-specific language for incremental story sifting. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. vol. 17, pp. 156–163 (2021)
15. Kreminski, M., Dickinson, M., Wardrip-Fruin, N.: Felt: A simple story sifter. In: International Conference on Interactive Digital Storytelling. pp. 267–281. Springer (2019)
16. Leece, M.A., Jhala, A.: Sequential pattern mining in starcraft: Brood war for short and long-term goals. In: Tenth Artificial Intelligence and Interactive Digital Entertainment Conference (2014)
17. Marsella, S.C., Pynadath, D.V., Read, S.J.: Psychsim: Agent-based modeling of social interactions and influence. In: Proceedings of the international conference on cognitive modeling. vol. 36, pp. 243–248 (2004)
18. Mateas, M., Stern, A.: Façade: An experiment in building a fully-realized interactive drama. In: Game developers conference. vol. 2, pp. 4–8 (2003)
19. Mathewson, K.W.: Humour-in-the-loop: Improvised theatre with interactive machine learning systems (2019)

20. McCoy, J., Treanor, M., Samuel, B., Reed, A.A., Wardrip-Fruin, N., Mateas, M.: Prom week. In: Proceedings of the International Conference on the Foundations of Digital Games. pp. 235–237 (2012)
21. McCoy, J., Treanor, M., Samuel, B., Reed, A.A., Mateas, M., Wardrip-Fruin, N.: Social story worlds with comme il faut. IEEE Transactions on Computational intelligence and AI in Games **6**(2), 97–112 (2014)
22. Meehan, J.R.: Tale-spin, an interactive program that writes stories. In: Proc. of the 5th International Joint Conference on Articial Intelligence, Aug. 1977. vol. 1, pp. 91–98 (1977)
23. Nelson, M.J., Mateas, M., Roberts, D.L., Isbell Jr, C.L.: Declarative optimization-based drama management (dodm) in the interactive fiction anchorhead. Computer Graphics and Applications **26**(3), 32–41 (2006)
24. Osborn, J.C., Samuel, B., McCoy, J.A., Mateas, M.: Evaluating play trace (dis) similarity metrics. In: Tenth Artificial Intelligence and Interactive Digital Entertainment Conference (2014)
25. Padovani, R.R., Ferreira, L.N., Lelis, L.H.: Bardo: Emotion-based music recommendation for tabletop role-playing games. In: Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference (2017)
26. Pareto, V.: Cours d'économie politique: professé à l'Université de Lausanne, vol. 1. F. Rouge (1896)
27. Rameshkumar, R., Bailey, P.: Storytelling with dialogue: A critical role dungeons and dragons dataset. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 5121–5134 (2020)
28. Roberts, D.L., Isbell, C.L.: A survey and qualitative analysis of recent advances in drama management. International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning **4**(2), 61–75 (2008)
29. Ryan, J.: Curating simulated storyworlds. Ph.D. thesis, UC Santa Cruz (2018)
30. Ryan, J.O., Summerville, A., Mateas, M., Wardrip-Fruin, N.: Toward characters who observe, tell, misremember, and lie. In: Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference (2015)
31. Ryan, J.O., Summerville, A.J., Samuel, B.: Bad news: A game of death and communication. In: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems. pp. 160–163 (2016)
32. Ryan, M.L.: From narrative games to playable stories: Toward a poetics of interactive narrative. Storyworlds: A Journal of Narrative Studies **1**, 43–59 (2009)
33. Sadati, S.H., Mitchell, C.: Serious game design as research-creation to address sexual and gender-based violence. International Journal of Qualitative Methods **20**, 16094069211046130 (2021)
34. Samuel, B., Ryan, J., Summerville, A.J., Mateas, M., Wardrip-Fruin, N.: Bad news: An experiment in computationally assisted performance. In: International Conference on Interactive Digital Storytelling. pp. 108–120. Springer (2016)
35. Shibolet, Y., Knoller, N., Koenitz, H.: A framework for classifying and describing authoring tools for interactive digital narrative. In: International Conference on Interactive Digital Storytelling. pp. 523–533. Springer (2018)
36. Smith, A.M., Lewis, C., Hullett, K., Smith, G., Sullivan, A.: An inclusive taxonomy of player modeling. University of California, Santa Cruz, Tech. Rep. UCSC-SOE-11-13 (2011)
37. Smith, G., Whitehead, J.: Analyzing the expressive range of a level generator. In: Proceedings of the 2010 Workshop on Procedural Content Generation in Games. pp. 1–7 (2010)

38. Summerville, A., Samuel, B.: Kismet: a small social simulation language. In: Summerville, A., & Samuel, B.(2020, September). Kismet: a Small Social Simulation Language. In 2020 International Conference on Computational Creativity (ICCC).(Casual Creator Workshop). ACC (2020)
39. Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A.K., Isaksen, A., Nealen, A., Togelius, J.: Procedural content generation via machine learning (pcgml). IEEE Transactions on Games **10**(3), 257–270 (2018)
40. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games **3**(3), 172–186 (2011)
41. Worthen, W.B.: " the written troubles of the brain":" sleep no more" and the space of character. Theatre Journal pp. 79–97 (2012)