# TaleBrush: Sketching Stories with Generative Pretrained Language Models

John Joon Young Chung
jjyc@umich.edu
University of Michigan
Ann Arbor, MI, USA

Wooseok Kim
ooooseok@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Kang Min Yoo
kangmin.yoo@navercorp.com
Naver AI LAB
Seongnam, Republic of Korea

Hwaran Lee
hwaran.lee@navercorp.com
Naver AI LAB
Seongnam, Republic of Korea

Eytan Adar
eadar@umich.edu
University of Michigan
Ann Arbor, MI, USA

Minsuk Chang
minsuk.chang@navercorp.com
Naver AI LAB
Seongnam, Republic of Korea

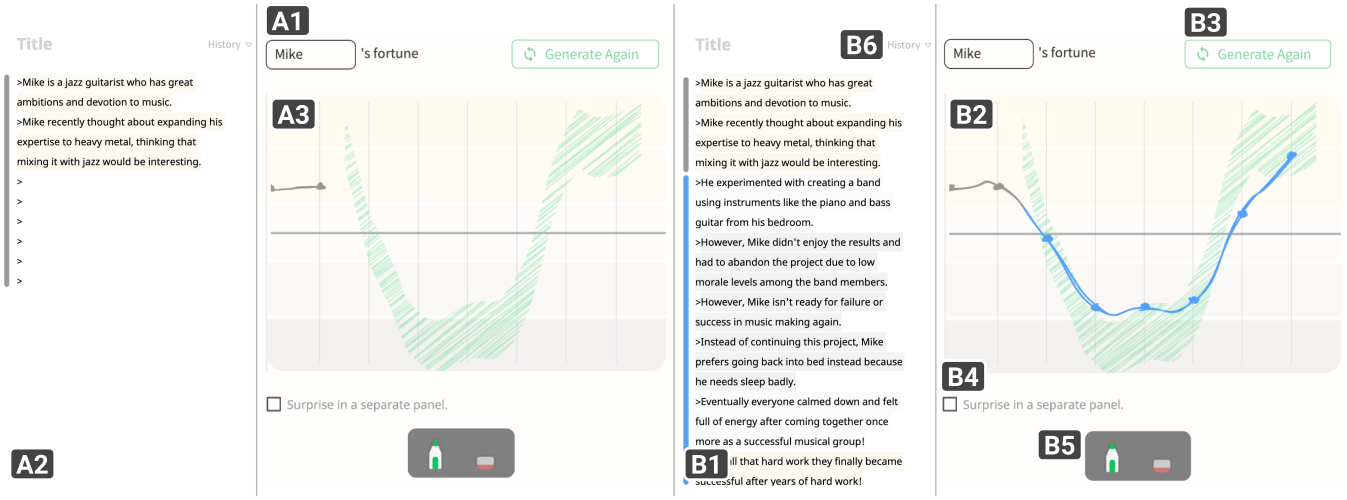


**Figure 1: TaleBrush uses a line sketching interaction for intuitive control and sensemaking of story generation with GPT-based language models, closing the gap in the iterative co-creation process between humans and AI. In our prototype, the protagonist's 'fortune' is controllable. The writer first decides the protagonist's name (A1), writes a portion of a story (A2), and then sketches how the protagonist's fortune should change in the story (A3, green shaded area). The $x$ position of the sketched line indicates the chronological position in the story (sentences are the units). The $y$ position shows how good or bad the protagonist's fortune should be (higher, better). The width of the line indicates the possible variance in the fortune of the generated sentences. Given the line sketch, TaleBrush will generate story sentences (B1, indicated with blue) and visualize the result on the original sketch (B2, the blue line and dots). The writer can always directly edit the generated text. They can also iterate by generating new text for the same sketch (B3, clicking on 'Generating Again') or revising their sketch and generating new text. Additional options allow the writer to specify how 'surprising' the generation should be (B4) or using the eraser tool (B5) to erase their sketch. Where the line is erased, TaleBrush generates unconstrained sentences. A history dropdown (B6) allows the writer to browse previously generated sentences.**


*CHI '22, April 29-May 5, 2022, New Orleans, LA, USA*

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3501819

## ABSTRACT

While advanced text generation algorithms (e.g., GPT-3) have enabled writers to co-create stories with an AI, guiding the narrative remains a challenge. Existing systems often leverage simple turn-taking between the writer and the AI in story development. However, writers remain unsupported in intuitively understanding the AI's actions or steering the iterative generation. We introduce TaleBrush, a generative story ideation tool that uses line sketching interactions with a GPT-based language model for control and sensemaking of a protagonist's fortune in co-created stories. Our

empirical evaluation found our pipeline reliably controls story generation while maintaining the novelty of generated sentences. In a user study with 14 participants with diverse writing experiences, we found participants successfully leveraged sketching to iteratively explore and write stories according to their intentions about the character's fortune while taking inspiration from generated stories. We conclude with a reflection on how sketching interactions can facilitate the iterative human-AI co-creation process.

## CCS CONCEPTS

• **Human-centered computing → Interactive systems and tools**;
• **Computing methodologies → Natural language generation**.

## KEYWORDS

story writing, sketching, creativity support tool, story generation, controlled generation

**ACM Reference Format:**
John Joon Young Chung, Wooseok Kim, Kang Min Yoo, Hwaran Lee, Eytan Adar, and Minsuk Chang. 2022. TaleBrush: Sketching Stories with Generative Pretrained Language Models. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3491102.3501819

## 1 INTRODUCTION

Advances in pretrained generative language models, such as Open AI's GPT-3 [119], have enabled new kinds of human-AI story co-creation tools [24, 27–29, 61, 62]. In many of these tools, the story co-creation process is iterative [24, 28, 29, 51, 143]: first, a writer gives the initial story sentences, such as "Melissa fought a dragon." Then, using the writer's sentence as a prompt, the AI tool appends story sentences or phrases. For example, the tool can generate: "She used magic to create a barrier around her." The writer can modify the story directly, start from scratch, hoping for better results, or continue the iterative process by adding another sentence.

However, consider a case where the writer has some idea of how the story should unfold. They would like to build tension for Melissa by giving her some bad luck with the dragon and only in the end boost her fortune. Unfortunately, steering the text generation in this way is hard both from the algorithmic and interaction perspective. First, most controlled story generation algorithms, which can be guided with inputs like topical keywords, apply the same control to the whole generated story. However, as the writer wanted Melissa to experience bad luck first and then end up with a happy ending, they would need to specify different parameters to different parts of the story. Such granular sequence control is not yet possible in many controlled story generation algorithms. Second, even when the generation algorithm allows such granular sequence control, existing interfaces (e.g., text inputs or sliders) are not intuitive or useful for controlling or sensemaking in the generative process. For example, when there is a series of granular controls (e.g., sliders), specifying them can be cumbersome (e.g., inputting multiple numerical control values). Sensemaking, or understanding what the algorithm will do, or has done, is similarly difficult. To understand if the generated story follows the given parameters, writers have to compare each generated story part with each granular parameter. These frictions make iterative co-creation slow and effortful.

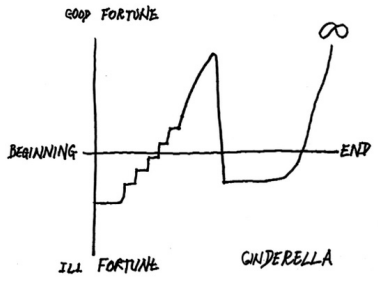


**Figure 2: Kurt Vonnegut's sketched line drawing of *Cinderella's* fortune over the course of her story [142].**

In this work, we propose that visual sketching interactions can facilitate granular sequence control and sensemaking in iterative story generation. Sequential properties of stories have frequently been visually expressed for intuitive understanding or planning [59, 71, 79, 92, 118, 121, 137, 138, 142]. For example, Kurt Vonnegut [142] drew out how a character's fortune changes in a simple time-series line drawing (Figure 2). This line drawing approach has inspired writers to adopt visual arcs to plan out their stories [34, 100, 122].

With this visualization as inspiration we created TaleBrush (Figure 1). TaleBrush is focused on human and AI co-writing short story outlines by allowing writers to control the generation according to the level of fortune the protagonist experiences [34, 121, 142]. Writers can specify the sequence of a character's fortune with a *simple* and *expressive* interaction (Figure 1A3). Using a single stroke (*simple*), the writer defines the fortune over the entire protagonist's 'time-series' (*expressive*). The sentences generated by TaleBrush are reflected back to the user as a line drawn on top of the original sketch (Figure 1B2). The lightweight and intentionally ambiguous nature of sketching [49, 84] manages the writer's expectation regarding the fidelity of control, making uncertainties and errors in controllability acceptable. To realize this interaction, we implemented a technical architecture that achieves steerable story generation by training 1) learnable prompts that guide a language model to serve different story generation tasks [87] and 2) a control module that steers the language model to generate stories according to the given sketch [83].

We conducted a technical evaluation of TaleBrush and found that our approach can reliably steer generations that are still novel and tolerably coherent in flow and grammar. From a user study on 14 participants with varying expertise in story writing, we found that participants could use the sketching control to iteratively steer the generation to find inspiring stories that align with their intentions about the protagonist's fortune. Some participants also used line sketches as a plan when editing the generated texts. In our discussion section, we lay out the design elements for interactions that facilitate iterative human-AI co-creation. We also reflect on how visual sketching interactions can be extended to other story-writing attributes and domains while supporting reliable steerability even under the uncertainty of machine learning algorithms. This work

opens up a way of using visual controls and visualizations to make iterative human-AI co-creation intuitive and frictionless.

To summarize, our work contributes: 1) A line sketching interaction for control and sensemaking of story generation with an abstract graphical representation of sequential attributes. 2) A GPT-based controllable language model architecture that generates story sentences with the sketching input on the protagonist's fortune. 3) TaleBrush, a system that enables human-AI story co-creation with the protagonist's fortune arc by combining line sketching interaction with GPT-based controllable language model. 4) Reflections on generalizable design elements of control interaction for iterative human-AI co-creation and how line sketching interaction can be expanded to other generative contexts.

## 2 RELATED WORK

We review research on four main areas related to TaleBrush: 1) writing support tools, 2) story generation, 3) visual expressions of stories, and 4) sketching.

### 2.1 Writing Support Tools

Many tools support different aspects of writing. These range from spelling and grammar correction [85, 115], to thesauruses [45], to crowd-powered editors [11, 108]. As writing tasks and styles are often domain-specific, tools can be similarly specialized: email message phrasing [23], help requests in professional contexts [69], mental support [114], affectionate messaging [80], education [22], and journalism [95]. Of specific interest to us are tools for creative writing tasks. These run the gamut from suggesting metaphors [46] to helping with song lyrics [147]. Commercial tools such as Dramatica or Plottr are largely human-driven and help the author *structure* their plot or narrative [19, 43].

To *produce* content, some tools connect the author to other humans or support collective story writing by writing stories with the crowd. These systems can structure creative leadership [77] and decision-making [78], or even simulate the characters with crowds [63]. Newer tools have adopted novel machine learning (ML) and natural language processing (NLP) techniques [153]. Among those techniques, ML's generative functions offer a powerful alternative approach. These systems append or suggest generated texts to the writer's text using sophisticated language models [24, 28, 29, 129, 143]. In theory, these algorithms can act as writing support tools. However, limited controls and interactions—largely rephrasing prompts and contexts—constrain their applicability. Our goal with TaleBrush is to leverage these language models but provide an alternative control strategy using both text and abstract visual representations and interactions.

### 2.2 Machine Story Generation

Story generation is one of the grand challenges in artificial intelligence (AI) with multiple practical applications. These range from entertainment [124] and education [98, 124] to our target, writing support [24, 28]. Early approaches included the use of story block templates that could be put together sequentially [30]. With computational techniques, autonomous story generation became feasible. One technique is computational planning, where the computer does symbolic planning to accomplish a given goal [86, 101, 113, 125, 146]. Case-based reasoning techniques enable story generation by adapting stored stories to new contexts [47, 117, 123, 135, 139]. Alternative strategies involve character-based simulation, where characters and the world are simulated to unfold the story with the given facts and underlying beliefs [25, 94].

Advances in ML, largely in language models, have introduced more opportunities in story generation [5, 6, 53, 64, 70, 89, 90, 99, 136, 145]. Specifically, Transformer-based Language Models (LMs) [21, 119, 140] probabilistically sample continuing stories [41] or infilling sentences [5, 35, 67, 70, 89, 104, 111, 145] based on the given story context. While imperfect in generating coherent text, they hold promise for improved story generation.

Controllability is a key requirement for all these approaches. However, LM models, in particular, utilize structures that are much harder to inspect and explain. For the algorithms to be useful and usable, new control mechanisms are needed [28]. The most basic, and perhaps obvious, controls are driven by natural language prompts [36] (e.g., "Tell me a happy story"). To better guide LMs, researchers investigated ways to automatically learn these prompts [87, 88, 93, 131]. Other language-based approaches for control involve specifying keywords to prime the generated text [40, 70, 134, 152]. Alternative approaches, like Intent-Guided Authoring (IGA), provide some degree of control over output sentences [134]. The author can use a number of pre-specified tags (e.g., *cause* or *effect*) to describe the desired relationship of output to input. Researchers also introduced algorithms that generate stories while considering the character goals or abilities [3, 136]. Control codes, such as genre (e.g., is the story fantasy or science fiction?), can also provide some guidance. Technically, this approach has been enabled by training models with control codes [54, 75], or manipulating intermediate representation layers of LMs [26, 31]. Researchers have also introduced the approach of first acquiring keywords about the code, and generating the story based on them [82].

As LMs grow with more parameters, adopting these approaches became more challenging, as most LM parameters need to be tuned. To overcome this, researchers train smaller models that guide the large LMs with control codes [83]. While many approaches apply the control to the whole story (e.g., 'the entire story should be science fiction'), a few approaches demonstrated how different parameters can be applied to different parts of the story [90, 120, 154]. However, they did not consider the interaction of the user control generation algorithms. Building on this prior work, we extend both the technical and interaction sides of controllable story generation. We designed our story generation model to be controlled with real-valued sequence inputs from line sketching interaction. There has been a relevant thread of work called visual story telling [10, 60–62, 66, 144], that writes stories about visual scenes (e.g., a photo of a dog running becomes "A dog was chasing down a rabbit"). In contrast, TaleBrush uses sketches of an *abstract* attribute of the story (e.g., the protagonist's fortune).

### 2.3 Visualizing and Visually Expressing Stories

To allow users to specify sequential attributes of the story and capture how texts are generated with controls, TaleBrush adopts abstract visual representations. Visual techniques have been used to show high-level attributes of documents [73] and stories (e.g., [17]).

Digital humanities researchers have used visualizations to understand stories–a form of "distant reading" [71]. Through "distant reading", diverse story components can be visualized. These range from character interactions [12, 15, 72, 110, 151], event progression [59, 92, 137, 138], character emotions and sentiment [7, 12, 32, 57, 107, 121, 127]. As some stories can be non-linear in their telling, researchers have found ways of representing broader narrative structures [79, 130]. Our goal is not simply to represent the story through the visualization but to allow for manipulation through that representation. However, not all visualization formats are naturally amenable to direct manipulation (beyond the standard interaction techniques [155]).

With TaleBrush, we focus on visual encodings that can both express high-level changes in the story's progression but also be manipulable. Kurt Vonnegut's story arcs fit these requirements. The arcs were used by Vonnegut to express canonical story types with diagrams that show how a protagonist's fortune changed in the story [142]. For example, in Figure 2, *Cinderella's fortune* is expressed with a curve that rises, drastically falls, and then rises again. Writers have since leveraged this type of visual expression to plan out story writing [34, 100, 122]. TaleBrush leverages this approach to allow for drawing a property, such as fortune, over time as a way of guiding the generative process.

## 2.4 Sketching Interactions

To visually express sequential attributes of stories, TaleBrush uses sketching interaction. Sketching is a flexible and lightweight way to convey the user's high-level intentions [38] with its roughness, uncertainty, and ambiguity [16, 49, 84]. Because of these features, various sketching tools have been built to embody these characteristics [49] or act as mechanisms to transform rough ideas into precise representations (e.g., visual renderings [84]). This interaction style has some similarity to TaleBrush, but without the direct mapping between objects (e.g., draw a cat to get a photorealistic cat). In the case of TaleBrush, we are creating a sketch in the abstract time-series space that will generate a corresponding story in the 'concrete' text space.

While sketching is most often used in a 'direct' way (e.g., drawing a sketch), it also has been used for constructing examples (e.g., for image [39, 65] or 3D model search [81, 112]). More relevant to our approach are tools for sketching time-series 'queries' for finding relevant items in time-series datasets. These can be used to find everything from fixed patterns [74] to constrained inputs [58, 126] to free sketches [37, 42, 96, 132, 148, 157]. However, most systems here assume the existence of matching datasets. In our case, the dataset does not yet exist. Thus, our approach does not sit cleanly in the regular taxonomy of visualization interaction [55, 155]. With TaleBrush, we leverage recent work on interactive visualization tools for data generation. However, these are largely focused on numerical approaches rather than text (e.g., generating a dataset that matches a sketched histogram [97]).

# 3 SKETCHING SEQUENCE CONTROL

Though narratives may not be linear, stories are inherently formed in a sequence, progressing from the story beginning to the ending. Within that sequence, attributes of the story or characters (e.g., the protagonist's fortune) can change. Our goal is to allow the user to use these attributes to guide the generated sequence. Specifically, we seek a lightweight mechanism to quickly specify multiple temporally-varying attributes. In this work, we consider line sketching interaction as the modality. There are a number of features of this interaction modality that make the approach appropriate. First, line sketching in time series visualization can be *expressive* enough to convey sequence and attribute information [1, 106]. For example, in Figure 2, the sequence can be shown on the $x$-axis while expressing the protagonist's fortune on the $y$-axis. Second, line sketches are *simple*. In Figure 2, the protagonist's fortune is expressed just in a single stroke of a line. Though we could build a similar time series through sliders, text entry of coordinates into a spreadsheet or clicking on points, a drawn line in a time-series is lightweight and intuitively understood. This design is well suited for continuous values (e.g., the fortune level). However, one could imagine 'sketching' discrete/categorical values as well by drawing disconnected horizontal lines. By default, a simple swipe can allow the author to indicate the attribute, for how long it should last, and where in the story it should exist.

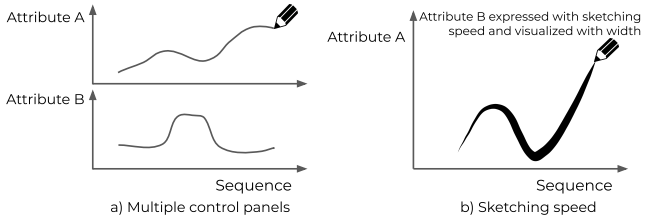


**Figure 3: More than one story attribute can be expressed with a) multiple control panels or b) the use of sketching speed.**

While **sequence** can be expressed in various ways, we choose to use the familiar linear $x$ position (with attribute values on the $y$). Linear positions have been most frequently used to visualize sequence and it can promote accurate perceptual judgment of time [1, 18]. Rotated (e.g., clock) or spiral-format temporal representations are also possible [149] but are best suited for representing cycles or seasonality in temporal data—something uncommon in stories.

In some cases, we may want to express **multiple story attributes**. For example, the user might want to specify two characters' fortunes. In this situation, the multiple time series can either be sketched within the same $x - y$ space (assuming the scales are the same) or drawn in multiple spaces (e.g., Figure 3a). To draw two attributes (along with time) in a single plot is possible but requires additional encodings. For example, a connected scatterplot (see [52]) draws the values for the two attributes on the $x$ and $y$ axes respectively with time encoded using marks on the lines themselves (e.g., arrows or graduated thickness) that indicates sequence. One could also imagine using a feature of the sketching interaction (e.g., pressure or the time it takes to draw a segment) to encode values. For example, in Figure 3b, time is displayed on the $x$-axis, the value for attribute $A$ on the $y$-axis and the time used to draw the stroke (or pressure) to encode the $B$ attribute (visually these can be encoded using stroke size or color). Drawing speed is naturally combined with stroked lines and allows for a semantic connection

between fast-slow drawing speeds, and high-low attribute values. Additionally, users may have ambiguous perceptions of speed and time [116]. This might further the idea that the generated story will only roughly follow the input. Ultimately, this encoding is likely inappropriate. It still limits us to two attributes and is likely complex and discordant as two similar variables (i.e., attributes $A$ and $B$) are created and visualized in two different ways.

A potentially better use of the drawing speed (or pen pressure) is to convey **ambiguity and error-tolerance**. In most cases, a generative algorithm will not be able to generate a sentence that precisely matches the attribute value. This is due to two factors. First, is the randomness in the generation (i.e., we may not be able to generate a sentence with a character's fortune at exactly 0.58). Second, is the inherent noise in any classifier that judges attribute values based on the text. By using 'sketchiness,' we are already conveying some level of ambiguity. However, it would be ideal to allow the end-user to be able to indicate how accurately they want the generated sentence to map to their sketch. To allow this specification, sketch speed or pen pressure may useful. Both actions have a natural mapping to 'dropping more ink' in natural pen interactions. The size of the sketched line at a particular time indicates the desired bounds–smaller, indicating more constrained. Notably, the specific semantics of slower motion can be switched. Drawing slower can be made to produce 'more ink' and therefore higher tolerance. Conversely, people often draw more slowly when they want an 'accurate' shape. Thus, a slower draw speed can semantically map to a more 'accurate' (i.e., tighter/thinner) line being drawn.

Though there is a fairly broad design space for sketched input, the combination of interactive controls, representation effectiveness, and flexibility seem best addressed through simpler representations: standard time-series for a single attribute and additional panels for multiple attributes. However, stroke speed or pressure, represented as line thickness or color, allows us to encode additional information such as error-tolerance.

# 4 TALEBRUSH: INTERFACE

In this section, we motivate our selection of controlled story attributes and explain interactions in TaleBrush's interface.

## 4.1 Iterative Co-creation With Line Sketching

TaleBrush is designed to help writers co-create a short storyline with AI through sketching interactions. TaleBrush supports the planning stage of the writing process [44, 48] by giving ideas that can be novel to writers. We chose to support this specific stage, as writers would likely accept and benefit from diverging generations. After a story is generated, writers can freely edit it or even try the generation again to see another story from TaleBrush. TaleBrush can support writers who would use generated sentences as writing prompts, or even potentially help them overcome writer's block. For novice writers, TaleBrush would demonstrate a story writing that would allow them to easily jump into writing.

In TaleBrush, writers iterate on the story generation to reach the desired state. Through our interactive controls, our goal is to allow writers to reach this state with a small number of iterations. As introduced in Section 3, sketching controls (green shaded area in Figure 1A3 and B2) allow for *simple* and *expressive* control interaction. TaleBrush visualizes generated stories in the same abstract visualization (blue line in Figure 1B2). Through this, writers can more easily understand the relation of their input to the generated content. We believe that the intuitiveness of these natural controls and visualizations will ultimately allow writers to more easily iterate on the story generation.

In our initial implementation, TaleBrush specifically focuses on two controllable attributes. The first is **the level of the protagonist's fortune**. This is based on the existing practice of writers expressing and planning stories based on character or emotional arcs [34, 150] or more generally on a character's fortune [142]. TaleBrush is designed to allow the control of the protagonist's fortune in chronological order. Non-linear narratives, such as flashbacks, are not currently supported. The user can also specify how tightly TaleBrush would follow these given fortune parameters at the cost of generation time. The second attribute is **the level of surprise** in the generation, or how unexpected the generation should be.

## 4.2 Interface

*4.2.1 Text Editor and Canvas.* TaleBrush has a text editor (Figure 1A2 and B1) and a canvas for drawing and visualizing the story arc (Figure 1A3 and B2). The text editor has multiple 'bullet points', each of which stands for a single sequence step, most often a sentence. The writer can add their own sequence sentences or ask TaleBrush to generate them. As with a standard editor, the writer can add new sequence sentences (simply by hitting enter), or remove them by deleting the bullet point line. TaleBrush will visualize each bullet point line as a dot in the canvas and sequential dots will be graphically connected with lines. Thus, there is a correspondence between bullet point order and $x$ position of those sentences in the visualization. Missing sentences (i.e., blank bullet points) are reflected with a visual gap. The protagonist's fortune in a given sentence is determined by an ML recognition algorithm.

*4.2.2 Generate By Sketching.* After the writer sketches a character's fortune (as described in Section 3), TaleBrush generates text for those sentences for "sketched" sentence slots. For example, the writer can enter one or two sentences, for which the fortune is calculated and displayed. They can then sketch the rest of the sequence. TaleBrush also supports 'infilling' the story. For example, if the writer enters a sentence at the start and end of the sentence list, this will appear as a broken line in the visualization. The writer can sketch how they want the story to behave in this empty part and TaleBrush will create suitable sentences. The writer can also draw 'broken' lines by erasing part of the sketched line. In this mode, TaleBrush will generate a suitable sentence but without any specific fortune 'target.' This allows writers to explore stories even when they do not have a specific idea on what the character's fortune should do. Once the sketch is drawn, the writer can perform generation again with the same parameters by clicking the "Generate Again" button (Figure 1B3). They can also redraw a line only on a part of the sequence to see other generated sentences on the redrawn part. In all cases, the writer also can revise generated sentences directly in the text editor.

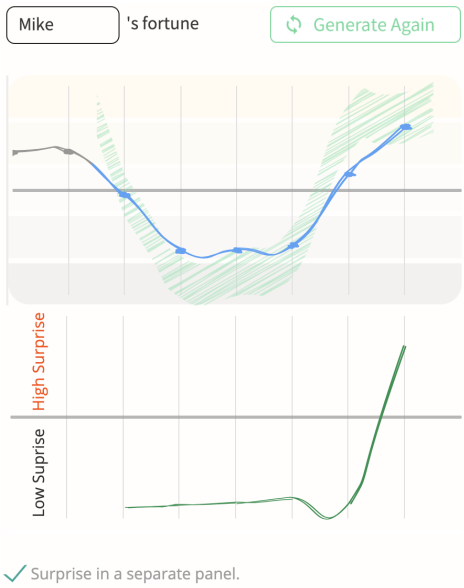


**Figure 4: TaleBrush allows writers to specify the level of 'surprise' for each generated sentence. A separate canvas can be opened where the writer can sketch a thin green line indicating the desired level of surprise. Here, we see a fortune arc for Mike (top) starting high, going low, and ending high. A corresponding surprise sketch (bottom) indicates that the author wanted the first part of the story to be more controlled but the end should be very surprising.**

*Sketching Speed for Generation Control Tightness.* As the controlled generation has uncertainty in how the control would apply, with the sketching speed, the writer can set how closely generated sentences should follow the given sketch. With slow sketching, TaleBrush tries more generations to find the sentence that better matches the given fortune parameter. In our experience, we found that the *fast → less accurate* and *slow → more accurate* mappings were easier to understand. The drawn sketch can be perceived as an 'envelope' representing the bounds in which the generated sentences would be more likely to fall (Figure 1A3 and B2). We detail how the specific width is determined in Section 6. The width of the sketched line gradually changes visually (i.e., it is not disjoint). This is more appealing visually. However, the actual envelope is determined using sentence 'units.' Subtle vertical lines indicated where these transitions are, thus allowing the user to vary their drawing speed in each segment.

*Surprise Level Control.* As we describe below, the specific language model we used also allowed us to change parameters that roughly translated to how 'surprising' a sentence would be in the following prior sentences. For example, with the beginning sentence of "Melissa fought a dragon", the low surprise may generate "The dragon was a big, green, scaly beast", but the high surprise may result in "Melissa killed a giant robot with a robot dragon". To experiment with this feature, we added a "surprise in a separate panel" control. This would open up another time series canvas

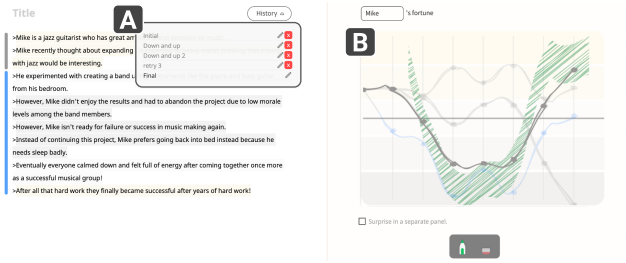


**Figure 5: TaleBrush stores generated stories in a list (A) and allows writers to compare them on the canvas by visualizing the fortune arcs of all stored stories in low opacity (B). The writer can hover over stored stories in the list to see their text and story arc visualization.**

where the writer could draw the desired level of surprise for each sentence segment (see Figure 4).

*4.2.3 Multi-Story Management.* As the writer generates multiple stories, TaleBrush stores each. Users can compare and choose among them (Figure 5). A dropdown menu lists the generated stories, which will be displayed on the canvas with low opacity. As the writer moves the mouse over the list, the hovered story is shown in the text box and highlighted on the canvas. The writer can select one of these to 'roll back' to a past generation.

## 4.3 Implementation

The interface for TaleBrush is implemented as a web application, using HTML, CSS, JavaScript, and React. The language model operations for text generation are implemented in a back-end server. A Flask-based REST API is used to connect the front– and back–end sub-systems.

# 5 TALEBRUSH: TECHNICAL DETAILS

There are two principle technical components to TaleBrush: sentence **recognition**, which determines a fortune level given an input sentence; and sentence **generation**, which creates new sentences. We begin by explaining our annotated training data, and then cover each of recognition and generation modules.

## 5.1 Collecting Fortune of the Protagonist

We adopted crowdsourcing to create an annotated fortune dataset, as there is no readily available data of this type. We define the protagonist's fortune as *level of 'goodness' or 'badness' of the fortune experienced by the protagonist, as perceived by the readers.* This construct is naturally affected by the character's status, emotional and physical well-being, or spatial, mental, and emotional proximity to the character's goal [34]. While line sketching interaction would require fortune annotations to be real values, due to the subjectivity in perceiving this concept, directly collecting real-valued annotations can result in a high variance. As people can make comparisons reliably on subjective annotations [33, 68], we took the approach similar to previous work's [33]: turning crowdsourced comparisons into real-valued annotations.

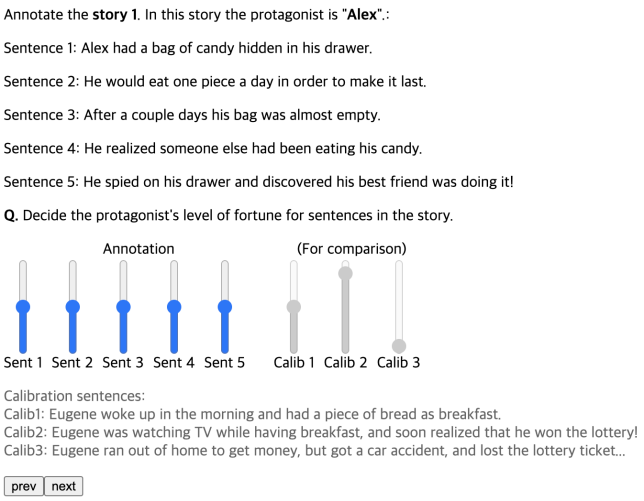


**Figure 6: Interface for annotating the protagonist's fortune level. The annotator can make comparisons between sentences in the given story. Calibration sentences help for making comparisons between stories.**

We collected fortune annotations on a subset of the ROCStories dataset [105]. Items in this dataset are short five-sentence stories. We randomly sampled 2200 stories, using 2000 for training and 200 as a test set. We assigned three annotators for each story. During data collection, we explicitly specified the protagonist of the story. Annotators were asked to specify the level of the protagonist's fortune for each sentence using a continuous slider (Figure 6). While this annotation approach is mainly designed to collect real-valued annotations, the interface also enables annotators to encode comparisons between sentences. Workers were not explicitly asked to compare sentences between different stories. However, they were asked to annotate using the "calibrating sentences" as benchmarks. One of the authors crafted these calibrating sentences from a three-sentence story to ensure a unique, single-sentence example for three fortune 'types': good, bad, and neutral. Our annotators were asked to label these calibrating sentences at the start of the task and could see them as they worked (see Figure 6).

To ensure high annotation quality, we included one gold standard question with sentences of obvious good and bad fortunes. We filtered out workers who annotated such questions in opposite fortune directions. We recruited annotators from Amazon Mechanical Turk, who are in the US and had 97% acceptance rate with 1000 or more accepted tasks. Each worker was asked to annotate 11 stories including the calibrating story and was paid $2.50 (about $10/hr payment rate).

To learn the reliability of annotations, we analyzed agreements between pairs of annotators. We measured Spearman's $\rho$ in ranking five sentences in a story according to the fortune level (i.e., with perfect agreement $\rho = 1$ or $-1$ with perfect disagreement). The average and median of $\rho$ values between pairs were 0.54 and 0.7, respectively. Though clearly imperfect, annotators did display agreement in their annotations.

To turn comparisons into real-valued labels, we used the TrueSkill algorithm [33, 56, 102]. The algorithm can calculate the real-valued fortune score of each sentence out of all comparisons from our crowdsourced annotations. While this algorithm is originally designed to infer rankings of game players out of all individual game results, like who won or lost, we adapted it to get rankings of fortunes out of all comparison annotations. With this algorithm, sentences frequently annotated to have a higher fortune than other sentences would have a high fortune ranking. We normalized this ranking information to have a uniformly distributed dataset on the scale of 0 to 1 and used the resulting annotations.

## 5.2 Technical Architecture

A high level architecture for the recognition and generation elements of TaleBrush is depicted in Figure 7.

*5.2.1 Recognition Module.* TaleBrush recognizes the protagonist's fortune to map sentences in the visualization. To match the training dataset for the recognition and controlled generation, we trained our own recognition model. Specifically, we use prompting approaches of GPT-based models [21, 119]. For instance, to recognize the protagonist's fortune, to the model, we can input the prompt "Estimate fortune:" with the subject sentence appended to it. However, hand-designing optimal prompts is known to be challenging [87, 93]. Hence, we used the soft prompt approach [87, 88, 93], which learns these prompts on the continuous input embedding space. These soft prompts replace "discrete" natural language prompts. In other words, we can replace "Estimate fortune:" with several learnable soft prompt embeddings. We took the soft prompt tuning approach [87] instead of fine-tuning the whole parameters of the LM, as trained soft prompts require less memory (about $\times 10^5$ less size than the whole parameters). Fine-tuning multiple whole-parameter models for multiple tasks (e.g., generation), would require significantly more computation resources.

We trained the soft prompts to minimize the regression loss on the protagonist's fortune label of an estimated sentence. As a language model, we used a GPT-Neo [13], which is a GPT-based auto-regressive model [21, 119] with 2.7 billion parameters. We included the context sentences previous to the target sentence in the input data, as they can have critical information. This previous context includes, at maximum, three sentences. We picked three as it had the smallest loss in the test dataset. We also included the protagonist's name in the context. The input and output labels are formatted as follows:

- Input: $[P^1_{prev}]$, ..., $[P^n_{prev}]$, (previous context), $[P^1_{prot}]$, ..., $[P^n_{prot}]$, (protagonist), $[P^1_{tar}]$, ..., $[P^n_{tar}]$, (target sentence)
- Label: $y \in [0, 1]$

$[P^i_r]$ stands for the placeholder for soft prompt embeddings, with $r$ and $i$ indicating the role and the index of soft prompts, respectively. For each prompt role, we chose three prompt tokens ($n = 3$) which fit with that of initialization prompts we used ('Previous Context:' for $P_{prev}$, 'Protagonist:' for $P_{prot}$, and 'Sentence:' for $P_{tar}$). With trained prompts, we could recognize the protagonist's fortune, as demonstrated on *Cinderella* in Figure 8.

*5.2.2 Generation Module.* Our goal is for TaleBrush's generation to be steerable with a sequence of fortune values. For this, we modified

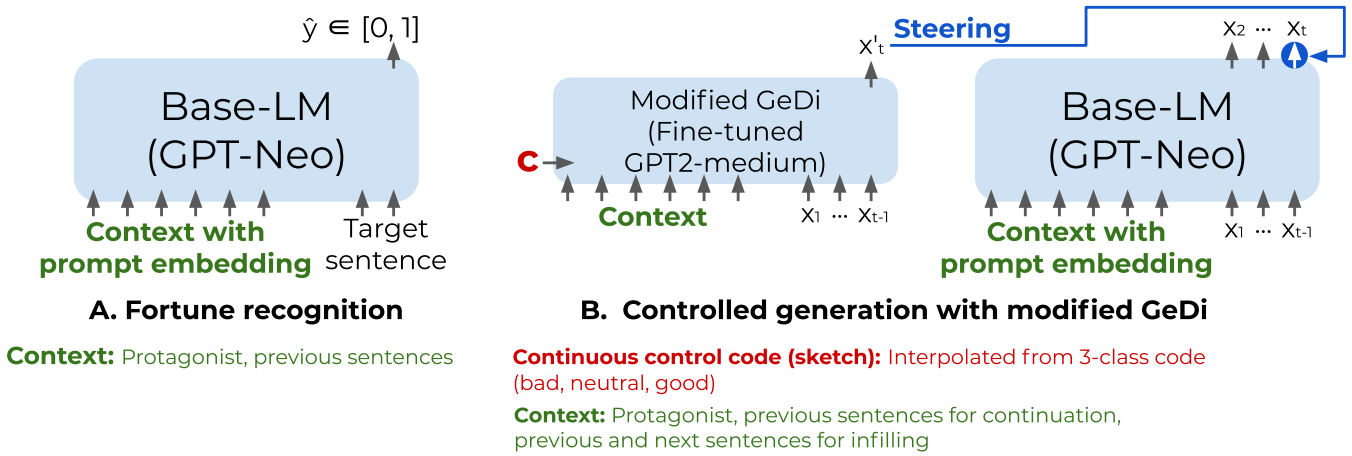


**Figure 7: The technical architecture of TaleBrush. To serve recognition and generation, the prompts of a Base-LM are tuned. These prompts are used interchangeably for a single LM. For fortune recognition, context, prompt embeddings, and target sentence are given as inputs and the fortune level is predicted as a real value between 0 and 1. For controlled generation with the fortune sketch, a modified version of GeDi is used. GeDi is a fine-tuned model that steers the generation according to real-valued fortune input, or continuous control code. Context and prompt embeddings are also used as input. For a sentence to be generated, the $y$-position of a point in a sketch is used as a continuous control code. This continuous control code has a value between 0 and 1 and its input embedding is interpolated with three class codes (bad for 0, neutral for 0.5, and good for 1) before being input into the modified GeDi. Base-LM used GPT-Neo while the modified GeDi used GPT2-medium.**

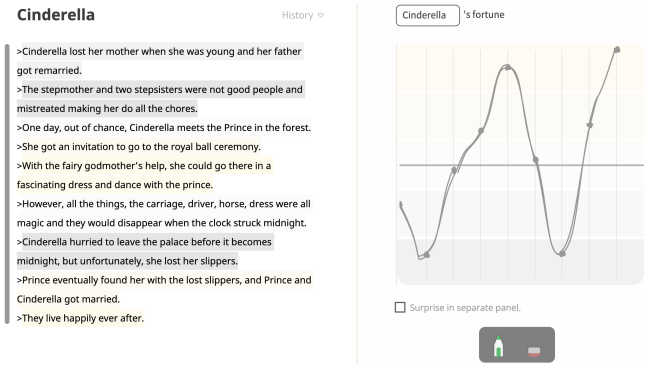


**Figure 8: How the protagonist's fortune is recognized in the summary of *Cinderella*. The visualized recognition results are similar to Vonnegut's original (Figure 2).**

the approach used in GeDi [83], which uses a fine-tuned smaller model to guide a bigger language model with a control input. In order to explain the details of our GeDi variant, we first describe how generation can be done with LMs. Then, we describe our base LM that serves story generation tasks and GeDi model that steers stories with the control code.

*Generation with Language Model.* When an auto-regressive LM receives a sequence of tokens ($x_{1:T} = \{x_1, \ldots, x_T\}$), it calculates the probabilities for the next tokens with a chain rule ($P_\theta(x_{1:T}) = \prod_{t=1}^{T} P_\theta(x_t|x_{<t})$). With this probability, we can sample out the next likely token. By inputting the sampled token back into the LM, we can also calculate the probability for the token that comes after the lastly sampled token.

*Base Language Model for Story Generation.* As a base LM for language generation, TaleBrush uses the same GPT-Neo the recognition module uses. However, different soft prompt embeddings are trained for the generation tasks. The benefit is that one large pre-trained LM can be utilized for different tasks just by replacing prompts and contexts according to the task purpose. We trained soft prompts for story continuation and infilling. For continuation (Figure 9a), the previous context and protagonist are fed into the language model as input and the model generates the continuing sentences. We appended the character information as models without character information tend to introduce random characters. We placed the character information right before the text to be generated, to make it more likely that character information is considered during the continuing generation. Accordingly, the input data is formulated as the following:

- Continuation: $[P^1_{prev}]$, ..., $[P^n_{prev}]$, (previous context), $[P^1_{prot}]$, ..., $[P^n_{prot}]$, (protagonist), $[P^1_{cont}]$, ..., $[P^n_{cont}]$

Similar to recognition prompts, we chose the number of prompts that matches the initializing prompt ($n = 3$ with 'Beginning Story:' for $P_{prev}$, 'Story Character:' for $P_{prot}$, and 'Continue Story:'for $P_{cont}$).

For infilling (Figure 9b and c), we included both the previous and next contexts and the protagonist as the model input. We trained two types of prompts for infilling: one for filling after the previous context, and the other for filling backward from the next context. When running infilling generation, these two prompts are used in turn, switching from one to the other for every generated infilling sentence. For infilling from the previous, we placed the previous

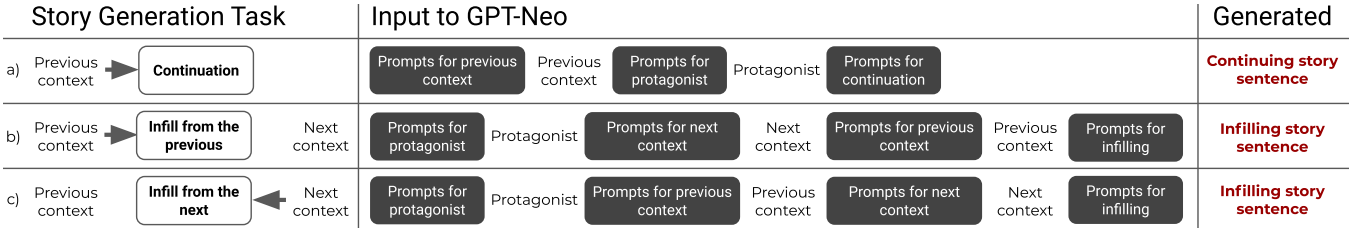


**Figure 9: TaleBrush performs three story generation tasks: a) continuation, b) infill from previous, and c) infill from next. For different generation tasks, how contexts and prompts are structured for inputs differs. In *Input to GPT-Neo*, dark boxes indicate continuous prompts. In *Generated*, red text stand for generated sentences.**

context right before the text to be generated, as the generated text will be appended right next to the previous context. With a similar rationale, for infilling from next, we placed the next context right before the text to be generated. We placed protagonist information at the front-most prompt, as infilling tend to digress less with the protagonist compared to continuation. This likely occurs as the protagonist is mentioned in both the 'previous' and 'next' contexts. The input data for infilling was formulated as follows:

- Infilling from the previous: $[P^1_{prot}]$, ..., $[P^n_{prot}]$, (protagonist), $[P^1_{next}]$, ..., $[P^n_{next}]$, (next context), $[P^1_{prev}]$, ..., $[P^n_{prev}]$, (previous context) $[P^1_{infill}]$, ..., $[P^n_{infill}]$
- Infilling from the next: $[P^1_{prot}]$, ..., $[P^n_{prot}]$, (protagonist), $[P^1_{prev}]$, ..., $[P^n_{prev}]$, (previous context), $[P^1_{next}]$, ..., $[P^n_{next}]$, (next context) $[P^1_{infill}]$, ..., $[P^n_{infill}]$

We chose the number of prompts according to that of the initializing prompt. ($n = 3$ with 'Story Character:' for $P_{prot}$, 'Beginning Story:' for $P_{prev}$, and 'Ending Story:' for $P_{next}$, and $n = 4$ with 'Infilled Story:' for $P_{infill}$).

*Controlling Generation with GeDi.* TaleBrush adopts the GeDi approach. This model guides an LM model (base LM) with a smaller Class-conditional LM (CC-LM) that is fine-tuned to generate using a user-specified control code (e.g., generate a happy story with the control code of "good"). Similar to standard LMs, CC-LM calculates the probability of the next token given a sequence of tokens. However, it also considers a specific *control code* ($c$) given to the CC-LM ($P_\theta(x_{1:T}|c) = \prod_{t=1}^{T} P_\theta(x_t|x_{<t,c})$). With CC-LM's probabilities for different codes and the Bayes rule, we also can calculate the probability of each sequence of tokens being classified into a control code:

$$P_\theta(c|x_{1:t}) = \frac{P(c)P_\theta(x_{1:t}|c)^{\alpha/t}}{\Sigma_{c' \in \{c,\bar{c}\}} P(c')P_\theta(x_{1:t}|c')^{\alpha/t}}, \quad (1)$$

where $\alpha$ is a learnable scale parameter and $\bar{c}$ is codes other than the selected one. This probability can be combined with that of base LMs, to decide which token would lead to the coherent story that follows the control code:

$$P_\omega(x_t|x_{<t}, c) \propto P_{LM}(x_t|x_{<t})P_\theta(c|x_t, x_{<t})^\omega \quad (2)$$

Note that $\omega$ serves as a parameter that decides how much steering will be done with GeDi (higher → more steering). GeDi is faster than other control approaches that directly manipulate weights [31]

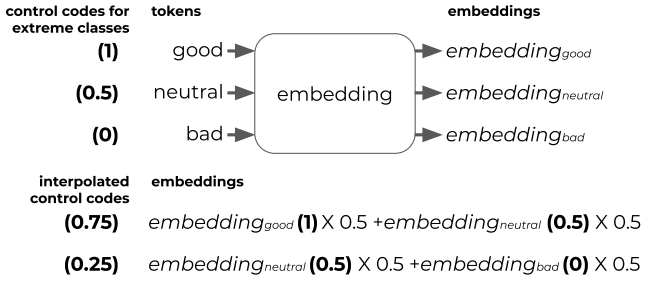


**Figure 10: An approach to turn continuous control codes into embeddings that can be input to CC-LM. For three extreme classes (1, 0.5, 0), we use embeddings of the tokens, *good*, *neutral*, and *bad*, respectively. For other control codes, we interpolated the embeddings of these extreme classes.**

or that filter from many generated texts. Thus it is better suited for interactive applications.

*Modified GeDi for Continuous-Value Control.* We modified GeDi to control generation with continuous-valued control codes. To enable continuous control between good and bad fortunes, we interpolated between extreme fortune classes. We first trained CC-LM with three topic classes, each mapping to extremely *bad* fortune, *neutral* fortune, and extremely *good* fortune. When training the CC-LM with these three classes, we took the multi-class topic control approach introduced in GeDi [83]. With this approach, the given "main" codes are *true* and *false*, while three classes of *bad*, *neutral*, and *good* are given as the secondary codes. When given *true*, CC-LM trains to generate texts according to the given secondary code, and with *false*, CC-LM would not follow the secondary code. We adopted this multi-class topic control approach as it has benefit in computation time, with fewer "main" control codes to be computed to get the probability from Equation 1. We also included the protagonist's name at the beginning of the input as contexts. A part of training dataset might look like:

- <true> <good> Protagonist: Mark # Mark was happy to pass the exam that he wanted to pass.
- <false> <bad> Protagonist: Mark # Mark was happy to pass the exam that he wanted to pass.

For the training data, we used only some of the annotations from our collection: those in the top, middle, and bottom 10% percentile. That is, we mapped ranges of values to three discrete classes to

assure we have enough training data. To train this CC-LM, we fine-tuned the GPT2-medium model. After the CC-LM is trained, when an arbitrary value between 0 and 1 is used as a code, we interpolate embeddings between three discrete classes considering each of them as having 0, 0.5, and 1 (Figure 10). Then, we used the interpolated embedding as the input embedding. For example, when the control code is 0.25 (halfway between the bad and neutral fortunes), we calcualted the weighted sum of the the learned *bad* and *neutral* embeddings with the weight of 0.5 each.

While GeDi steers the generation with the protagonist's fortune, it can potentially make errors, producing sentences outside of the desired fortune level. To handle these errors, we adopted the approach of regenerating sentences if they were significantly out of the range of the specified code. Specifically, TaleBrush's recognition module measures the generated sentence's level of fortune, and if it disagrees with the control value, TaleBrush tries to generate again. We set the threshold of error for regeneration as 0.2 (i.e., $|y - \hat{y}| > 0.2$). TaleBrush tries to regenerate the text 1-2 more times. Again, this is based on the desired 'closeness' of the generated fortune to the writer's intent (as expressed by the speed of the writer's drawn stroke). After the maximum number of regeneration has been tried, TaleBrush picks the generated sentence that has minimal fortune level error with the given control value. The surprise level control value is used as the temperature of the softmax function of the base language model. To implement our architecture, we used models from Huggingface[1][2].

# 6 TECHNICAL EVALUATION

We conducted technical evaluations on the recognition and generation modules. For recognition, we measured how accurately a character's fortune is recognized in a sentence. For the generation module, we focused on how controllability impacts other qualities of the story. Controllability is known to have trade-offs with some story qualities like coherence [90]. Ideally, our control approach would allow the writer to steer the generation without hurting other metrics. We conducted three evaluations: 1) recognition performance, 2) an automated evaluation on generation, to find the parameter that can achieve the 'sweet spot' between controllability and coherence, and 3) human evaluation on generation, to show that our approach allows controllability while maintaining other story qualities.

## 6.1 Fortune Recognition Evaluation

To evaluate the recognition module, we focused on recognition error, the difference between the recognized fortune value and the gold standard labels. This was done on the 200 annotated examples reserved for testing. The mean and median errors were 0.179 and 0.159 respectively on a 0 to 1 scale. We considered this result to be reliable. Concepts similar to fortune, such as sentiment, are usually measured through an ordinal five-level scale. Human raters can easily discern concepts with this number of levels [133]. If we view the five levels on a 0 to 1 scale, with the assumption of the uniform interval between levels (which is widely adopted in ML [133, 156]), then the gap between levels will be 0.25. Our average error (.179) is below this 0.25 gap so most differences would not be easily discernible.

## 6.2 Automated Generation Evaluation

We conducted the automated evaluation to identify the parameter that balances coherence and controllability. Here, the parameter of interest is $\omega$: how intensely the steering control is applied. Past work has demonstrated that a higher $\omega$ value leads to better controllability, but worse coherence [90]. As we modified the GeDi approach to our context, we reinvestigated how $\omega$ impacts coherence and controllability.

For coherence, we calculated perplexity, which is a measure of how well a probability model predicts to generate a sequence of text tokens [20]. To evaluate the controllability, we measured the control difference: the difference between an input fortune value and the estimated fortune level of the text generated. To estimate the fortune level of generated text, we used our fortune recognition module. We measured these two metrics with our 200 test data points. For the continuation task, we considered the first sentence as the initial context and generated four more sentences. For infilling, we considered the first and the last sentences as the initial context and generated middle three more sentences. For the controlled generation, we gave fortune annotations in our test data as control values and measured perplexity and control difference, while varying $\omega$ value. Recall that we allow for 'regenerating' the text up to two times to improve the generated text's fit to the desired fortune (selecting the best of the generated options). We also simulated this regeneration up to two times to determine the impact of this on our metrics.

With controlled continuation (Figure 11a), the perplexity tends to increase with the increase of $\omega$. The increases in median and $25^{th}$ percentile were relatively stable compared to that of $75^{th}$ percentile. The control difference tended to decrease very slightly with the increase of $\omega$. The regeneration approach tends to decrease not only the median of control differences but also the $75^{th}$ percentile of control difference by a large amount. Regenerating once (two stories to pick from) resulted in a 30%-40% increase in generation time, while regenerating twice (three stories) resulted in a 50%-60% increase. The rate of performing regeneration decreased very slightly with the increase of $\omega$.

With infilling, the change of $\omega$ did not change perplexity much (Figure 11b). This might be because infilling considers both beginning and end contexts, and would diverge less from the gold standard sentences. On the other hand, for continuation, as there is no end context to consider, the generated text can diverge significantly from the gold standard sentences. The control difference and regeneration rate did not have a discernible trend with changes to $\omega$. Without regeneration, the control difference of infilling was larger than that of continuation. Similar to continuation generation, the addition of regeneration leads to decreased control difference. The rate of regeneration was 40%-50% when regenerated once and 50%-65% when regenerated twice, which was higher than those of the continuation.

From the results, we find that a small $\omega$ value did not hurt controllability much. Moreover, a low $\omega$ value assured low perplexity

[1]https://huggingface.co/
[2]Code for TaleBrush is shared in https://johnr0.github.io/publications/TaleBrush_CHI2022.

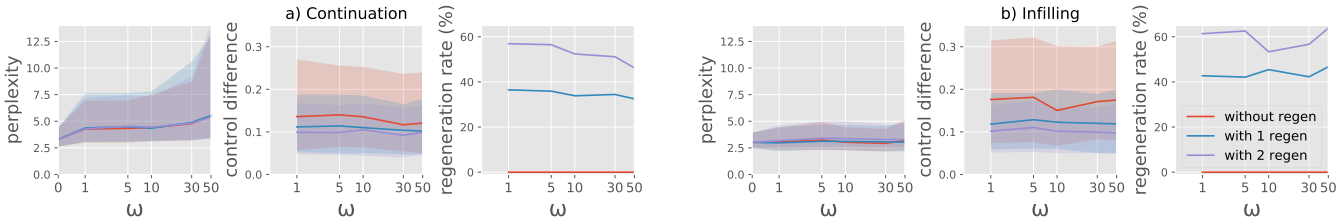


**Figure 11: Automated evaluation results on controlled story a) continuation and b) infilling. Median of perplexity and control difference (between given control value and output fortune) are plotted for the range between $25^{th}$ and $75^{th}$ percentile. In the regeneration condition, we measured the rate of sentences that were regenerated to minimize control difference. The $x$-axis is presented in log-scale.**

in continuation tasks. If regenerating once, the control difference decrease at a cost of 30%-50% more generation time. Considering the findings, in TaleBrush, we use $\omega$ value of 1 and try regeneration once as a baseline. Recall that we still limit regeneration to at most twice based on the writer's input. While additional regeneration may improve the results, the response time costs may be too high for an interactive application with the current architecture.

## 6.3 Human Evaluation on Generation

In addition to the automated test, we conducted a human evaluation to see if stories generated by TaleBrush are perceived as coherent, novel, and controlled. For controllability, we showed evaluators two sentences generated with two different fortune levels and asked them to decide in which sentence the protagonist has a better fortune. Evaluators were not shown the input fortune values. If an evaluator chose the sentence generated with a higher fortune control value, we considered the answer correct. All fortune values in the human evaluation were randomly sampled. However, in the controllability study, we excluded control values that were too similar (a difference smaller than 0.25), as such similar controls would be even difficult for humans to differentiate. We decided this threshold as 0.25 as fortune level differences larger than this value would be discernible to evaluators (i.e., five-level Likert scales are frequently used for sentiment annotation). We additionally investigated the novelty of generated texts as TaleBrush is designed to aid story ideation. We showed evaluators two versions of generated stories: one 'with' controls (i.e., using the GeDi model and character fortune values); and the other without controls (only with the prompt-tuned GPT-Neo model). We asked the evaluators to decide which story was more coherent or novel. For coherence, we specifically asked: 1) if story flow is coherent (story coherence) and 2) if sentences are grammatically coherent (grammaticality). While there can be more specific incoherent patterns [50], we chose story coherence and grammaticality as they each can tell us about the global and local coherence of the story. Evaluators could choose one of the generated stories with better coherence or novelty (CX for "without controls" and CO for "with controls" in Figure 12), or decide that they are similar in quality (S in Figure 12). Evaluators were not aware of which story is generated with fortune controls. For each criterion, we evaluated 100 story pairs, 50 for continuation and the other 50 for infilling. When generating these stories, we applied the regeneration strategy once and used $\omega = 1$. We hired crowd workers as evaluators, as we wanted to know if even non-experts in story writing perceive TaleBrush's generations as coherent, controlled, and novel. For each assessment question, we assigned three crowd workers to assure reliable evaluation. We recruited crowd workers from Amazon Mechanical Turk, who are in the US, with acceptance rates higher than 97% and more than 1000 tasks accepted. Each worker evaluated one criterion, for eleven pairs of stories including one gold standard question. We paid them $3, which is over a $10/hr payment rate.

Figure 12 summarizes our results. Evaluators recognized which sentence is generated with a higher fortune control value 88.0% of time for continuation (84.7% for infilling). Stories generated without fortune controls (CX) were chosen most frequently to have better story coherence and grammaticality. In continuation, their ratio was below 50% (45.3% and 46.0% for story coherence and grammaticality, respectively). That is, when using continuation with fortune controls, in more than half of cases, users would observe stories with similar (S) or better (CO) story coherence and grammaticality compared to those generated without controls. With infilling algorithms, similar to continuation, stories generated without controls (CX) showed better story coherence and grammaticality most frequently, but their frequencies were higher than continuation (62.7% and 76.7% for story coherence and grammaticality, respectively). One potential reason for this result might be the fundamental difficulty in generating coherent infilling stories when a given fortune sequence is not plausible (e.g., drastic fluctuation in fortune). For novelty, with continuation algorithm, different conditions showed similar performances. With the infilling algorithm, stories generated with controls (CO) are perceived to be more novel more frequently. In summary, we found that novelty is not hurt with fortune controls. We also find that our control approach can steer the generation according to the control value, but at some cost to story coherence and grammaticality. The cost is lower in the continuation algorithm than in infilling one. Regardless of the strategy, no generative algorithm can generate perfect coherence or grammaticality. Thus, it is critical to ensure that the writer can edit the final text.

# 7 USER STUDY

We conducted a user study to gain insights and feedback on the potential, limitations, and future opportunities of sketching interactions for iterative human-AI story co-creation. We focused on

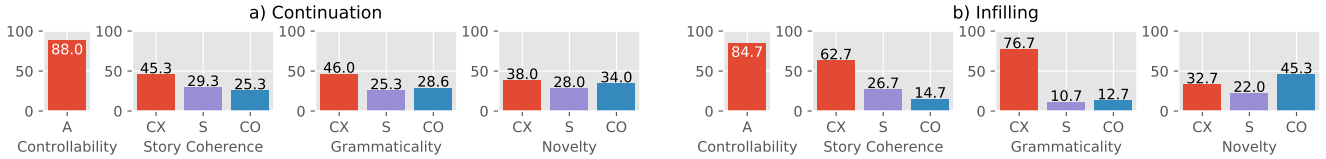


**Figure 12: Human technical evaluation results on a) continuing and b) infilling generation. For controllability, the accuracy (A) in estimating which sentence is generated with a higher control value is reported. For coherence, grammaticality, and novelty, the ratio of evaluators choosing which condition has better quality is reported (CX for generation without controls, S for quality being similar, and CO for generation with controls).**

learning how our novel sketching control facilitates iterative exploration of story ideas and story co-creation. As this study seeks to understand how users leveraged our novel interactions, we did not conduct a formal comparison to existing human-AI story co-creation tools.

## 7.1 Participants

We recruited 14 participants (7 female and 7 male) by word-of-mouth and through online advertisements on social media and communities of universities. Participants completed a pre-survey on their background before the study. Our participants had a range of expertise: novices, hobbyists, and experts. We classified participants who do not write stories occasionally as novices. Others were classified as either hobbyists or experts, and experts considered themselves as story-writing professionals. Participants were fluent in using English. We detailed participants in Table 1.

## 7.2 Procedure

We conducted a remote study with Zoom[3]. Participants were first given an overview of the study (15 min), then went through a tutorial on the tool (15 min). Participants could access TaleBrush directly through a URL we shared. In the tutorial, we explained that better fortune was represented with higher points on the $y$-axis. However, we allowed participants to use the tool to specifically learn how the mapping worked as there are no explicit axis labels. After the tutorial, we asked participants to use TaleBrush in two tasks. First, they were given a specific story beginning and were asked to generate a continuing story with TaleBrush until they found one that appealed as a draft story. After participants picked this draft story, they were asked to edit the text directly without additional generation (20 min). This task allowed us to observe how participants drew fortune arcs and which of the generated elements they retained or modified. For the second task, we asked participants to freely use the tool, bringing in their own story from the very beginning (20 min). We did not restrict how and when they used generation. Whereas the first task roughly corresponded to one 'turn' (machine then human), the second allowed us to observe an iterative co-creation process. In both tasks, we asked participants to share the screen of the interface. We also asked participants to think aloud while using TaleBrush to learn their rationales and reactions in using the tool. Lastly, we conducted a short interview, asking the experience of using TaleBrush, such as how they used the line sketching, how they incorporated stories generated by the tool, and how they would adopt TaleBrush to their practice (20 min). The whole session was video-recorded.

[3]https://zoom.us/

## 7.3 Results

We qualitatively analyzed screen recordings, think-aloud statements, and interviews. One of the authors analyzed data by iterative coding with inductive analysis, and coded results were reviewed with two other authors. Note that unless explicitly stated, results are about the participants across all expertise levels.

### 7.3.1 *Use of Line Sketches.*

*Line sketching facilitates iterative co-creation with intuitive controls.* Overall, participants indicated that line sketching was a simple and expressive way to specify the protagonist's fortune. As critically, they understood how to use sketching to steer text generation. Participants also thought that fortune was an effective attribute to control the overall story flow. During the study, writers experimented and iterated on multiple line sketches. Some tried generation on a sub-part of the story to only change that part. Others tried generation in the middle of editing the story. For example, P6 repetitively tried generation to find infilling sentences that would go along well with the ending sentence previously generated from TaleBrush. During the process, P6 also edited generated sentences and used them as the context for the next generation. Participants also mentioned that sketching allowed them to specify the arc without the pressure of being precise. For example, P6 mentioned: *"If I should have input each number, I might have been more reluctant to complete the story. Because they are specifying more detailed values. So, I think drawing would be more helpful in generating the stories."*

*Surprise control and controllability in small changes could be improved.* A few participants used the surprise control, but the perceived effectiveness diverged between participants. Additionally, when participants used sketching for small and detailed fortune changes, they felt that their control was not reflected well on the generation. The feature of specifying control fidelity with sketching speed might have been one approach to solve this problem, but it was not effective enough in some cases. Participants speculated that the feeling of lack of control might be due to disagreement between their perception of fortune and the fortune level calculated by TaleBrush. The current generation approach of replacing the whole sentence may have also exacerbated this problem, as the user expected a slight change in the text when they are making

**Table 1: Participants of the user study. Their domain of interest and years of experience in story writing is shown.**

| Expertise | | Domain | Year | Expertise | | Domain | Year |
|---|---|---|---|---|---|---|---|
| Novice | P1-5 | N/A | N/A | Hobbyist | P10 | Fantasy | 2 |
| Hobbyist | P6 | Play script | 2 | Expert | P11 | Sci-fi feature films | 50 |
| | P7 | Time travel story | 3 | | P12 | Literary/Psychological fiction | 15 |
| | P8 | Sports story | 0.5 | | P13 | Fiction, Fantasy | 4 |
| | P9 | Hard SF, Dungeon & Dragon | 3 | | P14 | User/Personal/Fictional stories | 8 |

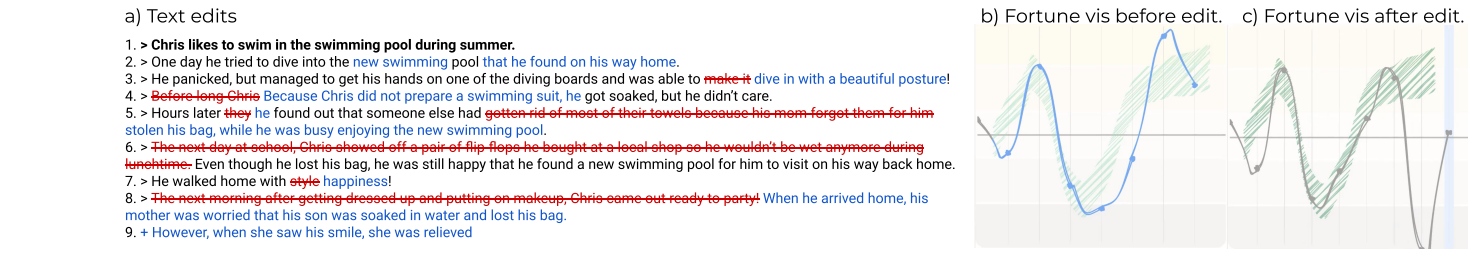


**Figure 13: P2's work: a) Edit of a generated story. Bolded, red, and blue text indicates initial context input, removed parts, and added texts, respectively. b) Line sketch and fortune arc of the generated story before editing. c) Fortune arc of the edited story.**

small changes in the sketch. While the new sentence may be at the appropriate fortune level, the content may have been too radically different from the sentence at that time point.

*Line sketching facilitated the planning of writing to some participants.* Within the iterative co-creation process, the line sketch could also influence participants to edit the story to follow the sketched fortune arc. For example, in Figure 13, P2 edited generated text to follow the line sketch that P2 drew in most sentences (from b to c). Because the current story was constantly compared to the drawn sketch, the participant would try to adjust the story so that it was more aligned with the sketch. Participants also mentioned that the sketch served as a plan or a guideline. For example, P7 mentioned: *"As I draw graphs, it is intuitive, I could easily see how I would compose the storyline and how I should structure it."* However, some participants who deeply focused on the story itself did not consider the sketch when editing the text.

#### 7.3.2 *Use of Generated Texts.*

*Generated texts are used as ideation materials.* From the automatically generated stories, participants adopted characters, settings, short expressions, sentences, events, or even the overall story flow. For example, in Figure 13a, P2 liked the event described in the third sentence and edited the story to go along with the sentence. Novice participants mentioned that generated texts lowered the barrier on starting writing, as they would have more frustration if they start from a blank paper.

*Incoherent generations are targets for revision.* Unsurprisingly, participants frequently revised incoherent parts. For example, the sixth sentence of Figure 13a was taken off because mentioning flip-flops was perceived as a drastic context change. Participants also added more details to the story to make it more reasonable and concrete. For example, in Figure 13a, P2 added details on the 'pool' in the second sentence that it is a new swimming pool on the way to the protagonist's home. Participants also added new sentences to make the story flow natural. For instance, in Figure 13a, P2 added the last sentence to make the ending more natural. Interestingly, the editing pattern of novices, hobbyists, and experts did not differ much, potentially due to the novelty of the tool's functionality to participants.

*Incoherent generations serve as the reason to exert creativity for some participants.* While incoherent or ambiguous story elements require participants to revise sentences, some participants appreciated this as a 'prompt' to apply their own creativity and revise the text. For example, P1 mentioned: *"If, there is nothing like that [incoherence], I would not do something creative. However, with something that needs to be fixed, and if you kind of get that they need to be revised, I feel like I can exert my creativity... So I don't think these are downsides."* Moreover, some 'incoherence' was perceived as novel elements to participants. This 'bug' thus becomes a 'feature' in the context of ideation support.

#### 7.3.3 *Potential of TaleBrush in practice and suggestions for improvements.*

*Potential for quick and iterative ideation.* Experts and hobbyists mentioned that they would use TaleBrush to quickly iterate through diverse ideas. Collected ideas can be used in the ideation stage or to overcome writer's block. This benefit would be maximized in settings where the writer should bring up multiple stories, such as tabletop role-playing games.

*Supporting more attributes, longer texts, and various types of writings.* Participants mentioned that they would like to see controls for more story attributes, such as other characters' fortunes, settings, or genres. They reasoned that more controls would likely lead to a more desired text that they would adopt to theirs. Some participants indicated that the visualization and control granularity might need to change if TaleBrush expands to generate longer

texts, as users would want to give paragraph-level or chapter-level specifications. Experts noted that TaleBrush needs to be adapted to each usage context. For example, P11, a screenwriter, stated that the tool should be able to express character emotions or states only with observational statements, as those need to be shown, not told, in screen writings. Finally, participants mentioned the potential of using sketching to control generation of texts other than stories, such as controlling tone in speech or editorials.

*Addressing bias.* Potential bias in the story generation and control was brought up as an issue. For example, P5 perceived that the algorithm generates stories that are more likely written in western culture. P10 mentioned that the controlled generation did not align with what they expected when the protagonist is an anti-hero, signaling potential bias. An important starting point would be in ensuring a more diverse training set, as is broader testing.

# 8 DISCUSSION

While TaleBrush is currently focused on character fortune, we reflect lessons learned for control design and for iterative human-AI co-creation.

## 8.1 The Gulf of Human-AI Co-Creation

Our design for interactive controls addresses challenges in the gulf of execution and evaluation [109] between the user and the generative AI systems. A key feature is reducing the friction in the iterative co-creation process. First, the **control interaction** of TaleBrush is designed to be *expressive* and *easy*. These are often conflicting goals: simple interfaces have limited expressiveness and expressive interfaces are not simple. However, this balance is clearly needed for closing the gulf of execution but is missing in many existing interactive approaches to co-creation. Second, to minimize the gulf of evaluation, **sensemaking** of generated results should be facilitated. Users should be able to more easily understand how the given controls relate to what the AI generated. This is often missed in 'explainable' AI approaches. The explanation does not map to either the user's intent or how they interface with the software.

## 8.2 Generalizing Line Sketching Control

Beyond story generation, line sketching can potentially be adopted for other applications that generate content with sequence attributes. This is particularly true in domains where we can operationalize and encode features of the content as drawings. For example, poetry [14] and lyrics [147] can be visualized based on moods or rhythms. One indicator of whether the line sketching approach is appropriate for a generative context is if time-series style visualizations already exist in that medium. For example, in audio, we can visualize (and partially control) changes in volume over time. This can be extended to support more powerful types of guidance. For example, a composer could control rhythmic density or melodic contour [2] of generated outputs. Similarly, a sketch line could be used with a text-to-speech generator for flexible micro expression control (e.g., pace or aggressiveness) [128].

## 8.3 Disagreement Between Human and AI

In our experiments, we found situations in which the controlled generation do not satisfy a user's expectation, and the generated text's fortune level might not accord with the given controls. This can be due to algorithmic error, either from the controlled generation or recognition. However, we found that in some cases this was a result of an implausible request (e.g., drastic fluctuation of fortune). Dramatic fluctuations are not often observed in real text and the generative algorithms may not be able to generate suitable text. To address this, we might limit how much the sketch can move between time steps (e.g., a smoothing). Alternatively, we can leverage the ambiguity and undetermined nature of sketching [49, 84] to communicate uncertainty in control. In our prototype, as the user sketches, we show a wide range where the generated output would likely fall. While these design elements manage user expectations on the tightness of the control, our user study found that users would want more sensitive controls when they iterate over generations with small changes in parameters. The sketch-speed approach is one way to allow users to encode how 'tightly' they want the story to fit. However, it may be worth considering alternative approaches in the future (e.g., explicitly selecting different 'pen' widths or experimenting with things like pen pressure).

As our user study revealed, the writer's perception of the protagonist's fortune can also be misaligned with how the tool recognizes it. Adapting the machine's fortune scale to the user's perception can be a way to solve this. For example, we can allow users to revise the machine recognition results based on their perception of the text (e.g., by moving the line the machine drew). This can be fed back to the underlying classifier to learn the user's scale of fortune. To enable such interactions with low user effort, it should be technically possible to adjust models with few data instances from the user. Recent language model tuning approaches, including soft prompt tuning [87, 93], may support this approach. In some situations, the user's expectation may be mismatched due to the potential bias in the model or training data. For example, our study participants mentioned the model seemed to show biased behaviors, such as generating stories more likely written in western cultures, which might be due to skewed distribution in the variety of stories [9]. To alleviate this problem, we can try pretraining big language models with balanced data [9] or having a controllable module that tries to control these biases [76, 91].

## 8.4 Further Support in Story Co-creation

Story attributes other than the protagonist's fortune (e.g., conflict, settings) can also steer the story flow. To decide which attribute to support, the writer's specific needs should be understood first. Based on the identified attribute, visual controls can be designed to assure expressive and simple control for the attribute. For example, if writers want textual keywords to be used as control [152], placing keywords along the story sequence axis can be one design option. Alternatively, if a writer wants to convey condensed information about character interaction, diagramming a character network graph can be a control option [4]. For example, one could imagine drawing multiple instances of a network diagram to guide the AI by which characters in a story should interact and how. Alternative sketched representations are worth studying both for

text generation [103] and beyond. To support these, the dataset and algorithmic pipeline would need to be expanded based on the attribute to be controlled.

For specific story domains, such as screenwriting, generation can be extended to adopt the "show, not tell" approach. For such extensions, the dataset would need to be constructed with screen scripts while the algorithms would also need to consider the latent fortune states throughout the story.

TaleBrush can potentially be extended to support the authoring of high-fidelity, longer text. To provide such support, the tool would need to consider writers in the translation stage [44, 48], where ideas and plans are transformed into detailed text. Hence, generative tools should be able to follow the writer's specifications. The annotation approach would also need to be reconsidered, such as annotating the fortune of the character after summarizing the long text with crowdsourcing [141] or algorithms [8]. Moreover, the control and sensemaking should be re-designed for longer text. For example, a tool might allow the writer to first sketch an outline and then a detailed story [40]. A more complete tool might enable a long-term study of how co-creation impacts writing.

### 8.5 Limitation and Future Work

In this work, we qualitatively investigated how writers would use TaleBrush and sketching interactions. Future work can investigate how proposed interactions impact the story writing compared to writers' current practices or when controlled generation is used without sketching interactions. Moreover, we did not look into how each function impacts story writing separately. For example, how would the control of surprise impact the writing experience alone? To answer such questions, more controlled user experiments would be needed. While our annotation approach produced usable data, we identified ways it could be improved in the future. While collecting comparison information with the slider interface, we saw that how calibrating sentences were annotated would impact the annotation results. For example, if one calibrating sentence has been annotated with maximum fortune and the annotator later realizes that one of the task sentences has higher fortune, the annotator would either not annotate accurate comparison information or need to re-annotate the calibrating sentence. One possible approach is to modify the interface so that sliders can be extended without having maximum or minimum ends. Finally, while the training data specified which character's fortune should be steered, the trained model frequently coupled multiple characters' fortunes. For example, in the story about Bob and Alice, it is hard to make Bob have a good fortune and Alice have a bad one. Explicit control of multiple characters' fortune can be one future work direction to handle this limitation.

## 9 CONCLUSION

We introduce TaleBrush, a human-AI story co-creation tool that allows writers to sketch out textual stories by visually controlling the protagonist's fortune. The visual sketch is used as an input to the GPT-based story generation model and the model generates stories according to the specified protagonist's fortune. To help make sense of the generated story, the protagonist's fortune is visualized using the same abstract representation as the drawn sketch. From our technical evaluation and user study, we found TaleBrush has reliable controllability, while its sketching interaction facilitates participants to iterate on generation to collect novel ideas that align with their intentions. With the recent advances in generative ML algorithms, we hope TaleBrush opens up new ways to provide frictionless and intuitive controls.

## ACKNOWLEDGMENTS

We want to thank Naver AI Lab for supporting the work. We also thank Tal August, Young Wook Do, Kenneth Huang, Forrest Huang, Eun Jeong Kang, Haesoo Kim, Rebecca Krosnick, Yoonjoo Lee, Hee-Seung Moon, and Michael Nebeling for valuable feedback and discussion on the work. Lastly, we thank crowd workers and user study participants for their time and effort.

## REFERENCES

[1] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. 2011. *Visualization of Time-Oriented Data* (1st ed.). Springer Publishing Company, Incorporated.

[2] Taketo Akama. 2019. Controlling Symbolic Music Generation based on Concept Learning from Domain Knowledge. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, Arthur Flexer, Geoffroy Peeters, Julián Urbano, and Anja Volk (Eds.). 816–823. http://archives.ismir.net/ismir2019/paper/000100.pdf

[3] Nader Akoury, Shufan Wang, Josh Whiting, Stephen Hood, Nanyun Peng, and Mohit Iyyer. 2020. STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6470–6484. https://doi.org/10.18653/v1/2020.emnlp-main.525

[4] Prithviraj Ammanabrolu, William Broniec, Alex Mueller, Jeremy Paul, and Mark Riedl. 2019. Toward Automated Quest Generation in Text-Adventure Games. In *Proceedings of the 4th Workshop on Computational Creativity in Language Generation*. Association for Computational Linguistics, Tokyo, Japan, 1–12. https://aclanthology.org/2019.ccnlg-1.1

[5] Prithviraj Ammanabrolu, Wesley Cheung, William Broniec, and Mark O. Riedl. 2020. Automated Storytelling via Causal, Commonsense Plot Ordering. *CoRR* abs/2009.00829 (2020). https://arxiv.org/abs/2009.00829

[6] Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara Martin, and Mark Riedl. 2019. Guided Neural Language Generation for Automated Storytelling. In *Proceedings of the Second Workshop on Storytelling*. Association for Computational Linguistics, Florence, Italy, 46–55. https://doi.org/10.18653/v1/W19-3405

[7] Kitti Balogh, Krisztina Szucs, Viktoria Verecze, and Zoltan Varju. [n.d.]. Visualizing Star Wars Movie Scripts. https://tas.precognox.com/labs/star-wars-visualization/

[8] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).

[9] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (Virtual Event, Canada) *(FAccT '21)*. Association for Computing Machinery, New York, NY, USA, 610–623. https://doi.org/10.1145/3442188.3445922

[10] Eden Bensaid, Mauro Martino, Benjamin Hoover, Jacob Andreas, and Hendrik Strobelt. 2021. FairyTailor: A Multimodal Generative Framework for Storytelling. arXiv:2108.04324 [cs.CL]

[11] Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. *Soylent: A Word Processor with a Crowd Inside*. Association for Computing Machinery, New York, NY, USA, 313–322. https://doi.org/10.1145/1866029.1866078

[12] Natalia Bilenko. 2016. *The narrative explorer*. Ph.D. Dissertation. Master's thesis, EECS Department, University of California, Berkeley.

[13] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow*. http://github.com/eleutherai/gpt-neo

[14] Kyle Booten and Katy Ilonka Gero. 2021. Poetry Machines: Eliciting Designs for Interactive Writing Tools from Poets. In *Creativity and Cognition*. 1–5.

[15] Mike Bostock. 2012. Les Misérables Co-occurrence. https://bost.ocks.org/mike/miserables/

[16] Nadia Boukhelifa, Anastasia Bezerianos, Tobias Isenberg, and Jean-Daniel Fekete. 2012. Evaluating Sketchiness as a Visual Variable for the Depiction of Qualitative Uncertainty. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2769–2778. https://doi.org/10.1109/TVCG.2012.220

[17] Richard Brath. 2021. Surveying Wonderland for many more literature visualization techniques. *CoRR* abs/2110.08584 (2021). arXiv:2110.08584 https://arxiv.org/abs/2110.08584

[18] Matthew Brehmer, Bongshin Lee, Benjamin Bach, Nathalie Henry Riche, and Tamara Munzner. 2017. Timelines Revisited: A Design Space and Considerations for Expressive Storytelling. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 23 (2017), 2151–2164. Issue 9. https://doi.org/10.1109/TVCG.2016.2614803

[19] Write Brothers. 1994. Dramatica®The Next Chapter in Story Development. https://dramatica.com/

[20] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. 1992. An Estimate of an Upper Bound for the Entropy of English. *Computational Linguistics* 18, 1 (1992), 31–40. https://aclanthology.org/J92-1002

[21] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]

[22] Jill Burstein, Beata Beigman Klebanov, Norbert Elliot, and Hillary Molloy. 2016. A Left Turn: Automated Feedback and Activity Generation for Student Writers. In *Language Teaching, Learning and Technology*. 6–13. https://doi.org/10.21437/LTLT.2016-2

[23] Daniel Buschek, Martin Zürn, and Malin Eiband. 2021. The Impact of Multiple Parallel Phrase Suggestions on Email Input and Composition Behaviour of Native and Non-Native English Writers. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 732, 13 pages. https://doi.org/10.1145/3411764.3445372

[24] Alex Calderwood, Vivian Qiu, Katy Ilonka Gero, and Lydia B Chilton. 2020. How Novelists Use Generative Language Models: An Exploratory User Study.. In *HAI-GEN+ user2agent@ IUI*.

[25] Marc Cavazza, Fred Charles, and Steven J. Mead. 2001. Characters in Search of an Author: AI-Based Virtual Storytelling. In *Proceedings of the International Conference on Virtual Storytelling: Using Virtual Reality Technologies for Storytelling (ICVS '01)*. Springer-Verlag, Berlin, Heidelberg, 145–154.

[26] Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. 2021. CoCon: A Self-Supervised Approach for Controlled Text Generation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=VD_ozqvBy4W

[27] John Joon Young Chung, Shiqing He, and Eytan Adar. 2021. The Intersection of Users, Roles, Interactions, and Technologies in Creativity Support Tools. In *Conference on Designing Interactive Systems*. ACM, 1817–1833.

[28] Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 2018. Creative Writing with a Machine in the Loop: Case Studies on Slogans and Stories. In *23rd International Conference on Intelligent User Interfaces* (Tokyo, Japan) *(IUI '18)*. Association for Computing Machinery, New York, NY, USA, 329–340. https://doi.org/10.1145/3172944.3172983

[29] Andy Coenen, Luke Davis, Daphne Ippolito, Emily Reif, and Ann Yuan. 2021. Wordcraft: a Human-AI Collaborative Editor for Story Writing. *arXiv preprint arXiv:2107.07430* (2021).

[30] W.W. Cook and P. Collins. 2016. *Plotto: The Master Book of All Plots*. Tin House Books. https://books.google.co.kr/books?id=xOCMDAEACAAJ

[31] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=H1edEyBKDS

[32] Alexandre Denis, Samuel Cruz-Lara, Nadia Bellalem, and Lotfi Bellalem. 2014. Visualization of affect in movie scripts. In *Empatex, 1st International Workshop on Empathic Television Experiences at TVX 2014*. Newcastle, United Kingdom. https://hal.archives-ouvertes.fr/hal-01099668

[33] Ruta Desai, Fraser Anderson, Justin Matejka, Stelian Coros, James McCann, George Fitzmaurice, and Tovi Grossman. 2019. *Geppetto: Enabling Semantic Design of Expressive Robot Behaviors*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300599

[34] Eva Deverell. 2020. Character Arc Plot & Kurt Vonnegut's Story Shapes. https://www.eadeverell.com/character-arc/

[35] Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling Language Models to Fill in the Blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2492–2501. https://doi.org/10.18653/v1/2020.acl-main.225

[36] Alexandre Duval, Thomas Lamson, Gaël de Léséleuc de Kérouara, and Matthias Gallé. 2021. Breaking Writer's Block: Low-cost Fine-tuning of Natural Language Generation Models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Online, 278–287. https://aclanthology.org/2021.eacl-demos.33

[37] Philipp Eichmann and Emanuel Zgraggen. 2015. Evaluating Subjective Accuracy in Time Series Pattern-Matching Using Human-Annotated Rankings. In *Proceedings of the 20th International Conference on Intelligent User Interfaces* (Atlanta, Georgia, USA) *(IUI '15)*. Association for Computing Machinery, New York, NY, USA, 28–37. https://doi.org/10.1145/2678025.2701379

[38] Mathias Eitz, James Hays, and Marc Alexa. 2012. How Do Humans Sketch Objects? *ACM Trans. Graph. (Proc. SIGGRAPH)* 31, 4 (2012), 44:1–44:10.

[39] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. 2009. PhotoSketch: A Sketch Based Image Query and Compositing System. In *SIGGRAPH 2009: Talks* (New Orleans, Louisiana) *(SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 60, 1 pages. https://doi.org/10.1145/1597990.1598050

[40] Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 889–898. https://doi.org/10.18653/v1/P18-1082

[41] Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for Structuring Story Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 2650–2660. https://doi.org/10.18653/v1/P19-1254

[42] Chaoran Fan, Kresimir Matkovic, and Helwig Hauser. 2020. Sketch-based fast and accurate querying of time series using parameter-sharing LSTM networks. *IEEE Transactions on Visualization and Computer Graphics* (2020), 1–1. https://doi.org/10.1109/TVCG.2020.3002950

[43] Fictional Devices. 2021. Plottr. https://plottr.com/

[44] Linda Flower and John R. Hayes. 1981. A Cognitive Process Theory of Writing. *College Composition and Communication* 32, 4 (1981), 365–387. http://www.jstor.org/stable/356600

[45] Katy Ilonka Gero and Lydia B. Chilton. 2019. How a Stylistic, Machine-Generated Thesaurus Impacts a Writer's Process. In *Proceedings of the 2019 on Creativity and Cognition* (San Diego, CA, USA) *(C&C '19)*. Association for Computing Machinery, New York, NY, USA, 597–603. https://doi.org/10.1145/3325480.3326573

[46] Katy Ilonka Gero and Lydia B. Chilton. 2019. *Metaphoria: An Algorithmic Companion for Metaphor Creation*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300526

[47] Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. 2005. Story Plot Generation Based on CBR. *Know.-Based Syst.* 18, 4–5 (Aug. 2005), 235–242. https://doi.org/10.1016/j.knosys.2004.10.011

[48] Nick Greer, Jaime Teevan, and Shamsi Iqbal. 2016. *An Introduction to Technological Support for Writing*. Technical Report MSR-TR-2016-1. https://www.microsoft.com/en-us/research/publication/an-introduction-to-technological-support-for-writing/

[49] Mark D. Gross and Ellen Yi-Luen Do. 1996. Ambiguous Intentions: A Paper-like Interface for Creative Design. In *ACM Symposium on User Interface Software and Technology*. ACM, 183–192.

[50] Jian Guan, Zhexin Zhang, Zhuoer Feng, Zitao Liu, Wenbiao Ding, Xiaoxi Mao, Changjie Fan, and Minlie Huang. 2021. OpenMEVA: A Benchmark for Evaluating Open-ended Story Generation Metrics. arXiv:2105.08920 [cs.CL]

[51] Matthew Guzdial and Mark Riedl. 2019. An Interaction Framework for Studying Co-Creative AI. arXiv:1903.09709 [cs.HC]

[52] Steve Haroz, Robert Kosara, and Steven L. Franconeri. 2016. The Connected Scatterplot for Presenting Paired Time Series. *IEEE Transactions on Visualization and Computer Graphics* 22, 9 (2016), 2174–2186. https://doi.org/10.1109/TVCG.2015.2502587

[53] Brent Harrison, Christopher Purdy, and Mark Riedl. 2021. Toward Automated Story Generation with Markov Chain Monte Carlo Methods and Deep Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 13, 2 (Jun. 2021), 191–197. https://ojs.aaai.org/index.php/AIIDE/article/view/13003

[54] Devamanyu Hazarika, Mahdi Namazifar, and Dilek Hakkani-Tür. 2021. Zero-Shot Controlled Generation with Encoder-Decoder Transformers. arXiv:2106.06411 [cs.CL]

[55] Jeffrey Heer and Ben Shneiderman. 2012. Interactive Dynamics for Visual Analysis. *Commun. ACM* 55, 4 (April 2012), 45–54. https://doi.org/10.1145/2133806.2133821

[56] Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. TrueSkill™: A Bayesian Skill Rating System. In *Proceedings of the 19th International Conference on Neural Information Processing Systems* (Canada) *(NIPS'06)*. MIT Press, Cambridge, MA, USA, 569–576.

[57] Will E. Hipson and Saif M. Mohammad. 2021. Emotion Dynamics in Movie Dialogues. arXiv:2103.01345 [cs.CL]

[58] Christian Holz and Steven Feiner. 2009. Relaxed Selection Techniques for Querying Time-Series Graphs. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology* (Victoria, BC, Canada) *(UIST '09)*. Association for Computing Machinery, New York, NY, USA, 213–222. https://doi.org/10.1145/1622176.1622217

[59] Eric Hoyt, Kevin Ponto, and Carrie Roy. 2014. Visualizing and Analyzing the Hollywood Screenplay with ScripThreads. *Digit. Humanit. Q.* 8, 4 (2014). http://www.digitalhumanities.org/dhq/vol/8/4/000190/000190.html

[60] Chi-yang Hsu, Yun-Wei Chu, Ting-Hao Huang, and Lun-Wei Ku. 2021. Plot and Rework: Modeling Storylines for Visual Storytelling. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 4443–4453. https://doi.org/10.18653/v1/2021.findings-acl.390

[61] Ting-Yao Hsu, Yen-Chia Hsu, and Ting-Hao (Kenneth) Huang. 2019. On How Users Edit Computer-Generated Visual Stories. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI EA '19)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3290607.3312965

[62] Ting-Yao Hsu, Chieh-Yang Huang, Yen-Chia Hsu, and Ting-Hao Huang. 2019. Visual Story Post-Editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 6581–6586. https://doi.org/10.18653/v1/P19-1658

[63] Chieh-Yang Huang, Shih-Hong Huang, and Ting-Hao Kenneth Huang. 2020. Heteroglossia: In-situ story ideation with the crowd. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[64] Chieh-Yang Huang and Ting-Hao Huang. 2021. Semantic Frame Forecast. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 2702–2713. https://doi.org/10.18653/v1/2021.naacl-main.215

[65] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. *Swire: Sketch-Based User Interface Retrieval*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3290605.3300334

[66] Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual Storytelling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, 1233–1239. https://doi.org/10.18653/v1/N16-1147

[67] Yichen Huang, Yizhe Zhang, Oussama Elachqar, and Yu Cheng. 2020. INSET: Sentence Infilling with INter-SEntential Transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2502–2515. https://doi.org/10.18653/v1/2020.acl-main.226

[68] Jordan S. Huffaker, Jonathan K. Kummerfeld, Walter S. Lasecki, and Mark S. Ackerman. 2020. *Crowdsourced Detection of Emotionally Manipulative Language*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376375

[69] Julie S. Hui, Darren Gergle, and Elizabeth M. Gerber. 2018. *IntroAssist: A Tool to Support Writing Introductory Help Requests*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173596

[70] Daphne Ippolito, David Grangier, Chris Callison-Burch, and Douglas Eck. 2019. Unsupervised Hierarchical Story Infilling. In *Proceedings of the First Workshop on Narrative Understanding*. Association for Computational Linguistics, Minneapolis, Minnesota, 37–43. https://doi.org/10.18653/v1/W19-2405

[71] Stefan Jänicke, Greta Franzini, Muhammad Faisal Cheema, and Gerik Scheuermann. 2015. On Close and Distant Reading in Digital Humanities: A Survey and Future Challenges. In *Eurographics Conference on Visualization (EuroVis) - STARs*, R. Borgo, F. Ganovelli, and I. Viola (Eds.). The Eurographics Association. https://doi.org/10.2312/eurovisstar.20151113

[72] Jermain Kaminski, Michael Schober, Raymond Albaladejo, Oleksandr Zastupailo, and César Hidalgo. 2018. Moviegalaxies-social networks in movies. (2018).

[73] D. Kaufer, C. Geisler, Pantelis Vlachos, and S. Ishizaki. 2006. Mining textual knowledge for writing education and research: The DocuScope project. *Studies in Writing* (01 2006), 115–129. https://doi.org/10.1163/9781849508209_011

[74] Eamonn Keogh and Padhraic Smyth. 1997. A Probabilistic Approach to Fast Pattern Matching in Time Series Databases. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (Newport Beach, CA) *(KDD'97)*. AAAI Press, 24–30.

[75] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. arXiv:1909.05858 [cs.CL]

[76] Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. 2021. A Distributional Approach to Controlled Text Generation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=jWkw45-9AbL

[77] Joy Kim, Justin Cheng, and Michael S. Bernstein. 2014. Ensemble: Exploring Complementary Strengths of Leaders and Crowds in Creative Collaboration. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Baltimore, Maryland, USA) *(CSCW '14)*. Association for Computing Machinery, New York, NY, USA, 745–755. https://doi.org/10.1145/2531602.2531638

[78] Joy Kim, Sarah Sterman, Allegra Argent Beal Cohen, and Michael S. Bernstein. 2017. Mechanical Novel: Crowdsourcing Complex Work through Reflection and Revision. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17)*. Association for Computing Machinery, New York, NY, USA, 233–245. https://doi.org/10.1145/2998181.2998196

[79] Nam Wook Kim, Benjamin Bach, Hyejin Im, Sasha Schriber, Markus Gross, and Hanspeter Pfister. 2018. Visualizing Nonlinear Narratives with Story Curves. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 595–604. https://doi.org/10.1109/TVCG.2017.2744118

[80] Taewook Kim, Jung Soo Lee, Zhenhui Peng, and Xiaojuan Ma. 2019. Love in Lyrics: An Exploration of Supporting Textual Manifestation of Affection in Social Messaging. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 79 (nov 2019), 27 pages. https://doi.org/10.1145/3359181

[81] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Stephen DiVerdi, and Thomas Funkhouser. 2012. Exploring Collections of 3D Models Using Fuzzy Correspondences. *ACM Trans. Graph.* 31, 4, Article 54 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185550

[82] Xiangzhe Kong, Jialiang Huang, Ziquan Tung, Jian Guan, and Minlie Huang. 2021. Stylized Story Generation with Style-Guided Planning. arXiv:2105.08625 [cs.CL]

[83] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. GeDi: Generative Discriminator Guided Sequence Generation. arXiv:2009.06367 [cs.CL]

[84] J.A. Landay and B.A. Myers. 2001. Sketching interfaces: toward more human interface design. *Computer* 34, 3 (2001), 56–64. https://doi.org/10.1109/2.910894

[85] Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. *Automated Grammatical Error Detection for Language Learners*. Morgan and Claypool Publishers.

[86] Michael Lebowitz. 1983. Creating a Story-Telling Universe. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 1* (Karlsruhe, West Germany) *(IJCAI'83)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 63–65.

[87] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. arXiv:2104.08691 [cs.CL]

[88] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. arXiv:2101.00190 [cs.CL]

[89] Shih-Ting Lin, Nathanael Chambers, and Greg Durrett. 2021. Conditional Generation of Temporally-ordered Event Sequences. arXiv:2012.15786 [cs.CL]

[90] Zhiyu Lin and Mark Riedl. 2021. Plug-and-Blend: A Framework for Controllable Story Generation with Blended Control Codes. arXiv:2104.04039 [cs.CL]

[91] Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-Time Controlled Text Generation with Experts and Anti-Experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 6691–6706. https://doi.org/10.18653/v1/2021.acl-long.522

[92] Shixia Liu, Yingcai Wu, Enxun Wei, Mengchen Liu, and Yang Liu. 2013. StoryFlow: Tracking the Evolution of Stories. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2436–2445. https://doi.org/10.1109/TVCG.2013.196

[93] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. arXiv:2103.10385 [cs.CL]

[94] Sandy Louchart and Ruth Aylett. 2007. Building Synthetic Actors for Interactive Dramas.. In *AAAI Fall Symposium: Intelligent Narrative Technologies*. 63–70.

[95] Neil Maiden, Konstantinos Zachos, Amanda Brown, George Brock, Lars Nyre, Aleksander Nygård Tonheim, Dimitris Apsotolou, and Jeremy Evans. 2018. *Making the News: Digital Creativity Support for Journalists*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3173574.3174049

[96] Miro Mannino and Azza Abouzied. 2018. *Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173962

[97] Miro Mannino and Azza Abouzied. 2019. Is This Real? Generating Synthetic Data That Looks Real. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 549–561. https://doi.org/10.1145/3332165.3347866

[98] Stacy C. Marsella, W. Lewis Johnson, and Catherine LaBore. 2000. Interactive Pedagogical Drama. In *Proceedings of the Fourth International Conference on*

*Autonomous Agents* (Barcelona, Spain) *(AGENTS '00)*. Association for Computing Machinery, New York, NY, USA, 301–308. https://doi.org/10.1145/336595.337507

[99] Lara Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[100] MasterClass. 2020. Learn About Narrative Arcs: Definition, Examples, and How to Create a Narrative Arc in Your Writing - 2021. https://www.masterclass.com/articles/what-are-the-elements-of-a-narrative-arc-and-how-do-you-create-one-in-writing#how-to-create-a-narrative-arc-in-4-easy-steps

[101] James R. Meehan. 1977. TALE-SPIN, an Interactive Program That Writes Stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1* (Cambridge, USA) *(IJCAI'77)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 91–98.

[102] Microsoft Research. 2017. TrueSkill. https://trueskill.org/

[103] Franco Moretti. 2005. *Graphs, maps, trees: abstract models for a literary history*. Verso.

[104] Yusuke Mori, Hiroaki Yamane, Yusuke Mukuta, and Tatsuya Harada. 2020. Finding and Generating a Missing Part for Story Completion. In *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*. International Committee on Computational Linguistics, Online, 156–166. https://aclanthology.org/2020.latechclfl-1.19

[105] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, 839–849. https://doi.org/10.18653/v1/N16-1098

[106] T. Munzner. 2015. *Visualization Analysis and Design*. CRC Press. https://books.google.de/books?id=NfkYCwAAQBAJ

[107] Eric T. Nalisnick and Henry S. Baird. 2013. Character-to-Character Sentiment Analysis in Shakespeare's Plays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 479–483. https://aclanthology.org/P13-2085

[108] Michael Nebeling, Alexandra To, Anhong Guo, Adrian A. de Freitas, Jaime Teevan, Steven P. Dow, and Jeffrey P. Bigham. 2016. *WearWrite: Crowd-Assisted Writing from Smartwatches*. Association for Computing Machinery, New York, NY, USA, 3834–3846. https://doi.org/10.1145/2858036.2858169

[109] Donald A. Norman. 2002. *The design of everyday things*. Basic Books, [New York]. http://www.amazon.de/The-Design-Everyday-Things-Norman/dp/0465067107/ref=wl_it_dp_o_pC_S_nC?ie=UTF8&colid=151193SNGKJT9&coliid=I262V9ZRW8HR2C

[110] Santiago Ortiz. 2012. Lostalgic. http://intuitionanalytics.com/other/lostalgic/

[111] Jiefu Ou, Nathaniel Weir, Anton Belyy, Felix Yu, and Benjamin Van Durme. 2021. InFillmore: Frame-Guided Language Generation with Bidirectional Context. In *Proceedings of *SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, Online, 129–142. https://doi.org/10.18653/v1/2021.starsem-1.12

[112] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. 2011. Exploration of Continuous Variability in Collections of 3D Shapes. *ACM Trans. Graph.* 30, 4, Article 33 (July 2011), 10 pages. https://doi.org/10.1145/2010324.1964928

[113] J. Scott Penberthy and Daniel S. Weld. 1992. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning* (Cambridge, MA) *(KR'92)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 103–114.

[114] Zhenhui Peng, Qingyu Guo, Ka Wing Tsang, and Xiaojuan Ma. 2020. Exploring the Effects of Technological Writing Assistance for Support Providers in Online Mental Health Community. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3313831.3376695

[115] James L. Peterson. 1980. Computer Programs for Detecting and Correcting Spelling Errors. *Commun. ACM* 23, 12 (Dec. 1980), 676–687. https://doi.org/10.1145/359038.359041

[116] Ernst Pöppel. 1978. *Time Perception*. Springer Berlin Heidelberg, Berlin, Heidelberg, 713–729. https://doi.org/10.1007/978-3-642-46354-9_23

[117] Rafael PÉrez Ý PÉrez and Mike Sharples. 2001. MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence* 13, 2 (2001), 119–139. https://doi.org/10.1080/09528130010029820 arXiv:https://doi.org/10.1080/09528130010029820

[118] Lu Qiang, Chai Bingjie, and Zhang Haibo. 2017. Storytelling by the StoryCake Visualization. *Vis. Comput.* 33, 10 (Oct. 2017), 1241–1252. https://doi.org/10.1007/s00371-017-1409-2

[119] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[120] Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 4274–4295. https://doi.org/10.18653/v1/2020.emnlp-main.349

[121] Andrew J. Reagan, Lewis Mitchell, Dilan Kiley, Christopher M. Danforth, and Peter Sheridan Dodds. 2016. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science* 5, 1 (04 Nov 2016), 31. https://doi.org/10.1140/epjds/s13688-016-0093-1

[122] Reedsy. 2017. What is a Narrative Arc? • A Guide to Storytelling Structure. https://blog.reedsy.com/narrative-arc/

[123] Mark O Riedl. 2008. Vignette-based story planning: Creativity through exploration and retrieval. In *Proceedings of the 5th International Joint Workshop on Computational Creativity*. 41–50.

[124] Mark Owen Riedl and Vadim Bulitko. 2012. Interactive Narrative: An Intelligent Systems Approach. *AI Magazine* 34, 1 (Dec. 2012), 67. https://doi.org/10.1609/aimag.v34i1.2449

[125] Mark O. Riedl and R. Michael Young. 2010. Narrative Planning: Balancing Plot and Character. *J. Artif. Int. Res.* 39, 1 (Sept. 2010), 217–268.

[126] Kathy Ryall, Neal Lesh, Tom Lanning, Darren Leigh, Hiroaki Miyashita, and Shigeru Makino. 2005. *QueryLines: Approximate Query for Visual Browsing*. Association for Computing Machinery, New York, NY, USA, 1765–1768. https://doi.org/10.1145/1056808.1057017

[127] Thomas Schmidt. 2019. Distant Reading Sentiments and Emotions in Historic German Plays. In *Abstract Booklet, DH_Budapest_2019*. Budapest, Hungary, 57–60. https://epub.uni-regensburg.de/43592/

[128] Eric Hal Schwartz. 2020. Resemble AI Can Now Clone Your Voice to Speak New Languages. https://voicebot.ai/2020/10/15/resemble-ai-can-now-clone-your-voice-to-speak-new-languages/

[129] Ashish Sharma, Inna W. Lin, Adam S. Miner, David C. Atkins, and Tim Althoff. 2021. Towards Facilitating Empathic Conversations in Online Mental Health Support: A Reinforcement Learning Approach. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 194–205. https://doi.org/10.1145/3442381.3450097

[130] Rasagy Sharma and Venkatesh Rajamanickam. 2013. Using interactive data visualization to explore non-linear movie narratives. *Parsons Journal for Information Mapping* (2013).

[131] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 4222–4235. https://doi.org/10.18653/v1/2020.emnlp-main.346

[132] Tarique Siddiqui, Paul Luh, Zesheng Wang, Karrie Karahalios, and Aditya Parameswaran. 2020. ShapeSearch: A Flexible and Efficient System for Shape-Based Exploration of Trendlines. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 51–65. https://doi.org/10.1145/3318464.3389722

[133] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, 1631–1642. https://aclanthology.org/D13-1170

[134] Simeng Sun, Wenlong Zhao, Varun Manjunatha, Rajiv Jain, Vlad Morariu, Franck Dernoncourt, Balaji Vasan Srinivasan, and Mohit Iyyer. 2021. IGA: An Intent-Guided Authoring Assistant. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 5972–5985. https://aclanthology.org/2021.emnlp-main.483

[135] Reid Swanson and Andrew S. Gordon. 2012. Say Anything: Using Textual Case-Based Reasoning to Enable Open-Domain Interactive Storytelling. *ACM Trans. Interact. Intell. Syst.* 2, 3, Article 16 (Sept. 2012), 35 pages. https://doi.org/10.1145/2362394.2362398

[136] Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J. Martin, Animesh Mehta, Brent Harrison, and Mark O. Riedl. 2019. Controllable Neural Story Plot Generation via Reward Shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 5982–5988. https://doi.org/10.24963/ijcai.2019/829

[137] Yuzuru Tanahashi and Kwan-Liu Ma. 2012. Design Considerations for Optimizing Storyline Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2679–2688. https://doi.org/10.1109/TVCG.2012.212

[138] Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. 2014. StoryGraphs: Visualizing Character Interactions as a Timeline. In *2014 IEEE Conference on*

*Computer Vision and Pattern Recognition*. 827–834. https://doi.org/10.1109/CVPR.2014.111

[139] S.R. Turner. 1992. *MINSTREL, a Computer Model of Creativity and Storytelling*. University of California (Los Angeles). Computer Science Department. https://books.google.co.kr/books?id=krAJjwEACAAJ

[140] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[141] Vasilis Verroios and Michael S Bernstein. 2014. Context trees: Crowdsourcing global understanding from local views. In *Second AAAI Conference on Human Computation and Crowdsourcing*.

[142] K. Vonnegut and D. Simon. 2011. *A Man Without a Country*. Seven Stories Press. https://books.google.co.kr/books?id=T7J-Xg2bYKAC

[143] Nick Walton. 2019. AI Dungeon 2. https://aidungeon.cc/

[144] Jing Wang, Jianlong Fu, Jinhui Tang, Zechao Li, and Tao Mei. 2018. Show, Reward and Tell: Automatic Generation of Narrative Paragraph From Photo Stream by Adversarial Training.. In *AAAI*. 7396–7403. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17049

[145] Su Wang, Greg Durrett, and Katrin Erk. 2020. Narrative Interpolation for Generating and Understanding Stories. arXiv:2008.07466 [cs.CL]

[146] Stephen G. Ware, R. Michael Young, Brent Harrison, and David L. Roberts. 2014. A Computational Model of Plan-Based Narrative Conflict at the Fabula Level. *IEEE Transactions on Computational Intelligence and AI in Games* 6, 3 (2014), 271–288. https://doi.org/10.1109/TCIAIG.2013.2277051

[147] Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, Tomoyasu Nakano, Satoru Fukayama, and Masataka Goto. 2017. LyriSys: An Interactive Support System for Writing Lyrics Based on Topic Transition. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces* (Limassol, Cyprus) *(IUI '17)*. Association for Computing Machinery, New York, NY, USA, 559–563. https://doi.org/10.1145/3025171.3025194

[148] Martin Wattenberg. 2001. Sketching a Graph to Query a Time-Series Database. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems* (Seattle, Washington) *(CHI EA '01)*. Association for Computing Machinery, New York, NY, USA, 381–382. https://doi.org/10.1145/634067.634292

[149] Marc Weber, Marc Alexa, and Wolfgang Müller. 2001. Visualizing Time-Series on Spirals. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01) (INFOVIS '01)*. IEEE Computer Society, USA, 7.

[150] KM Weiland. 2017. *Creating Character Arcs: The Masterful Author's Guide to Uniting Story Structure, Plot, and Character Development*. Smashwords Edition.

[151] Thomas Wilhelm, Manuel Burghardt, and Christian Wolff. 2013. " To See or Not to See"-An Interactive Tool for the Visualization and Analysis of Shakespeare Plays. (2013).

[152] Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 2831–2845. https://doi.org/10.18653/v1/2020.emnlp-main.226

[153] Qian Yang, Justin Cranshaw, Saleema Amershi, Shamsi T. Iqbal, and Jaime Teevan. 2019. *Sketching NLP: A Case Study of Exploring the Right Things To Design with Language Intelligence*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300415

[154] Lili Yao, Nanyun Peng, Weischedel Ralph, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-And-Write: Towards Better Automatic Storytelling. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*.

[155] Ji Soo Yi, Youn ah Kang, John Stasko, and J.A. Jacko. 2007. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1224–1231. https://doi.org/10.1109/TVCG.2007.70515

[156] Biqiao Zhang, Georg Essl, and Emily Mower Provost. 2017. Predicting the Distribution of Emotion Perception: Capturing Inter-Rater Variability. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction* (Glasgow, UK) *(ICMI '17)*. Association for Computing Machinery, New York, NY, USA, 51–59. https://doi.org/10.1145/3136755.3136792

[157] Kostas Zoumpatianos, Stratos Idreos, and Themis Palpanas. 2015. RINSE: Interactive Data Series Exploration with ADS+. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1912–1915. https://doi.org/10.14778/2824032.2824099