TaleBrush: Visual Sketching of Story Generation with 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

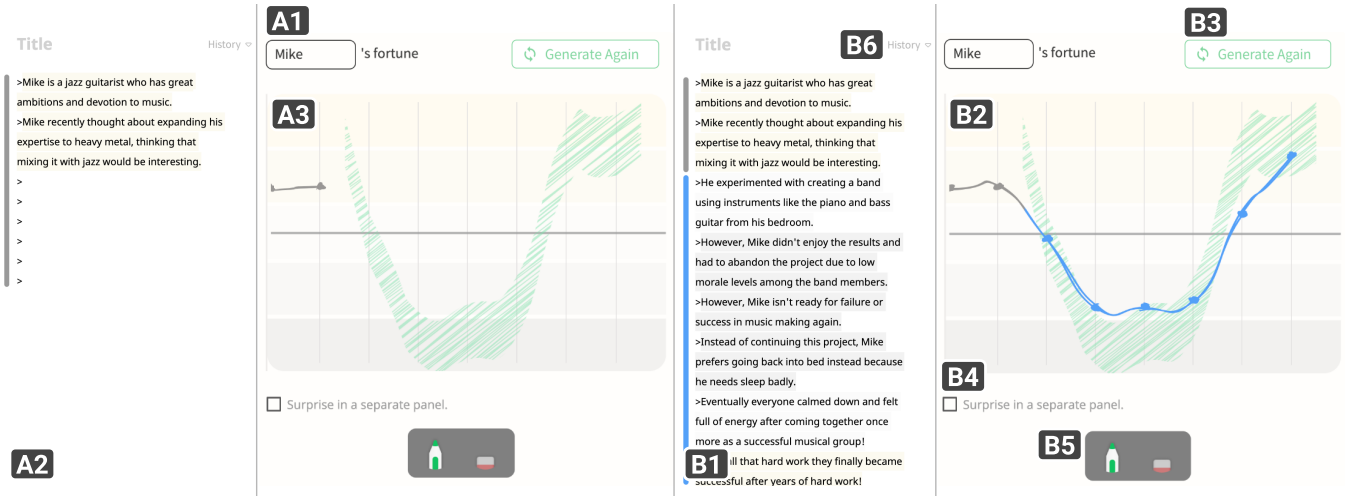

Figure 1: TaleBrush allows users to intuitively control language-model-based story generation by sketching the protagonist's fortune. The sketch (green shaded area in A3) indicates the protagonist's fortune (y-axis) over the chronological sequence of the story (x-axis). The higher the line, the better the fortune. With the width of the sketch, the writer can indicate a 'tolerance' for how much the generated sentence can deviate from the drawn input. The user can set the protagonist's name (A1) and the initial portion of a story (A2) before giving sketching input. After sketching, TaleBrush updates with new story sentences (the blue vertical line, B1) and calculates the fortune visualization (the blue line and dots in B2). The writer can iterate with direct editing or re-sketching. A "Generate Again" button (B3) will produce new sentences for the same sketch. The eraser tool allows writers to remove a portion of the drawn sketch and generate sentences without explicit limits for the erased part. The writer can also set constraints on how surprising the generation should be by opening a separate panel (B4). A history tool (B6) allows writers to access past generations.

ABSTRACT

Advancing text generation algorithms (e.g., GPT-3) have led to new kinds of human-AI story co-creation tools. However, it is difficult for authors to guide this generation and understand the relationship between input controls and generated output. In response, we introduce TaleBrush, a GPT-based tool that uses abstract visualizations and sketched inputs. The tool allows writers to draw out the protagonist's fortune with a simple and expressive interaction. The visualization of the fortune serves both as input control and representation of what the algorithm generated (a story with varying fortune levels). We hope this demonstration leads the community to consider novel controls and sensemaking interactions for human-AI co-creation.

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

ACM ISBN 978-1-4503-9156-6/22/04.
https://doi.org/10.1145/3491101.3519873

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: Visual Sketching of Story Generation with Pretrained Language Models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI '22 Extended Abstracts), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3491101.3519873

1 INTRODUCTION

Pretrained generative language models, such as OpenAI's GPT-3 [15], are rapidly advancing and enabling new types of human-AI story co-creation tools [2–5, 9, 10]. Story co-creation is often iterative. A writer sets the initial story sentence that is used as a prompt for the AI to generate new sentences (which then become new prompts). For example, the user can write "Melissa fought a dragon." The AI can append a sentence: "She used magic to create a barrier around her." The writer can allow the AI to continue by using this new sentence as input (with or without modification) or re-start the process to try to get to a reasonable narrative.

One limitation of this approach is that they are not easily steered given some 'vision' the writer has. For example, the writer might want Melissa to begin with bad luck but eventually reach a happy ending. However, such story dynamics are challenging to encode with existing story co-creation systems. Many algorithms do not yet allow specifications that change over different parts of the story. Existing interfaces (e.g., text inputs or sliders) are cumbersome in controlling time-varying parameters. AI generation is also 'unreliable' in that it may not produce exactly what the writer wanted (e.g., a fortune that varies from very low to very high). To validate that the output text was close to the input parameters may require the writer to manually compare one representation (e.g., a numerical fortune level) to another (the text). With these frictions, fine control is infeasible, making iteration slow and effortful.

In this work, we present TaleBrush, a story generation system that leverages sketching and visualizations for intuitive control and understanding of generated texts (Figure 1). With simple sketched lines, writers can *easily* and *expressively* specify a sequence of fortune (Figure 1A3). That is, the protagonist's fortune is specified as a time series (*expressive*) that can be created in a single stroke (*easy*). The lightweight and ambiguous nature of sketching [8, 12] also manages the writer's tolerance of uncertainty and errors in control, making them more acceptable. TaleBrush employs the same time-series visualizations (Figure 1B2) to support sensemaking of generated results. For these interactions, we implemented steerable story generation architecture. First, we trained prompts that guide the language model to perform different story generation tasks [13]. Second, we built a module that steers the language model to generate according to the input sketch [11]. We hope this demonstration can stimulate the CHI community to leverage the potential of using visual controls and visualizations to make iterative human-AI co-creation intuitive and frictionless.

2 TALEBRUSH: INTERACTION

Below, we briefly motivate our selection of controlled story attributes and explain interactions in TaleBrush's interface.

2.1 Iterative Co-creation With Line Sketching

TaleBrush is designed for generative human-AI co-creation of a short storyline. TaleBrush uses sketched lines as a control mechanism and visualizes generated stories to support sensemaking. Internally, the system is powered by a pretrained large language model, GPT-Neo [1]. GPT-Neo serves story generation with prompts tuned in the continuous token embedding space [13, 14]. TaleBrush also includes a GeDi-based control module, which uses a smaller language model finetuned to steer the generation with control input [11]. With generations as potential ideas, the system supports the planning stage of the writing process [6, 7]. As writers might not yet have a concrete direction in this stage, we expect them to be more accepting of generated results. Writers would use generated sentences as writing prompts or as a means to overcome writer's block. Novice writers can use stories generated from TaleBrush as demonstrations that can jumpstart their writing.

Due to algorithmic uncertainty in ML-based generation, we expect that writers would use TaleBrush iteratively. TaleBrush uses interactive controls to help writers to reach what they favor with a small number of iterations. Specifically, TaleBrush allows writers to specify how the protagonist's fortune should change within the generated story. To specify these sequential attributes, TaleBrush leverages sketching-based control (green shaded area in Figure 1 A3 and B2). This interaction is *simple* and *expressive*, as a series of values can be easily expressed with a single line drawing. The visualization of the generated story (vertical blue line in Figure 1 B2) helps writers to understand how the control specification is applied to the generation—they can simply compare the input sketch with the visualization. Intuitive control and sensemaking interactions would ultimately help writers with the iterative use of the system.

TaleBrush mainly focuses on controlling **the level of the protagonist's fortune** in chronological order. For example, if the main character of the story experiences their career high, they would be in good fortune. On the other hand, if they lose their close friends, they would be in a bad fortune. TaleBrush does not yet support the generation of non-linear narratives, such as flashbacks. The writer can also specify how tightly fortune control would apply in the generation at the cost of generation time.

TaleBrush also allows control of **the level of surprise**, or how unexpected the generation should be. For example, an unsurprising sentence after "Melissa fought a dragon" can be "The dragon was a big, green, scaly beast". On the other hand, a surprising one can be "Melissa killed a giant robot with a robot dragon".

2.2 Interface

2.2.1 Text Editor and Canvas. TaleBrush is composed of a text editor (Figure 1A2 and B1) and a canvas. The canvas supports both fortune sketching as well as visualization of the generated output (Figure 1A3 and B2). The text editor contains multiple 'bullet points', each standing for a single sentence. To these bullet points, the writer can either add their sentences or use TaleBrush to generate sentences. They can also add new bullet points by hitting enter.

Or, they can erase some by deleting the bullet point marker. Each bullet point will be visualized as a dot in the canvas, and sequential dots will be connected with lines. The x and y positions of dots will be decided with the sequential position of the sentence in the story and the protagonist's fortune, respectively. The visualized protagonist's fortune is calculated by an ML recognition algorithm. If no sentence exists in a bullet point, there will be a visual gap in the corresponding position of the canvas.

2.2.2 Generate By Sketching. TaleBrush allows writers to control the generation by manipulating the fortune level time series with sketching. The writer can sketch the arc of how the character's fortune should change. Similar to fortune visualization, x and y position of points in the sketched arc specifies the desired sequence and fortune level. After the sketch is drawn for some sentences, TaleBrush generates story texts on those sentences. When there is no sentence after the sketched part, TaleBrush tries to continue the story. However, if there are sentences after the sketch (an 'end' context), TaleBrush attempts to infill (i.e., take into account the sentences before *and* after the 'blank' part of the story). The writer can also erase portions of their sketch (Figure 1B5). In those spots, TaleBrush generates sentences without fortune constraints. With this function, even without concrete ideas on the character's fortune, writers can try a wide range of generated sentences. The writer can always ask for new sentences for the same sketch by clicking the "Generate Again" button (Figure 1B3). The writer can iterate on generation by redrawing only a portion of the sketch. They also can directly edit the generated text in the text editor.

Sketching Speed for Generation Control Tightness. With text generation, there is always some randomness that limits the ability to perfectly match the desired control. That is, the generative algorithm can't produce a sentence with an exact fortune level and still guarantee coherence. To address this, the text generation system can produce multiple outputs and provide the best match to the writer. This, naturally, comes at the cost of increased generation time. TaleBrush allows users to describe how well they would like generated sentences to follow the given fortune sketch. When the writer sketches slowly, TaleBrush tries more generations to find the one that more matches the given fortune parameter. The slow → accurate mapping was selected to be semantically oriented to the idea that we draw more carefully and slowly when we want an accurate line. The sketch visualizes this controlled precision with a width, which indicates where the generated sentences would be more likely to fall (Figure 1 A3 and B2). The width follows the median of control errors shown from our test data, with the maximum count of tried generation decided by the sketching speed. Hence, the range will be tighter with slower sketching. This speed-based width specification is decided in the unit of a sentence. Hence, if one drew faster on one sentence in a single stroke, the width will be tighter on the quickly drawn part.

Surprise Level Control. TaleBrush allows specification of surprise levels for generations. When the writer checks "surprise in a separate panel", another control panel opens where the writer can draw surprise levels (Figure 2).

2.2.3 Multi-Story Management. If the writer tries generation multiple times, TaleBrush stores each attempt so that the writer can

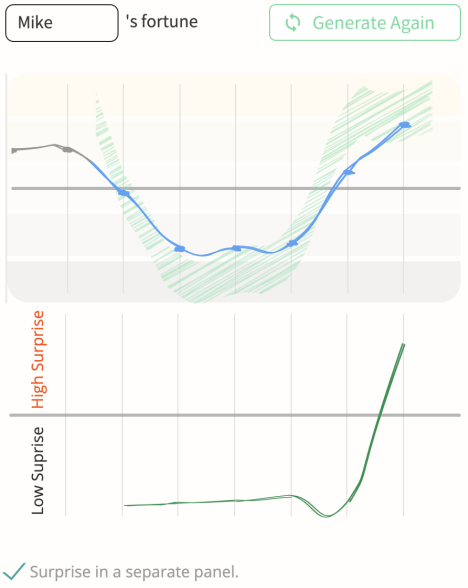

Figure 2: The writer can specify the level of surprise for the generation. When "Surprise in a separate panel" is checked, TaleBrush shows a panel where they can draw a thin green line to specify the surprise level.

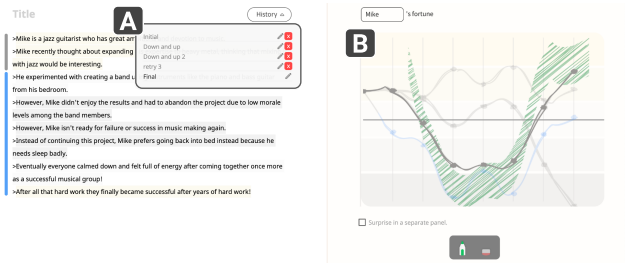

Figure 3: The writer can compare previously generated stories with a drop-down list (A). When the drop-down is open, the fortune arcs of all stored stories are visualized in low opacity on the canvas (B). The writers can check each stored story text and arc visualization by hovering their mouse over the item in the list.

compare and choose any version (Figure 3). They can browse the list of generated stories with the drop-down menu. As they open the menu, all previously generated stories will be shown on the canvas with low opacity. The writer can hover their mouse over an item, and the textbox will show the corresponding story. Moreover, the story's visualization will be highlighted on the canvas. The writer can roll back to one of the past generations by clicking the item in the list.

3 CONCLUSION

We presented TaleBrush, a human-AI story co-creation tool that allows writers to control the generation of textual stories with the sketched arc of the protagonist's fortune. Our GPT-based story

generation architecture uses sketching of the protagonist's fortune as an input to steer the story. To allow the writer to better understand generated output, TaleBrush provides the visualization of the output upon the sketch input. This allows the writer to compare the output to specification, and refine as needed. By leveraging the abstract representation and sketch interactions, TaleBrush was built to introduce a novel way to support frictionless and intuitive human-AI co-creation.

REFERENCES

[1] 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

[2] 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.*

[3] 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.

[4] 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

[5] 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).

[6] 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

[7] 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/

[8] 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.

[9] 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

[10] 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

[11] 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]

[12] 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

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

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

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