Proceedings of the Eighteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment
(AIIDE 2022)

# A Hybrid Approach to Co-creative Story Authoring Using Grammars and Language Models

**Adam Riddle**

Independent Scholar
realtimeriddle@gmail.com

## Abstract

Large language models are powerful tools for story generation, but are difficult to control. Story grammars are a more controllable tool for story generation, but require a large amount of upfront work and tend to create predictable results. We present a story generation tool that combines the positives of both methods using language models to aid in writing a grammar, and using the output of that grammar to generate controlled text from language models. Our approach combines the structure of a grammar with the unexpectedness of a large language model.

## Introduction

Story grammars and large language models are two popular tools used in the co-creative computational generation of stories. Though large language models can quickly create grammatically appropriate text, their output is difficult to control to the degree that authors can not easily use them to express their preexisting ideas for the progression of a story. Several tools for writing with large language models currently exist, usually in the form of feeding some text into a program and then having the model generate a chunk of text which the author can ether approve of, modify, or discard completely and regenerate a new block of text in hopes of producing a more preferable result. With these tools the author has little, if any, control over generation.

Story grammars, on the other hand, can easily be engineered to conform to the author's needs and expectations, with the trade-offs of the grammar's output being predictable and an exponentially increasing amount of work that must go into creating a complex grammar. Tools for creating story grammars are also abundant, usually differing in the authoring style of the grammars, and with what systems the finished grammars are meant to interact. These tools focus on generating random output from the grammars, with all of the uniqueness, direction, and content coming from the work of the author. Combining a grammar creation tool with an existing corpus of data is an option, but can be difficult to incorporate into a project as the author must ensure that all of the data is appropriate and corresponds to their vision.

Our co-creative story authoring tool utilizes large language models and story grammars in a way that emphasises both methods' strengths while softening their weaknesses and guiding authors through the process of using both sets of tools. Using language models to not only both generate and classify text to aid the author in writing a grammar, but also relying on language models to progressively generate the final text, greatly decreases the workload of creating a complex grammar. Simultaneously, using a grammar to direct the output of language models gives the author the opportunity to funnel the generation process to one that fits with their creative vision.[1]



## Design

Using our tool, the author starts by creating a grammar with the replacement grammar library Tracery (Compton, Kybartas, and Mateas 2015). To aid in this process, the author has access to both a zero-shot classifier model to identify appropriate material and a causal language model to both create and inspire new text for creating a grammar. Upon completion of the grammar, the author can feed it's output of partial text into multiple language models, edit the resulting string, append more partial text, and feed all of it back into the generating language models. That cycle continues until the story is complete.

Our software provides two tools to aids in grammar creation. The first tool is a zero-shot classifier that can aid the author in sorting out lists of possible words and sentences. The classifier creates new sub-lists and gives the author the ability to define their own tags and if multiple labels may apply. Authors can also choose to specify a hypothesis template, which lets the classifier model look at their labels as a hypothesis, instead of just a word. Hypotheses templates are always sentences that include {} to denote a blank space where the possible label is supposed to fit in[2].

The second tool gives the author the ability to generate output sentences from a fine-tuned causal language model. This tool lets the author input preceding text and then move the outputs they deem acceptable to the proper grammar. The author can now run the grammar until they receive an output they approve.

[1] https://github.com/realtimeriddle/co-creative-authoring-tool
Video Demo: https://youtu.be/pUbJb09bzCQ

[2] An example hypothesis template: "This label is an example of {}."

The output text is generated to a text editing box and consists of words or phrases made mostly of relevant nouns, verbs, adjectives, pronouns, and punctuation separated by a special separation token <|sepofcond|>. The phrases in between the separation tokens remain as they are throughout the generation process, unless manually edited by the author. The author can send the unprocessed text through two fine-tuned language models in which the first adds words that are not from parts of speech already in the string of text, and the second which adds stop-words and removes the separator tokens creating finished text. The author can choose what text to process by highlighting the text they would like processed and pressing the Process button. The processed text will be added to the text box. The author can now edit the processed text and send the next batch of unprocessed text appended to the newly edited text repeating the cycle until the story is completely written.

## Implementation

### Grammar Creation

The user interface is implemented with PySimpleGUI. The grammar is implemented with Allison Parrish's python port of Tracery, pytracery[3].

### Zero-Shot Classifier

The model used for zero-shot classification is Bart-large (Lewis et al. 2019) fine-tuned on MultiNLI Data-set (Williams, Nangia, and Bowman 2018). Zero-shot classification is handled by the transformers library produced by artificial intelligence company, Hugging Face.[4]

### Conditional Text Generation

The model used for conditional text generation is GPT2-XL (Radford et al. 2019). We fine-tuned the base model for 2 epochs with 10000 samples from the Reddit Writing Prompt Dataset (Fan, Lewis, and Dauphin 2018). Each sample was tokenized into sentences with Spacy (Honnibal and Montani 2017) and each word was tagged by the flair library (Akbik, Blythe, and Vollgraf 2018). Any words not tagged as adjectives, nouns, pronouns, proper nouns, verbs, punctuation and words that were found in NLTK's (Bird, Klein, and Loper 2009) stop-words corpus were removed and replaced with a special token, <|sepofcond|>, separating groups of remaining words.

### Progressive Text Generation

The two models used for progressive text generation (Tan et al. 2020) were two large Bart models both fine-tuned for 2 epochs on 10000 samples from the Reddit writing prompt corpus prepared similarly to the last section. The only difference was only stop-words were excluded from both the output of the first model and input of the second Bart model. The output of the second Bart model contained unaltered text from the corpus. During generation the input string, formatted again as described in the previous section was fed into the first Bart model, whose output was decoded and fed into the second Bart model. The resulting output is fully realized text.

[3]https://github.com/aparrish/pytracery

[4]https://github.com/huggingface/transformers

## Related Work

Researchers from Columbia University conducted an exploratory user study examining how four published novelists interacted with generative language models when such models were introduced into their workflow. (Calderwood et al. 2020) The researchers found that ease of use, a rapid iteration cycle, and little overhead in interacting with language models are important in co-authoring tools. Researchers also found that the unpredictability of language models can be a strength as the model will challenge the preconceived notions of what the author was planning, but there should be at least some constraints as to ensure the outputs of language models tend more toward the author's intended use. Our co-authoring tool follows these guidelines by controlling the output of language models with a grammar with the ability to classify the author's intent, and easy editing of both the input and output texts.

Two other co-authoring tools that attempt to aid authors in writing using large language models are a tool from the researchers a the University of Hamburg that focuses on an agent acting as a teammate with a group of human authors (Wiethof, Tavanapour, and Bittner 2021), and Wordcraft, a co-authoring tool that uses a single language model to aid the author in multiple tasks including continuation, in-filling, elaboration, and rewriting of text. Both tools suffered from the common downsides of large language models previously discussed, but using human-like agents and dialogue models fine-tuned for multiple tasks could be useful in future iterations of our tool, aiding the in the co-authoring process by not only controlling the language model, but also communicating with it.

While other co-creative authoring tools do exist, none we found attempt to combine large language models and grammars as we have. Two surveys done of recent work in authoring tools, one focusing on the cognitive needs of writers (Gero et al. 2022), and another focusing more on gaps in the creative process not currently seeing much support from current co-creative authoring tools (Kreminski and Martens 2022). Both surveys cover problems with many existing solutions such as, highly constrained methods to aid in writing, or simply getting writer unstuck, as well as problems with little to no current solutions like, aiding in expressing intent, or turning an outline into prose. Our tool could be a good first step in improving solutions to already solved problems and creating solutions for unsolved ones, such as expressing intent and an outline in a grammar and using that grammar to direct a language model. Using this approach, our tool has the benefit of better solved problems to tackle the less firmly solved.

## Acknowledgements

Adam Riddle thanks Max Kreminski for their invaluable assistance and guidance in the creation of this work.

## References


Akbik, A.; Blythe, D.; and Vollgraf, R. 2018. Contextual String Embeddings for Sequence Labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, 1638–1649.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Calderwood, A.; Qiu, V.; Gero, K. I.; and Chilton, L. B. 2020. How Novelists Use Generative Language Models: An Exploratory User Study. In *HAI-GEN + user2agent @ IUI*.

Compton, K.; Kybartas, B.; and Mateas, M. 2015. Tracery: an author-focused generative text tool. In *International Conference on Interactive Digital Storytelling*, 154–161. Springer.

Fan, A.; Lewis, M.; and Dauphin, Y. 2018. Hierarchical Neural Story Generation. arXiv:1805.04833.

Gero, K.; Calderwood, A.; Li, C.; and Chilton, L. 2022. A Design Space for Writing Support Tools Using a Cognitive Process Model of Writing. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, 11–24.

Honnibal, M.; and Montani, I. 2017. Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. https://spacy.io. Accessed: 2022-06-02.

Kreminski, M.; and Martens, C. 2022. Unmet Creativity Support Needs in Computationally Supported Creative Writing. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, 74–82.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv:1910.13461.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Tan, B.; Yang, Z.; AI-Shedivat, M.; Xing, E. P.; and Hu, Z. 2020. Progressive Generation of Long Text with Pretrained Language Models. arXiv:2006.15720.

Wiethof, C.; Tavanapour, N.; and Bittner, E. 2021. Implementing an intelligent collaborative agent as teammate in collaborative writing: toward a synergy of humans and AI. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, 400.

Williams, A.; Nangia, N.; and Bowman, S. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1112–1122. Association for Computational Linguistics.