Winnie-the-Pooh Bedtime Story Generator

Overview

Bedtime stories are an integral part of many children's daily routines. I have spent a significant amount of time with my two-year-old niece, who loves having stories read to her. However, she often becomes bored and impatient when the same stories are read repeatedly night after night. While libraries are a great option to find a variety of children's books, the access to these books are not instantaneous and there can still be a limited selection. Having a program that can produce new content at the whim of a child can assist parents and caregivers to engage children with their stories.

The goal of this project is to build this kind of interactive program that generates new stories customized to the user's prompt, including a main character and setting/events. As a starting point, this will be limited to the style and world of the original Winnie the Pooh stories. Further development can expand into different styles and genres.

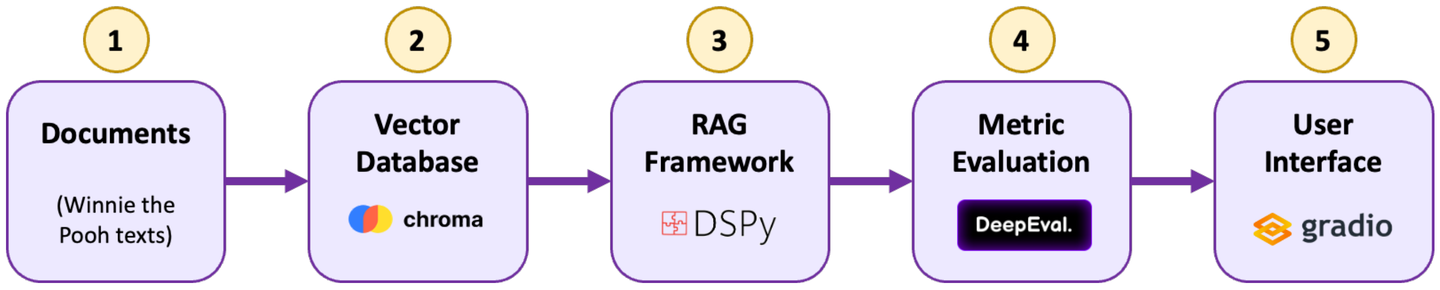
Data

The original Winnie the Pooh stories are currently public domain, and are therefore are readily available in digital forms. I sourced the texts from Project Gutenberg – a free digital library – because it provides relatively clean text files. A challenge with children's texts is the illustrations and visual style within the books; with many pictures and inconsistent text formatting, reading PDFs can become an involved process. With the texts already extracted by Project Gutenberg, this is no longer a concern.

These texts will ultimately be cleaned, chunked, and embedded into a vector database for future retrieval, a process that is described in the following section.

Project Approach

This project was built using five primary stages:



In addition to these five components used to build the application, the flow of the application is completed when user input is added. These inputs are a character name and story prompt, both of which are directed to the retrieval-augmented generation – or RAG – framework (3), and the prompt is sent to the vector database (2) to retrieve the context. Then the flow continues with metric evaluation (4) and everything is wrapped up in the UI (5).

1. Preparing Data/Texts

As previously noted, the Winnie the Pooh stories are already in a .txt file format, and therefore straightforward to import. The following steps were taken to clean and prepare the texts to be embedded in a vector database.

I first inspected the text to checked if there are any obvious text characters that are superfluous to the stories. This started by counting non-alphanumeric characters and looking at exerts that included unexpected punctuation. In this case, asterisks (*) and underscores (_) were included as emphasis, but did not pertain to the context of the stories. Therefore, I determined these characters can be removed before embedding.

Next, I defined new functions to aid in the processing:

- ***load_documents()*** removes whitespace, underscores and asterisks, while also splitting the text file into chapters and removing title pages and publication information that are presented before the first chapter.
- ***split_text()*** the breaks the chapters into smaller chunks (200 characters) using Langchain's *RecursiveCharacterTextSplitter.* For each chunk, metadata is also recorded to include the title, author, chapter and chunk number for future reference.

After the Winnie the Pooh text was loaded and chunked, I checked that all ten original chapters/stories were saved in the dataframe, and each chapter contained anywhere from 46-110 chunks.

2. Vector Database (ChromaDB)

I chose to use Chroma as the vector database primarily due to its ease of use, open-sourced nature and minimal cost. Chroma's existing compatibility and integration with popular frameworks including DSPy and Langchain, also allowed for flexibility in the planning stages of this project.

The ***save_to_chroma()*** function embeds and saves the text chunks and their associated metadata to a persistent ChromaDB collection. This process involves:

- Clearing any existing database to ensure there is no previous data in the database that would
- Creating a new Chroma vector database and collection
- Embedding documents (chunks) with their metadata and generated UUIDs using Chroma's default embedding function

When testing the database by entering a sample query for “honey” (Pooh’s favorite treat), the top three results were returned. The results confimed the cosine similarity distances, corresponding text chunks, and saved metadata were all retrieved.

3. RAG Framework (DSPy)

DSPy Model Configuration

- The script configures OpenAI's gpt-4o-mini model as the large language model (LLM). This LLM is sufficient for these purposes and minimizes the cost to run.
- It sets up a ChromaDB client and retriever using DSPy’s built-in *ChromadbRM*, which will be used to fetch relevant context from the Winnie the Pooh stories.
- DSPy settings are configured with the LLM and retriever.

Signature: *GenerateStory*

- Inputs: name, prompt, context → Output: story
- Name and prompt are provided by the user, while the context is retrieved from the database based on the prompt.

Module: *StoryGenerator*

- Generates using DSPy’s *ChainOfThought* module with the customized *GenerateStory* signature as defined above
- Retrieves 8 chunks from the vector database most similar to the prompt, and defined that as the context.
- Finally returns the story generated based on the name, prompt, and retrieved context.

4. Evaluation Metrics (DeepEval)

Choosing Metrics

Common evaluation metrics for NLP and text generation, such as accuracy and relevancy, are not so applicable for this project. Fictional stories are not intended to be accurate, and the generated text is not meant to strictly adhere to the relevancy of the context retrieved. With the goal of this project focused on the appropriateness of stories for children, I primarily considered readability metrics as the most appropriate measure of the generated outputs.

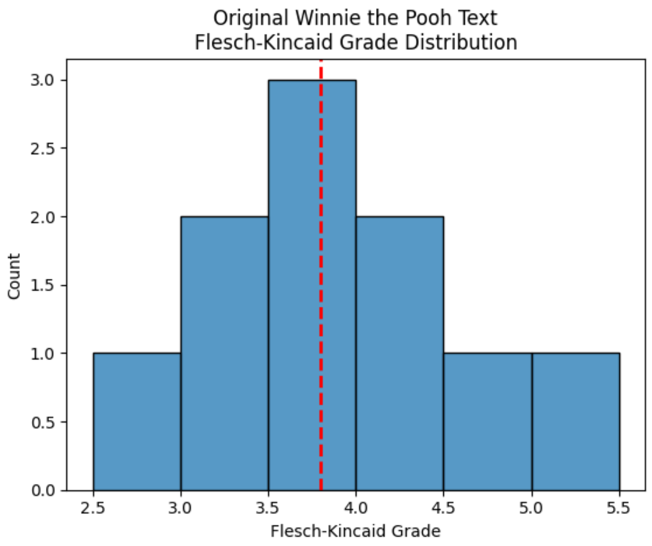
The Flesch–Kincaid readability test is a well-established, yet simple, measure of reading complexity based on word and sentence lengths:

$$0.39\left(\frac{total\ words}{total\ sentences}\right) + \ 11.80\left(\frac{total\ syllables}{total\ words}\right) - 15.59$$

When applying this formula to the original texts, the mean grade level score of the original texts is 3.8, with a standard deviation 0.8. My goal is for the generated stories to be within the range 3.0 - 4.6, or within one standard deviation of the original mean.

Original Winnie the Pooh Text
Flesch-Kincaid Grade Metrics

Mean:	3.8
Median:	3.7
Standard deviation:	0.8
Maximum:	5.5



Evaluation with DeepEval

Defining a custom DeepEval class based on the Flesch-Kincaid Grade Level:

- *ReadabilityMetric* class is defined with high and low thresholds
- It aims to keep the generated stories within one standard deviation of the mean readability score of the original Winnie the Pooh stories (target: less than 4.6).
- *ReadabilityMetric.score()* returns the calculated Flesch-Kincaid Grade Level
- *ReadabilityMetric.measure()* returns whether the score is within the set threshold

Story Evaluation Function

- *evaluate_readability* function generates a story, initializes the *ReadabilityMetric,* and returns the readability and the story output.
- *print_story* function attempts to generate a story that passes the readability metric. If the first attempt fails the metric, it tries again with a modified prompt to use simplified words and sentences. The function returns the generated story if it passes, or a message that it could not generate an appropriate story if failed twice.

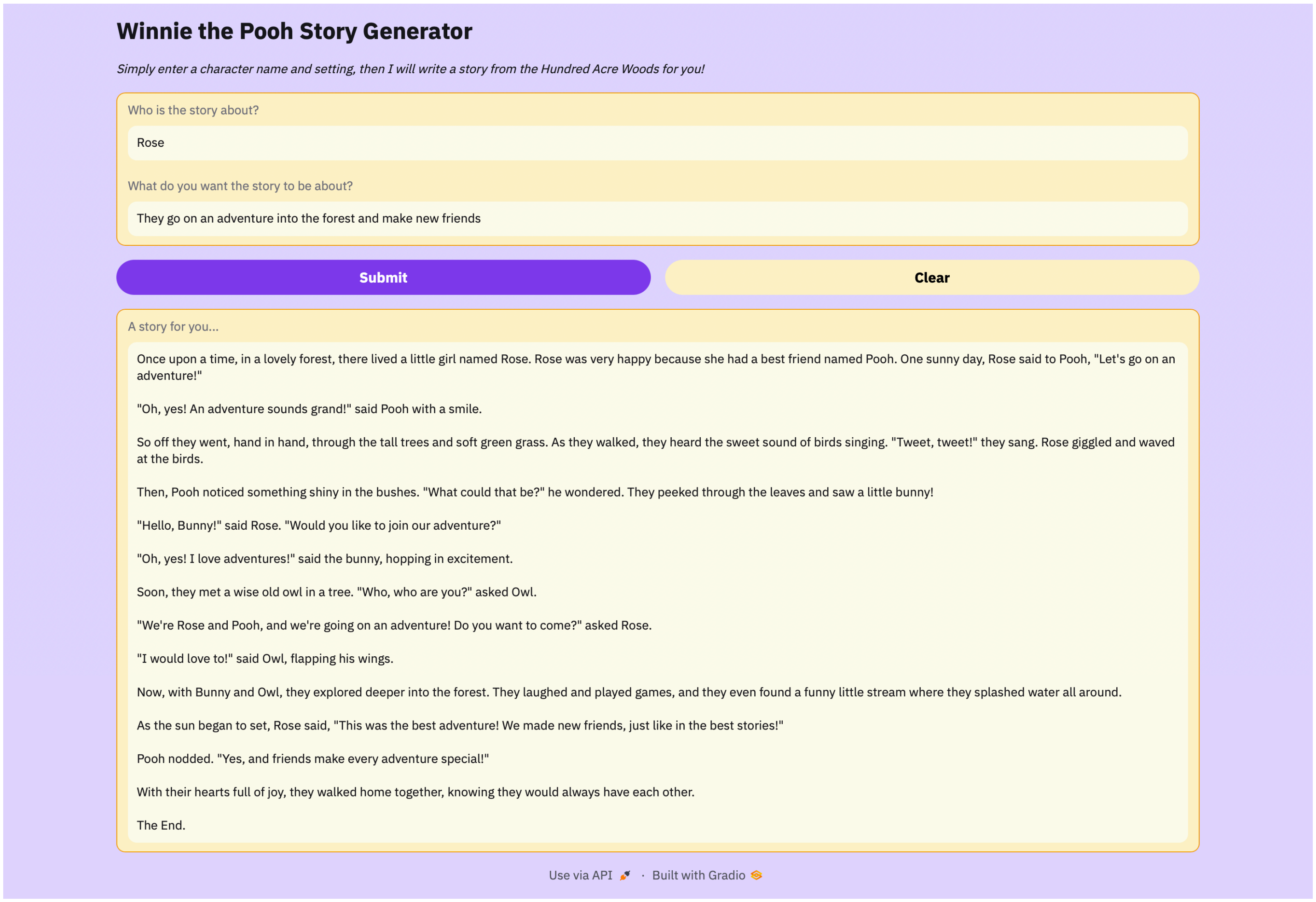
This process creates an interactive system that generates Winnie the Pooh-style stories based on user inputs, while ensuring the readability of the generated content is appropriate for a children's story. The use of RAG allows the system to incorporate elements and style from the original Winnie the Pooh stories, while the LLM enables creative generation of new content.

5. User Interface (Gradio)

Using a customized a Gradio-based user interface:

- Users enters a character name and a story prompt

- The interface has a submit button to generate the story and a clear button to reset inputs
- If submitted, the UI calls ***print_story*** with the provided inputs
- The generated story output is returned and presented in an output text box.



Winnie the Pooh Story Generator

Simply enter a character name and setting, then I will write a story from the Hundred Acre Woods for you!

Who is the story about?

Rose

What do you want the story to be about?

They go on an adventure into the forest and make new friends

A story for you...

Once upon a time, in a lovely forest, there lived a little girl named Rose. Rose was very happy because she had a best friend named Pooh. One sunny day, Rose said to Pooh, "Let's go on an adventure!"

"Oh, yes! An adventure sounds grand!" said Pooh with a smile.

So off they went, hand in hand, through the tall trees and soft green grass. As they walked, they heard the sweet sound of birds singing. "Tweet, tweet!" they sang. Rose giggled and waved at the birds.

Then, Pooh noticed something shiny in the bushes. "What could that be?" he wondered. They peeked through the leaves and saw a little bunny!

"Hello, Bunny!" said Rose. "Would you like to join our adventure?"

"Oh, yes! I love adventures!" said the bunny, hopping in excitement.

Soon, they met a wise old owl in a tree. "Who, who are you?" asked Owl.

"We're Rose and Pooh, and we're going on an adventure! Do you want to come?" asked Rose.

"I would love to!" said Owl, flapping his wings.

Now, with Bunny and Owl, they explored deeper into the forest. They laughed and played games, and they even found a funny little stream where they splashed water all around.

As the sun began to set, Rose said, "This was the best adventure! We made new friends, just like in the best stories!"

Pooh nodded. "Yes, and friends make every adventure special!"

With their hearts full of joy, they walked home together, knowing they would always have each other.

The End.

Future Development

There are two areas in which I would like to further develop and expand:

1. Explore performance of different LLMs.
   My selection of the GPT-4o mini LLM was driven by the balance of its cost-effectiveness and high performance. However, I would be interested in comparing how other LLM performances compared in terms of latency, noticeable differences in output, etc.

2. Add more variation to children's stories to the database.
   Increasing the scope beyond Winnie the Pooh would allow for a broader use and could be expanded to different age groups. Additional genres could be fairy tales, fables, and other classic story collections.