Excel@FIT 2025
http://excel.fit.vutbr.cz

# AI-generated fairy tales

Barbora Šmondrková*

**Abstract**

The aim of this work is to simplify the creation of personalized fairy tales using artificial intelligence. Existing tools for story generation often require extensive customization, making them less accessible to younger or less experienced users. This paper presents a modular application that enables users to generate AI-driven fairy tales from a simple keyword or short description, combined with voice cloning for audio output. The system consists of four interconnected modules: client, server, text-to-speech, and language model. The user interface is designed to be intuitive and accessible for all age groups. To assess whether the generated content qualifies as a fairy tale, both human evaluation based on a custom definition and automated metrics were applied. Results show that the system produces coherent, readable texts that align with the characteristics of traditional fairy tales, offering a user-friendly solution for AI storytelling.

*xsmond00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

## 1. Introduction

The goal is to make the creation of fairy tales customizable, simple and enjoyable. Traditional storytelling relies on time and skill in writing, while the proposed system empowers anyone to create custom stories in just a couple of steps. By combining AI-generated fairy tales with voice cloning and intuitive controls, the system allows anyone to create personalized fairy tales in both text and audio formats.

The aim for the system is to be suitable for all age groups, including children. It could also give an idea of what AI is capable of even to a younger audience. The system allows for exploring the creative possibilities of AI in fairy tale generation.

Existing solutions like the *Fairytale Generator* [1] are skilled at creating personalized fairy tales, but they involve complicated and in-depth customization. The process can be time-consuming, as users must tailor each aspect of the story, including settings, characters, genres, and more. While they now also offer narration in a selected voice, they don't support the upload of personal voices.

My solution aims to simplify the customization process, making it easy enough for children to use without becoming overwhelmed by unnecessary options. Instead of requiring extensive input, my application generates fairy tales based on just a single word, a few keywords, or a short description. Additionally, users will have the option to choose from a selection of prepared voices. They will also be able to upload a voice sample or record their own voice, allowing them to narrate the story in any voice or character they choose. The solution also explores possible methods to ensure the generated text meets the characteristics of a true fairy tale.

## 2. Solution

The system is a modular web application where all modules operate independently while communicating with each other.

### 2.1 Architecture

The application is built from four modules: client, server, text-to-speech (TTS) and large language model (LLM) Figure 4.

Client (the frontend) is developed using React and provides a simple and intuitive user interface. It serves as the entry point for users interacting with the application. Server (the backend) is a Node.js Express server that acts as the central coordinator between the client and the services. All business logic, routing, and API handling are handled within this module. The TTS and LLM are treated as services. The LLM service relies on an external LLM API provider, such as the OpenAI API [2], and generates story text based on the user input included in a structured prompt. The TTS service runs on a Flask server using Python, loads the XTTSv2 model [3], and generates speech from text using either a selected speaker or a provided voice sample.

### 2.2 Generation process

As mentioned, the important part of this system is the ability to customize the output to the user's preferences. Text can be customized by providing keywords or a short description that can be imported with the component Figure 1. The user can choose three keywords, and each emoji corresponds to one keyword, such as the duck emoji represents the word duck. This component was designed with kids in mind and to simplify the process of creating an idea for a fairy tale. Even adults can prefer this component as it does not require coming up with their own idea and simply choosing from what is prepared. The user can also import a description using classic text input.

Audio can be customized by selecting from predefined speakers, uploading a voice sample, or recording a voice. For kids to make it more interesting, there are voices inspired by popular characters to make the narrative more exciting Figure 2. In this case, voice samples are saved within the browser, and voice cloning creates the narration. The same goes for the user's voice samples or their own voice recording. The user can also choose from predefined speakers provided by the used TTS system.

Once the user customization is provided, the story creation begins Figure 3. The LLM creates a story text that can be read while the TTS system generates the audio. Once completed, the finished audio is available for playback with player controls.

### 2.3 Generation approaches

Two approaches are used to generate the text, one focusing on the use of only one single prompt (single-prompt) and the other dividing the generation into multiple smaller tasks (subtasks) Figure 5.

A single-prompt approach collects the user input (keywords or short description) and adds it to a prepared prompt template. This prompt template consists of a task specification, a fairy tale definition, and guidelines for generation. This prompt template is then provided to the LLM for generation.

The subtasks approach, in the end, uses the same prompt as the single-prompt approach, but before this final full story prompt, it generates story-specific attributes that are then connected to the final prompt. It generates space, character, event, and plot, each on its own, and always passes the output from the prompt as input to the next one, along with user input.

## 3. Evaluation

The problem with fairy tales is that everyone can imagine something different under this term. For example, some people require magic in every fairy tale, while others do not. For this reason, I created a definition of fairy tale based on which the story is generated and evaluated.

The generated text can be evaluated by the mentioned definition, where humans are the judges. Based on this definition, they evaluate the text and consider if the generated text passes the provided definition Figure 6. A checklist of attributes is derived from this definition to simplify the evaluation. Additionally, they evaluate preferences for the text generated by different approaches (single-prompt or subtasks).

Another approach is to use automatic evaluation metrics, such as BERTScore or Readability. BERTScore compares two texts based on their semantic similarity [4]. Readability evaluates how easily a text can be understood by sentence length and the number of syllables in words [5]. I used these metrics on a 100 classic fairy tales from the Fairy Tale dataset acquired from Hugging Face to determine the typical values for fairy tales. I compared each fairytale to every other in the dataset, and then, with statistics methods, IQR or mean and standard deviation [6, 7], defined the ranges. These ranges are considered the typical fairy tale values in my work Figure 7.

The generated text is evaluated using the same metrics as the 100 fairy tales. The metrics are then averaged and compared to see if they fall within the typical ranges. If the scores of the generated text fall within these ranges, it is possible to determine that the generated text shares a sentence structure and vocabulary similar to fairy tales in the dataset.

## 4. Conclusions

I believe I have developed a user-friendly application that simplifies the generation process for all ages. I identified a suitable text evaluation method and explored various approaches to AI fairy tale creation. In the next steps, I aim to improve voice cloning and make the generation even more interesting while keeping it simple.

## References

[1] Fairytale generator – create your own ai-powered fairytales. online, 2024. https://fairytalegenerator.com/.

[2] OpenAI. Openai api. online, 2024. https://openai.com/index/openai-api/.

[3] CoquiAI. Xtts-v2. online, 2024. https://huggingface.co/coqui/XTTS-v2.

[4] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*, 2020.

[5] William H DuBay. The principles of readability. *Online submission*, 2004. ERIC Document ED490073.

[6] Songwon Seo. A review and comparison of methods for detecting outliers in univariate data sets. Master's thesis, University of Pittsburgh, Graduate School of Public Health, 2006.

[7] Marcin Rutecki. Outlier detection methods. online, 2022. https://www.kaggle.com/code/marcinrutecki/outlier-detection-methods.