




When Content Speaks Volumes: Podcastfy — An Open Source Python Package Bridging Multimodal Data and Conversational Audio with GenAI

Tharsis T. P. Souza ^{1,2}

¹ Columbia University in the City of New York ² Instituto Federal de Educacao, Ciencia e Tecnologia do Sul de Minas (IFSULDEMINAS)

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Abstract

Podcastfy is an open-source Python framework that programmatically transforms multisourced, multimodal content into multilingual, natural-sounding audio conversations using generative AI. By converting various types of digital content - including images, websites, YouTube videos, and PDFs - into conversational audio formats, Podcastfy enhances accessibility, engagement, and usability for a wide range of users. As an open-source project, Podcastfy benefits from continuous community-driven improvements, enhancing its adaptability to evolving user requirements and accessibility standards.

Statement of Need

The rapid expansion of digital content across various formats has intensified the need for tools capable of converting diverse information into accessible and digestible forms ([Chen & Wu, 2023](#); [Johnson & Smith, 2023](#); [McCune & Brown, 2023](#)). Existing solutions often fall short due to their proprietary nature, limited multimodal support, or inadequate accessibility features ([Gupta & Lee, 2023](#); [Marcus & Zhang, 2019](#); [Peterson & Allen, 2023](#)).

Podcastfy addresses this gap with an open-source solution that supports multimodal input processing and generates natural-sounding, summarized conversational content. Leveraging advances in large language models (LLMs) and text-to-speech (TTS) synthesis, Podcastfy aims to benefit a diverse group of users — including content creators, educators, researchers, and accessibility advocates — by providing a customizable solution that transforms digital content into multilingual textual and auditory formats, enhancing accessibility and engagement.

Features

- Generate conversational content from multiple sources and formats (images, websites, YouTube, and PDFs).
- Customize transcript and audio generation (e.g., style, language, structure, length).
- Create podcasts from pre-existing or edited transcripts.
- Leverage cloud-based and local LLMs for transcript generation (increased privacy and control).
- Integrate with advanced text-to-speech models (OpenAI, ElevenLabs, and Microsoft Edge).
- Provide multi-language support for global content creation and enhanced accessibility.
- Integrate seamlessly with CLI and Python packages for automated workflows.

See [audio samples](#).

Implementation and Architecture

Podcastfy implements a modular architecture designed for flexibility and extensibility through five main components, as shown in Figure 1.

1. Client Interface

- Provides both CLI (Command-Line Interface) and API interfaces.
- Coordinates the workflow between processing layers.
- Implements a unified interface for podcast generation through the `generate_podcast()` method.

2. Configuration Management

- Offers extensive customization options through a dedicated module.
- Manages system settings and user preferences, such as podcast name, language, style, and structure.
- Controls the behavior of all processing layers.

3. Content Extraction Layer

- Extracts content from various sources, including websites, PDFs, and YouTube videos.
- The `ContentExtractor` class coordinates three specialized extractors:
 - `PDFExtractor`: Handles PDF document processing.
 - `WebsiteExtractor`: Manages website content extraction.
 - `YouTubeTranscriber`: Processes YouTube video content.
- Serves as the entry point for all input types, providing standardized text output to the transcript generator.

4. LLM-based Transcript Generation Layer

- Uses large language models to generate natural-sounding conversations from extracted content.
- The `ContentGenerator` class manages conversation generation using different LLM backends:
 - Integrates with `LangChain` to implement prompt management and common LLM access through the `BaseChatModel` interface.
 - Supports both local (`LlamaFile`) and cloud-based models.
 - Uses `ChatGoogleGenerativeAI` for cloud-based LLM services.
- Allows customization of conversation style, roles, and dialogue structure.
- Outputs structured conversations in text format.

5. Text-to-Speech (TTS) Layer

- Converts input transcripts into audio using various TTS models.
- The `TextToSpeech` class implements a factory pattern:
 - The `TTSFactory` creates appropriate providers based on configuration.
 - Supports multiple backends (`OpenAI`, `ElevenLabs`, and `Microsoft Edge`) through the `TTSPProvider` interface.
- Produces the final podcast audio output.

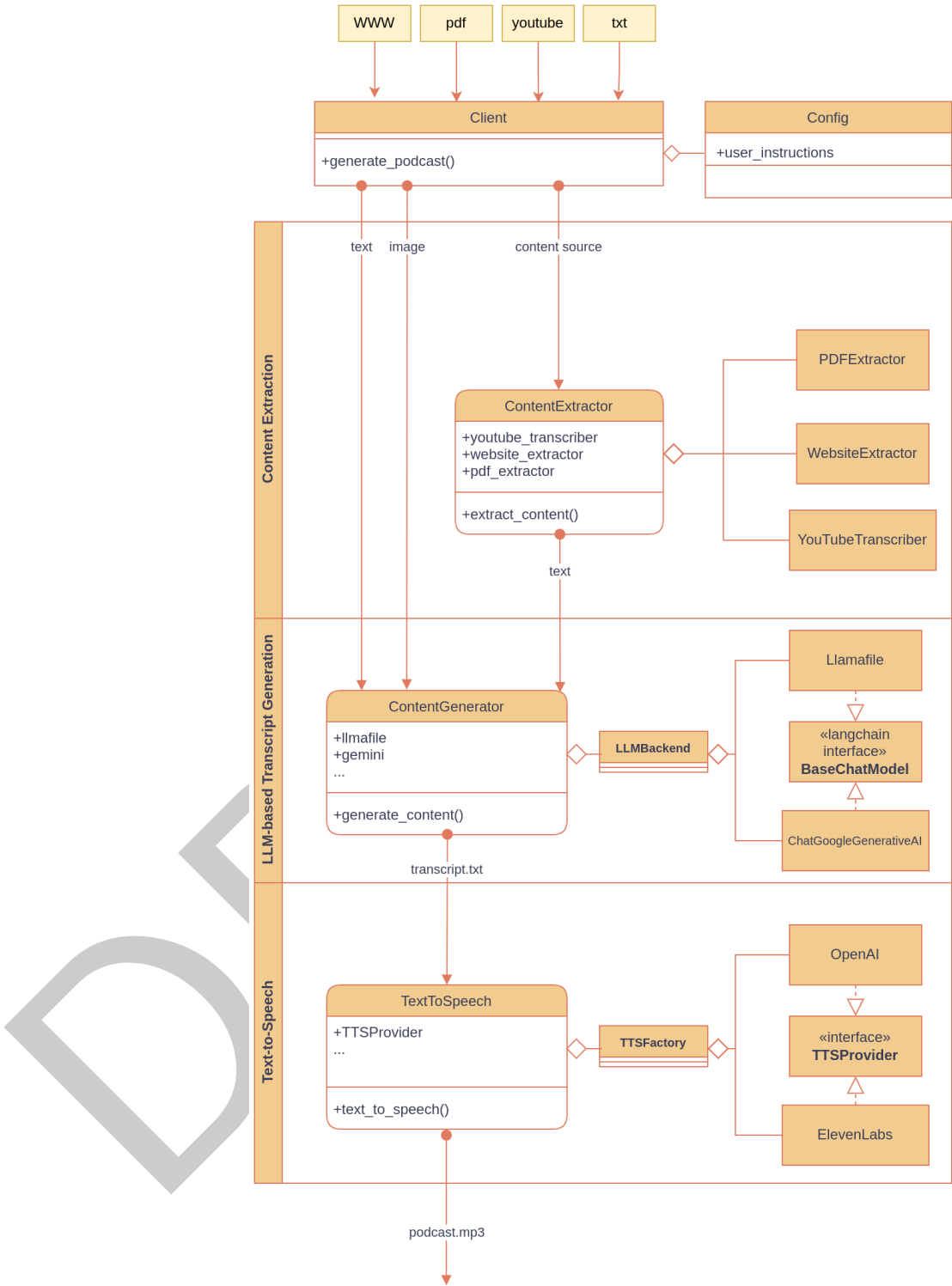


Figure 1: Podcastfy's simplified architecture and workflow diagram showing the main components and their interactions.

79 The modular architecture enables independent development and maintenance of each compo-
80 nent. This pipeline design ensures a clean separation of concerns while maintaining seamless
81 data transformation between stages. This modular approach also facilitates easy updates and

82 extensions to individual components without affecting the rest of the system.

83 The framework is offered as a Python package, with a command-line interface as well as a

84 REST API, making it accessible to users with different technical backgrounds and requirements.

85 Quick Start

86 Prerequisites

- 87 ▪ Python 3.11 or higher
- 88 ▪ `$ pip install ffmpeg` (for audio processing)

89 Setup

- 90 1. Install from PyPI `$ pip install podcastfy`
- 91 2. Set up [API keys](#)

92 Python

```
from podcastfy.client import generate_podcast

audio_file = generate_podcast(urls=["<url1>", "<url2>"])
```

93 CLI

```
94 python -m podcastfy.client --url <url1> --url <url2>
```

95 Customization Examples

96 Podcastfy offers various customization options that make it versatile for different types of

97 content transformation. To accomplish that, we leverage LangChain's ([LangChain, 2024](#))

98 prompt management capabilities to dynamically construct prompts for the LLM, adjusting

99 conversation characteristics such as style, roles, and dialogue structure. Below are some

100 examples that demonstrate its capabilities.

101 Academic Debate

102 The following Python code demonstrates how to configure Podcastfy for an academic debate:

```
from podcastfy import generate_podcast

debate_config = {
    "conversation_style": ["formal", "debate"],
    "roles_person1": "main presenter",
    "roles_person2": "opposing viewpoint",
    "dialogue_structure": ["Introduction", "Argument Presentation", "Counterarguments"],
}

generate_podcast(
    urls=["PATH/T0/academic-article.pdf"],
    conversation_config=debate_config
)
```

103 In this example, the roles are set to “main presenter” and “opposing viewpoint” to simulate an

104 academic debate between two speakers on a chosen topic. This approach is especially useful

105 for educational content that aims to present multiple perspectives on a topic. The output is
106 structured with clear sections such as introduction, argument presentation, counterarguments,
107 and conclusion, allowing listeners to follow complex ideas easily.

108 Technical Tutorial

109 In this example, the configuration is optimized for creating technical tutorial content.

```
tutorial_config = {  
    "word_count": 2500,  
    "conversation_style": ["instructional", "step-by-step"],  
    "roles_person1": "expert developer",  
    "roles_person2": "learning developer",  
    "dialogue_structure": [  
        "Concept Introduction",  
        "Technical Background",  
        "Implementation Steps",  
        "Common Pitfalls",  
        "Best Practices"  
    ],  
    "engagement_techniques": [  
        "code examples",  
        "real-world applications",  
        "troubleshooting tips"  
    ],  
    "creativity": 0.4  
}  
  
generate_podcast(  
    urls=["https://tech-blog.com/tutorial"],  
    conversation_config=tutorial_config  
)
```

110 The roles are set to “expert developer” and “learning developer” to create a natural teaching
111 dynamic. The dialogue structure follows a logical progression from concept introduction through
112 implementation and best practices. The engagement_techniques parameter ensures the content
113 remains practical and applicable by incorporating code examples, real-world applications, and
114 troubleshooting guidance. A moderate creativity setting (0.4) maintains technical accuracy
115 while allowing for engaging explanations and examples.

116 Limitations

117 Podcastfy faces several key limitations in its current implementation. The accuracy and
118 quality of generated content heavily depends on the underlying LLMs, with complex technical
119 content potentially being misinterpreted. Additionally, while multilingual support is available,
120 performance varies across languages, with less common languages having limited TTS voice
121 options. The framework also relies on third-party APIs which introduces service availability
122 risks, and local LLM options require significant computational resources.

123 These limitations highlight areas for future development and improvement of the framework.
124 Users should carefully consider these constraints when implementing Podcastfy for their
125 specific use cases and requirements.

Conclusion

Podcastfy contributes to multimodal content accessibility by enabling the programmatic transformation of digital content into conversational audio. The framework addresses accessibility needs through automated content summarization and natural-sounding speech synthesis. Its modular design and configurable options allow for flexible content processing and audio generation workflows that can be adapted for different use cases and requirements.

Acknowledgements

We acknowledge the open-source community and the developers of the various libraries and tools that make Podcastfy possible. Special thanks to the developers of LangChain, Llamafile and HuggingFace. We are particularly grateful to all our [contributors](#) who have helped improve this project.

References

- Chen, R., & Wu, Y. (2023). Digital accessibility tools for multimodal content processing: A systematic review. *Digital Transformation Review*, 5(3), 91–109.
- Gupta, S., & Lee, A. (2023). Advances in adaptive user interfaces for enhanced accessibility. *Journal of Accessibility and User Experience*, 14(3), 203–215.
- Johnson, L., & Smith, K. (2023). Adaptive user interfaces for accessibility across digital content modalities. *Journal of Multimodal Accessibility*, 17(2), 43–58. <https://link.springer.com/article/10.1007/s12193-023-00427-4>
- LangChain. (2024). *LangChain: Building applications with LLMs through composability*. <https://www.langchain.com/>
- Marcus, A., & Zhang, T. (2019). Design for multimodal human-computer interaction. In *Lecture notes in computer science* (Vol. 1157, pp. 160–176). https://link.springer.com/chapter/10.1007/978-3-319-52162-6_18
- McCune, B., & Brown, L. (2023). Accessibility of digital information: Standards, frameworks, and tools related to information literacy and information technology. *Journal of Information Literacy and Accessibility*, 9(4), 112–129.
- Peterson, L., & Allen, J. (2023). Web accessibility and multimodal digital engagement. *Technology Accessibility Journal*, 12(1), 23–34.