Towards LLM-based Configuration and Generation of Books

Jovan Mihajlovic[1,*], Alexander Felfernig[1]

[1]Graz University of Technology, Inffeldgasse 16b, Graz, 8010, Austria

Abstract

Large Language Models (LLMs) can support a wide range of content generation tasks. Interaction with LLMs can occur either through user-friendly web interfaces or via provided APIs. Our work focuses on content generation for a specific use case: creating lecture books from recorded lectures. To support this goal, a web application with a simple configuration interface has been developed. Users can include transcripts of recordings, configure options, and generate books through the application. It allows for flexibility by offering different prompts and the ability to select among various LLMs. Initial results demonstrate that LLMs can generate books from the recorded lectures, however, evaluation results show varying output quality depending on the selected configuration.

Keywords

Large Language Models, Configuration, Generation of Books

1. Introduction

Configuration can be regarded as a specialized form of design activity in which the final product is assembled from a predefined set of component types, all of which must comply with a corresponding set of domain-specific constraints [1, 2, 3, 4]. In this paper, we demonstrate how Large Language Models (LLMs) can be leveraged for the automated generation of university lecture books. This generation is based on a collection of video recordings (of lecture units) [5]. Our application enables the configuration of key properties relevant to book generation and automatically produces a draft book proposal from the transcripts of these lectures. The major motivation for our work is to exploit simple ways to provide students with additional learning contents that help to improve the overall learning experience and to accelerate learning. The focus of our work is to apply Large Language Models (LLMs) [6] to generate book contents based on LLM prompts which are themselves generated on the basis of configured book and generation process properties.

There are multiple ways to interact with LLMs. First, queries can be defined in user interfaces of LLM providers. Second, models can be accessed via APIs offered by these providers, allowing the development of customized applications tailored to specific tasks such as generating book content from lecture transcripts. Our book generator application serves as an intermediary between the LLM and the user – it facilitates the process of transforming lecture transcripts into book content. It simplifies the interaction with LLMs by abstracting the underlying complexity of prompt engineering. Generating structured book content involves first establishing context and defining the generation goal, then supplying the transcript text along with relevant options and constraints. While crafting a prompt manually might be effective for single-use scenarios, repeating this process for multiple transcripts is labor-intensive. Our system addresses this by using pre-defined prompt templates.

Due to their generative nature, large language models can be applied in various generation-related tasks. Related examples in the configuration context are the generation of explanations [7] (e.g., sustainability-aware explanations nudging configurator users towards more sustainable consumption patterns), the generation of configuration knowledge bases [8] (which helps to reduce efforts in the

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy

*Corresponding author.

jovan.mihajlovic@student.tugraz.at (J. Mihajlovic); alexander.felfernig@ist.tugraz.at (A. Felfernig)

https://ase.sai.tugraz.at/ (A. Felfernig)

0000-0003-0108-3146 (A. Felfernig)

context of configuration knowledge base development and maintenance), and interactive configuration [9] where LLMs can be applied to support interactive chat-based interfaces that support product and service configuration by allowing users to describe their requirements/preferences in natural language (and to generate a set of solver-understandable preferences thereof). In contrast to existing work in the context of combining LLMs with configuration technologies, the work presented in this paper focuses on the generation of artifacts (in our case, books) on the basis of pre-configured parameters representing intended book properties and also technical properties relevant for the book generation phase.

The major contribution of this paper is to present an initial idea of a book generation interface based on configured parameter settings. Furthermore, we summarize initial insights from the analysis of a first prototype implementation which is currently not available for productive use.

The remainder of this paper is organized as follows. Section 2 explains how the implemented web application automates the book generation workflow. Section 3 discusses the outcomes obtained using real-world lecture recordings. Finally, Sections 4 and 5 outline directions for future development and summarize our findings.

2. LLM-based Book Generator

The book generator application is structured into a backend, implemented using the NestJS framework[1] and a frontend which uses the React[2] library. It generates LLM prompts from input data (e.g., transcripts from lecture videos) and selected (configured) options. The generated prompts are forwarded to the LLM via API. The input data consists of the following elements:

- one or more lecture video transcript files (textual)
- *(optional)* a LaTeX template file as basis for guiding the LLM-based book generation (in LaTeX format).
- *(optional)* parameters (which prompt to use, book title, chapter size and other parameters)

The user gets a result either as a single LaTeX file, or a ZIP file containing multiple output files. There are three "pipelines" available for the user to choose, these are described in the following. Each pipeline presents one or more prompts used to generate a book from the input data (see Table 1).

Table 1
LLM pipelines available in the current book generator prototype where *name* denotes the name of the pipeline, *input* denotes the number of transcript files processes at a time (n represents the number of input transcripts), *parameters* indicates whether configuration parameters can be specified within this selected pipeline, *concepts* indicates whether LLM-identified key concepts should be used for organizing individual sections, and *#chap* indicates whether one or more chapters are generated based on the selected pipeline and the provided further input.

name	input	parameters	concepts	#chap
T2SC	n	no	no	n
C2C	n	no	yes	#concepts
C2CO	n	yes	yes	#concepts

2.1. Transcript to Single Chapter (T2SC)

This pipeline focuses on generating a single LaTeX book chapter for each provided transcript file, which is derived from the video of an individual lecture unit. The LLM is instructed to create one cohesive chapter. Structuring elements to be used (e.g., sections and subsections) are enumerated explicitly. To ensure consistency, each chapter is generated based on the same prompt template.

[1] https://nestjs.com, accessed 16-June-2025
[2] https://react.dev, accessed 16-June-2025

The primary focus of each chapter is to explain the core concepts of the corresponding lecture unit in detail. *Examples* can be included if they help in clarifying the discussed concepts. The style in which examples are presented is not strictly specified. At the end of each chapter, self-evaluation questions are included to encourage further learning. In the current version of the book generator, the LLM is explicitly instructed to generate all book contents in English. To assure consistency in formatting, each generated chapter is generated on the basis of a basic LaTeX template.

The following is the full prompt (transcript contents are formatted/represented as "[...]"):

> An audio transcript of a lecture is provided within the quotes at the end of this message. Generate a LaTeX chapter for this lecture. The chapter should start with \chapter (you can use sections, subsections and everything else provided in standard LaTeX). Focus on explaining the main concepts of the lecture. Use examples if they can help in explaining. Add questions for self preparation at the end of the chapter. Include some concepts which are related, but not mentioned in the transcript. The output language should be English. The transcript is: [...]

2.2. Concepts to Chapters (C2C)

Compared to T2SC, the C2C pipeline includes an additional step for generating chapters. Rather than creating individual chapters directly from each transcript derived from a lecture unit video, the LLM is first asked to identify the key concepts within the transcripts. Once the key concepts are identified, the LLM is instructed to enhance the presentation of the content using lists, text styling, structural elements, and similar formatting techniques. Each concept should be clearly explained to the reader, with related concepts further elaborated upon in corresponding subsections. As in T2SC, the LLM is also directed to include relevant examples to aid in the explanation.

The following is the prompt, where CONCEPT_NAME corresponds to the target concept the prompt is applied to:

> Within the context of a Computer Science lecture called "Software development processes", a concept called "CONCEPT_NAME" is present. Write a LaTeX chapter about that concept. The output language should be English.
> [list of detailed instructions]

2.3. C2C with Options (C2CO)

The main difference between C2CO and the previous pipelines is the provision of options (parameters) that help to further configure the pipeline – see a corresponding example user interface in Figure 1. These options can be used to further tailor the generated contents.

The idea of the C2CO pipeline is the same as C2C, however, the used prompts are different (as a direct consequence of the provided options). The probably most influential option is to allow the usage of a customized LaTeX template (see Section 3). Selected options are represented as a list of rules part of the prompt. For example, the first rule to follow when generating a chapter is the following:

> "the chapter starts with introductory paragraph which explains the concept concisely"

Irrelevant elements in provided custom templates (e.g., texts and bibliographic entries from other articles written on the basis of this template) are first removed by the LLM. The purpose of this step is to ease further use of the template by the LLM (e.g., texts from other papers have the risk to confuse the LLM and let the LLM integrate related contents into the generated book). The remaining steps are quite similar to the C2C pipeline – the major difference are the inserted rules which are defined by the user via book generator user interface.

The used prompt is the following:

Your task is to generate a LaTeX chapter for the following concept (within the context of a Computer Science lecture called "Software development processes"): "CONCEPT_NAME"
You should follow these rules: [a list of rules like the one stated previously]

The returned content should be a LaTeX \chapter which can be included in a template.

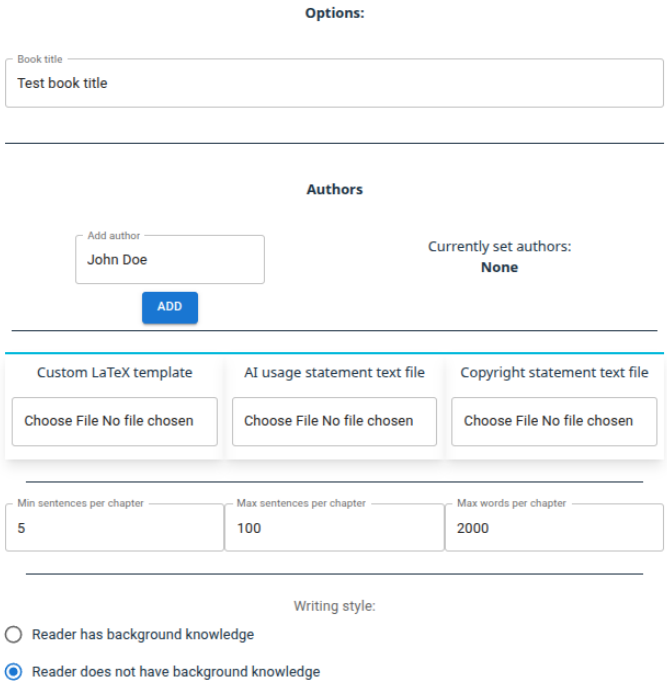


Figure 1: Options provided by the book generator UI (C2CO). The book title and a list of authors can be defined. Different files like a custom LaTeX template, AI usage statement and copyright statement can be uploaded. Lastly, constraints for the length of each chapter can be set, as well as a choice between two writing styles.

3. Preliminary Evaluation

The following subsections discuss first results of applying our LLM-based approach to video transcripts generated from a lecture on the topic of *software development processes*. In this context, each individual subsection discusses the observed results of applying an individual pipeline. In its current version, our book generator application lacks automated quality assurance mechanisms. The generation of the transcripts was performed on the basis of the OpenAI Whisper model.[3]

Our initial evaluation of the LLM-generated outputs (books) has been performed manually and the corresponding results (feedback of two persons) are presented in an aggregated fashion in the following paragraphs. For evaluation purposes, the generation focused on an individual lecture unit related to different techniques in the context of the topic of *software requirements prioritization* including subtopics such as *release planning* and *minimum viable products*.

An overview of the different applied LLMs is provided in Table 2.

Table 2
Large Language Models (LLMs) used by the book generator application for content generation purposes.

Name	link
gemini-2.0-flash	https://cloud.google.com/vertex-ai/generative-ai
lama-3.3-70b-versatile	https://www.llama.com

[3]https://huggingface.co/openai/whisper-large-v3

In the following, we summarize the initial evaluation results of the LLM-generated outputs on the basis of following four evaluation dimensions:

- **Understandability** of the content
- **Completeness**: degree to which transcript contents are covered
- **Example quality**: quality of the included (generated) examples
- **Additional content**: presence of additional information related to transcript contents, but not contained in the transcript

3.1. T2SC Pipeline

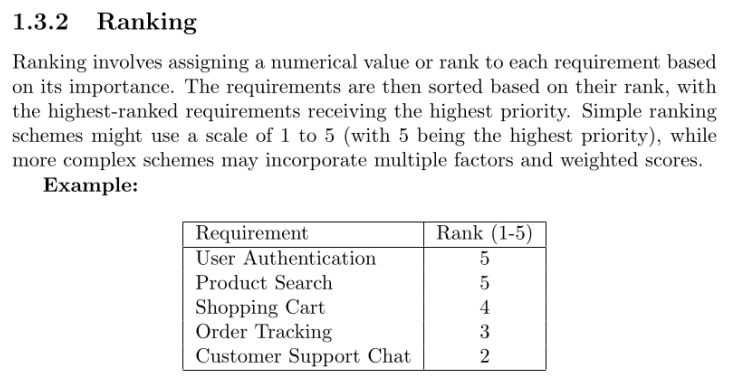
gemini-2.0-flash The content is clear and easy to understand, and it is provided in the requested language. The coverage of the transcript content is comprehensive, including a thorough introduction and an in-depth classification of *requirement prioritization approaches* (this is the lecture topic). The examples provided are generally of high quality, with a few that could benefit from improvements in readability and formatting. Compared to Llama, the quality and quantity of examples are significantly better. The subsection on additional content lists seven related concepts, each accompanied by a concise description. However, there is a need for better structure, as the concepts are currently presented in a single paragraph without clear separation.

llama-3.3-70b-versatile The content is generated in German, despite the instruction specifying English. Additionally, some sections are a bit unclear—specifically, the subsections on "Basic Release Planning" and "Integrated Release Planning" both contain identical sentences. Only the latter mentions additional factors, which suggests there is some intended difference between the two but fails to clarify it adequately. The overall completeness is in the lower-medium range. Key concepts like "Minimum Viable Product," which are included in the other model, are missing. The chapter is concise, fitting onto just two pages, including the self-preparation questions, but lacks depth in certain areas. The examples provided are of low quality, with only one example given in two short sentences, which fails to adequately illustrate the concept. This model performed poorly in terms of additional content, as it did not provide any supplementary material in this area.

3.2. C2C Pipeline

gemini-2.0-flash The C2C pipeline generates multiple chapters along with corresponding sections and subsections, for which a table of contents can be generated. The content is generally clear and well-presented. There are some formatting issues with the questions that could hinder readability, but the questions themselves are understandable.. The transcript is largely covered, but some concepts, such as basic/integrated release planning (despite the presence of a "Release Planning" chapter) and utility-based prioritization, are notably missing. The examples, primarily presented in table form, are easy to understand. A snippet of a subsection with an example is shown in Figure 2. However, many of other example tables overflow the page, which can make them difficult to read. The additional content is excellent, with well-organized per-chapter sections and separate chapters that enrich the material.

llama-3.3-70b-versatile The content is clear and easy to understand, with no notable issues. The transcript is well-covered, though the "Release Planning" chapter lacks references to basic/integrated release planning, similar to the Gemini model. On the positive side, it does include a chapter on utility-based prioritization, which the other model does not. The quality of examples is mixed: the textual example in Chapter 2 is strong, while the one in Chapter 3 is quite brief and doesn't fully explore the concept. The remaining examples are decent, though it's difficult to assess them fully due to tables spilling out of the page, making the examples incomplete or hard to view. The additional content has a satisfactory level of depth and organization.



1.3.2 Ranking

Ranking involves assigning a numerical value or rank to each requirement based on its importance. The requirements are then sorted based on their rank, with the highest-ranked requirements receiving the highest priority. Simple ranking schemes might use a scale of 1 to 5 (with 5 being the highest priority), while more complex schemes may incorporate multiple factors and weighted scores.

Example:

Requirement	Rank (1-5)
User Authentication	5
Product Search	5
Shopping Cart	4
Order Tracking	3
Customer Support Chat	2

Figure 2: A subsection containing text which explains how requirements can be ranked.

3.3. C2CO Pipeline

gemini-2.0-flash For the chosen template used in evaluation, the title page clearly presents the title of the book in bold text, with authors directly below. The header on the page following the title page suffers from broken formatting, though it remains legible with some effort. Otherwise, the content is generally easy to understand, though there are a few areas where improved formatting could enhance clarity. The completeness is similar to the C2C case, as the approach is quite comparable, with only minor differences in the prompts used for generating the concept chapter. There are numerous examples presented in table format, all of good quality. Some tables extend beyond the page boundaries so significant parts of examples are lost. Each chapter references related concepts for additional content, but the formatting is inconsistent. In some instances, these concepts are listed clearly, while in others, they are presented continuously in a single paragraph. The prompt could benefit from a more consistent structure for how additional content should be organized.

llama-3.3-70b-versatile The content is generally clear and easy to follow, though there are some issues with tables extending beyond the page, similar to the issues found in the Gemini case. The level of completeness is comparable to the C2C case for the same reasons mentioned in the previous section. The quality of examples is similar to that of Gemini: some tables are well-structured and present clear values, while others overflow the page. One example is supposed to demonstrate how to calculate opportunity costs using the Kano model (as indicated by its title), but it only provides the final values, without showing the calculations involved. Additional content is provided, but it appears somewhat brief and/or poorly formatted, which could benefit from further refinement.

Table 3
Evaluation of the generated book chapters on the basis of the evaluation dimensions *understandability*, *completeness*, *example quality*, and *additional content* with a rating scale (1 (poor) .. 5 (excellent). Evaluated LLMs: gemini-2.0-flash (GF) and lama-3.3-70b-versatile (LV).

LLM	understandability			completeness			example quality			additional content		
	T2SC	C2C	C2CO	T2SC	C2C	C2CO	T2SC	C2C	C2CO	T2SC	C2C	C2CO
GF	4.5	4.5	4	4.5	3.5	4	4	4	4	4.5	4.5	4
LV	3	4.5	4	3	3.5	4	2	3	3	1	4.5	3

Initial Summary The Gemini model shows most consistent performance across different dimensions and pipelines. The approach used by C2C/C2CO pipelines seems to be the right direction, as it allows more control over structure and length of generated content. Based on our evaluation provided in Table 3, the two analyzed LLMs show major differences in terms of the evaluation dimensions *example quality* and *additional content*.

4. Future Work

The current version of this application provides a solid foundation with consistent results, but there is considerable room for improvement. The next logical step is to evolve the content generation process into a more structured, configurable approach that allows for step-by-step refinement.

By adding more customizable options, we can address a broader range of users. For instance, default settings would meet the needs of those seeking quick, approximate results, while users who require greater control can directly adjust specific parameters. Introducing more interactivity between the user and the model would provide even finer control over the generated content.

One approach could involve having the model analyze the input data without producing results immediately. Instead, it could prompt the user for additional options or guidance before proceeding, in contrast to the current method where the user interacts only once, at the data input stage.

Another crucial feature currently missing is quality assurance. At present, results are generated by the model and returned directly to the user without any validation. To address this, we will introduce a quality-check mechanism either at the end of the process, between various steps, or ideally, both. These quality checks will provide valuable feedback to users, allowing them to review suggested improvements or request more detailed revisions.

An important open issue in this context is also to include mechanisms that help to include information about information sources used by the LLM – primarily for the purpose of preventing violations of copyrights and similar issues. Also this aspect has not been taken into account in the current version of the book generator which is also the reason why the system currently is not applied in productive use.

Finally, for future versions, we also plan to include the possibility of defining seed knowledge, i.e., already proposing a basic structure of the book which is then enriched by the LLM.

By incorporating these changes, we hope to not only enhance the content generation process but also to foster greater user involvement and satisfaction.

5. Conclusion

The presented application demonstrates the potentials of LLMs in supporting the automated generation of books where different LLMs and prompt configurations produce varying outcomes. This sets the stage for further development to explore the extent to which these results can be refined. The current approach leaves room for enhancements, particularly with new features added, opportunities to improve both, functionality and quality, are given.

Declaration on Generative AI

The authors used ChatGPT-4o[4] for grammar checking, spellchecking, and improving the formulation of the text. All AI-generated suggestions were carefully reviewed and edited by the authors, who take full responsibility for the content of this publication.

References

[1] D. Sabin, R. Weigel, Product configuration frameworks-a survey, IEEE Intelligent Systems 13 (1998) 42–49. doi:10.1109/5254.708432.

[2] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration: From Research to Business Cases, 1 ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[3] A. Felfernig, A. Falkner, D. Benavides, Feature Models – AI Driven Design, Analysis, and Applications, Springer, Cham, 2024.

[4] https://chatgpt.com

[4] A. Popescu, S. Polat-Erdeniz, A. Felfernig, M. Uta, M. Atas, V.-M. Le, K. Pilsl, M. Enzelsberger, T. N. T. Tran, An overview of machine learning techniques in constraint solving, J Intell Inf Syst 58 (2022) 91–118. doi:10.1007/s10844-021-00666-5.

[5] S. Lubos, A. Felfernig, D. Garber, V.-M. Le, M. Henrich, R. Willfort, J. Fuchs, Towards Group Decision Support with LLM-based Meeting Analysis, in: 33rd ACM Conference on User Modeling, Adaptation and Personalization, UMAP Adjunct '25, ACM, New York, NY, USA, 2025, pp. 331–335. URL: https://doi.org/10.1145/3708319.3733646. doi:10.1145/3708319.3733646.

[6] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, X. Hu, Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond, ACM Trans. Knowl. Discov. Data 18 (2024). doi:10.1145/3649506.

[7] S. Lubos, A. Felfernig, L. Hotz, T. N. T. Tran, S. P. Erdeniz, V. Le, D. Garber, M. E. Mansi, Responsible Configuration Using LLM-based Sustainability-Aware Explanations, in: É. Vareilles, C. Grosso, J. M. Horcas, A. Felfernig (Eds.), 26th International Workshop on Configuration (ConfWS 2024), CEUR-WS.org, 2024, pp. 68–73.

[8] L. Hotz, C. Bähnisch, S. Lubos, A. Felfernig, A. Haag, J. Twiefel, Exploiting Large Language Models for the Automated Generation of Constraint Satisfaction Problems, in: ConfWS'24, volume 3812, CEUR, 2024, pp. 91–100.

[9] P. Kogler, W. Chen, A. Falkner, A. Haselboeck, S. Wallner, Configuration Copilot: Towards Integrating Large Language Models and Constraints, in: ConfWS'24, volume 3812, CEUR, 2024, pp. 101–110.