Applying Retrieval-Augmented Generation for Academic Discipline Development: Insights from Zero-Shot to Tree-of-Thought Prompting

Polina Shnaider, Anastasiia Chernysheva, Anton Govorov, Maksim Khlopotov, Anna Nikiforova
ITMO University
Saint Petersburg, Russia
polina.in.tech@gmail.com

***Abstract*— This study assesses the efficiency of large language models (LLMs) in generating university course structures, comparing traditional methods with Retrieval-Augmented Generation (RAG). It involves a comparative analysis across twelve courses using four LLMs: starling-lm-7b-alpha, openchat_3.5, saiga2_13b, and gpt-3.5-turbo, with four distinct prompting approaches. Findings indicate that advanced prompting techniques significantly influence model performance and response variability. The study underscores the importance of selecting appropriate LLMs and prompting strategies to optimize educational outcomes, highlighting RAG's role in enhancing data retrieval accuracy in educational technology.**

I. Introduction

Large language models (LLMs) significantly impact natural language processing and machine learning, offering capabilities that range from writing coherent articles to engaging in complex dialogues. In education, LLMs enhance learning by supporting personalized and multilingual education, thereby increasing accessibility and inclusivity. They adapt content to diverse learning styles and needs, effectively bridging knowledge gaps and enriching learner engagement [1]

Additionally, LLMs help automate educational content creation, aligning with curricular goals and reducing educators' workloads, thereby boosting efficiency in educational administration [2, 3]. Recent progress in generative AI and large language models has propelled numerous innovations in educational technology aimed at automating the tedious tasks of creating and analyzing text-based content, such as formulating open-ended questions and evaluating responses from student feedback surveys [4], [5].

According to recent studies, such LLMs as BERT, GPT-2, GPT-3, OpenAI's Codex, and T5 were used in automating various educational tasks [1]. BERT variants are extensively utilized for tasks like content profiling and labeling, while GPT models excel in content generation, such as creating math problems or evaluating student responses [6]. Despite their capabilities, the adoption of newer models like GPT-3 is limited due to the high costs associated with their commercial use, reflecting a broader trend in educational technology where the potential of LLMs to support teaching, learning, and administrative processes is balanced against the challenges of implementation and the need for model fine-tuning [7].

The present paper explores the transformative potential of large language models in designing university course structures. This study employs the Retrieval-Augmented Generation (RAG) technique, utilizing a university courses database to provide relevant course context through keyword-specific searches. The research encompasses an experimental comparison of four prompting approaches (zero-shot, few-shot, chain-of-thought, and tree-of-thought) across 12 university courses executed on four large language models: starling-lm-7b-alpha, openchat_3.5, saiga2_13b, and gpt-3.5-turbo. Starling-7b and OpenChat-3.5 were taken as the most successful ones in the previous study [8]. The experiment also checks the amount of topics proposed for various levels of education and evaluates them using embeddings measuring number of subject areas covered and proximity to the existing courses in the university.

II. Methods

A. Prompt Engineering

Large language models (LLMs) demonstrate significant efficacy in content generation due to their sophisticated design, which mimics human's creative processes. However, to achieve specific outcomes, it's crucial to guide these models accurately. One effective method for directing LLMs is through prompt engineering, which involves strategic formulation of queries that precisely influence the model's output, ensuring that the content produced adheres to specific requirements and relevancy for the intended tasks. It might include selecting specific words, phrases, amplifiers and contextual cues that enhance the AI's understanding of the query.

In the conducted experiment, four prompting techniques were employed: zero-shot, few-shot, chain-of-thought, and tree-of-thought. Each method offers a distinct manner of presenting information to the large language model.

In the zero-shot approach, a large language model receives a task without additional examples and attempts to solve it using its prior experience. An example prompt might be:

"Describe the structure of a course on Web programming" — without any preliminary examples or explanations.

In the few-shot approach, the model is provided with

several examples which aid LLM to understand the context and the desired response format. This approach is effective when it is necessary to guide the model towards a specific response style or when the model needs to understand the specific context of the task.

Below is an example of a prompt for an LLM that could be used for generating a web development course program using a few-shot approach. The examples provided are based on topics from previously developed similar courses at the university. From these examples, the LLM is expected to understand the desired level of detail for topics in the course, as well as the balance between depth and breadth of the material.

"You are a teaching assistant. Develop a course structure for the 'Web Programming' discipline for undergraduate students with an emphasis on the practical aspects. The course should include the study of Django and Vue.js frameworks and cover topics such as working with sockets, the OSI network model, Apache vs Nginx, OOP, and design patterns including MVC. To better understand the structure, include the following examples in your request:

Example 1: Basics of Web Programming Introduction to HTML, CSS, and JavaScript Web page structure, Basics of styling, Basics of programming in JavaScript

Example 2: Advanced Web Programming Using Django and Vue.js frameworks Django architecture, Basics of Vue.js, Data binding and component-based approach"

The chain-of-thought approach encourages models to sequentially explain their thought process. This technique is particularly valuable for tasks requiring deep logical analysis, such as in mathematics and when dealing with abstract concepts. In the context of curriculum development, it helps form logical connections between sections and topics of a course. It also ensures structured learning for students, ensuring that each subsequent module builds on the knowledge acquired in previous sections. Below is an example of a request using this technique.

"You are a teaching assistant. Develop a course structure for the discipline "Web programming". Start by analyzing what knowledge, skills and abilities students already have before starting the course. Then, for each section of the course, explain why you chose those topics and subtopics and how they relate to students' prior knowledge. Include reflection on how each topic prepares students for subsequent topics, justifying why you believe that after studying one topic, students are ready to move on to the next. Provide examples of assignments or projects that will help consolidate acquired knowledge and skills. Summarize by explaining how the entire course structure contributes to the educational goals of the discipline"

The tree-of-thought approach can be seen as an extension of the chain-of-thought concept. It allows the model to simultaneously consider multiple reasoning pathways, similar to a tree structure. This method is used for highly complex problems that have multiple possible solutions or require a comprehensive analysis of different aspects of the issue. Below is an example prompt incorporating elements of tree-of-thought, where the model simulates a competitive discussion aimed at achieving a consensus on the themes to be included in the course structure.

"You are a teaching assistant. Develop a course structure for the discipline "Web programming". Simulate a situation where 100 experts create a course in a discipline. Each expert includes from 5 to 7 topics in his course. Your task is to create a list of the main topics that must be mastered. Start by identifying topics that appear in at least 5 programs. Consider why these topics are often chosen by experts: perhaps they are critical to understanding the discipline. Then analyze how these popular topics are interconnected and how the exclusion of less popular topics might affect the overall understanding of the discipline. Based on the results of the analysis, propose a final list of topics, explaining how each of them contributes to the achievement of the educational goals of the course."

B. Large Language Models Used

There was conducted an experiment within which were generated 12 academic course programs from various subject areas, including Computer Networks, Information Security Management, Foreign Language in Professional Activity, Web Programming, Fundamentals of Economics, Technological Foresight, Mathematical Linguistics, UML Analysis and Design, Agile Management, Laser Physics, Art, Science and Technology, and Biometrics and Neurotechnology.

Then, for every course, four distinct prompts were devised utilizing established prompt-engineering techniques. For the few-shot method, examples from current university courses crafted by faculty were incorporated. This collection of prompts was then presented to four LLMs. Before text was proposed to each LLM, it underwent several preprocessing steps:

1) **Language Detection:** The *langdetect* library identified whether the text was in Russian or English.
2) **Stopword Removal:** Based on the detected language, NLTK's predefined stopword lists were applied. Additionally, an *IDF-based* method identified high-frequency, low-informative words specific to the dataset, which were added to the stopword list.
3) **Punctuation Removal:** All punctuation marks were removed for cleaner tokenization.
4) **Lemmatization:** For Russian text, *pymorphy2* was used to convert words to their base forms.
5) **Tokenization:** NLTK was employed to split the text into tokens.

These steps ensured clean, relevant inputs tailored to the language and context, optimizing the model's performance.

Given the constraints on computational resources, the LLMs used in the experiment were either optimized for quicker response times, such as GPT-3.5 Turbo from OpenAI, or were quantized models like starling-lm-7b-alpha, openchat_3.5, and saiga2_13b.

In the realm of large language models, quantization involves adjusting model weights to a lower precision format, such as converting to 8-bit integers from 32-bit or 16-bit floating points. This process reduces memory usage and boosts processing speed, vital for deployment in environments with limited resources. Although quantization can marginally lower the quality of the model outputs, the decrease in precision is often insignificant compared to the advantages of reduced size and increased speed [9].

In the context of our specific task—automating the generation of course programs using large language models (LLMs) enhanced by Retrieval-Augmented Generation (RAG)—we selected four models based on their ability to address domain-specific content requirements while balancing computational efficiency. Furthermore, the selection was influenced by findings from a previous study, which demonstrated these models' effectiveness in similar educational content generation tasks. The model's ability to effectively integrate retrieved educational content (via RAG) with generative capabilities makes it especially valuable for creating domain-specific curricula. In our prior research, OpenChat_3.5 consistently performed well, especially in content generation tasks where it was required to align educational content with specific university guidelines.

Many quantized large language models, despite their smaller size, demonstrate impressive results. OpenChat 3.5 7B is noted for its excellent natural language processing performance. This model employs the 'Q5_K_M' quantization method, allowing for efficient 5-bit quantization while maintaining a compact footprint—7 billion parameters and a size of 5.13 GB, requiring up to 7.63 GB of RAM for operation [10].

Similarly, Starling LM 7B Alpha, developed by the Berkeley-Nest team, leverages the berkeley-nest/Nectar training dataset and the Advantage-Induced Policy Alignment (APA) policy optimization method. It also uses 'Q5_K_M' quantization, resulting in significant performance with minimal quality loss. The Starling LM 7B Alpha has achieved notable scores, surpassing many models except GPT-4 and GPT-4 Turbo in the MT-Bench evaluations, showcasing its capability for extensive tasks without substantial quality degradation [11]. The model is particularly effective in scenarios where speed and scalability are essential, such as when dealing with large sets of educational materials across multiple subjects. In our previous research, Starling-lm-7b-alpha achieved high performance across a range of educational tasks, generating content that closely matched predefined curricula based on cosine similarity metrics (Fig. 1 from the study). This further validates its suitability for our current project.

The experiment also included Saiga2_13b, a model tailored for the Russian language, enhancing its proficiency in processing and generating Russian text. Although previously conducted experiments [8] indicated it as one of the lower-performing models, its inclusion in this research aimed to explore its capabilities in a Retrieval-Augmented Generation (RAG) context, assessing its utility in a new application framework.

The study also involves ChatGPT-3.5 Turbo. As the only commercial LLM used, its performance was crucial, particularly following the previously superior results of ChatGPT-4 [8]. This research aimed to determine if quantized LLMs could perform comparably to more resource-intensive models using RAG and prompt-engineering. Results indicated that less resource-intensive models could deliver similar outcomes in specific contexts, validating the practicality of employing more accessible models [12].

While not the most resource-efficient model, GPT-3.5 Turbo provides the highest content quality and is often used to validate and compare the outputs of more efficient, quantized models. In scenarios where educational program generation requires precision and depth, GPT-3.5 Turbo serves as the gold standard for comparison.

The selection of these models reflects a balance between performance, resource efficiency, and domain-specific capabilities. Table I presents a comparative analysis of all four models.

TABLE I. COMPARATIVE ANALYSIS OF THE SELECTED MODELS

Model	Parameters	Pre-training Focus	Key Strengths	Limitations
Starling-lm-7b-alpha	7 billion	Open-domain, policy alignment	High performance with low resource use (quantized)	Limited pre-training on education-specific data
OpenChat_3.5	3.5 billion	Dialogue and generation	Efficient natural language generation and quantized for faster performance	Slightly weaker in domain-specific adaptation
Saiga2_13b	13 billion	Russian-language content	Excellent handling of Russian educational content	High resource requirements
GPT-3.5 Turbo	175 billion	General-purpose text	Superior language understanding and complex generation	Very resource-intensive

The quantized models (Starling-lm-7b-alpha and OpenChat_3.5) excel in resource-constrained environments, making them highly practical for scalable educational program generation. Saiga2_13b was chosen for its language-specific focus, while GPT-3.5 Turbo serves as a baseline for understanding the upper limits of performance at the cost of higher computational requirements. Each model brings its own advantages and trade-offs, enabling a comprehensive comparison and further development of the RAG methodology in educational content generation.

The dataset used in the experimental design possessed the following characteristics:

1) **Average Size of Reference Texts:** The average length of the reference texts in the dataset is approximately *139 words* and *1272 characters.* Additionally, the median text length is *82 words* and *736.5 characters*. The difference between the median and the average indicates the presence of some longer texts in the dataset, which skew the average upward, while most texts are shorter, as indicated by the median. This suggests that the dataset contains a diverse range of text lengths, from concise summaries to more detailed descriptions.

2) **Text Generation Limitations:** The generated text by the models was limited by a maximum token count parameter, typically set to *1024 tokens,* during generation. This token limit was chosen to ensure that the models provided sufficiently detailed responses while remaining concise and coherent. Additionally, the *temperature* parameter was set to *0.7* to balance creativity and precision, avoiding overly deterministic outputs while maintaining relevance. *Top-p* was set to *1,* allowing for the full range of possible tokens, which ensured diversity in the generation while keeping the generated text aligned with the context.

These generation parameters helped to ensure that the model outputs were well-structured, contextually relevant, and limited to a manageable size, aligning with the diverse range of reference texts in the dataset. For all four models hyperparameters remain the same, with the primary variation being the prompt wrapping technique. For instance, while *Saiga* uses the *[INST] ... [/INST]* format, *Starling* and *OpenChat* implement a dialogue-based wrapper like GPT3.5 User: ... <|*end_of_turn*|> GPT3.*5 Assistant:*. These wrappers ensure compatibility with each model's expected input format, maximizing the effectiveness of the same core parameters.

C. RAG

Retrieval-Augmented Generation (RAG) combines the capabilities of large language models (LLMs) with information retrieval from external databases to enhance content generation. This approach utilizes pre-trained models' vast knowledge while incorporating precise, context-specific data from external sources, thus significantly enhancing response accuracy and relevance in complex tasks [13].

Studies show RAG's effectiveness in various domains, including a custom Singlish-speaking chatbot, Professor Leodar, which improved student engagement at Nanyang Technological University [14]. Another application involved domain-specific question answering, where the "RAG-end2end" model demonstrated notable performance enhancements by integrating domain-specific knowledge into the retrieval process [15]

The present study employs Retrieval-Augmented Generation (RAG) to facilitate the creation of academic course structures, leveraging approved educational programs available within the university's database. To enable efficient access to relevant data, a system for indexing and retrieving information from these programs was developed, utilizing Elasticsearch alongside morphological analysis techniques.

This system transforms educational program data into a format suitable for analysis by conducting data cleansing, text normalization, noise reduction, tokenization, and lemmatization. It improves the alignment of user queries with the actual course content through the use of lemmatized text versions and the application of filters such as stop-word removal and text conversion to lowercase. Additionally, the system identifies and removes specific stop-words that are commonly found in course descriptions but do not contribute to understanding the specific subject area.

The retrieval system efficiently indexes course-related fields like titles, descriptions, sections, and topics, facilitating precise search capabilities. These fields are weighted according to their relevance, with titles receiving the highest due to their direct reflection of course content. Utilizing a multi-match query approach enhances the ability to search across these fields, improving result accuracy and relevance. This setup ensures users can locate the most appropriate courses, further enriched by allowing users to input keywords that reflect the educational competencies to be included in the course.

The structure for requesting data retrieval is outlined as follows

```
{
   "title": "Web-development",
   "keywords": "HTML, CSS, JavaScript"
}
```

Below is represented example of retrieved database content

```
{
 "retrieved_data": "Development of Backend services, Development of SSI services, Development of Frontend services. OSI network model. TCP vs UDP, Templates, Design patterns, Working with data models, Django REST Framework, APiView, CVB, OOP principles, Apach vs. Nginx, Angular vs. React, Django settings, migrations, superuser, SPA, Working with sockets, JWT, Djoser",
}
```

As observed, the output often includes more information than initially requested because it pulls the entire structure from the most relevant course program. It's essential to recognize that while this additional context can provide valuable insights, it may also mislead the language model (LLM). Inaccurate or irrelevant data can significantly reduce the quality of the response.

Additionally, if the retrieval system doesn't capture all user-provided keywords—perhaps reflecting only some needed skills—it's crucial for the LLM to prioritize covering the specified topics regardless of the provided context. In such cases, employing a zero-shot approach within the RAG framework might look like:

"You are a teaching assistant. Using information about the courses available at the university: [retrieved_data], Develop a course structure for the discipline "Web programming". The teacher asked to include the following topics: HTML, CSS, JavaScript"

In other techniques such as few-shot, chain-of-thought, and tree-of-thoughts, the prompts were constructed similarly: the language model was provided with context and specifically instructed to incorporate the topics explicitly specified by the user. This approach ensures that the LLM remains focused on the relevant content while integrating the necessary details into the structured output.

III. Experiment

During the experiment, a total of 384 text fragments describing course structures were generated. This included 12 disciplines across 4 prompting approaches, 4 large language models, and 2 methodologies (RAG and non-RAG). In addition to human expert evaluations, an embedding generation algorithm was utilized to assess the quality of the LLM outputs.

The construction of a discipline's embedding is performed through a graph of educational entities, representing the set of prerequisites and learning outcomes—skills possessed by students before and after the course. A link exists between two vertices if the corresponding entities appear in the description of a discipline. The embedding is formed in several stages (see Table II):

1) A graph of subject areas is created by clustering the graph of educational entities. A cluster containing more than 10 educational entities is considered a subject area.

2) Educational entities from smaller clusters (up to 10 entities) are distributed among the formed subject areas based on contextual proximity of tokens from the educational entity to tokens from the subject area, calculated using embeddings from a Word2Vec model trained on data from 8699 disciplines implemented at ITMO University from 2018 to 2023.

3) The number n of communities in the resulting subject area graph is counted, and a zero vector D_j is formed, where $j = 0,...,n$.

4) The number of educational entities from each subject area included in the description of each discipline C_i is calculated, where $i=0,...,m$, and m is a total number of disciplines. The general form of the embedding set for the disciplines is tabulated.

TABLE II. Example of Embedding Creation

	D_0	D_1	D_2	…	D_n
C_0	2	6	0	…	3
⋮	⋮	⋮	⋮	⋱	⋮
C_m	*1*	9	0	…	0

5) Finally, the discipline embedding is normalized.

Such an approach to constructing embeddings allows for capturing and assessing the overall subject orientation of a course without delving into the details of terminology. This is particularly important when comparing extensive texts that may use different concepts to describe similar ideas. The embedding formed from the generated text is compared to the embedding of an existing discipline through cosine similarity; the closer to one, the more similar the vectors are.

Fig. 1 and 2 illustrate the average cosine similarity of embeddings compared to a reference discipline. As shown in Fig. 1, the best result was achieved by the LLM chatgpt-3.5 turbo, while the worst performance was observed with saiga2_13b.

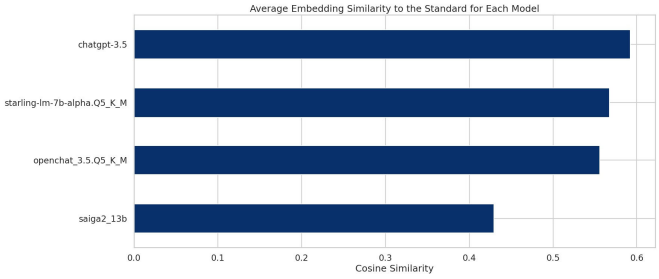

Fig. 1. Average cosine similarity of generated content embeddings compared to the reference for each model.

Fig. 2 displays the average cosine similarity of embeddings compared to a standard reference across different prompting methods. The few-shot approach demonstrated the highest similarity scores, followed by the zero-shot method. The tree-of-thoughts approach showed the lowest performance in terms of similarity to the reference.

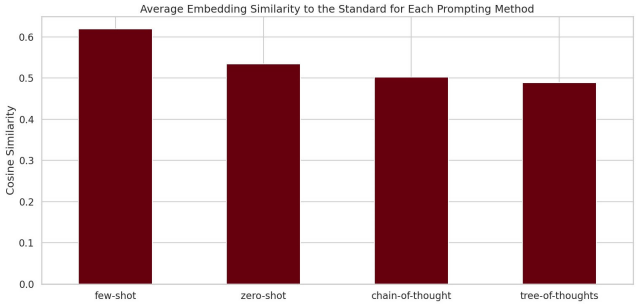

Fig. 2. Average cosine similarity of generated content embeddings compared to the reference for each prompting method

Fig. 3 shows the performance results for each model using different prompting methods. This comprehensive comparison allows for an analysis of how well each combination of model and method can generate content that aligns with the reference standards set in the study.

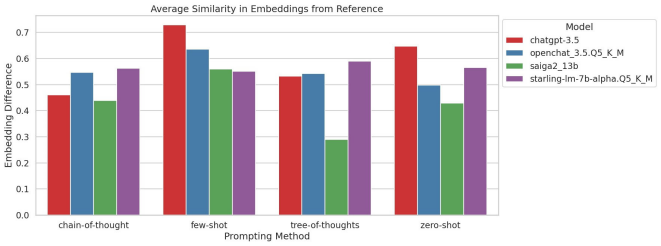

Fig. 3. Summary diagram of embedding differences from the reference across models and prompting methods

Fig. 3 illustrates how prediction success depends on the prompting technique and the specific LLM used. The few-shot method shows superior performance, partly because it incorporates examples from the reference course into the model's inputs. Notably, models like OpenChat and Starling perform well across all prompting methods, sometimes even surpassing ChatGPT-3.5 turbo. However, the Saiga model struggles particularly with the tree-of-thoughts technique, likely due to its more limited capacity for complex reasoning, which stems from its training and focus.

The outcomes highlight that a model's size and number of parameters significantly impact its ability to process context and perform logical reasoning, essential for chain-of-thought and tree-of-thoughts techniques. Model effectiveness also heavily depends on its training and fine-tuning. It's important to consider that prompts were not tailored to any specific LLM, which suggests that more customized queries might have yielded better results. Model performance might decrease if not explicitly trained for tasks requiring intricate reasoning. The specialization of a model like Saiga is critical, as its unique setup and training determine its effectiveness in applying specific techniques [16].

Moreover, embedding similarity should not be the sole measure of the quality of generated course structures. A minimal overlap of generated topics with those of an existing course does not necessarily imply incorrectness. Courses with the same title can cover different aspects, and academic course development is largely a creative endeavor.

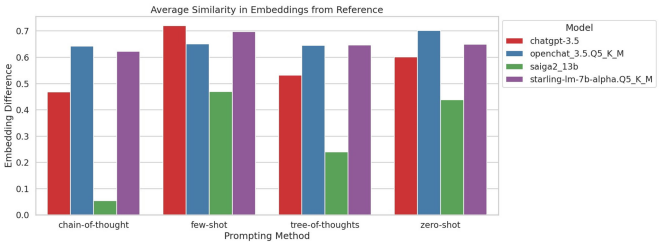

Fig. 4. Summary diagram of embedding differences from the reference across models and prompting methods (RAG approach)

Fig. 4 illustrates the results of the same experiment incorporating the Retrieval-Augmented Generation (RAG) approach. When comparing Fig. 3 and 4, it is apparent that RAG generally boosts the performance of all prompting methods across various models.

- *ChatGPT-3.5*: Shows consistent improvements across all prompting methods with RAG, maintaining a high similarity in embeddings.
- *OpenChat-3.5*: Similar to ChatGPT-3.5, demonstrates improved performance under RAG across all prompting methods.
- *Saiga2_13b*: Rather than experiencing an improvement, this model shows a performance decline. The additional context proved too ambiguous and complex for it to handle effectively.
- *Starling-lm-7b-alpha*: Benefits from RAG in all prompting methods, with notable improvements in embedding similarity.

This indicates that the additional context provided by RAG is beneficial in creating content that closely mirrors the reference, thereby enhancing the models' understanding and portrayal of the course material. Despite the potential for imperfect relevance in data retrieved from university databases, the results demonstrate that even a basic implementation of RAG can positively influence LLM performance. Further, the results highlight that quantized LLMs like Starling-lm-7b-alpha and OpenChat-3.5, when supplemented with relevant context, are capable of excelling in course generation tasks, negating the need for more expensive commercial LLMs like ChatGPT-4 from OpenAI.

To delve deeper into the capabilities of LLMs and RAG, additional experiments were performed, with results depicted in Fig. 5. LLMs were tasked with creating course programs for both bachelor and master's students, using hints tailored to their educational level to guide the content generation. Unlike previous assessments that relied on embedding differences, this evaluation focused on the number of subject area entities extractable from the LLMs' responses. This method aimed to assess the relevance and richness of the content generated relative to the specified educational level.

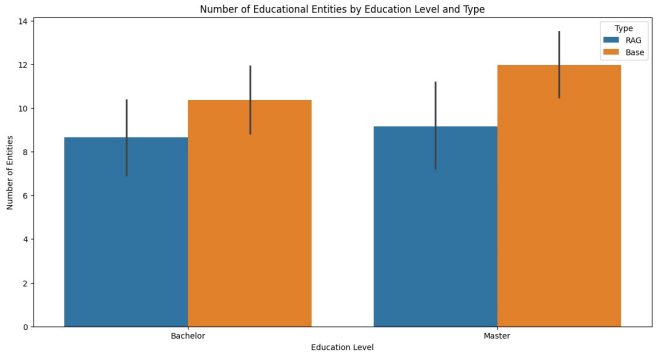

Fig. 5. Number of educational entities containing in academic course based on educational level

For the RAG approach, the LLM prompt was adapted from the previous experiment to include both the retrieved relevant content and additional keywords relating to subject area aspects specified by the teacher. Fig. 5 illustrates that the large language models effectively interpret these cues, incorporating more comprehensive material for master's level students

compared to bachelors. However, the RAG variant tends to indicated that the RAG model focuses intensely on the provided keywords, leading to less creativity in generating independent content ideas. This focus is advantageous when the educator has a clear vision of the course scope and should not be considered when seeking creative input from the model.

Across models and subjects, the average improvement in cosine similarity was +0.08 (8% increase in relevance) when RAG was employed. This demonstrates the importance of integrating retrieval-augmented data for domain-specific content. Without RAG models struggled to maintain high relevance in complex, institution-specific queries. This highlights the limitations of relying purely on LLM generative capacities without incorporating domain-relevant data.

The combination of prompting techniques, RAG, and LLMs improved the accuracy and relevance of generated educational content by approximately 15-20% across subject areas. The most significant improvements were seen when using few-shot and chain-of-thought prompting with the RAG approach.

VI. Conclusion and Future Work

This study explored the influence of various prompting methods and the Retrieval-Augmented Generation (RAG) technique on the quality of educational content generated by large language models (LLMs). The findings suggest that integrating RAG enhances the consistency and predictability of model outputs, thus increasing their reliability for educational applications. Notably, the study also demonstrated that quantized LLMs, which are resource-efficient, and can deliver impressive results with proper prompting techniques.

While the chosen models show promise in automating the generation of educational programs, several challenges remain:

- Improved Adaptability to Specific Standards: One challenge is ensuring that the generated programs fully align with specific educational standards at ITMO University or other institutions. While models like Saiga2_13b handle language-specific requirements, additional alignment methods (such as our proposed methodology for aligning LLMs with institutional data) will further enhance the models' relevance and adaptability.
- Further RAG Enhancements: Our research has shown that RAG significantly improves content retrieval for educational purposes, but limitations remain, particularly in retrieving highly domain-specific content. Future work should focus on refining the retrieval mechanisms to ensure that highly relevant educational material is retrieved based on the unique needs of each institution.

Future efforts will focus on refining the retrieval process to fetch more relevant content, thus enhancing the precision and utility of the generated materials. Plans also include implementing this system within a university setting, offering different generative modes to address diverse creative and include fewer topics overall. Analysis of LLM responses directive needs. Additionally, the development of advanced evaluation metrics beyond cosine similarity of embeddings is planned, aiming to provide a deeper insight into the models' capability to capture and reproduce educational content intricacies.

References

[1] *L. Yan, L. Sha, L. Zhao, Y. Li, R. Martinez-Maldonado, G. Chen, X. Li, Y. Jin, and D. Gašević, "Practical and ethical challenges of large language models in education: A systematic scoping review," British Journal of Educational Technology, vol. 55, pp. 90–112, 2024.*

[2] *A. Carroll, K. Forrest, E. Sanders-O'Connor, L. Flynn, J. M. Bower, S. Fynes-Clinton, A. York, and M. Ziaei, "Teacher stress and burnout in Australia: Examining the role of intrapersonal and environmental factors," Social Psychology of Education, vol. 25, pp. 441–469, 2022.*

[3] *D. Ramesh and S. K. Sanampudi, "An automated essay scoring systems: A systematic literature review," Artificial Intelligence Review, vol. 55, pp. 2495–2527, 2022.*

[4] *E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel, J. Pfeffer, O. Poquet, M. Sailer, A. Schmidt, T. Seidel, et al., "ChatGPT for good? On opportunities and challenges of large language models for education," Learning and Individual Differences, vol. 103, 102274, 2023.*

[5] *S. Wollny, J. Schneider, D. Di Mitri, J. Weidlich, M. Rittberger, and H. Drachsler, "Are we there yet?—A systematic literature review on chatbots in education," Frontiers in Artificial Intelligence, vol. 4, 654924, 2021.*

[6] *I. Drori, S. Zhang, R. Shuttleworth, L. Tang, A. Lu, E. Ke, K. Liu, L. Chen, S. Tran, N. Cheng, R. Wang, N. Singh, T. L. Patti, J. Lynch, A. Shporer, N. Verma, E. Wu, and G. Strang, "A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level," Proceedings of the National Academy of Sciences, vol. 119, e2123433119, 2022.*

[7] *T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, et al., "Language models are few-shot learners," Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020.*

[8] *P. Shnaider, A. Chernysheva, A. Nikiforova, A. Govorov, and M. Khlopotov, "Exploring the effectiveness of prompt engineering and quantized large language models in the development of academic courses," Computer Tools in Education, no. 1, pp. 32–44, May 2024.*

[9] *S. Li, X. Ning, H. Ke, T. Liu, L. Wang, X. Li, K. Zhong, G. Dai, H. Yang, and Y. Wang, "LLM-MQ: Mixed-precision quantization for efficient LLM deployment," 2023.*

[10] *G. Wang, S. Cheng, X. Zhan, X. Li, S. Song, and Y. Liu, "OpenChat: Advancing open-source language models with mixed-quality data," 2023. Available: doi.org/10.48550/arXiv.2309.11235.*

[11] *B. Zhu, E. Frick, T. Wu, H. Zhu, and J. Jiao, "Starling-7B: Improving LLM helpfulness & harmlessness with RLAIF," 2023.*

[12] *C. Irugalbandara, A. Mahendra, R. Daynauth, T. Kasthuri Arachchige, K. Flautner, L. Tang, Y. Kang, and J. Mars, "A trade-off analysis of replacing proprietary LLMs with open source SLMs in production," 2024. Available: doi.org/10.48550/arXiv.2312.14972.*

[13] *J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking large language models in retrieval-augmented generation," 2024.*

[14] *M. Thway, J. Recatala-Gomez, F. S. Lim, K. Hippalgaonkar, and L. W. T. Ng, "Battling Botpoop using GenAI for higher education: A study of a retrieval augmented generation chatbots impact on learning," 2023.*

[15] *S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, "Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering," Transactions of the Association for Computational Linguistics, vol. 11, pp. 1–17, 2023.*

[16] *M. Tikhomirov and D. Chernyshev, "Impact of tokenization on LLaMa Russian adaptation," 2023.*