

Operational efficiency in Large Language Models: A token optimization approach for high-volume data/query systems

Adrian BESIMI*, Xhemal ZENUNI, Florije ISMAILI, Nuhi BESIMI, Artir DAUDI

South East European University, Contemporary Sciences and Technologies, Tetovo, North Macedonia

a.besimi@seeu.edu.mk (*Corresponding author), xh.zenuni@seeu.edu.mk, f.ismaili@seeu.edu.mk, n.besimi@seeu.edu.mk, ad28783@seeu.edu.mk

Abstract: This paper presents a two-layer token optimization approach applied in AI@SEEU: AI for Staff. The approach focuses on both the LLM Tool Selector input and the output structure of MCP tools. The aim of this work is to reduce token consumption in a private institutional deployment while preserving the information required to answer academic staff queries. The first optimization targets the Tool Selector prompt, which initially included extensive tool descriptions and redundant schema definitions that increased input token consumption. The second optimization restructures the output of internal academic tools by removing repeated contextual metadata and introducing a compact hierarchical format. The approach was evaluated across eight representative academic queries and resulted in an overall token reduction of 54.30%. The results demonstrate substantial reductions in token consumption; however, functional correctness, latency, cost, and generalizability are discussed as areas that require further validation.

Keywords: Large Language Models, Token optimization, Tool selection, MCP tools, AI.

1. Introduction

Large language models (LLMs) have become increasingly important in research, education, and institutional digital transformation. Their adoption has accelerated as new model families and deployment options have become available, and universities are now exploring how LLM-based systems can support teaching, administration, and access to institutional services.

Foundational research was conducted before 2018, beginning with the Transformer architecture proposed in 2017 by Vaswani et al. (2017) whose paper has received more than 180,000 citations, followed by BERT or Bidirectional Encoder Representations from Transformers by Devlin et al. (2019) and of course the paper on generative pre-trained language models by OpenAI, titled "Improving Language Understanding by Generative Pre-Training" by Radford et al. (2018), titled "Improving Language Understanding by Generative Pre-Training".

Moving forward, the focus on LLMs, GPTs and OpenAI has grown from the research perspective, including more than 189,000 publications across various platforms and publishers, and more than 13,000 peer-reviewed articles (Google Scholar data). In addition to ongoing fundamental research and improvements in models and algorithms, interest has also shifted towards evaluating GPTs performance, results, costs, and other related aspects.

In our recent publication (Besimi et al., 2025), the focus was on evaluating the feasibility of running large language models (LLMs) locally, with a focus on cost, whereas in this paper, the focus is on optimization issues specifically related to OpenAI models that are of particular interest to us. As both researchers and developers of custom in-house platforms, we consider these findings important during the implementation phase and they may also serve future scholars as important references where performance is also compared to the price of running GPT models.

Large language models (LLMs) are powerful but expensive systems whose cost scales with the number of tokens processed. When tools and routing logic are integrated into conversational AI agents, long prompts with detailed descriptions can quickly increase request sizes and degrade latency. Organizations must understand the key drivers of LLM pricing and employ strategies such as using smaller models for simple tasks, selecting different models for different tasks, context-aware querying, token reduction during model use, refinements, etc.

The use of chatbots in various organizations is considered an added value. Different organizations use different approaches address this digital transformation with AI enablers to retrieve relevant information from organizational data, both public and private. ChatGPT or similar tools raise concerns regarding data privacy, and maintaining control over this data may be difficult. Several organizations tend to implement private chatbots based on various LLMs, either cloud-based or based on in-house models. Duțescu et al. (2024) present an experimental chatbot based on fine-tuned RoGPT2 models to accelerate digital transformation in Romanian public institutions by simplifying citizen interactions with bureaucratic procedures. This approach has also been experimented with by our university; however, the deployment of the in-house model caused issues related to infrastructure upgrades and required substantial investment costs. Therefore, the approach at our university was to deploy a private GPT instance on Azure provided by OpenAI, with custom MCP server and client implementations developed in-house. The issue of security has not been covered in our approach, but further recommendations from Dinu et al (2024) suggest that transformations such as this AI implementation will bring new issues that need to be overcome as well as new opportunities to improve cybersecurity, requiring a multidisciplinary approach incorporating engineers, ethicists, policymakers, and various other actors to ensure that AI systems are designed and used responsibly.

The private Azure deployment was also motivated by findings reported by Ravšelj et al. (2025), in which SEEU students participated. Two relevant findings for academic chatbot use were:

- 41% of respondents believed that ChatGPT can provide reliable information;
- 66% of respondents indicated that students should take appropriate measures to protect their personal information and data when using public chatbots, specifically ChatGPT.

These findings helped guide the decision to focus on a private Azure deployment organized into three pods: one for public use with publicly available data, one for students with access to private and study-related information, and one for staff serving as a staff assistant tool. This deployment is part of AI@SEEU, an in-house SEEU initiative that integrates modern LLMs into university academic workflows in a controlled environment. Sensitive academic data are accessed through authenticated internal APIs, and the chatbot is designed so that private data remain within the institutional Azure environment rather than being exposed through public chatbot services.

The focus of the paper is AI for staff (the staff assistant), an in-house tool, but the same approach has been used in all developed pods.

In the context of AI@SEEU's staff assistant, the LLM Tool Selector, an MCP tool selector, translates natural-language queries into calls to internal academic tools. This introduction sets the stage for optimizing that selector by reviewing existing token-reduction techniques and outlining our motivation for minimizing input length while preserving the information needed for correct tool selection in the evaluated cases.

2. LLM token optimization in the literature

The literature on reducing token consumption for LLMs has burgeoned recently. We have summarized at least twenty papers into the following fields: general performance measures and techniques, token-level reduction combined with prompt engineering, context-aware prompt enhancement and context-aware querying, and finally, prompt compression methods.

2.1. General performance measures and techniques

Research on reducing token consumption in Large Language Models has expanded rapidly, mainly because token usage affects cost, latency, memory consumption, and scalability. In this review, more than twenty papers are grouped into three main areas: performance measures, token-level reduction and prompt engineering, and context-aware or compression-based methods.

Reliable evaluation of LLM performance requires not only accuracy-based metrics, but also efficiency indicators such as latency, memory use, token consumption, and generation speed.

Sagi (2025) highlights “Generation Time,” defined as the time between the first and final generated token. This metric is relevant to our implementation, where response delay was one of the main issues. Therefore, reducing token usage became a key motivation for the optimization approach proposed in this paper.

2.1.1. Token-level reduction and prompt engineering

Several studies focus on reducing unnecessary input tokens while preserving model performance. Wang et al. (2025) introduce LeMo, a “Contextual Token Sparsity” method that removes redundant tokens during fine-tuning to reduce memory use and improve inference speed. Huang et al. (2025) propose Reinforced Token-Level Input Refinement, in which a reinforcement learning agent decides which tokens should be retained or discarded to improve accuracy and efficiency.

Prompt engineering also contributes to token reduction. By removing redundant wording, defining clear role constraints, and adding focused contextual information, prompts can become shorter and more effective. Hu et al. (2025) show that adding relevant context, such as file paths, module names, or tool names, helps the model avoid unnecessary reasoning over irrelevant information. Similarly, Song et al. (2025) demonstrate that simple token-pruning strategies can be effective, particularly in multimodal prompts.

2.1.2. Context-aware and query-aware methods

Another research direction focuses on selecting only the context that is relevant to a specific query. Arefeen et al. (2024) propose LeanContext, which uses reinforcement learning to reduce context length by 37% to 67% without reducing accuracy. Liu et al. (2024) introduce prompt-aware sparsity, which scores tokens based on their relevance and accelerates long-context inference. Liu and Yu (2025) further extend this idea through query-aware cache selection, in which only relevant key-value cache blocks are loaded during decoding.

Efficiency can also be improved through model selection. Maurya, Srivatsa & Kochmar (2024) propose SelectLLM, which routes each request to the most suitable model instead of using the largest model for all tasks. Li et al. (2025) also examine prompt compression for retrieval-augmented generation, focusing on how relevant passages can be selected to improve prompt efficiency.

2.1.3. Prompt compression methods

Prompt compression has become an important method for improving LLM efficiency. Li et al. (2024) classify compression techniques into hard methods, such as token pruning, summarization, and paraphrasing, and soft methods, which use learned vector representations. Nagle et al. (2024) add a theoretical perspective by using rate-distortion principles to define the limits of prompt compression in black-box LLMs.

Zhang et al. (2025) compare six prompt compression methods across thirteen datasets and find that moderate compression can preserve, and sometimes improve, performance in long-context tasks. Recent methods such as Dynamic Compressing Prompts and ICPC further explore learned compact context representations for more efficient inference (Hu et al., 2025; Yu & Liu, 2025).

Overall, these studies show that carefully designed token reduction and prompt compression can improve LLM efficiency while maintaining useful output quality. The approach proposed in this paper draws on three of these strands and applies them to an operational MCP-based system. From the token-pruning literature (Huang et al., 2025; Song et al., 2025), we take the principle that a substantial share of prompt content can be removed without losing task-relevant information, and we apply it to tool descriptions and docstrings. From the context-aware literature (Arefeen et al., 2024; Hu et al., 2025a) we take the principle of supplying only query-relevant context, and we apply it to the structure of tool outputs. From the model-routing literature (Maurya et al., 2024) we take the use of a smaller model for the selection step.

3. AI custom implementation

The custom AI implementation at SEEU (AI@SEEU) is based on Chainlit and Python-based MCP servers. The MCP server queries in-house API endpoints for data retrieval and exposes specialized academic tools to the LLM-based staff assistant.

The following tools were developed to facilitate retrieval-augmented generation (RAG) in the chatbot and enhance user experience:

- **Exam Tools:** Lists of teacher exams; Lists of enrolled students and grades; Historical exam records, etc.
- **Schedule Tools:** Current teacher schedule; student enrollments, etc.
- **Profile & Research Tools:** Teacher academic profile; research publications; research interest areas, FRASCATI classification codes and descriptions, etc
- **Student Attendance Tools:** Student attendance for the current term; other student attendance records, etc.
- **Teacher Attendance Tools:** Current attendance record; topics or subjects taught in the course, etc.
- **Other tools:** Structured course report; student statistics; exam statistics; per-student reports, etc.

All tools have arguments that enable optional filtering (e.g., the number of students with grade 10, by language, term, campus, degree, etc) and some have metrics built-in such as the average grade of a course.

4. Optimization approach

The purpose of this optimization was to **reduce token consumption** in the institutional staff assistant workflow. Before optimization, many tool responses contained repeated contextual data, including teacher details, academic year, term, and course information, for every student record. The goal was to restructure this data so that shared fields were returned only once while preserving the informational value needed to answer the user query.

Driven by the need for optimization, in this paper we have addressed several issues identified in the literature review. One of the main metrics motivating the need for optimization was “Generation Time”, as introduced by Sagi (2025), with computational costs being the cause effect of the shorter execution time; thus, the optimization has a two-fold effect.

In our approach, further explained in the following section, we try to use recommendations from literature. Specifically, we follow the approach described by Song et al. (2025) in their article titled “Less Is More”, where we tend to have simple token pruning, then adding context awareness, as proposed by Hu et al. (2025), to the tool selector as well as to the prompt itself. In combination with the MCP capabilities, these techniques improve efficiency of tool selection and data selection.

A further improvement in our methodology is based on Maurya et al. (2024), in which all requests are initially processed by a smaller LLM, intended for the MCP tool selector, and subsequent requests use larger models to provide better responses to users, thus reducing computation.

In the next subsection we will elaborate on the LLM Tool Selector optimization that was undertaken with eight sample prompts, examples of before and after token responses.

Following this, the next optimization elaborated in the subsequent section is the LLM Tool’s Output Optimization process.

4.1. Optimization 1: LLM tool selector input optimization

LLM tool selector is specifically designed to choose the appropriate MCP tool running on the server, thus mapping the natural-language query as input and identifying the right output. In this subsection, we will provide one example of tool selector prompt optimization, showing the before and after changes.

4.1.1. Problem: Excessive token usage in the tool selector input

The LLM Tool Selector is responsible for analyzing the professor’s natural-language query and determining which internal tool should be executed. Before optimization, the input provided to the Tool Selector contained **extensive and highly redundant information**, including:

- Long and detailed tool descriptions
- Full return schemas for each tool
- Additional docstring notes and explanations
- Repeated arguments and examples
- An inline JSON schema describing the tool-selection response format
- All available tools listed with lengthy explanations

4.1.2. Optimization steps overview

The table below summarizes the main steps involved in optimizing the Tool Selector input. Each step targets a specific source of redundancy or inefficiency, helping to transform the original prompt into a more compact and focused structure. This process greatly contributed to the overall reduction in token consumption (see the steps in Table 1 steps).

Table 1. Steps undertaken to apply Optimization 1: LLM Tool Selector Input Optimization

Nr	Step	Description
1	Edit tool Returns	Removed descriptions of key listings returns.
2	Edit tool Returns	Remove keys from the return and keep only a short summary of what the tool returns.
3	Edit Docstring’s Notes	Remove additional notes that are not necessary.
4	Edit tool args	Remove notes that are not necessary at docstrings’s args
5	Edit instruction prompt	Remove the json schema from the instruction prompt. Reason now we are using the structured responses with fixed json_schema.
6	Final Approach	

4.1.3. Tool selector prompt optimization

To understand the impact of the optimization, the following examples show what the Tool Selector prompt looked like originally and how it was redesigned. This side-by-side view makes it clear where the biggest reductions came from and why the new version is more efficient.

Before Optimization: <pre>""""# Identity You are an assistant helping SEEU professors with access to academic tools. # Instructions * Analyze the user's request and determine which tools are needed to answer the question. * Use the following format exactly: [[{"tool": "tool-name", "arguments": {"argument-name": "value"}}}] * If no tools are needed, return: [] Current teacher id is {teacher_id}. Current Date: {datetime.now().strftime('%A, %d %B %Y')}\nAcad emic Year: {get_current_academic_year()}\nTerm:</pre>	After Optimization: <pre>""""# Identity You are an assistant helping SEEU professors with access to academic tools. # Instructions * Analyze the user's request and determine which tools are needed to answer the question. Current teacher id is {teacher_id}. Current Date: {datetime.now().strftime('%A, %d %B %Y')}\nAcad emic Year: {get_current_academic_year()}\nTerm: {get_current_term()} *Your only task is to decide which tool(s) to call based on the user's request. If the user asks for counts, totals, or numbers</pre>
---	---

<pre>{get_current_term()} *Your only task is to decide which tool(s) to call based on the user's request. * Respond with only the minified JSON string - no additional explanation or formatting. # Output Expectations * Your response must be a single-line, minified JSON string. * Do not include any explanation, formatting, or extra text. * Do not include trailing commas, line breaks, or Markdown formatting. * Return exact argument names as listed in the tool descriptions. {tools_description}""""</pre>	<pre>related to students, schedules, exams, or research, always call the count_data tool. Never infer or generate counts manually. If teacher profile or schedule data is already provided in context, use that information directly instead of calling any tool that fetches it again. * Return exact argument names as listed in the tool descriptions. # Tools Available {tools_description}""""</pre>
--	---

4.1.4. Tool description optimization

Another part of the optimization involved simplifying the tool descriptions sent to the LLM. The original docstrings contained extensive explanations, full schema representations, and detailed argument lists. Although technically informative, much of this content was unnecessary for tool selection and significantly increased the number of tokens processed in each request.

The redesigned docstrings follow a minimal and functional structure. Each tool description now includes only a concise explanation of the tool's purpose, a short list of essential arguments, and a high-level description of the expected output. All verbose details, internal field definitions, and schema listings were removed.

This refinement resulted in a **60–70% reduction** in the size of the `{tools_description}` block, representing one of the most impactful contributors to overall token savings in the system.

4.2. Optimization 2: Tool's output optimization

Before optimization, several tools within the system were responsible for returning detailed information about students and their enrollments. These tools produced large data outputs containing repetitive contextual information related to the teacher, course, and session details for every individual student record.

4.2.1. Problem: Redundant data structure (before optimization)

Before optimization, each record in the returned dataset represented **one student** record. All contextual information, such as course title, term, teacher name, location, and schedule, was repeated for **every single student**, even though these fields were identical across all rows. The same **course/exam and session metadata**, were duplicated for every student. In classes with dozens or hundreds of students, this duplication resulted in a substantial increase in JSON size. Every duplicate key and value consumes tokens when the LLM reads the tool's output. It also makes post-processing slower and more error-prone. For example, if 60 students attend a single course, the LLM must process $60 \times 15 \approx 900$ **duplicated fields**.

Example of initial output from the MCP tool	After optimization of the MCPs Tool Output
<pre>[{ "TermCode": "TermCode", "StudentID": 000000, "StudentName": "Name Surname", "StudyProgramCode": "StudyProgramCode", "DegreeLevel": "DegreeLevel", "AcademicYear": "YY/YY", "Term": "Term", "CourseCode": "CourseCode",</pre>	<pre>{ "GeneralInfo": { "CourseTitle": "CourseTitle", "CourseCode": "CourseCode", "Teacher": "Teacher", "TeacherName": "TeacherName", "AcademicYear": "AcademicYear", "TermCode": "TermCode", "Term": "Term",</pre>

<pre>"CourseTitle": "CourseTitle", "Campus": "Campus", "StudyMode": "StudyMode", "ClassLanguage": "ClassLanguage", "ClassType": "ClassType", "Teacher": "Teacher", "TeacherName": "TeacherName", "Semester": "Semester", "Day": "Day", "StartTime": "StartTime", "EndTime": "EndTime", "Location": "Location" },</pre>	<pre>"CourseTitle": "CourseTitle", "Campus": "Campus", "StudyMode": "StudyMode", "ClassLanguage": "ClassLanguage", "ClassType": "ClassType", "Teacher": "Teacher", "TeacherName": "TeacherName", "Semester": "Semester", "Day": "Day", "StartTime": "StartTime", "EndTime": "EndTime", "Location": "Location" }</pre>	<pre>"DegreeLevel": "DegreeLevel", "Semester": "Semester" }, "Sessions": [{ "ClassType": "Day": "Day", "StartTime": "EndTime": "EndTime", "Location": "Location", "Campus": "Campus", "ClassLanguage": "ClassLanguage", "Students": [{ "StudentID": StudentID, "StudentName": "StudentName" }, { "StudentID": StudentID, "StudentName": "StudentName" }}] }</pre>
---	--	--

4.2.2. Solution: Hierarchical grouping (after optimization)

In order to eliminate redundancy in the tool-selector prompt, we restructured the input schema by grouping static metadata (e.g., course details and teacher identifiers) under higher-level sections such as **GeneralInfo**, **Sessions**, and **Students**. This hierarchical organization ensures that constant attributes are specified once, while student-specific fields are nested within their respective groups.

The optimized structure groups shared course and session details and separates them from per-student data. The new structure introduces three main layers:

- **GeneralInfo** → Constant metadata shared by all students in the dataset (e.g., course title, academic year, teacher).
- **Sessions** → A list of unique class sessions (day, time, room, class type, etc.), where each session contains a dedicated list of students who attend that specific session.
- **Students** → A simplified per-student representation that includes only the fields that vary between students (e.g., StudentID, StudentName), placed within the appropriate session rather than repeated globally.

The steps undertaken and the purpose of each step are shown in Table 2.

Table 2. Steps undertaken to apply Tool’s Output Optimization

Step	Description	Purpose
1. Extract General Info	The first row of the dataset is used to capture invariant metadata such as term, academic year, teacher, and course.	Eliminates repeated contextual data.
2. Identify Unique Sessions	A session is defined by CourseCode, ClassType, Day, Time, Location, Campus, ClassLanguage.	Groups students who share the same lecture/lab.
3. Build Session Dictionary	If a session key doesn’t exist, initialize it; otherwise, append the student to that group.	Efficient grouping of students per class session.
4. Construct Final Response	Combine GeneralInfo and all session objects into a single structured JSON.	Compact, logical, and LLM-friendly format.

5. Optimization results

To evaluate the optimization approach, we measured token usage before and after optimization for eight representative academic queries. The evaluation focuses on token consumption across the tool-selection and answer-generation stages. It should therefore be

interpreted as a token-efficiency case study rather than a full benchmark of response quality, latency, or hallucination reduction.

Token reduction is important for Azure OpenAI deployments because the service uses pay-as-you-go billing for input and output tokens, with prices varying by model and deployment type. Therefore, token reductions can reduce operating costs, but the exact cost impact depends on the model used, the separate input/output token prices, caching, deployment configuration, and any enterprise agreement. The cost results in this paper are therefore reported as token-consumption reductions unless model-specific pricing is explicitly calculated.

For reproducibility, the experimental setup should be read as follows. The system exposes specialized MCP tools for exams, schedules, profiles, research, student attendance, teacher attendance, counting, averaging, and course or student reports. The test set consists of eight structurally similar academic queries involving course enrolment, schedules, exams, and grades. The reported values were obtained from the measured executions of these queries in the AI@SEEU staff-assistant workflow. Future versions of the study should additionally specify the exact model used in each phase, generation parameters such as temperature and maximum output tokens, and whether repeated-run averages were collected.

The tables below report the number of input and output tokens consumed during the tool-selection phase (Table 3) and the subsequent answer-generation phase (Table 4). Table 5 presents the cumulative total across the full multi-step workflow. This organization separates the two optimization layers and makes clear which stage contributes to each reported reduction.

The table organization measures improvements for the two phases of the prompt execution, as follows:

- Optimization 1: LLM Tool Selector Input Token Optimization (Table 3);
- Optimization 2: Tool's Output Token Optimization (LLM Answer Generation) (Table 4).

Table 3. LLM Tool Selector Input Token Optimization for 8 prompt samples

Optimization #1 - LLM Tool Selector (First OpenAI Request)							
Prompt	Before (Tokens)			After (Tokens)			% reduction
	Input	Output	Total	Input	Output	Total	
Show me students that are enrolled in Web Programming course	9.654	29	9.683	5.897	31	5.928	-38,78%
Show me students enrolled in Web Programming course that is held on Tuesday	9.660	33	9.693	5.900	35	5.935	-38,77%
Show me students that are enrolled in Distributed Systems for academic year 24/25 term Spring	9.670	40	9.710	5.905	42	5.947	-38,75%
Show me students enrolled in Game Programming for academic year 24/25 term Spring scheduled on Monday	9.672	44	9.716	5.906	46	5.952	-38,74%
Show me students and their grades in Game Programming exam	9.654	30	9.684	5.897	32	5.929	-38,78%
Show me students who got grade 6 in Distributed Systems	9.656	35	9.691	5.898	37	5.935	-38,76%
Show me students and their grades in Web Programming exam for academic year 23/24, term Spring	9.674	41	9.715	5.907	43	5.950	-38,75%
Show me students who got grade 6 in Game Programming exam for academic year 23/24, term Spring	9.678	46	9.724	5.909	48	5.957	-38,74%
Average token reduction (percentage):							-38,76%

In Table 3, the data on token usage show that Before (Tokens) usage was extremely stable, with variation of only ± 7 tokens across all prompts, due to the nature of the request composition. The same case is shown in After (Tokens) usage, which is also stable, confirming that the optimization method produces deterministic reductions. The average reduction rate is $\sim 38.8\%$ for

this stage of the workflow, with total tokens per operation dropping significantly. This pattern is consistent across prompts with very low variance, showing predictable performance.

In Table 4, token usage varies substantially across prompt requests and MCP server responses because the amount of returned data differs by query. Before optimization, input tokens range from 3,988 to 32,593; after optimization, input tokens drop to a range of 2,209 to 6,262. Output-token counts also differ by query. In the first row, output tokens increase from 1,213 to 2,054 after optimization, which may reflect a longer generated answer based on a more compact but differently organized context. This increase does not eliminate the overall reduction for that query because input-token savings are much larger than the output-token increase.

Table 4. Tool's Output (LLM Answer Generation) Token optimization for 8 prompt samples

Optimization #2 - Tool's Output Optimization (LLM Answer Generation) Token optimization							
Prompt	Before (Tokens)			After (Tokens)			%
	Input	Output	Total	Input	Output	Total	
Show me students that are enrolled in Web Programming course	32.593	1.213	33.806	6.262	2.054	8.316	-75,40%
Show me students enrolled in Web Programming course that is held on Tuesday	23.676	4.358	28.034	4.893	1.561	6.454	-76,98%
Show me students that are enrolled in Distributed Systems for academic year 24/25 term Spring	22.494	4.760	27.254	4.628	1.360	5.988	-78,03%
Show me students enrolled in Game Programming for academic year 24/25 term Spring scheduled on Monday	17.653	2.230	19.883	4.085	1.179	5.264	-73,53%
Show me students and their grades in Game Programming exam	7.680	1.024	8.704	2.953	754	3.707	-57,41%
Show me students who got grade 6 in Distributed Systems	3.988	790	4.778	2.209	584	2.793	-41,54%
Show me students and their grades in Web Programming exam for academic year 23/24, term Spring	5.019	1.097	6.116	2.482	542	3.024	-50,56%
Show me students who got grade 6 in Game Programming exam for academic year 23/24, term Spring	3.992	917	4.909	2.292	594	2.886	-41,21%
Average token reduction (percentage):							-61,83%

The system sends more task-relevant and compact data to the LLM instead of full student lists with repeated metadata or verbose descriptions. This optimization step focuses specifically on the answer-generation phase, where token savings come primarily from restructuring tool output and reducing redundant context. The average token reduction for this stage is 61.83%. Because Azure OpenAI bills input and output tokens separately, this percentage should be interpreted as a reduction in token consumption rather than as a direct percentage reduction in cost.

Table 5 presents a comparative analysis of total token consumption before and after the full optimization process (which includes both input context optimization and LLM answer-generation optimization). The values represent the total tokens billed for each prompt, i.e., input + output tokens across the entire multi-step tool execution pipeline.

Across all eight sample prompts, the results show a consistent and substantial reduction in token usage, with an average overall reduction of 54.30%. This means that the optimized workflow consumes nearly half the tokens compared to the baseline method, which can support lower operating cost and improved scalability when applied in high-volume institutional settings.

The more complex or context-heavy a prompt is, the greater the token savings achieved through optimization. The average reduction of 54.30% across all prompts highlights the strong performance of the optimization pipeline in a realistic, multi-query educational information system.

Table 5. Overall Token reduction for 8 prompt samples

Prompt	Before Total	After Total	Token reduction
Show me students that are enrolled in Web Programming course	43.489	14.244	-67,25%
Show me students enrolled in Web Programming course that is held on Tuesday	37.727	12.389	-67,16%
Show me students that are enrolled in Distributed Systems for academic year 24/25 term Spring	36.964	11.935	-67,71%
Show me students enrolled in Game Programming for academic year 24/25 term Spring scheduled on Monday	29.599	11.216	-62,11%
Show me students and their grades in Game Programming exam	18.388	9.636	-47,60%
Show me students who got grade 6 in Distributed Systems	14.469	8.728	-39,68%
Show me students and their grades in Web Programming exam for academic year 23/24, term Spring	15.831	8.974	-43,31%
Show me students who got grade 6 in Game Programming exam for academic year 23/24, term Spring	14.633	8.843	-39,57%
Average token reduction (percentage)			-54,30%

6. Findings

The empirical results demonstrate that the implemented optimization approach yields substantial improvements in token efficiency across the evaluated LLM-assisted workflow. Across eight representative academic queries, the system consistently reduces total token consumption by a large margin, achieving an average overall reduction of 54.30% and an answer-generation-stage reduction of 61.83%. In several cases, the overall reduction exceeds 67%, indicating that the approach is particularly effective for context-heavy tasks such as course enrollment queries, academic scheduling, and grade retrieval.

A central outcome of this study the finding that token optimization is both effective and stable across diverse prompt types. Regardless of differences in information density, query formulation, or result set size, the optimized pipeline reliably decreases both input and output token usage. The system avoids unnecessary narrative explanations and limits its outputs to task-relevant, structured content, which directly reduces the computational burden on the model.

An important observation is that token savings scale with the complexity of the queried information. Prompts requiring retrieval of larger amounts of student data or multiple filtering conditions benefited the most from optimization, reflecting the advantage of routing queries through a tool-based system with minimal contextual overhead.

The findings from our optimization approach show improvements in the following areas:

- **Cost optimization** - Azure OpenAI charges separately for input and output tokens. The observed token reductions are expected to reduce operating cost, but the exact cost reduction depends on the model, deployment type, regional pricing, caching, and enterprise agreement. Therefore, this paper reports token-consumption reductions rather than reduction in financial cost.
- **Latency improvements** - Fewer processed tokens may reduce generation time and improve responsiveness, particularly under load. However, this study did not measure latency directly, but as a result of the overall token reduction.
- **Answer quality and consistency** - The more compact and structured outputs may support clearer answers by reducing irrelevant context and repeated metadata. However, this study did not include a formal hallucination or answer-quality benchmark.

Table 6. Compact functional validation of tool-call correctness for the eight evaluation queries

Query	Query type	Expected tool category	Required arguments	Correctness		
				Tool	Args.	Output
Q1	Current course enrolment	Current scheduled-session enrolment tool	teacher_id; course_title=Web Programming; current academic year; current term	Yes	Yes	Yes
Q2	Current course enrolment with day filter	Current scheduled-session enrolment tool	teacher_id; course_title=Web Programming; day=Tuesday; current academic year; current term	Yes	Yes	Yes
Q3	Historical course enrolment	Teaching-history or historical enrolment tool	teacher_id; course_title=Distributed Systems; academic_year=24/25; term=Spring	Yes	Yes	Yes
Q4	Historical enrolment with day filter	Teaching-history or historical enrolment tool	teacher_id; course_title=Game Programming; academic_year=24/25; term=Spring; day=Monday	Yes	Yes	Yes
Q5	Current exam grades	Current exam-with-students-and-grades tool	teacher_id; course_title=Game Programming; current academic year; current term	Yes	Yes	Yes
Q6	Current exam grades with grade filter	Exam-with-grades tool with grade filtering	teacher_id; course_title=Distributed Systems; grade=6; current academic year; current term	Yes	Yes	Yes
Q7	Historical exam grades	Historical exam-records tool	teacher_id; course_title=Web Programming; academic_year=23/24; term=Spring	Yes	Yes	Yes
Q8	Historical exam grades with grade filter	Historical exam-records tool with grade filtering	teacher_id; course_title=Game Programming; academic_year=23/24; term=Spring; grade=6	Yes	Yes	Yes

To complement the token-consumption measurements, we conducted a compact functional validation of the eight evaluation queries (Table 6). For each query, the expected tool category, required arguments, argument correctness, and sufficiency of the optimized output were reviewed. The validation focuses on functional equivalence rather than subjective answer quality or broader response-evaluation metrics.

In the evaluated cases, the optimized selector selected the expected tool category and preserved the required arguments for all eight test queries. This result should be interpreted as a basic functional check for the selected case-study queries, not as a general benchmark of tool-selection accuracy across all possible academic requests.

7. Conclusions

Optimizing the AI@SEEU Tool Selector prompt demonstrates that careful prompt engineering and structured tool-output design can substantially reduce token usage in an institutional LLM workflow. By trimming unnecessary descriptions, simplifying return structures, and restructuring repeated data into hierarchical outputs, the implementation achieved an average overall token reduction of 54.30% across eight representative academic queries. The results support the usefulness of the proposed approach as an applied case study, while broader claims about response quality, latency, hallucination reduction, and cost savings require additional dedicated evaluation.

The applied techniques reduce output redundancy while preserving the core information required for the evaluated query types. The largest improvements occur when initial token usage is high, because repeated metadata and large result sets create more opportunity for compression. However, the evaluation is limited to eight structurally similar queries from a single institutional deployment, so the findings should be interpreted cautiously and validated on broader query sets and in additional institutional contexts.

These techniques align with broader recommendations to use smaller models, limit context length, cache prompt templates and outputs, and refine prompts, as also discussed by Faschilin (2025) and in the articles reviewed in the literature section.

Collectively, the findings show that the proposed optimization approach can support more efficient and scalable use of LLMs in institutional workflows. The implementation is especially relevant for systems that rely on specialized tools to retrieve structured academic data. Future work should extend the evaluation with repeated runs, latency measurements, model-specific cost calculations, functional correctness checks, and comparisons against alternative prompt-compression, context-pruning, and tool-routing strategies.

Future work will further analyze prompt compression and related methods proposed by Li et al. (2024), Nagle et al. (2024), and Zhang et al. (2025), as well as evaluate how these methods can be combined with the in-house AI@SEEU platform for additional token optimization.

Author contributions

Conceptualization: A.B., XH.Z. and F.I. Data curation: A.B. and A.D. Project administration: F.I. Validation: A.B., N.B. and A.D. Writing—original draft: A.B. Writing—review and editing: A.B., N.B., XH.Z. and F.I. All authors have read and approved the version of the manuscript intended for publication.

Submission received: 18 May 2026; Revised: 05 August 2026; Accepted: 21 August 2026; Published: 30 September 2026.

REFERENCES

- Arefeen, A., Debnath, B. & Chakradhar, S. (2024) LeanContext: cost-efficient domain-specific question answering using LLMs. *Natural Language Processing Journal*. 7, 100065. <https://doi.org/10.1016/j.nlp.2024.100065>.
- Besimi, A., Besimi, N. & Caushi, A. (2025) Evaluating the feasibility of running AI large language models locally: performance, cost, and strategic insights. *2025 IEEE International Conference on AI and Data Analytics (ICAD)*, Medford, MA, USA. pp. 1-7. <https://doi.org/10.1109/ICAD65464.2025.11114035>.
- Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019) BERT: pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019, Minneapolis, MN*. pp. 4171-4186. <https://doi.org/10.18653/v1/N19-1423>.
- Dinu, A., Vasile, P. C., Georgescu, A. (2024) AI-driven solutions for cybersecurity: comparative analysis and ethical aspects. *Romanian Journal of Information Technology and Automatic Control*, 34(3), 35-48. <https://doi.org/10.33436/v34i3y202403>
- Duțescu, R.A., Apostol, M. & Barbu, M. (2024) Experimental chatbot for accelerated digital transformation in public institutions. *Romanian Journal of Information Technology and Automatic Control*. 34(3), 105-116.
- Faschilin, A. (2024) How to reduce LLM costs: 10 proven strategies for budget-friendly AI. Available at: <https://www.uptech.team/blog/how-to-reduce-llm-costs>.
- Hu, J., Wang, Y., Zhang, W. & Du, Q. (2025a) Optimizing token consumption in LLMs: a nano surge approach for code reasoning efficiency. *arXiv preprint arXiv:2504.15989*. <https://doi.org/10.48550/arXiv.2504.15989>.

- Hu, J., Zhang, W., Wang, Y., et al. (2025b) Dynamic compressing prompts for efficient inference of large language models. *arXiv preprint* arXiv:2504.11004. <https://doi.org/10.48550/arXiv.2504.11004>.
- Huang, G., Li, Y., Chen, X. & Zhang, H. (2025) Beyond prompt engineering: a reinforced token-level input refinement for large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*. 39(22), 24113-24121. <https://doi.org/10.1609/aaai.v39i22.34586>.
- Li, Z., Liu, Y., Su, Y. & Collier, N. (2024) Prompt compression for large language models: a survey. *arXiv preprint* arXiv:2410.12388. <https://doi.org/10.48550/arXiv.2410.12388>.
- Li, Z., Long, D.X., Dinh, D. et al. (2025) What makes a good natural language prompt? *arXiv preprint* arXiv:2506.06950. <https://doi.org/10.48550/arXiv.2506.06950>.
- Liu, D. & Yu, Y. (2025) TinyServe: query-aware cache selection for efficient LLM serving. *Proceedings of the ACM*. <https://doi.org/10.1145/3746027.3758181>.
- Liu, Y., Wu, F., Li, R., Tang, Z. & Li, K. (2024) PAR: prompt-aware token reduction method for efficient large multimodal models. *arXiv preprint* arXiv:2410.07278. <https://doi.org/10.48550/arXiv.2410.07278>.
- Maurya, K.K., Srivatsa, K.A. & Kochmar, E. (2024) SelectLLM: query-aware efficient selection algorithm for large language models. *arXiv preprint* arXiv:2408.08545. <https://doi.org/10.48550/arXiv.2408.08545>.
- Nagle, A., Girish, A., Bondaschi, M. et al. (2024) Fundamental limits of prompt compression: a rate-distortion framework for black-box language models. *arXiv preprint* arXiv:2407.15504. <https://doi.org/10.48550/arXiv.2407.15504>.
- Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. (2018) *Improving language understanding by generative pre-training*. *OpenAI Technical Report*. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
- Ravšelj, D., Keržič, D., Tomaževič, N. et al. (2025) Higher education students' perceptions of ChatGPT: a global study of early reactions. *PLoS ONE*. 20(2), e0315011. <https://doi.org/10.1371/journal.pone.0315011>.
- Sagi, S. (2025) Optimizing LLM inference: metrics that matter for real-time applications. *Journal of Artificial Intelligence & Cloud Computing*. 4(4). [https://doi.org/10.47363/JAICC/2025\(4\)446](https://doi.org/10.47363/JAICC/2025(4)446).
- Song, D., Wang, W., Chen, S. et al. (2025) Less is more: a simple yet effective token reduction method for efficient multi-modal LLMs. *Proceedings of the 31st International Conference on Computational Linguistics*. pp. 7614-7623.
- Vaswani, A., Shazeer, N., Parmar, N. et al. (2017) Attention is all you need. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. pp. 5998-6008.
- Wang, T., Li, Z., Chen, X. & Zhao, Y. (2025) LeMo: enabling less token involvement for more context fine-tuning. *arXiv preprint* arXiv:2501.09767. <https://doi.org/10.48550/arXiv.2501.09767>.
- Yu, Z. & Liu, Y. (2025) ICPC: in-context prompt compression with faster inference. *arXiv preprint* arXiv:2501.01625. <https://doi.org/10.48550/arXiv.2501.01625>.
- Zhang, Z., Li, J., Lan, Y., Wang, X. & Wang, H. (2025) An empirical study on prompt compression for large language models. *arXiv preprint* arXiv:2505.00019. <https://doi.org/10.48550/arXiv.2505.00019>.



Adrian BESIMI holds a Bachelor's degree in Computer Science from South East European University, a Master of Science degree in Computer Science from Purdue University, and a Ph.D. in Computer Science from South East European University. His research and professional activities focus on Artificial Intelligence, Data Analytics, Digital Transformation, and AI integration in higher education and enterprise environments. His areas of specialization include Generative AI, AI governance, business intelligence, and digital innovation ecosystems.

He is a Professor of Computer Science at South East European University and a former Vice-Rector for Academic Planning and Digitalisation. He currently serves as the National Contact Point for Horizon Europe Cluster 4 – Digital, Industry and Space, and as co-coordinator of the DigitMak European Digital Innovation Hub (EDIH) in North Macedonia. He has contributed to numerous international research and innovation projects related to AI, cybersecurity, digital education, and enterprise digitalisation.



Xhemal ZENUNI holds a Bachelor's degree in Computer Science from South East European University, a Master of Science degree in Informatics from New Bulgarian University, and a PhD in Computer Science from the Technical University of Sofia. He is a Full Professor of Computer Science and Dean of the Contemporary Sciences and Technologies Faculty at South East European University. His research and professional activities focus on Artificial Intelligence, Machine Learning, Distributed Systems, Cloud Computing, Data Analytics, Web Technologies, and Software Engineering. His areas of specialization include AI-based predictive systems, semantic web technologies, and cloud-based digital transformation solutions.

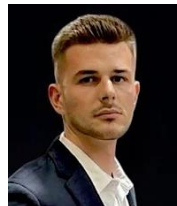


Florije ISMAILI holds a Bachelor's degree in Mathematics, with a major in Computer Science from EGE University, a Master of Science degree in Informatics from New Bulgarian University, and a PhD in Computer Science from Technical University of Sofia. She is Full Professor at the Faculty of Contemporary Sciences and Technologies and currently serves as Advisor for Academic Planning and Digitalization at South East European University. She is also a Member of the Accreditation Board for Higher Education in North Macedonia. Her research and professional activities focus on Artificial Intelligence, Machine Learning, Distributed Systems, Digital Transformation, and AI integration in higher education and industry.



Nuhi BESIMI holds a Bachelor's degree in Computer Science from South East European University, a master's degree in Computer Science from the University of Debrecen, and a PhD in Computer Science from South East European University. His academic and professional work focuses on Artificial Intelligence, Machine Learning, Data Engineering, Big Data systems, and digital transformation. His areas of interest include applied AI, data analytics, distributed systems, IoT, cloud technologies, and AI integration in education and industry, where he has authored and co-authored several scientific papers.

He is an Associate Professor at the Faculty of Contemporary Sciences and Technologies at South East European University. In addition to his academic work, he has been involved in various industry projects related to data platforms, enterprise AI systems, cloud-based solutions, and digital transformation. He has also participated in several research, innovation, and technology-transfer projects, including initiatives related to AI in higher education, extended reality, digital health, and applied data science.



Artir DAUDI is an MSc candidate and software engineer. He holds a Bachelor's degree in Computer Engineering from South East European University and is currently pursuing master's degree in Data Science. His academic and professional interests focus on Artificial Intelligence, Software Engineering, Digital Transformation, and Enterprise Systems.

He is engaged in the design and development of digital transformation solutions through AD Solutions, with a focus on AI integration, process automation, and enterprise digitalization. His work includes the development of intelligent software systems, data-driven applications, and scalable platforms aimed at improving operational efficiency and digital innovation.



This is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.