Evaluating and Training a Conversational QA System for Lecture Videos

Bachelor's Thesis of

Elitsa Dimitrova

Artificial Intelligence for Language Technologies (AI4LT) Lab
Institut for Anthropomatics and Robotics (IAR)
KIT Department of Informatics

Reviewer:	Prof. Dr. Jan Niehues
Second reviewer:	Prof. Dr. Barbara Bruno
Advisor:	M.Sc. Lukas Hilgert

01. May 2025 – 29. August 2025

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 29. August 2025

. . . .

(Elitsa Dimitrova)

Abstract

The increasing digitalization of education has made online resources, such as video lectures, a crucial part of modern learning environments, offering students flexible access to educational content. However, to be fully effective, video lectures must be enhanced with question answering (QA) capabilities to address challenges such as limited interactivity and time-consuming information retrieval. We investigate whether the open-source large language models (LLMs) Llama-3.1-8B and Qwen3-8B can be utilized to create a conversational QA system based on lecture transcripts, which we collect from the KIT Lecture Translator. The transcripts average ~15,000 tokens, with some exceeding 130,000, which creates significant challenges for processing within typical LLM context windows. To evaluate and improve the models, we use the Qasper dataset for long-context academic papers and implement an automatic question generation pipeline that creates 4,695 domain-specific questions. We design retrieval-augmented generation (RAG) pipelines and utilize zero-shot prompt engineering techniques to improve the performance on domain-specific data. Our experimental results show that baseline models achieve moderate performance on academic documents (~55% F_1-score) but struggle with abstractive questions (~30% F_1-score). With QLoRA fine-tuning on various training configurations, consisting of Qasper, SQuAD, and generated questions, we achieve a 2x improvement of the F_1-score (Llama: 37.78%, Qwen: 40.40%) compared to baseline models (Llama: 18.06%, Qwen: 19.13%) when evaluating on the generated questions. The effectiveness of our approach is further confirmed through the evaluation on real student questions, where the fine-tuned models consistently outperform baseline models, indicating the effectiveness of domain-specific training and the applicability of LLM-based QA systems in an educational context.

Zusammenfassung

Mit der zunehmenden Digitalisierung der Bildung sind Ressourcen wie Videovorlesungen zu einem wichtigen Bestandteil moderner Lernumgebungen geworden und bieten Studierenden flexiblen Zugang zu Bildungsinhalten. Um jedoch vollständig effektiv zu sein, müssen Videovorlesungen um Question-Answering-Fähigkeiten erweitert werden, um Herausforderungen wie mangelnde Interaktivität und zeitaufwendige Informationsbeschaffung zu bewältigen. Wir untersuchen, ob die Open-Source Large Language Models (LLMs) Llama-3.1-8B und Qwen3-8B zur Erstellung eines konversationalen QA-Systems basierend auf Vorlesungstranskripten genutzt werden können, die wir vom KIT Lecture Translator sammeln. Die Transkripte umfassen durchschnittlich ∼15,000 Token, wobei einige 130,000 überschreiten, was erhebliche Herausforderungen bei der Verarbeitung innerhalb typischer LLM-Kontextfenster schafft. Zur Evaluierung und Verbesserung der Modelle verwenden wir den Qasper-Datensatz für wissenschaftliche Artikel mit längerem Kontext und implementieren eine automatische Fragengenerierungspipeline, die 4,695 domänenspezifische Fragen erstellt. Wir entwickeln Retrieval-Augmented Generation (RAG) Pipelines und nutzen Zero-Shot Prompt Engineering-Techniken zur Leistungsverbesserung auf domänenspezifischen Daten. Unsere experimentellen Ergebnisse zeigen, dass Baseline-Modelle mittelmäßige Leistung bei wissenschaftlichen Dokumenten (~55% F_1-score) erreichen, jedoch bei abstraktiven Fragen (~30% F_1-score) Schwierigkeiten haben. Durch Fine-Tuning mit QLoRA mit verschiedenen Trainingskonfigurationen bestehend aus Qasper, SQuAD und generierten Fragen, erreichen wir eine 2x Verbesserung des F_1-scores (Llama: 37.78%, Qwen: 40.40%) im Vergleich zu Baseline-Modellen (Llama: 18.06%, Qwen: 19.13%) bei der Evaluierung an generierten Fragen. Die Wirksamkeit unseres Ansatzes wird durch die Evaluierung an echten Studentenfragen bestätigt, wo die feinabgestimmten Modelle durchgängig die Baseline-Modelle übertreffen und die Effektivität domänenspezifischen Trainings sowie die Anwendbarkeit LLM-basierter QA-Systeme im Bildungskontext zeigen.

Contents

List of Figures

List of Tables

1. Introduction

Over the past two decades, technology has been considered an important enabler of educational transformation [33]. The rise in digital research has led to the creation of modern learning environments that differ from traditional face-to-face classroom approaches [44]. The COVID-19 pandemic significantly accelerated this shift, leading to rapid improvement of remote learning platforms, video conferencing tools, and virtual educational resources.

Even though most universities still prioritize in-person teaching approaches, many are starting to provide students with online access to course material and resources. For example, many courses now offer lecture recordings, which combine video of the presentation with a voiceover from the instructor. Having access to them, students can benefit from additional time to understand the lecture content by replaying key segments, while also having the flexibility to review the provided information at any time. Research on the effectiveness of video lecture materials showed that even though having both video and slides can increase the cognitive load, in general, it results in higher student satisfaction, enhanced understanding of content, and improved accuracy of course notes, especially for non-native speakers [8]. However, video lectures create significant challenges for information retrieval. Students often struggle to locate specific information within lengthy recordings and have to re-watch entire sections to find answers to their particular questions.

Furthermore, in order for video lecture platforms to replicate the traditional classroom experience, which includes students asking questions and teachers answering them, they need to be extended with question-answering (QA) capabilities. Traditional rule-based QA systems are usually task- and domain-specific, making it challenging to adapt them to any lecture topic. Large Language Models (LLMs) offer a promising alternative approach, due to their general-purpose capabilities and ability to generate relevant answers for questions they were not explicitly trained on [47]. An LLM-based QA approach would further improve video lecture platforms and address the information retrieval challenge with long videos that students face.

1.1. Problem Statements & Research Questions

In this thesis, we want to research whether we can develop a QA system that complements video lectures from the KIT Lecture Translator [13]. We will experiment with small open-source LLMs, which will be further improved with state-of-the-art fine-tuning and retrieval techniques. Even so, several challenges need to be addressed so that we can integrate long lectures that exceed model context limits.

The lectures from the KIT Lecture Translator [13] have English or German as the original language, and these are the transcripts that we will be extracting and providing QA functionality for. Analyzing the available lectures in the Translator, we see that they contain, on average, 11,703 words in English (12,201 words in German), with some transcripts exceeding 70,000 words. To estimate the computational requirements, we tokenized the transcripts in both languages using the "bert-base-multilingual-cased" tokenizer [15], resulting in an average token count of 14,877 for English lectures and 18,436 for German lectures. The maximum token count is 131,611 for a lecture in German. These lecture lengths create a significant challenge because the context window of an LLM (the amount of text, measured in tokens, that the model can consider and process at any time) is typically much smaller than the length of the transcripts. Even though recent open-source models such as Llama-3 [16] offer context windows up to 128K tokens, evaluation frameworks like RULER [22] show that the effective context length of these models is often limited to 32K/64K tokens. Additionally, processing sequences with more than 100K tokens remains computationally expensive with performance degradation in the middle of the long context [31].

Research Question 1: How effectively do open-source large language models handle question answering tasks on long-context academic documents?

Before we examine task-specific model performance, it is useful to understand the baseline capabilities of open-source LLMs on academic content. Academic and educational documents are more challenging than a general-knowledge context due to the domain-specific vocabulary and complex reasoning, which differ from the typical pre-training data of LLMs.

Research Question 2: Can we create an effective QA system for lecture transcripts with limited domain-specific training data?

A significant limitation of our research is the absence of domain-specific question-answer pairs, which would serve as crucial training and evaluation data for the language models. We will explore different approaches to try to compensate for the missing data and evaluate their effectiveness for the task.

Research Question 3: How effective are enhancement techniques such as retrieval-augmented generation (RAG), prompt engineering, and parameter-efficient finetuning for improving QA performance on lecture transcripts in comparison to the base model?

Furthermore, we will evaluate the impact of enhancement techniques, which are designed to improve model performance on domain-specific tasks.

1.2. Thesis Outline

After we introduced the problem and highlighted the main questions we aim to answer, the second chapter will focus on providing background for the most important concepts used throughout this work. In the third chapter, we will describe our approach and methods. The fourth chapter includes our experimental setup and the results we achieved. In the final fifth chapter, we will summarize our findings as answers to the research questions and discuss future research directions.

2. Background and Related Work

In this chapter, we will introduce all the relevant concepts for this thesis. Firstly, we will start by describing fundamental concepts such as neural networks and sequence representation (subsection 2.1.1), then continue with language models and the Transformer architecture (subsection 2.1.2), concluding with an overview of Large Language Models (LLMs) (subsection 2.1.3) and the multi-stage training process, including pre-training and post-training (subsection 2.1.4). We will also highlight efficiency techniques (subsection 2.1.5), more specifically QLoRA and FlashAttention2, and describe the open-source LLMs used for this thesis (subsection 2.1.6). The second section will cover Question Answering Systems (section 2.2), starting with the problem statement, architectures, approaches, and challenges (subsection 2.2.1). Moreover, we will describe the composition and properties of the specific QA datasets, utilized for training and evaluation (subsection 2.2.2). The third section will focus on enhancement techniques (section 2.3), including Retrieval-Augmented Generation for handling long contexts (subsection 2.3.1), and automatic question generation techniques for creating domain-specific data (subsection 2.3.2). The final section will explore various evaluation techniques (section 2.4), highlighting information retrieval (subsection 2.4.1), text-based (subsection 2.4.2), and LLM-based evaluation methods (subsection 2.4.3).

2.1. Language Models and Neural Architectures

This section will introduce the foundations for modern language processing systems, starting with basic neural networks and their application to sequential data. Then, we will highlight the Transformer architecture and its self-attention mechanism, which enables parallel processing of long sequences. Finally, we will cover Large Language Models (LLMs), which are built upon these architectures at a larger scale to improve language understanding and generation capabilities.

2.1.1. Neural Networks and Sequence Representation

Neural networks are computational models consisting of interconnected nodes (neurons), which are arranged in layers that learn to approximate complex functions f by composing several simpler functions [21]. A feedforward network, also known as a multilayer perceptron, consists of an input layer, one or more hidden layers, and an output layer. Each hidden layer represents a numerical vector and receives inputs x, applies a linear transformation with an activation function $f(Wx + b)$, and passes its outputs to the next layer. W represents the weight matrix, consisting of learnable parameters that determine the strength of connections between neurons, whereas b stands for the bias terms - special weight parameters that allow the activation function to shift, providing flexibility in modelling [21]. During training, neural networks learn by computing a loss function that measures the difference between predicted and actual outputs, then use backpropagation to calculate gradients and adjust each parameter to minimize the loss.

Natural language features, such as words and linguistic information, cannot be used directly as input in a neural network and first have to be converted to an equivalent numerical representation. Initially, we apply tokenization, which translates and compresses the text into a sequence of distinct tokens [38]. One common technique is Byte Pair Encoding (BPE), which merges the most common pairs of adjacent tokens into single tokens until the desired vocabulary size is reached. Once the text is tokenized, we proceed with embedding, where each token is mapped to a dense vector representation in a high-dimensional space [20]. The feature embeddings capture semantic relationships between tokens and need to be trained with the other components of the neural network.

Traditional feedforward networks process each token independently, losing the sequential information important for understanding how words relate to each other in a sentence [20]. This led to the development of specialized architectures for sequence modeling. Prior to the introduction of Transformer-based architectures, sequence modeling relied on recurrent neural networks (RNNs) or convolutional neural networks (CNNs) [43]. For example, RNNs maintain a global context about the text by maintaining a hidden state h that carries information about previous time steps. At each time step t, the RNN computes a new state h_t based on the previous state h_{t-1} and the current input x_t. However, recurrent networks are suboptimal for long input texts since they prevent parallel processing, and the distance between positions increases linearly with the length of the sequence.

2.1.2. Language Models and the Transformer Architecture

Language models are statistical models that learn to predict the probability distributions of words or tokens in a sequence, allowing them to understand and generate human language [21]. However, traditional language models were restricted in their ability to capture long-range dependencies and parallelization potential [43]. The introduction of the Transformer architecture introduced significant improvements in this field by replacing traditional recurrent or convolutional connections with self-attention mechanisms, which create a relationship between different positions in a sequence through direct computation [43]. The goal is that each position learns how much attention to pay to every other position for understanding its own context. In order to compute an output vector containing this information, we need three matrices: queries (Q), keys (K), and values (V). For each position, we compute attention weights by measuring the compatibility through the dot product QK^T. To complete the Scaled-Dot-Product (Figure 2.1), we apply a scaling factor $\sqrt{d_k}$ in order to prevent very large values from causing vanishing gradients, and a softmax function, which normalizes the scores into probability distributions. At the end, to compute the final attention, we multiply the normalized weights with the values V (Equation 2.1).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Multi-Head Attention Multi-Head Attention (MHA) (Figure 2.1) improves the attention mechanism by running several attention functions over different parts of the input in parallel [43]. Instead of using only one set of Q, K, V matrices, MHA creates h different heads, where each head i has its own weight matrices W_i^Q, W_i^K, and W_i^V (Equation 2.2), learned during training. As a result, each head can specialize in capturing different kinds of relationships between positions (e.g., syntactic, semantic). Each attention head computes its own attention output separately, and the outputs of all heads are then concatenated and projected through a weighted layer W^O to produce the multi-head attention output (Equation 2.2).

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.2}$$

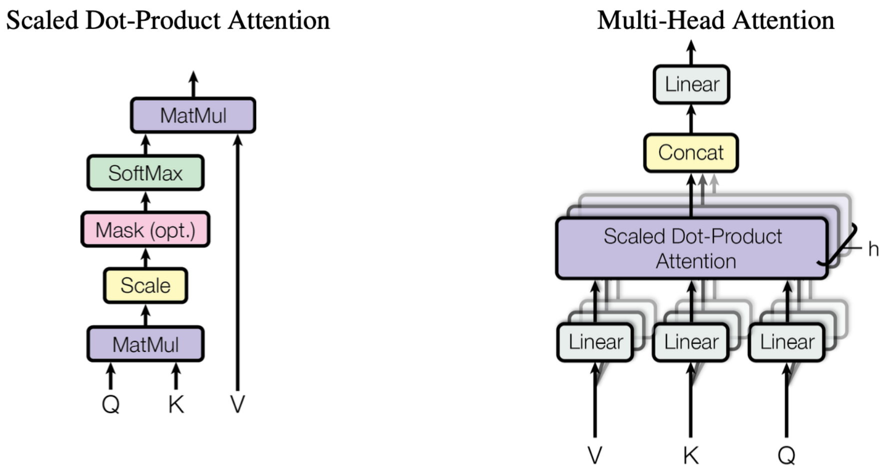

Figure 2.1.: Scaled-Dot-Product (left), Multi-Head Attention (right) [43]

2.1.3. Large Language Models

Large Language Models (LLMs) typically represent Transformer-based language models, which contain billions of parameters and are trained on massive amounts of text data [51]. Through this extensive training, they obtain comprehensive knowledge of grammar and semantics, making them suitable for solving Natural-Language Processing (NLP) tasks, including question-answering (QA) [47]. The key properties of LLMs are their larger scale in terms of parameters and training data, and their ability to process long contexts and sentences in parallel through stacked attention layers in deep neural networks [51].

Modern LLMs go through a multi-stage training process that includes pre-training on large text corpora for improving general language understanding, followed by post-training techniques such as parameter-efficient fine-tuning to adapt them for specific tasks, while maintaining computational efficiency [51]. Additionally, alignment approaches such as instruction tuning and reinforcement learning from human feedback (RLHF) ensure that LLMs can follow specific instructions and generate outputs aligned with user intentions and human values [34].

2.1.4. Pre-training and Post-training

In this subsection, we will highlight the multi-stage training process modern LLMs go through, starting from pre-training with the goal of establishing basic language understanding. However, in order to improve the models for a specific use case, post-training approaches are needed. These include supervised fine-tuning, instruction tuning, and reinforcement learning from human feedback (RLHF), which aim to align the model with the specific application requirements and human expectations.

Pre-training Pre-training language models focuses on learning general language representations from large unlabeled text corpora, which can later be applied to various downstream tasks [15]. Modern decoder-only models use causal (unidirectional) attention mechanisms with each token attending only previous ones and itself, which allows autoregressive text generation through next token prediction [51]. The input and output tokens are processed in the same way through stacked decoder layers, and the effectiveness of this approach is demonstrated through the impressive in-context learning capabilities of models like GPT-3 [4]. However, pre-training these decoder-only models involves heavy computational resources, because to achieve state-of-the-art performance, they need massive amounts of data, billions of parameters, and long training times [25].

Supervised Fine-Tuning Even though pre-trained models achieve promising results in few-shot settings, updating the parameters through fine-tuning allows specialization to the target domain, which significantly improves performance on user-specific tasks [4]. Fine-tuning involves updating the general representations learned during pre-training and updating them, using task-specific datasets [23]. However, full fine-tuning (Figure 2.2) pre-trained models is computationally heavy and time-consuming for models with many parameters, as the resulting model contains as many parameters as the original one, causing significant memory and deployment overhead [25].

Instruction Tuning Pre-trained models such as GPT-3 perform much worse in a zero-shot setting in comparison to few-shot for question-answering, translation, reading comprehension, and natural language inference (NLI) tasks [4]. This could be due to the fact that without a comprehensive example next to the task instruction, the model only relies on the pre-trained data, which could have a different format than the current task [45]. However, including a long context example in a few-shot setting adds token overhead and creates a potential mismatch between the task query and the provided context [6]. Instruction tuning is based on supervised learning with the goal to teach the language model to follow instructions even for unseen tasks [45]. A collection of datasets with natural language task descriptions is brought to an instructional format, consisting of a task-describing instruction, the input query, and the target response, and is then used to finetune the model using supervised learning. The Finetuned Language Net (FLAN) achieved better results than GPT-3 in a zero-shot setting on 20 of 25 datasets, specifically on tasks such as NLI, QA, and translation, whereas in common sense tasks, where the instructions are largely redundant, it was not as effective. Instruction-tuned models with more parameters demonstrated superior performance on unseen tasks in comparison to smaller models due to capacity constraints. Further improvements were achieved by adding chain-of-thought annotations to the instruction templates and showed an improved reasoning capability in models such as Flan-PaLM and Flan-T5 [9].

RLHF Pre-training techniques focus mostly on language modeling, which is technically useful, but sometimes clashes with the user objectives, including truthfulness, unbiasedness, helpfulness, and safety [34]. Reinforcement learning from human feedback (RLHF) is a technique that uses human-generated labels as reward signals to fine-tune base models in order to create a user-aligned resulting model. An example of this is InstructGPT, which was created in three steps by firstly fine-tuning GPT-3 on manually labeled instruction-following data using supervised learning. Then a reward model (RM) was trained on human preferences through labelers, who rated several model outputs for the same prompt, teaching the model to predict which responses humans prefer. These ratings were then used instead of the final unembedding layer of GPT-3 to predict scalar reward values for different responses. Lastly, the policy was further improved through Proximal Policy Optimization to maximize the reward scores, which resulted in outputs that were better aligned with human preferences.

2.1.5. Efficiency Techniques

In most use cases, the computational and memory resources to train and deploy a large language model are constrained. Therefore, we apply efficiency techniques to reduce the overhead while maintaining the desired performance. These approaches allow us to fine-tune and deploy models with billions of parameters by optimizing different steps of the training and inference pipeline.

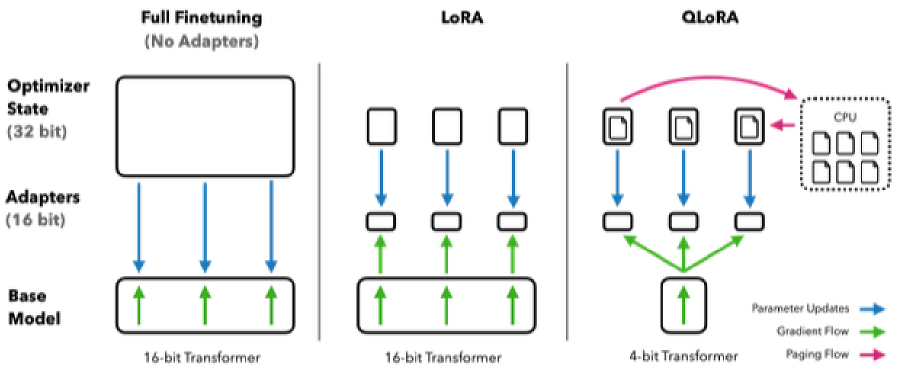

Figure 2.2.: Full Fine-tuning (left), LoRA (centre), QLoRA (right) [14]

LoRA Low-Rank Adaptation (LoRA) solves the computational and memory challenges of full fine-tuning by freezing pre-trained weights and adding small trainable matrices to each layer, which reduces the number of trainable parameters significantly [23]. Instead of updating the full weight matrix W^O, LoRA (Figure 2.2) decomposes the updates into low-rank matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ (called adapters) with $r \ll \min(d, k)$ (Equation 2.3). During training, W^O remains frozen while B and A receive gradient updates. In the case of GPT-3 175B [4], LoRA reduces the parameters by 10,000 times compared to full fine-tuning [23].

$$W^O + \Delta W = W^O + BA \tag{2.3}$$

QLoRA Quantized LoRA (QLoRA) extends LoRA by quantizing the frozen base model weights W^O to 4-bit precision, while keeping the adapters in higher precision [14]. To achieve that, 4-bit NormalFloat4 quantization is applied, which achieves better results than 4-bit integers/floats. During the computation, the 4-bit weights are temporarily dequantized to BFloat16 to maintain multiplication stability. Additionally, QLoRA (Figure 2.2) includes paged optimizers that dynamically offload optimizer states between GPU and CPU memory, preventing out-of-memory errors during training. The whole approach reduces memory usage from >780GB to <48GB for a 65B parameter model without performance loss.

FlashAttention Standard attention mechanisms in Transformers have quadratic complexity $O(n^2)$ with respect to the input sequence length, creating memory and computational bottlenecks for longer contexts [10]. FlashAttention [11] is a technique that reduces this memory usage to linear $O(n)$ while achieving 2-4x speedup through better utilization of the GPU memory hierarchy. However, this version of FlashAttention had limited throughput with forward passes achieving only 50% of device FLOPs/s, and backward passes only 35%. FlashAttention2 [10] improves this by optimizing matrix multiplications for GPU architectures, parallelizing both forward and backward passes, and implementing advanced partitioning techniques to minimize shared memory (SRAM) read/write operations.

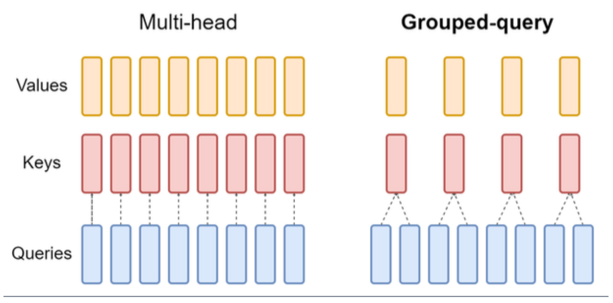

Figure 2.3.: MHA (left), GQA (right) [1]

GQA Grouped-Query Attention (GQA) is a memory-efficient attention mechanism that reduces the computational overhead of self-attention without significant quality loss [1]. GQA has been implemented in recent open-source LLMs such as LLaMA 3.1 [16], Gemma

3 [24], and Qwen3 [46]. Self-attention computes similarity pairwise for all of the tokens in the sequence, resulting in $O(n^2 \cdot d)$ complexity, where each of the n tokens is compared to all n tokens with d-dimensional representations [43]. This quadratic scaling in memory and computational requirements becomes problematic for longer contexts, particularly due to the memory needed to store the representations for all previous tokens. The main difference to the previously mentioned multi-head attention (MHA) (Figure 2.3) is that GQA shares key-value pairs across groups of query heads rather than using separate pairs for each head. While individual query heads provide high-quality results, they also require substantial memory to store key-value pairs [1]. By grouping heads to share key-value pairs, GQA significantly reduces memory overhead without significant quality loss.

2.1.6. Modern Open-Source LLMs

Closed-source LLMs, such as OpenAI's GPT and Antropic's Claude, offer state-of-the-art performance at the cost of reproducibility and research opportunities [32]. On the other hand, open-source LLMs allow researchers to access model weights and run experiments with the goals of transparency, inclusivity, and improvement through collaboration. This thesis focuses on two open-source model families with strong long-context abilities: Meta AI's Llama 3.1 and Alibaba Cloud's Qwen3.

LLama 3.1 Meta AI's Llama 3.1 model family was officially released in 2024 [16], building on its predecessors Llama [42] and Llama 2 [41]. While the model offers three parameter configurations (8B, 70B, and 405B), in this thesis, we will use the 8B configuration. The context window of Llama 3.1 has been extended to 128K tokens [16], in comparison to the window of Llama 2, which was only 4K tokens [41]. Additionally, Llama 3.1 implements GQA to improve inference speed and memory consumption [16]. The rotary positional embeddings, first implemented in Llama [42], have increased base frequency to support longer contexts [16]. The training approach for Llama 3.1 combines pre-training over a large multilingual text corpus with supervised fine-tuning on instruction-tuned data and RLHF.

Qwen3 Alibaba Cloud's Qwen3 model family was officially released in 2025 [46], represents the latest release from the company. The model is available in dense and Mixture-of-Expert (MoE) architectures with parameters ranging from 0.6B to 235B; here, we will focus on the 8B variant. A key innovation in Qwen3 is the dual-mode architecture: a thinking mode for complex, multi-step reasoning and a non-thinking mode for rapid and context-based responses. The context window spans 32K tokens, which is significantly lower than Llama 3.1's. Like Llama 3.1 [16], Qwen3 utilizes GQA and optimized tokenization to handle multilingual texts [46]. The training process starts with pre-training on 36 trillion tokens over 119 languages, followed by post-training with supervised fine-tuning and reinforcement learning techniques.

2.2. Question Answering Systems

Question Answering (QA) is a subfield of Natural Language Processing (NLP) to build systems that automatically answer user questions in natural language [5]. In this section, we will first define the question answering task and present different types of solutions and architectures. Moreover, we will explore the characteristics of several QA datasets and highlight the challenges for QA systems.

2.2.1. Fundamentals

There are different types of questions, depending on their complexity and the techniques used to answer them [48]. Factoid questions (what, who, when, which, how-quantity, quality) expect simple short answers, confirmation questions require a yes/no answer, causal questions (why, how) expect a more complex and detailed explanation relevant to the subject, whereas unanswerable questions have to be categorized as unanswerable.

Problem Statement Given an input question Q, a given context C and an answer sequence of T tokens $A = (a_1, a_2, \dots, a_T)$ [47], the QA task computes $P(Q|A)$, meaning the probability of generating the answer given the question. We will focus on QA as a sequence generation task, where A is unknown and the model has to generate each token a_t based on Q, C, and the previously generated tokens $a_1, a_2, \dots, a_{t-1}$ (Equation 2.4).

$$A = \arg\max_{A} \prod_{t=1}^{T} P(a_t|Q, C, a_1, a_2, \dots, a_{t-1}; \theta) \tag{2.4}$$

QA Architectures Depending on the underlying source of information, QA system architectures can be categorized into text-based, knowledge-based, and hybrid [5]. Text-based QA systems answer questions by finding the most similar answer from unstructured documents, whereas knowledge-based QA queries structured data in knowledge bases, which include relations and entities. Hybrid QA systems synthesize information from both types of data sources in order to maximize efficiency and answer accuracy.

Approaches Question answering approaches are typically grouped into traditional, machine learning-based, and deep learning-based. The traditional pipeline starts with Question Analysis (semantical, syntactical, etc.) with the goal of reducing the search space [5]. Afterwards, text-based architectures employ Passage Retrieval to find related passages ranked by relevance and Answer Extraction to generate and validate the final answer. Knowledge-based QA focuses on Information Retrieval to sort the candidate answers; however, this method struggles with complex questions and interpretability. Machine learning-based QA systems such as support vector machines, decision trees, and nearest neighbors were used for question classification; however, they require feature engineering in order to be effective [50]. Deep learning models, based on the Transformer architecture, particularly BERT, include bidirectional context processing through fine-tuning approaches [15]. Modern LLMs further enhanced these capabilities and were able to understand or generate answers through fine-tuning [51] or few-shot prompting [4]. Their main advantage is the ability to generate answers for questions they were not specifically trained on [47].

Challenges LLM-based QA systems face several limitations, including repetitions, coherence loss over longer contexts, contradictions, and limited interpretability in comparison to humans [4]. Moreover, it is hard for them to provide insights into their predictions and may produce "hallucinations", meaning that they generate false information, while maintaining confidence in their truthfulness [28]. Another challenge comes from the evaluation metrics used for LLM-based QA systems, which often focus only on the final answers, ignoring the reasoning process, and cannot properly rate questions that require specific expertise [47]. Additionally, reasoning capabilities still require significant improvements through the inclusion of more varied reasoning paths and learning from past interactions. Most LLMs nowadays treat each user input independently, often making the same mistakes rather than learning from past turns. Another significant challenge is the lack of autonomous decision-making abilities of LLMs in comparison to humans, which limits their applicability in real-world scenarios.

2.2.2. QA Datasets

This section offers an overview of the question-answering datasets used in this thesis by examining their composition, structure, and characteristics. Each of the presented datasets offers distinct properties in terms of question complexity and answer formats to collectively improve model training and evaluation for academic question answering tasks.

Qasper The Question Answering over Scientific Research Papers (Qasper) [12] dataset contains 5,049 questions from readers of 1,585 academic research papers about the topic of NLP. The questions were split into a train (2,593), validation (1,005), and test (1,451) set, with each paper appearing in only one of the sets. After reading only the title and the abstract, one group of NLP practitioners was responsible for asking a question about the paper. Afterwards, a different group of NLP practitioners had to answer the questions based on the full paper content by annotating the relevant evidence, which could include paragraphs and figures, or tables. The types of question-answer pairs were characterized as extractive (answer is a list of direct citations from the paper), abstractive (free-form text not explicitly cited from the paper), binary (yes/no answer), or unanswerable. To evaluate the correctness of the answers (concatenated with comma separation if multiple), Qasper uses a span-level F_1 score, giving partial credit for word matches if the answer is not an exact match.

SQuAD The Stanford Question Answering Dataset (SQuAD) [35] focuses more on improving machine reading comprehension and consists of 107,785 questions over 536 Wikipedia articles from a wide range of topics, from music to abstract concepts. The articles were preprocessed by extracting single paragraphs over 500 characters and removing images or figures, and tables. Afterwards, crowdworkers were assigned to write free-form questions about the paragraph and annotate the corresponding answers, or skip the annotation step if the question was unanswerable. The question-answer pairs, along with the relevant paragraph and answers field, were split into a training (87,599) and a validation (10,570) set. Similar to Qasper, SQuAD uses a span-level F_1 score, alongside an exact match metric to evaluate answer correctness.

PeerQA The Peer Question Answering (PeerQA) [3] dataset is mostly similar to Qasper, containing a set of 579 questions from peer reviews over 208 academic papers, mostly concerning machine learning and NLP topics. The peer reviews were preprocessed to be context-independent and manually filtered to specifically address the content of the paper. Papers were included in their full form, and the answers were annotated by the original authors of each paper. The answers can include highlighted text as evidence and free-form text that directly answers the question. Questions can be annotated as unanswerable, and the final dataset contains 414 questions with annotated evidence and 336 with a free-form answer. Alongside established metrics like the F_1-score, PeerQA also employs AlignScore to measure factual consistency of the answers.

2.3. Enhancement Techniques

This section focuses on various enhancement techniques, which will be used to adapt the general-purpose LLMs to the specific task of question answering on long lecture transcripts. Retrieval-Augmented Generation (RAG) addresses the long context problem by retrieving only relevant chunks from documents, rather than processing entire transcripts. Additionally, we will explore automatic question generation techniques for creating domain-specific training data that follows educational taxonomies.

2.3.1. Retrieval-Augmented Generation (RAG)

Pre-trained LLMs have a large parametric memory, which is stored in the model parameters, and they do not depend on any external sources [28]. However, this approach has limitations: the parametric memory cannot be updated or expanded without retraining, models struggle to provide clear justifications for their predictions, and they are prone to hallucinations. To address these challenges, hybrid models, such as REALM and ORQA, utilize their non-parametric or retrieval-based memory, which allows the knowledge to be accessed and interpreted. Retrieval-Augmented Generation (RAG) is a fine-tuning approach that combines pre-trained parametric memory with a non-parametric memory accessed with a pre-trained neural retriever. It is especially beneficial for knowledge-intensive tasks, which depend on external knowledge, and achieves SOTA results in open-domain QA, whereas for abstractive QA, it was shown that RAG produced more factual and less hallucinative responses.

RAG Architecture The naive RAG pipeline (Figure 2.4) consists of three stages: indexing documents into a searchable database, retrieving the relevant documents/chunks z based on the query x, and generating responses y using the retrieved context [19, 28]. Indexing involves pre-processing and formatting of the raw data, which is then segmented into smaller chunks in order to handle the context constraints of LLMs [19]. With the help of an embedding model emb, the chunks are then encoded into vector representations $d(z) = \text{emb}_d(z)$ and stored in the database [28, 19]. After receiving a user query x, the system encodes it using the same embedding model: $q(x) = \text{emb}_q(x)$. The retrieval component then computes the similarity between the query vector $q(x)$ and the chunk vector $d(z)$, e.g. $p_\eta(z|x) \propto \exp(d(z)^T q(x))$. Afterwards, the top k chunks with the highest similarity scores top-k$(p_\eta(\cdot|x))$ are selected and used as expanded context to the prompt. In the end, the generation component combines the query x and the retrieved chunks z into a single prompt submitted to the model. The LLM then generates the response autoregressively, where each token y_i is produced with $p_\omega(y_i|x, z, y_{1:i-1})$, conditioning on the previous tokens $y_{1:i-1}$, the original input x, and the retrieved chunks z [28].

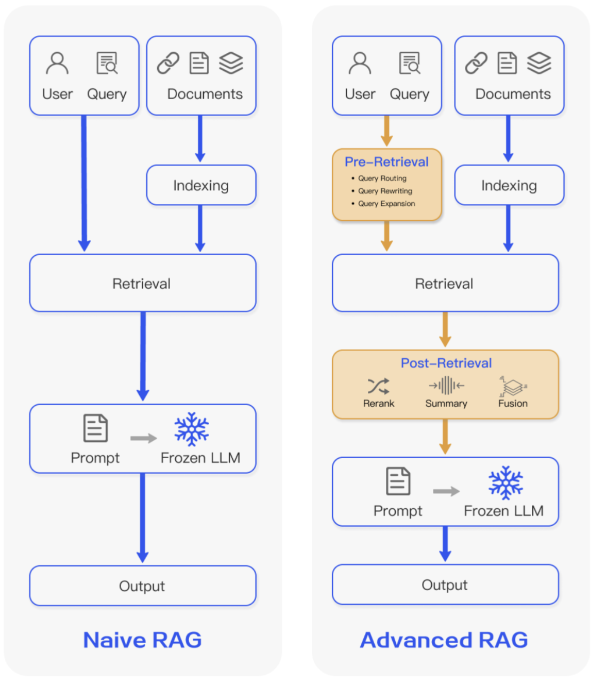

Figure 2.4.: Naive RAG (left), Advanced RAG (right) [19]

Challenges and Optimizations The naive RAG pipeline has some disadvantages: the retrieval component can select chunks irrelevant to the query and therefore miss out on key information, as well as select similar paragraphs from different sources, which could result in disjointed outputs [19]. In order to address these challenges, Advanced RAG includes various pre-retrieval and post-retrieval methods. Pre-retrieval optimization focuses on improving indexed content through better chunking strategies and metadata enrichment. Moreover, the user query can be rewritten or expanded to make it clearer for the specific retrieval task. In the post-retrieval stage, the emphasis is on improving the integration between the query and the retrieved context through re-ranking and context compression techniques. Frameworks such as LlamaIndex and LangChain implement post-retrieval methods to increase their effectiveness.

Chunking Techniques for Long Documents To handle long documents that exceed model context limits, chunking splits the text into fixed-size segments (100, 256, 512…) [19]. Larger chunks provide better contextual information, but require longer processing times and computational power. Smaller chunks reduce noise, but often do not contain the required context and may truncate sentences in the middle. Modern chunking approaches range from simple fixed-size splitting to semantic-based chunking methods, which group text by semantic similarity or use LLMs to determine the appropriate chunk boundaries [40]. Chunk quality can be improved through the addition of metadata information (timestamps, file names, author) or artificial metadata generation (summaries, hypothetical questions). Additionally, hierarchical indexing can create a tree-like structure, where documents are organized into parent-child relationships with the chunks, with each node containing a summary. This approach reduces retrieval errors caused by isolated chunks and improves the overall retrieval efficiency by providing information on contextual relationships.

Retrieval Methods Retrieval effectiveness depends on the similarity between the embeddings of the question query and the document chunks, so the choice of embedding model is crucial [19]. Embedding approaches can be sparse, like TF-IDF and BM25, which use inverted index matching, or dense, which embed the query and chunks into continuous vector spaces based on semantic similarity [17]. While sparse methods rely on query and data quality, dense methods allow training and are more adaptable. Dense retrieval typically uses a BERT-based bi-encoder architecture [17] (one encoder for query $q(x)$, one for the document $d(z)$), allowing similarity computation through dot products $d(z)^T q(x)$ [28]. Specialized dense models like Dense Passage Retrieval (DPR), which was pre-trained for open-domain QA, show superior performance in comparison to general-purpose embedders, such as Contriever and Spider [17].

RAG in Education In an educational setting, LLM-based systems aim to avoid hallucinations, provide accurate, explainable, and trustworthy responses, and allow personalization [30]. One way to achieve these goals is with the help of RAG, which improves the factual accuracy and transparency for educational tasks, where providing the correct and precise information is a priority. Educational QA systems such as Anatbuddy, which provide concrete anatomical explanations, had better accuracy and contextual relevance than general-purpose models. Educational chatbots like MoodleBot that retrieve course-specific materials focus more on the coherence of the conversation in order to assist learning. Furthermore, some chatbots, such as EduChat, include psychological and pedagogical methods to encourage critical thinking and further engagement with the course materials. Tutoring systems like OwlMentor apply logical retrieval routing to find the relevant scientific articles from the available literature. Moreover, multiple AI assistants support students by retrieving specific course materials. Such examples are Jill Watson, which answer student questions with DPR from the lecture notes and syllabus, and CS50 for the introductory computer science course at Harvard, which focuses on guiding the students during programming assignments rather than offering the solution directly.

2.3.2. Question Generation

This section explores automatic question generation techniques and their application in educational contexts. Additionally, we discuss prompting strategies that use pre-trained language models for question generation, including zero-shot and few-shot approaches, and their integration with Bloom's taxonomy.

Automatic Question Generation Generating questions for any educational domain requires linguistic proficiency, application of pedagogical techniques, and a deeper understanding of how learning works [18]. As a consequence, the need for systems specialized in this task to help educators and students emerged. Automatic Question Generation (AQG) is the process of generating syntactically and semantically fitting questions from the context. Based on the method of transforming input into questions, AQG can be characterized into template-based (fixed text with placeholders), rule-based (the rules specify the question type selection and the question construction logic), or statistical (learns how to transform a question from the training data) [26]. However, the first two require extensive manual work, whereas the statistical methods, such as sequence-to-sequence models with RNNs, struggle with long-distance dependencies and high computational costs [18]. With the introduction of Transformer-based models, recent research showed that models such as GPT3 and T5, combined with pre-training on large datasets and finetuning techniques, can generate questions, which are both semantically and syntactically better [18, 37].

Prompting Strategies Full-model training or fine-tuning demands extensive data and significant computational resources [37]. Prompt engineering offers an alternative by constructing prompts, which utilize pre-existing knowledge in LLMs to improve task understanding and enable efficient question generation [27]. In this way, as long as the prompt is constructed correctly, even with limited datasets and resources, the desired result can be achieved. Zero-shot prompting includes detailed task descriptions, while few-shot prompting additionally provides examples to improve the effectiveness of the model [37]. Techniques, such as enriching the prompt with CoT instructions and incorporating definitions from Bloom's taxonomy levels, can be used to improve question quality and model performance. Bloom's taxonomy characterizes educational objectives, from basic knowledge recall, understanding, and application to higher-order skills like analysis, evaluation, and creativity. Additionally, combining RAG with prompting allows question generation from the most relevant sections of longer educational materials [39].

2.4. Evaluation Techniques

In this section, we will look at some important metrics for evaluating the performance of a QA system. Firstly, we will explore information retrieval and text-based metrics. Then, we will describe AlignScore and the LLM-as-a-judge approach.

2.4.1. Information Retrieval Metrics

F_1-Score Accuracy is used to evaluate datasets with questions that have only one answer, whereas F-score considers multiple possible correct answers [5]. The F-score can be broken down into Precision, which measures the proportion of found answers that are correct, and Recall, which is the fraction of all correct answers that were successfully found (Equation 2.5).

$$\begin{aligned}
\text{Accuracy} &= \frac{\text{\# of Correctly Answered Qs}}{\text{\# of Answered Qs}} \\
\text{Precision} &= \frac{\text{\# of Correctly Answers Found}}{\text{\# of Answers Found}} \\
\text{Recall} &= \frac{\text{\# of Correct Answers Found}}{\text{\# of Correct Answers}} \\
\text{F-Score} &= 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\end{aligned} \tag{2.5}$$

2.4.2. Text-Based Metrics

ROUGE-L N-gram-based metrics, such as the F1 score, compute only n-gram similarity and therefore depend on exact word matching and struggle to capture context dependencies [7]. One such metric is the ROUGE-L, which measures the longest common subsequence (LCS) between two sentences [36]. ROUGE-L is computed as the F-Score with Precision and Recall being based on the length of the LCS, and β is a factor to control their weight ($\beta = 1$ equal distribution, $\beta > 1$ favors Recall, $\beta < 1$ favors Precision) (Equation 2.6).

$$\text{ROUGE-L} = \text{F-Score}_{\text{LCS}} = \frac{(1+\beta^2)\text{Precision}_{\text{LCS}} \times \text{Recall}_{\text{LCS}}}{\beta^2 \text{Precision}_{\text{LCS}} + \text{Recall}_{\text{LCS}}} \tag{2.6}$$

BERTScore BERTScore improves over n-gram matching by creating BERT token embeddings for the candidate and reference and then computing their cosine similarity [7]. Due to using word representations instead of exact match, BERTScore captures synonyms/paraphrases better than n-gram overlap. The calculation is expressed as an F_1-Score, where Precision averages the maximum cosine similarity of each token i from the reference r and Recall averages the maximum cosine similarity for each token j from the candidate p (Equation 2.7) [36]. However, standard BERTScore does not involve the context and question when evaluating QA systems, which are crucial for evaluating the answer [7]. Therefore, Conditional BERTScore concatenates the context, question, and answer as input to BERT, ensuring correct contextualization.

$$\begin{aligned}
\text{BERTScore} = \text{F-Score}_{\text{BERT}} &= 2\frac{\text{Precision}_{\text{BERT}} \cdot \text{Recall}_{\text{BERT}}}{\text{Recall}_{\text{BERT}} + \text{Precision}_{\text{BERT}}} \\
\text{Recall}_{\text{BERT}} &= \frac{1}{r} \sum_{i \in r} \max_{j \in p} \overrightarrow{i}^T \cdot \overrightarrow{j} \\
\text{Precision}_{\text{BERT}} &= \frac{1}{p} \sum_{j \in p} \max_{i \in r} \overrightarrow{i}^T \cdot \overrightarrow{j}
\end{aligned} \tag{2.7}$$

2.4.3. LLM-Based Metrics

AlignScore AlignScore is a factual consistency metric based on a unified alignment function trained on 15 datasets from 7 established language understanding tasks, including QA, NLI, and semantic similarity [49]. Firstly, each sample is converted into a text pair (a, b), and then the system predicts alignment labels using binary classification, 3-way classification, and regression. AlignScore is built upon RoBERTa architecture (125-355M parameters) and handles long contexts through splitting them into 350-token chunks. Afterwards, the alignment scores are aggregated by selecting the highest score for each sentence. AlignScore captures both semantic relationships and document-level consistency patterns, outperforming LLM-based evaluation metrics much larger in magnitude based on ChatGPT and GPT-4.

LLM-as-judge Although text-evaluation metrics are computationally efficient and widely used, they are not suited for dynamic scenarios and cannot capture attributes such as helpfulness and harmlessness [29]. The LLM-as-a-judge method uses LLMs to score, rank, and select from the available responses. LLMs can judge various attributes, including helpfulness, overall quality, reliability, relevance, feasibility, and harmlessness. In order to improve the judging capabilities of a general LLM, we can apply rule-augmented prompting by embedding principles and references in the prompt for the judge LLM. LLM-as-a-judge can be used for more adaptable custom evaluation for text generation or reasoning tasks.

3. Approach

The goal of this thesis is to improve the performance of general-purpose open-source LLMs on the task of question answering based on lecture transcripts. We will first explore task-agnostic techniques, which aim to generally improve the LLMs' instruction-following abilities, accuracy, and efficiency, independent of the task provided. In order to further enhance model performance for our use case, we will implement task-specific techniques to adapt the models to the academic context.

3.1. Task-agnostic Methods

We will compare the performance of two recent open-source LLMs (Llama-3.1-8B and Qwen3-8B) and analyze the impact of different architectures on question-answering tasks in scientific and academic contexts across different question types. Although both models support longer context windows (Llama: 128K, Qwen: 32K), we limit our experiments to 8K tokens to optimize memory consumption and computational resources. Additionally, this allows us to run more extensive experiments with more iterations, considering our research with limited GPU access. We apply parameter-efficient fine-tuning using QLoRA combined with FlashAttention2 to reduce memory usage and training time. Moreover, we use zero-shot prompt engineering to improve performance through better input formulation without parameter modifications, focusing on question-type-specific instructions.

Supervised Fine-tuning Even though pre-trained language models have shown impressive results on various NLP tasks, their performance on domain-specific problems, such as question answering, can be improved through fine-tuning (paragraph 2.1.4). In subsection 2.1.5, we described methods that improve on the standard training mechanisms and allow more efficient fine-tuning. We apply LoRA, which reduces the number of trainable parameters significantly by freezing the pre-trained weights and adding trainable low-rank matrices for weight updates. In order to further minimize memory consumption, we use QLoRA with double quantization, which quantizes the frozen weights to 4-bit NormalFloat4 precision and applies an additional step to the quantization constants themselves, while the trainable parameters keep their higher precision. Furthermore, we replace the standard attention mechanism of the transformer-based models with FlashAttention2 in order to reduce the memory usage to linear while providing significant speed improvements for both training and inference, which benefits our use case due to the long context of the transcripts.

Prompt Engineering Due to the length of the lecture transcripts, which could exceed 100K tokens, and the inability of smaller models to correctly process them, we cannot directly use few-shot prompting strategies with full examples, which would take up the space reserved for the actual context. Therefore, we focus on zero-shot prompts that include appropriate instructions that will handle different question types, regardless of the domain. We used prompting techniques that improve general model performance by providing explicit instructions for boolean questions ("Yes/No/Unanswerable" response), unanswerable questions (clear criteria to categorize to answer "Unanswerable" if context is insufficient), and for other questions to focus on precise and concise responses. Additionally, we created standardized prompt templates with consistent output format specifications to allow usage on different datasets and easier answer extraction.

3.2. Task-specific Methods

Even though task-agnostic methods improve general model capabilities, our educational domain presents further challenges that need to be addressed. The length of the lecture transcripts and the lack of domain-specific training data require task-specific solutions for handling long contexts and data augmentation. To address these challenges, we implement automatic question generation, multi-dataset training strategies, custom RAG pipelines, and specialized prompting techniques.

Automatic Question Generation One of the main challenges is that we do not have domain-specific training data based on the lecture transcripts. Therefore, we implement automatic question generation using a better model (Qwen3-32B) to create a dataset for improving and validating the models' performance on domain-specific data. By utilizing RAG on the transcripts, we identify the most educationally relevant passages from each lecture and generate questions using specialized prompts: Simple Question prompts for key concepts, Qasper-like prompts for the question types from the Qasper dataset, and Bloom's taxonomy prompts for the cognitive levels Remember, Understand, Apply, Analyze, Evaluate, and Create [37]. To evaluate our generated questions, we will use the LLM itself to perform self-evaluation based on verified criteria and rank the questions. The generated questions will be used both for training and evaluation with the goal of compensating for the missing domain-specific data.

Training Strategy We implement a multi-dataset training strategy to fine-tune the models specifically for our use case through four main training configurations. Our primary dataset is Qasper, which contains scientific papers that resemble the lecture transcripts in length and complexity. By consistently including Qasper in our training, we ensure that both question answering capabilities and long context handling are considered simultaneously. We first fine-tune exclusively on Qasper to establish baseline model performance on long-context scientific documents. Then, we extend the training data by combining Qasper with SQuAD in a 50/50 split, where we sample an equal number of QA pairs from each dataset. SQuAD contains much shorter paragraphs as context, allowing us to focus on improving the question answering abilities of the models. Additionally, we create a domain-specific dataset containing automatically generated questions based on the transcripts to directly influence models' performance on educational content. Finally, we experiment with a combined configuration that combines the generated questions with Qasper and SQuAD, allowing us to determine whether domain-specific training data works well with established QA datasets for our educational use case.

Prompt Engineering For our educational domain, we employed specialized prompting strategies to further improve model performance and instruction following. We utilized automatic question generation prompts across three categories: Simple Question prompts for extracting key concepts concisely, Bloom's Taxonomy prompts for generating structured questions across six cognitive levels (Remember, Understand, Apply, Analyze, Evaluate, Create), and Qasper-like prompts with four question classifications (Extractive, Abstractive, Boolean, Unanswerable) that correspond to the patterns in the Qasper dataset. Due to the availability of multilingual lecture content, we adapted all prompts for both English and German contexts, ensuring consistency for all lectures. Additionally, we used specific evaluation prompts for our LLM-as-judge approach, including Question-Answer Pair Quality Assessment prompts, which evaluate helpfulness, relevance, harmlessness, reliability, and feasibility, and Instructor Evaluation prompts with 12 criteria, adapted to the educational domain, such as educational value, cognitive engagement, and assessment utility. The domain-specific prompts allowed us to improve question generation and evaluation on the lecture transcripts, while ensuring that academic standards were held.

RAG We implement two custom retrieval-augmented generation (RAG) pipelines to handle the different structures of our data sources. For the Qasper scientific papers, we use LLamaIndex with a standard chunking approach that creates vector indices for each question-context pair, focusing on approach simplicity and computational efficiency over preserving the structure of the papers. This choice allowed us to experiment with a more straightforward implementation and analyze its fallbacks in comparison to a more complex implementation. For the lecture transcripts, we develop a structure-aware RAG pipeline, which builds on the inherent subtopic-based structure of the lecture transcripts. We implement a specialized pre-processing logic that preserves metadata information, such as subtopic names, IDs, and timestamps, from the original transcript structure. Lectures, which include specific subtopic divisions, are handled differently from unstructured transcripts, ensuring that related paragraphs stay grouped. In the case that subtopic chunks exceed the pre-determined context size limits, we apply further segmentation while maintaining subtopic-specific metadata to preserve the connection in a single subtopic. Both RAG pipelines use cosine similarity-based retrieval with a top-k parameter to identify the most relevant document chunks for each question. However, the transcripts RAG pipeline additionally benefits from the subtopic information to improve semantic coherence across retrieved passages.

Analysis and Evaluation of Student Questions In order to validate our QA system's performance in a real scenario, we conducted a user study with university students who are currently visiting the lectures for which we have available transcripts. We will collect question-answer pairs from them to create a small domain-specific test set that will allow us to evaluate our QA system's effectiveness in an actual learning context. The collected questions were manually preprocessed by us in order to ensure consistency and removal of unnecessary information. We evaluate the baseline models and our various training configurations on the human questions, using the same metrics we used on the verified datasets, to evaluate how well our system handles real student questions in comparison to synthetic evaluation data. Additionally, we conduct a qualitative analysis of the generated responses to identify weak points and assess how well the system fulfills the student inquiries that may not be reflected through quantitative metrics alone.

4. Experiments and Results

We will first describe our setup, including all settings and pre-processing procedures, in order to allow replication of our experiments. Then, we will list and discuss all of our results.

4.1. Experimental Setup

In this section, we will focus on the preparation for our experiments by providing a detailed description of the data pre-processing. Afterwards, we will list the hardware and software settings we used, including the training settings and a discussion about the models we chose to experiment with. When referring to token numbers, we will use binary prefixes[1], such as k, indicating 1,024 tokens.

4.1.1. Data Preprocessing

Our goal with the pre-processing step is to transform the raw datasets into a unified format $<$ context, question, answer $>$ that can be used both for training and for evaluation. Additionally, we extract relevant metadata from the lecture transcripts and ensure consistency across the human questions evaluation data.

Existing Datasets The pre-processing of Qasper training data was the most complex due to its nested structure. We extract the full paper context by concatenating all the paragraphs to use it as context for the unanswerable questions. For the training data answers, we have a hierarchical strategy by first matching the evidence sentences against the full context to extract the full paragraphs related to them. If no evidence is highlighted as an answer to the question, we convert unanswerable answers to "Unanswerable", boolean to "Yes/No/Unanswerable", and answerable questions receive either extractive spans or free-form answers in order of availability. The PeerQA paper content and question-answer pairs are initially split; therefore, we use the paper identifier from the QA file to join them. For answer formatting, we prioritize taking augmented answers enhanced by GPT-4o, followed by the original author's answers, and as a last alternative, have the highlighted evidence text. The SQuAD dataset requires minimal pre-processing of the answer field through a simple concatenation of the options, whereas the context field remains as given.

Lecture Transcripts There are two types of transcripts - unstructured (plain text without timestamps) and structured (with subtopics and timestamps). For structured transcripts,

[1]https://en.wikipedia.org/wiki/Binary_prefix

we extract and group the content by subtopics, preserving metadata such as subtopic names and IDs. We also include a language field that is contained in the metadata and specifies the language in which the transcript is. Our unique identifier for each lecture consists of a course name, semester, lecture title, and language. During pre-processing, we concatenate the content within each subtopic and then aggregate all subtopic content for each lecture. For structured transcripts with subtopic names, we use the format "SubtopicName: Content" (subsection A.3.1), whereas for unstructured transcripts, we format the content as "SubtopicId: Content" (subsection A.3.2). It is important to note that for the unstructured, there is only one unique SubtopicId, which is 0, so in the end, they are structured as "0: FullContent". We filter the data to include only English and German transcripts, as these are the languages the lectures are originally held in.

Student Questions The student questions evaluation dataset also required pre-processing to ensure consistent analysis. We manually mapped each question to its corresponding lecture based on topic content analysis and searching our transcript data. This step was needed in order for us to be able to extract the exact lecture identifier needed for our RAG pipeline. Furthermore, we cleaned the student answers by eliminating various introductory phrases ("Habe gelesen, dass...", "From the slides I understood that..."), uncertainties, references to external sources ("I used ChatGPT"), redundant expressions, and spelling errors. The pre-processed answers still included all technical content and explanations, while maintaining readability and consistency.

4.1.2. Hardware and Software

Due to the computational complexity of training LLMs, our experiments require significant GPU memory and power. Therefore, we conduct them on the bwUniCluster 3.0[2] using SLURM for job scheduling. Depending on the resource availability, we either use the NVIDIA A100 GPUs or the NVIDIA H100 GPUs. In order to integrate FlashAttention2, we used a PyTorch container[3] from NVIDIA with Docker[4] using the NVIDIA Enroot[5] functionality, which enables containerizing GPU applications.

Our software environment initially used PyTorch[6] as the primary deep learning framework, the Transformers library[7] and the TRL (Transformer Reinforcement Learning)[8] library's SFTTrainer[9] for supervised fine-tuning. However, to improve training consistency and reproducibility, we transitioned to the LLaMA-Factory[10] framework. It provides standardized templates and better training stability while still utilizing the same underlying dependencies, including BitsAndBytesConfig[11] for our QLoRA quantization configuration,

[2]https://wiki.bwhpc.de/e/BwUniCluster3.0
[3]https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch
[4]https://docs.docker.com/
[5]https://github.com/NVIDIA/enroot
[6]https://pytorch.org/
[7]https://huggingface.co/docs/transformers/en/index
[8]https://huggingface.co/docs/trl/index
[9]https://huggingface.co/docs/trl/en/sft_trainer
[10]https://github.com/hiyouga/LLaMA-Factory
[11]https://huggingface.co/docs/transformers/en/quantization/bitsandbytes

and FlashAttention-2 for memory-efficient attention computation on the A100 and H100 GPUs. We use LLamaIndex[12] for our RAG implementation, which provides automatic chunking, embedding generation, and retrieval mechanisms. For the pre-processing, we use the pola.rs[13] library, which provides efficient methods for DataFrame transformation, grouping, and filtering.

4.1.3. Models Choice

We decided to run our experiments on LLama 3.1-8B and Qwen3-8B firstly because both models are part of families that were published in the last two years, which means that they contain some of the latest advancements and are valuable to current research. Their improved long context capabilities in comparison to older models make them good candidates for our use case when dealing with long transcripts. Another advantage is the 8B parameter size, which allows time and resource-efficient fine-tuning, while maintaining performance.

4.1.4. Training Settings

For all our training experiments, we used a QLoRA configuration with FlashAttention2, in order to optimize the time to train and memory consumption of the process. Our training configuration evolved with time, as we improved our approach and changed data configurations based on the results. For the quantization, we used 4-bit NormalFloat4 with double quantization (which quantizes the quantization constants to save additional memory) enabled and a BFloat16 compute dtype to improve long context performance. We used a LoRA rank $r = 8$ and LoRA alpha of 16. The dropout was set to 0.05 and the warmup ratio to 0.03. We also halved the learning rate from the initial 1e-4 to 5e-5, which led to slightly slower but more stable learning. We focused mostly on training with 8K context length after 4K context experiments yielded suboptimal results. Our models were trained mostly on 1 or 3 epochs to prevent overfitting, which we noticed when we gradually evaluated the models on the validation sets during the process. For scheduling, we used a cosine scheduler with a 0.1 warm-up ratio, which was set in the standard template already. We did a couple of training runs with the linear scheduler for more stability, but there was no significant improvement.

4.1.5. Training Configurations

We experimented with three main dataset configurations to evaluate different training strategies for the task. The first configuration used the full Qasper dataset exclusively and served as the baseline due to the similarity between lecture transcripts and scientific papers, both in length and complexity. With this configuration, we wanted to evaluate the general model performance and create the foundation for our training process. Our second configuration combined Qasper with SQuAD in a 50/50 split to create a balanced

[12]https://docs.llamaindex.ai/en/stable/
[13]https://pola.rs/

training set, which includes both long-context scientific papers (from Qasper) and shorter extractive questions (from SQuAD). The goal of this configuration was to improve the models' general QA capabilities while maintaining good performance on long contexts. The third configuration combined the automatically generated questions from the lecture transcripts with the data from Qasper and SQuAD. This configuration was closest to our domain by including questions based on the transcripts, while not affecting the general QA and long-context capabilities. Initially, we wanted to include the PeerQA dataset in our training data; however, the dataset does not provide a designated training split but just a relatively small test split (579 questions). Therefore, we used PeerQA solely for evaluation purposes to benchmark our models against state-of-the-art results.

4.2. Evaluation Methods

In this section, we will describe all of the methods and metrics we used to evaluate the models' performance. We combine several metrics with various dataset configurations in order to measure various aspects thoroughly.

4.2.1. Metrics

Our primary evaluation metric is the token-level F_1-score, which measures the similarity between predicted answer and the ground truth by calculating the harmonic mean between precision and recall. In order to remain consistent, we used the evaluation script from the official SQuAD website[14] and its extended version that outputs the question-type specific F_1-scores for Qasper[15]. Additionally, we used external packages, which included implementations for ROUGE-L[16], and BERTScore[17] for semantic similarity. For PeerQA evaluation, we used some of the official evaluation scripts[18] to compute answerability, ROUGE-L for surface-level similarity, and AlignScore[19] for semantic alignment [3]. The answerability metric sees the task as a binary classification problem, in which the models predict if a question is answerable using the macro F_1 score to handle the imbalance in the amount of answerable (383) vs. unanswerable (112) questions.

4.2.2. Evaluation Datasets

We calculate the metrics on various datasets to draw conclusions about both the domain-specific task and long context handling. Firstly, we use the official Qasper test set to measure the trained models' ability to handle long scientific papers in comparison to the baseline models. After splitting the automatically generated questions into train/validation/test (70%/15%/15%), we use the test set to evaluate how well the models perform on domain-specific lecture content and to compare the effectiveness of different training configurations

[14]https://rajpurkar.github.io/SQuAD-explorer/
[15]https://github.com/allenai/qasper-led-baseline/blob/main/scripts/evaluator.py
[16]https://github.com/google-research/google-research/blob/master/rouge/rouge_scorer.py
[17]https://github.com/Tiiiger/bert_score
[18]https://github.com/UKPLab/PeerQA
[19]https://github.com/yuh-zha/AlignScore

on unseen lecture-based questions. We use the validation split during training for hyper-parameter tuning and early stopping, whereas the test split serves to assess final model performance on the generated questions. Additionally, we use the PeerQA test set as an independent benchmark to compare the fine-tuned models against established baselines and to performance between the different training configurations. Finally, we evaluate the models on the questions submitted by students to assess the applicability and performance in a real domain-specific scenario.

4.3. Results and Discussion

We ran the experiments using two models: Llama-3.1-8B and Qwen3-8B, both trained with a restricted context window of 8K tokens. We will refer to the models shortly as LLama and Qwen for simplicity.

4.3.1. Baseline Performance on Long Contexts

We start with evaluating the baseline performance of the models on the Qasper dataset in order to compare performance with the fine-tuned models later on. This evaluation helps us understand the built-in capabilities of Llama-3.1-8B and Qwen3-8B for question answering on long scientific documents before any domain-specific adaptations.

We ran both Llama and Qwen without constraining the context with a zero-shot prompt (Table 4.1), using the prompt template from LongBench [2] (subsection A.1.1). Without context constraints, the models had access to the full paper content in the prompt.

	Answer F_1		ROUGE-L		BERTScore		AlignScore	
Question Type	Llama	Qwen	Llama	Qwen	Llama	Qwen	Llama	Qwen
Overall	55.86	55.34	54.83	55.04	91.11	91.87	47.71	53.25
Extractive	60.21	56.64	58.95	56.31	91.32	91.30	46.81	52.30
Abstractive	32.17	29.11	30.56	28.53	87.70	87.28	27.73	29.20
Boolean	77.88	73.66	77.86	73.66	94.89	97.82	77.17	73.90
Unanswerable	55.37	81.13	55.63	81.21	92.60	96.70	57.86	82.71

Table 4.1.: Zero-shot baseline performance on Qasper test set using LongBench prompt [2], no context limit

The models had comparable overall performance, with Llama achieving 55.86% F_1-Score and Qwen achieving 55.34% F_1-Score using the LongBench (subsection A.1.1) prompt. However, the models showed different strengths and weaknesses across question types. Both models performed best on boolean questions (Llama: 77.88%, Qwen: 73.66%), which, compared to the F_1-score on abstractive questions (Llama: 32.17%, Qwen: 29.11%), shows that boolean classification is more accurate than information synthesis, represented by a difference of ~45%. However, the corresponding BERTScores for boolean (Llama: 94.89%, Qwen: 97.82%) and abstractive (Llama: 87.70%, Qwen: 87.28%) questions show a smaller performance gap of only ~7-10%, indicating that the models still generate semantically aligned responses but struggle with the information synthesis required for abstract questions.

The biggest difference in performance between the models was noted when handling unanswerable questions, with Qwen significantly outperforming Llama with an F_1-Score of 81.13% in comparison to Llama's 55.37%. This indicates that Qwen can better recognize when questions cannot be answered based on the provided context, which is very important for educational settings in order to avoid generating hallucinated responses. The BERTScore of both models was consistently high across all question types, typically exceeding 90%, which leads to the conclusion that both models generate semantically meaningful responses aligned with the expected answers. The AlignScore shows that Qwen generally outperforms LLama across most question types (53.25% vs. 47.71%), with both models achieving the highest scores on boolean questions (~75%) and the lowest on abstractive questions (~28%). This aligns with the trend of the F_1-score, asserting that models with higher AlignScores generally provide more factually consistent responses, particularly for unanswerable questions (Qwen: 82.71%).

4.3.2. Fine-Tuning Results

In this subsection, we will focus on the results after fine-tuning the models firstly fully on Qasper (subsubsection 4.3.2.1) and later on the combination of Qasper and SQuAD (subsubsection 4.3.2.2).

4.3.2.1. Fine-Tuning on Qasper

To evaluate the effectiveness of our fine-tuning approach and the ability of each model to adapt to the scientific domain without the influence of additional training data, we first trained and evaluated both Llama (Table 4.2) and Qwen (Table 4.3) fully on the Qasper dataset.

	Answer F_1-Score		ROUGE-L		BERTScore		AlignScore	
Type / Epochs	1	3	1	3	1	3	1	3
Overall	57.29	57.82	56.91	57.60	91.71	91.80	55.01	57.84
Extractive	60.02	61.33	59.72	60.98	91.33	91.52	55.25	59.65
Abstractive	23.83	23.70	22.79	23.38	85.56	85.62	24.86	27.33
Boolean	81.48	82.71	81.48	82.71	98.70	99.07	81.38	82.51
Unanswerable	76.16	74.15	76.16	74.75	96.10	95.66	76.93	76.20

Table 4.2.: Llama-3.1-8B fine-tuned on Qasper for 1 and 3 epochs, evaluated on Qasper using LongBench prompt [2], 8K context length

Based on the results, we can see that Llama showed steady but moderate improvement after fine-tuning on Qasper. The overall F_1-score increased from the baseline 55.86% (Table 4.1) to 57.29% after 1 epoch and 57.82% after 3 epochs. For extractive questions, all metrics showed a stable improvement with the training duration. The largest score increase was achieved on unanswerable questions, where the baseline of 55.37% jumped to 76.16% after one epoch of training. This points to the model's better instruction following abilities and more consistent recognition of unanswerable questions. The performance on abstractive responses declined significantly from the baseline F_1-Score of 32.17% (Table 4.1) to 23.70%, which could suggest overfitting to extractive patterns in the training data. Furthermore, the AlignScore increased by ~11% after 3 epochs of training, influenced by the improved factual consistency on extractive questions.

	Answer F_1-Score		ROUGE-L		BERTScore		AlignScore	
Type / Epochs	1	3	1	3	1	3	1	3
Overall	51.90	58.38	51.83	58.29	91.08	91.93	53.01	58.17
Extractive	53.28	60.12	53.36	60.20	90.61	91.33	53.92	58.44
Abstractive	18.01	25.46	17.31	24.51	84.98	86.07	20.87	27.85
Boolean	63.50	82.08	63.50	82.08	95.28	98.83	64.30	81.87
Unanswerable	91.89	83.66	92.22	84.33	98.65	97.27	92.19	85.56

Table 4.3.: Qwen3-8B fine-tuned on Qasper for 1 and 3 epochs, evaluated on Qasper using LongBench prompt [2], 8K context length

The training for Qwen showed a discontinuous pattern with performance decrease after 1 epoch to 51.90% compared to the baseline 55.34% (Table 4.1); however, after 3 epochs, the model achieved an F_1-score of 58.38%, surpassing Llama's 57.82%. The initial performance drop, especially influenced by the ~10% decrease in performance on abstractive questions, indicates that Qwen requires more training epochs to adapt to the Qasper scientific domain. On extractive questions, the model accomplished a ~4% increase in comparison to the baseline. Qwen achieved scores over 80% for both boolean and unanswerable questions, asserting its decision-making abilities. Similarly to Llama, the performance on abstractive questions did not surpass the baseline and remained a problematic tendency.

Comparing the results of both models after training for 3 epochs, we can conclude that the answer imprecision on abstractive questions remains, but recognition of unanswerable questions improved. Qwen's results indicate that the model requires more epochs to fully adapt, but it eventually achieves a slightly better overall F_1-Score than Llama. The consistently high BERTScores of both models demonstrate good semantic coherence, whereas the steadily improving AlignScores show that training improves the factual consistency for both models.

4.3.2.2. Fine-Tuning on Qasper and SQuAD

We transition to a more complex training configuration, which includes SQuAD in addition to Qasper, in order to improve general QA capabilities, while maintaining performance on long contexts. Firstly, we evaluated the performance on the Qasper dataset (Table 4.4); however, this did not provide enough insight into the effectiveness of the mixed training approach. Therefore, we evaluated the fine-tuned models Llama (Table 4.6) and Qwen (Table 4.7) on the PeerQA dataset in order to be able to compare their performance to state-of-the-art baselines.

Model	Llama-3.1-8B		Qwen3-8B	
Epochs	1	3	1	3
Answer F_1	57.19	57.49	49.67	58.31
Answer F_1 by type				
extractive	60.81	62.60	51.07	61.41
abstractive	22.65	23.70	15.52	22.75
boolean	83.72	77.88	52.31	79.25
none	73.15	72.67	96.06	82.32

Table 4.4.: Models fine-tuned on Qasper+SQuAD for 1 and 3 epochs, evaluated on Qasper using LongBench prompt [2], 8K context length

Multi-dataset training on both Qasper and SQuAD shows how dataset composition influences model capabilities in different ways. While the evaluation of the mixed training configuration after 3 epochs on Qasper demonstrated minimal overall performance decrease (<1%), we could determine its impact better when examining question types separately. The consistent improvement in extractive performance across both models (Llama: 60.21% → 62.60%, Qwen: 56.64% → 61.41%) indicates that SQuAD acts as a regularizer by enhancing surface-level extraction patterns, while preventing overfitting to Qasper's specific writing style. However, this caused a degradation in abstractive performance, due to SQuAD's predominant extractive question patterns, which biased both models toward direct extraction rather than synthesis. The slight extractive improvement (<5%) might not be valuable enough to justify the bigger abstractive loss (~7-9%), suggesting that SQuAD might not be the optimal supplementary dataset for a scientific domain setting. Additionally, evaluating on Qasper, which is also part of the mixed training data, may not reveal the full impact of the mixed training configuration on unseen academic tasks. Therefore, we require another evaluation benchmark, such as PeerQA.

For the PeerQA evaluation, we run the models with the PeerQA prompt (subsection A.1.2). We first present some baselines from the PeerQA paper [3] (Table 4.5) to compare with our fine-tuned and base models.

Metric	Answerability	Min Score	Max Score	Best Model
Precision	Answerable	78.5	91.0	Llama-3-IT-8B-8k
	Unanswerable	23.5	38.0	Command-R-v01-34B-128k
Recall	Answerable	23.0	92.5	Mistral-IT-v02-7B-32k
	Unanswerable	9.5	87.5	Llama-3-IT-8B-8k
Overall	Macro Avg F1	40.0	59.0	Command-R-v01-34B-128k
	ROUGE-L	12.5	20.5	Mistral-IT-v02-7B-32k
	AlignScore	34.5	40.0	GPT-3.5-Turbo-0613-16k

Table 4.5.: PeerQA Benchmark Reference Scores [3]

			Trained On	
Metric	Answerability	Base Model	Qasper	Qasper+SQuAD
Precision	Answerable	90.01	93.92	94.68
	Unanswerable	12.27	25.78	16.43
Recall	Answerable	58.58	79.61	57.30
	Unanswerable	47.37	57.89	73.68
Overall	Macro Avg F1	45.25	60.93	49.79
	ROUGE-L	28.55	19.31	16.70
	AlignScore	31.11	37.93	41.52

Table 4.6.: Llama-3.1-8B baseline and fine-tuned (8k, 3 epochs), evaluated on PeerQA using PeerQA prompt [3]

After the evaluation on PeerQA, we can see that the Llama trained only on Qasper achieved the highest Macro Average F_1-score (60.93%), significantly outperforming the base model (45.63%) and the mixed training approach (49.79%), and exceeding the state-of-the-art score of Command-R-v01-34B-128k (59%). However, the mixed training configuration achieved the highest AlignScore (41.52%) out of all configurations, surpassing the GPT-3.5-Turbo-0613-16k baseline (40.00%). Therefore, we can conclude that after including SQuAD in the training configuration, the models get better at grounding responses in source text but worse at selecting the correct information to include. Llama trained on Qasper and SQuAD also demonstrated better precision for answerable questions (94.68% vs. 93.92%) and better instruction following.

			Trained On	
Metric	Answerability	Baseline	Qasper	Qasper+SQuAD
Precision	Answerable	89.66	93.33	94.68
	Unanswerable	11.76	17.31	17.43
Recall	Answerable	61.37	63.09	57.30
	Unanswerable	42.11	63.16	73.68
Overall	Macro Avg F1	45.63	51.22	49.80
	ROUGE-L	22.35	18.09	15.40
	AlignScore	27.68	35.04	35.87

Table 4.7.: Qwen3-8B baseline and fine-tuned (8k, 3 epochs), evaluated on PeerQA using PeerQA prompt [3]

Qwen's performance on PeerQA was more stable in comparison to Llama's. While the Qasper configuration still achieved a better (51.22%) F_1-score than the base model (45.63%) and the model trained on both datasets (49.80%), the difference was smaller. Both training configurations showed high precision, similar to Llama. However, the AlignScore of the mixed configuration on Qwen (35.87%) did not reach the factual precision that Llama's (41.5%) did.

After having evaluated both models and training configurations on PeerQA, we can conclude that they demonstrate competitive performance against the benchmarks (Table 4.5).

Some of the scores surpassed the performance of models that had longer context length and more parameters; however, most of the scores were in the acceptable range of the baselines, showing that the models adapted well to the scientific domain. Moreover, the mixed training configurations displayed superior recall for unanswerable questions (73.68%) compared to the models trained only on Qasper (Llama: 57.89%, Qwen: 63.16%), indicating improved instruction following. In general, the results demonstrate that parameter-efficient fine-tuning can achieve competitive performance with significantly fewer resources, making it a viable option for the educational domain.

4.3.3. AQG Analysis and Evaluation

We implemented an automatic question generation (AQG) pipeline using Qwen3-32B as the generation model, combined with RAG-based context extraction on the lecture transcripts. The pipeline generates three distinct question types, each serving a different educational objective. Even though we extracted 200 distinct lectures, we wanted to ensure that the lecture is present in its original language, which is either English or German, resulting in a total of 390 lecture translations. There were also some lectures, such as "05-Wordrepresentation" from the course NLP, that had only a transcript in German, so we wanted to make sure to include both languages. We created questions based on the language, adapting the prompt to the language, which was synthetically added during collection as metadata information to each lecture.

Course	#Lectures	Simple	Bloom	Qasper	Total
Neural Networks	129	320	768	487	1575
NLP	107	300	642	419	1361
Artificial Intelligence 2	66	139	381	264	784
ASR	54	24	324	216	564
Machine Translation	34	94	184	133	411
#Questions	390	877	2299	1519	4695

Table 4.8.: Question types generated per course using Simple Question Prompt (subsection A.1.3), Bloom's Taxonomy Prompt [37], and Qasper-like Prompt (subsection A.1.5)

In total, we generated 4,695 questions across all courses (Table 4.8), split into Simple questions (877 questions, 19%), Qasper-like questions (1,519 questions, 32%), and Bloom's taxonomy questions (2,299 questions, 49%). The first type of questions we generated is called "Simple" and covers the most important concepts in the lecture, generating one question using the Simple Question Prompt (subsection A.1.3) for each retrieved lecture passage using the designated RAG queries (subsection A.2.1), depending on the language of the lecture. Bloom's Taxonomy Questions are organized into cognitive levels: Remember, Understand, Apply, Analyze, Evaluate, and Create. We generate Bloom questions for each lecture, corresponding to the levels, using our modified version of Bloom's Taxonomy Prompt (subsection A.1.4), which includes a description for each Bloom level [37], which we adapted to German [20]. The Qasper-like questions were generated for each lecture with the Qasper-like Prompt (subsection A.1.5), which is similar in structure to the Bloom one.

We evaluated the generated questions extensively, using two approaches with our LLM-as-judge: a quality assessment of the question-answer pairs, using the Question-Answer Pair Quality Assessment Prompt (subsection A.1.6), and pedagogical evaluation from an instructor's perspective, using our version of the Bloom's Evaluation Prompt [37] (subsection A.1.7). Both approaches use the 1-10 scale for judgments.

Metric	ASR	AI 2	Machine Translation	NLP	Neural Networks	Avg
Helpful	6.82	7.61	6.98	7.47	7.47	7.27
Relevant	7.33	8.16	7.56	7.93	7.91	7.78
Harmless	9.48	9.77	9.57	9.70	9.69	9.64
Reliable	6.40	7.58	6.52	7.28	7.34	7.02
Feasible	7.24	8.02	7.55	7.98	7.91	7.74
Quality	7.12	7.96	7.33	7.80	7.77	7.60
Total Avg	**7.40**	**8.18**	**7.59**	**7.86**	**7.85**	**7.84**

Table 4.9.: Average QA Scores by Course using the Question-Answer Pair Quality Assessment Prompt (subsection A.1.6)

The quality criteria of the Question-Answer Pair Quality Assessment Prompt are based on the attributes, which can be assessed by a judge LLM [29], specifically Helpfulness, Relevance, Harmlessness, Reliability, Feasibility, and Overall Quality (Table 4.9). We describe each property in more detail in the prompt to improve the instruction-following abilities of the LLM-judge. Harmlessness achieved the highest average score (9.64), meaning that the generated questions were educationally appropriate and almost free of harmful content. Relevance had a relatively high average score (7.78), indicating the questions were mostly based on the provided context. The scores of Helpfulness (7.27) and Feasibility (7.74) show that the questions assisted student understanding and are answerable by a student attending the lecture. Reliability, with a score of 7.02, was the worst-performing criterion, particularly the courses ASR and Machine Translation, which suggest challenges in maintaining factual consistency in the generated questions. However, overall, this was not as impactful due to the lower lecture count in these courses (54, 34) in comparison to courses such as Neural Networks and NLP, which included significantly more lectures (129, 107) (Table 4.8).

The instructor evaluation included 12 criteria (Table 4.10), used to assess pedagogical relevance and question quality [37]. The basic comprehension metrics (Understandable, Grammatical, Clear) achieved some of the highest scores out of all the criteria, indicating that the generated questions were syntactically correct and clear to students. Additionally, the context alignment scores (Context Related, Answerable, Central) all had a score over 7, confirming that most of the questions aligned with the provided context and were answerable. However, the advanced pedagogical criteria (Educational Value, Cognitive Engagement, Assessment Utility) with scores between 5.82 and 7.65 revealed that even though questions were comprehensible and relevant to the context, they often failed to

[20]https://de.wikipedia.org/wiki/Blooms_Taxonomie

Metric	ASR	AI 2	Machine Translation	NLP	Neural Networks	Avg
Understandable	7.86	8.54	7.88	8.35	8.48	8.22
Context Related	7.06	7.79	7.15	7.53	7.55	7.42
Grammar	7.39	8.58	7.19	8.32	8.64	8.02
Clear	7.14	8.04	7.13	7.74	7.99	7.61
Answerable	7.05	7.71	6.99	7.48	7.59	7.36
Central	7.61	8.29	7.76	8.04	8.09	7.96
Would Use It	6.88	7.79	6.90	7.52	7.61	7.34
Educational Value	7.21	8.03	7.23	7.81	7.95	7.65
Question Quality	6.02	7.27	6.10	6.87	7.20	6.69
Difficulty Appropriateness	5.51	6.52	5.58	6.12	6.29	6.00
Cognitive Engagement	5.36	6.26	5.38	5.84	6.03	5.77
Assessment Utility	5.37	6.32	5.31	5.92	6.18	5.82
Total Avg	**6.62**	**7.43**	**6.67**	**7.22**	**7.42**	**7.07**

Table 4.10.: Average Instructor Scores by Course using the Bloom Evaluation Prompt [37](subsection A.1.7)

provide substantial educational value, worth for university-level students. Nevertheless, the Would Use It criterion (7.34) shows that most of the questions were practically applicable, and an instructor would be willing to incorporate them in an actual teaching context. Overall, we see that from an instructor's point of view, the generated questions were evaluated as technically good but lacking educational effectiveness.

4.3.4. RAG Evaluation

This subsection focuses on the evaluation of our RAG pipelines: the first one handling the scientific papers from Qasper (Table 4.11) and the second one responsible for the lecture transcripts (Table 4.12). Apart from the F_1-score and its elements, precision and recall, we also included the RAG-specific evaluation metrics, containment and coverage. Containment measures how much of the gold evidence is found in the retrieved chunks, demonstrating the information capture ability. Coverage determines if the retrieved chunks overlap with any gold evidence, indicating the model's ability to identify relevant content.

Question Type	#Qs	F1	Recall	Precision	Containment	Coverage
Overall	1,451	12.32	90.42	12.91	90.42	93.18
Extractive	732	13.33	97.93	14.09	97.93	100.0
Abstractive	355	13.87	95.65	14.16	95.65	100.0
Boolean	198	11.65	90.79	12.42	90.79	92.93
None	166	5.37	45.66	5.66	45.66	48.80

Table 4.11.: RAG Evaluation on Qasper Scientific Papers

The scientific papers RAG pipeline was evaluated on the Qasper dataset and shows impressive recall (90.42%), coverage (93.18%), and containment (90.42%). However, the precision is significantly lower (12.32%), resulting in an overall F_1-score of 12.32%. The lack of relevance filtering is intentional and represents our design choice to include as much content as the context limit allows (in our case 8K tokens). This design decision was made due to the query (question) imprecision, specifically the fact that abstractive/boolean/unanswerable questions sometimes do not include the relevant keywords directly used in the original paper. Another reason is that abstractive and unanswerable questions often need an extensive background to be able to reason and synthesize, or determine that the answer is unattainable.

Examining the RAG scores for each question type, we can see that the system achieves its strongest performance on extractive and abstractive questions (Recall: ~95-97%, Coverage: 100%), indicating that all of the questions included the evidence needed to construct the answer. The slightly lower coverage of the boolean questions (92.93%) shows that some of them required paragraphs not directly contained in the query (question) for the final decision. The low coverage (48.80%) and containment (45.66%) of unanswerable questions represent the desired behaviour, where the RAG pipeline has difficulty in retrieving content when no answer exists in the original paper. The consistent pattern of high recall with low precision across all question types corresponds to our retrieval strategy, which uses similarity_top_k = 5 and a similarity cutoff at 20%.

The lecture transcripts RAG pipeline was evaluated on the automatically generated questions, showing better performance than our scientific paper pipeline. This can be seen through the 14.87% F_1-score and the overall coverage of 98.08%, which indicates the pipeline's better structure awareness. The performance improvement is mostly caused by the lecture-specific implementation, which includes subtopic-aware chunking to create semantically coherent passages and the inclusion of subtopic metadata, helping the system recognize the dependencies in the long lecture better. Precision also improved (16.99%) in comparison to the scientific papers RAG (12.91%); however, it is still on the lower end due to the same design decision we already described in detail.

Question Type	#Qs	F1	Recall	Precision	Containment	Coverage
Overall	729	14.87	86.87	16.99	87.33	98.08
Simple	163	32.60	92.15	33.10	92.18	99.39
Qasper-Like	210	10.18	89.29	12.53	89.95	98.57
Extractive	65	12.29	89.30	14.72	89.94	100.00
Abstractive	58	9.35	91.05	11.92	91.78	98.28
Boolean	51	9.25	89.06	11.37	89.80	98.04
Unanswerable	36	9.04	86.78	11.22	87.25	97.22
Bloom	362	9.50	83.01	12.21	83.56	97.24
Remember	59	10.57	89.18	13.43	89.88	100.00
Understand	57	10.45	82.00	13.50	82.63	92.98
Apply	68	10.12	84.50	12.77	85.07	98.53
Analyze	64	8.75	82.53	11.74	83.09	96.88
Evaluate	56	9.12	78.99	11.38	79.44	96.43
Create	52	7.90	80.17	10.38	80.51	98.08

Table 4.12.: RAG Evaluation on Lecture Transcripts

Examining each question type in detail, we see that simple questions achieved the best performance with an F_1-score of 32.60%, recall of 92.15%, and coverage of 99.39%. Qasper-like questions show consistent performance across all subcategories, with extractive questions achieving the highest score (F_1-Score: 12.29%, Coverage: 100%). The Bloom's taxonomy evaluation demonstrates the increase in complexity, with lower-order thinking skills (Remember: 10.57% F_1, Understand: 10.45% F_1) outperforming higher-order skills (Evaluate: 9.12% F_1, Create: 7.90% F_1). The decline in performance across the skill levels traces to the same reason seen in the abstractive and boolean questions in the scientific papers RAG pipeline, which is that complex reasoning tasks require a higher context variety to generate an appropriate answer.

4.3.5. Domain-Specific Evaluation

In order to evaluate the performance on the lecture transcripts, we first run the models on the test set of the automatically generated questions and calculate the F_1-Score of the answers. We experiment with three configurations: base models, models trained on the generated questions, and models trained on the combination of the generated questions and the datasets (Qasper and SQuAD). Additionally, we run all possible configurations on the preprocessed questions submitted from students in our survey.

4.3.5.1. Evaluation on Generated Questions

		Baseline		Generated Qs		Generated Qs + DS	
#Qs	Question Type	Llama	Qwen	Llama	Qwen	Llama	Qwen
729	Overall	18.06	19.13	36.09	30.68	37.78	40.40
157	Simple	14.69	15.03	32.47	25.69	38.26	36.63
210	Qasper-Like	22.43	21.95	43.06	35.82	45.62	52.10
65	Extractive	34.13	28.42	52.04	42.95	39.89	53.38
58	Abstractive	17.85	22.58	31.42	32.18	31.39	33.33
51	Boolean	25.72	25.31	37.15	35.58	35.83	42.95
36	Unanswerable	4.03	4.52	53.96	29.16	92.77	93.02
362	Bloom	16.98	19.28	33.62	29.85	33.03	35.25
59	Remember	28.82	26.82	44.67	42.48	37.58	48.50
57	Understand	19.22	21.01	37.27	33.06	36.00	37.26
68	Apply	14.04	18.58	31.83	28.39	34.00	36.10
64	Analyze	14.97	18.21	32.58	28.61	30.74	33.44
56	Evaluate	12.79	15.53	29.40	25.20	30.29	30.36
52	Create	12.32	15.24	26.18	20.83	29.35	24.71

Table 4.13.: Model F_1-Score Comparison evaluated on generated questions

The evaluation results demonstrate consistent improvement in model performance across training configurations, showing the models' ability to adapt to the educational domain of the lecture transcripts. The baseline models achieve moderate F_1-scores of 18.06% (Llama) and 19.13% (Qwen) on questions about the lecture transcripts, which are much lower in comparison to baseline models' capabilities on Qasper (~55%, Table 4.1). This performance decline demonstrates the significant domain gap between general academic content and lecture-specific materials, caused by the lack of structure in the conversational transcripts and the complex university-level terminology.

The most significant improvement occurs after training on the generated questions, doubling the F_1-scores to 36.09% and 30.68%, respectively. The combined training approach (generated questions + DS (= Qasper + SQuAD) yields the best overall performance, with Qwen achieving an F_1-score of 40.40%, indicating that combining verified QA datasets alongside domain-specific questions creates a more capable model that balances between general QA capabilities and specialized knowledge. Moreover, we notice that Qwen, trained on the mixed configuration, outperforms Llama on most question types, which reveals architectural superiority.

After examining each question type, we can see the models' abilities in more detail. The impressive increase in performance on unanswerable questions in the combined training configuration (4.03% $\longrightarrow$ 92.77% for Llama and 4.52% $\longrightarrow$ 93.02% for Qwen) reveals the improved instruction following abilities of the models after training on diverse data. However, the limited F_1-score improvement on higher-order Bloom's taxonomy question types, such as Create (Llama: 29.35%, Qwen: 24.71%) and Evaluate (Llama: 30.29%, Qwen: 30.36%), shows that the models still struggle with complex reasoning tasks that require synthesis and critical thinking. Overall, we can conclude that while domain-specific training significantly improves factual recall and comprehension (Qwen's performance on Remember questions improved from 28.82% to 48.50%), the models still struggle with the higher-order thinking skills that are essential for quality interaction with students.

4.3.5.2. Analysis and Evaluation of Student Questions

Our goal with the student question dataset was to evaluate the models on real-world examples, though the limited size of the collected questions proved to be a challenge. The dataset consists of questions submitted by university students, currently attending the lectures for which we have transcripts. We managed to collect only 15 question-answer pairs and had to do extensive manual pre-processing to adapt them to our prompt and RAG requirements (paragraph 3.2). The dataset contains 9 QA pairs in English and 6 in German, which is characteristic of the KIT multilingual environment, being a German university, while the lectures were held in English. The questions included topics such as technical implementation details ("What is the key property to have efficient sampling when denoising diffusion using DDPM?"), conceptual comparisons ("Was ist der Unterschied zwischen encoder-decoder und decoder-only models?") and theoretical understanding ("Why do we need both cross-attention and self-attention?"). An important property is that 7 out of the 15 questions began with "why", demonstrating the students' tendency for deeper understanding rather than factual recall. The answers revealed the learning experience of the students by including references to external tools ("I used ChatGPT"), comparison with other courses ("[...] in another lecture I saw"), and uncertain formulations, which were removed during pre-processing.

RAG Used	Trained On	Llama-3.1-8B		Qwen3-8B	
		F_1-Score	BERTScore	F_1-Score	BERTScore
No	-	12.79	80.84	18.27	82.43
Yes	-	11.58	80.55	16.47	82.65
Yes	Qasper	7.08	80.01	16.53	82.43
Yes	Qasper + SQuAD (DS)	8.71	79.88	14.56	82.11
Yes	Generated Questions	21.64	83.71	23.16	84.40
Yes	Generated Questions + DS	21.18	83.48	21.75	83.65

Table 4.14.: Model Performance Comparison evaluated on student questions

The results of the evaluation on student questions revealed different patterns in comparison to the evaluation on synthetic data, demonstrating the gap between automated and human-centered assessment. The baseline models without RAG achieve F_1-scores of 12.79% (Llama) and 18.27% (Qwen), which shows that Llama's performance differs significantly between automatic (18.06%) and human questions, whereas Qwen achieves a similar score (19.13%) (Table 4.13). Introducing RAG without domain-specific training slightly decreases the performance (Llama: 11.58%, Qwen: 16.47%); however, this can be due to the pre-processing step, in which we manually assign the lecture to the question, or to the fact that some questions expect cross-lecture knowledge.

The most important takeaway comes from the F_1-score increase when the models are trained on the generated questions, improving to 21.64% for Llama and 23.16% for Qwen. Even though the training data substantially differs from the authentic student questions in style, complexity, and focus, the domain-specific training successfully captures the main patterns of the educational context, which can then be transferred to real-world applications. The negligible difference between training on generated questions and training on a combination with QA datasets (21.18% vs. 21.64% for Llama, 21.75% vs. 23.16% for Qwen) points out that for real-world student questions, domain-specific data has a bigger influence than general QA abilities and scientific-domain knowledge. However, due to our small student questions dataset, this statement requires further validation with a higher-scale study. In comparison to the relatively low F_1-Scores, the BERTScores (80-84%) were consistently high across all training configurations, indicating that while the models may struggle with generating responses that exactly match the reference answers, they successfully capture the semantic meaning and contextual relevance.

5. Conclusion

In this chapter, we will address the research questions (section 1.1) and discuss future research directions for further improving conversational QA systems based on lecture transcripts.

5.1. Answers to Research Questions

Our experiments demonstrate that open-source LLMs show moderate but improvable performance on long-context documents. The baseline models evaluated on Qasper had similar mid-range F_1-Scores of ~55%, with satisfactory performance on extractive, binary, and unanswerable questions. However, on abstractive questions, the base models achieved F_1-scores of ~30%, which highlights the difficulties with questions that require complex reasoning and synthesis. This was further confirmed by the similar ROUGE-L and AlignScore results, which emphasize that factual consistency and syntactic similarity remain high for extractive questions, but decline when a more abstractive formulation is required. The high BERTScores (~90%) indicate that the generated answers were semantically similar to the gold answers, although they did not match them precisely. After fine-tuning on Qasper, the models showed consistent but moderate gains of ~2-3% in F_1 and ROUGE-L. This indicates that the performance on long documents can be improved, but the relatively small change points out that fine-tuning alone is insufficient to address limitations such as abstractive reasoning.

Comparing the single-dataset and mixed-dataset training configurations, we can conclude that incorporating SQuAD into the training data had a positive effect on the F_1-score of extractive (factual) questions, which shows that mixed training improved the models' information retrieval abilities. Evaluation on PeerQA further confirmed this, with Llama-3.1-8B achieving an AlignScore of 41.52%, indicating solid factual consistency. However, despite these improvements, most exact-match metrics remained higher for the models trained on Qasper-only, highlighting that SQuAD-enhanced training focused more on improving information retrieval rather than overall QA performance.

The biggest challenge we faced was the lack of domain-specific training data based on the lecture transcripts. Our approach for automatically generating questions using Qwen3-32B, RAG, and question-type-specific prompt engineering proved essential to the improvement of the model's performance on the transcripts. We generated 4,695 questions, split into diverse question types, ranging from simple factual questions to Bloom's taxonomy questions, which require higher-order thinking skills. The evaluation of the generated questions showed mixed results, with high scores for basic comprehension but lower performance on advanced instructor metrics, indicating that the questions were suitable for basic training purposes but lacked substantial educational value.

Evaluation of the models on the generated questions demonstrated substantial improvement across all question types, with the best scores achieved on the training configuration, which combines the generated questions with the existing datasets. The models had overall F_1-scores of 37.78% (Llama) and 40.04% (Qwen), which is approximately double the baseline performance of 18.06% and 19.13%, respectively. The most significant improvements were seen in the performance for unanswerable questions (~4% → ~93%), indicating better instruction following and recognition of unanswerable questions. Overall, the enrichment of the generated questions training dataset with existing QA datasets proved beneficial for maintaining general QA capabilities while enhancing domain-specific performance. Evaluation of student questions revealed that the models can improve their performance when trained on meaningful domain-specific data; however, due to the size of the collected real-world sample dataset, the conclusion was not as convincing.

The implemented enhancement techniques were essential for handling long contexts and domain-specific content. Our custom RAG pipelines for the Qasper papers and lecture transcripts showed excellent coverage (97-99%) and high containment scores (83-92%), demonstrating the retrieval effectiveness. Parameter-efficient fine-tuning using QLoRA showed only slight improvements on Qasper evaluation and satisfactory results on PeerQA. On the generated questions, it proved to be much more effective across all question types, with both the mixed and normal versions of the training data. Zero-shot prompt engineering techniques were crucial for maintaining consistency across all question types and datasets, while avoiding the context length limitations that would occur when using few-shot prompting with full lecture transcripts. Overall, the combination of the enhancement techniques proved to be most effective on domain-specific lecture content, showing that specialized approaches are needed in order to improve the performance of conversational QA systems.

5.2. Future Work

We will highlight various possibilities for future research, which we gathered based on our research and experiment results. To provide more context and visual information, the QA system would benefit from the implementation of multimodal support, which combines the lecture slides with the transcripts (which include timestamps). Another challenge we faced in our research was the consistently lower performance on abstractive questions, which could be improved through specialized training on reasoning and summarization datasets or through implementing Chain-of-Thought prompting techniques for multi-step reasoning. Additionally, exploring newer model capabilities such as Qwen's thinking mode could improve reasoning abilities. Another possible direction is the implementation of advanced RAG techniques, particularly hierarchical retrieval structures (paragraph 2.3.1), which would further improve the lecture-aware chunking strategy.

Bibliography

[1] Joshua Ainslie et al. “GQA: Training generalized multi-query transformer models from multi-head checkpoints”. In: *arXiv preprint arXiv:2305.13245* (2023).

[2] Yushi Bai et al. “LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding”. In: *CoRR* abs/2308.14508 (2023). DOI: 10.48550/ARXIV.2308.14508. arXiv: 2308.14508. URL: https://doi.org/10.48550/arXiv.2308.14508.

[3] Tim Baumgärtner, Ted Briscoe, and Iryna Gurevych. “PeerQA: A Scientific Question Answering Dataset from Peer Reviews”. In: *CoRR* abs/2502.13668 (2025). DOI: 10.48550/ARXIV.2502.13668. arXiv: 2502.13668. URL: https://doi.org/10.48550/arXiv.2502.13668.

[4] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.

[5] Michael Caballero. “A Brief Survey of Question Answering Systems”. In: *International Journal of Artificial Intelligence Applications* 12 (Sept. 2021), pp. 01–07. DOI: 10.5121/ijaia.2021.12501.

[6] Arie Cattan et al. “Can Few-shot Work in Long-Context? Recycling the Context to Generate Demonstrations”. In: *CoRR* abs/2406.13632 (2024). DOI: 10.48550/ARXIV.2406.13632. arXiv: 2406.13632. URL: https://doi.org/10.48550/arXiv.2406.13632.

[7] Anthony Chen et al. “Evaluating Question Answering Evaluation”. In: *Proceedings of the 2nd Workshop on Machine Reading for Question Answering, MRQA@EMNLP 2019, Hong Kong, China, November 4, 2019*. Ed. by Adam Fisch et al. Association for Computational Linguistics, 2019, pp. 119–124. DOI: 10.18653/V1/D19-5817. URL: https://doi.org/10.18653/v1/D19-5817.

[8] Chih-Ming Chen and Chung-Hsin Wu. “Effects of different video lecture types on sustained attention, emotion, cognitive load, and learning performance”. In: *Comput. Educ.* 80 (2015), pp. 108–121. DOI: 10.1016/J.COMPEDU.2014.08.015. URL: https://doi.org/10.1016/j.compedu.2014.08.015.

[9] Hyung Won Chung et al. “Scaling Instruction-Finetuned Language Models”. In: *J. Mach. Learn. Res.* 25 (2024), 70:1–70:53. URL: https://jmlr.org/papers/v25/23-0870.html.

[10] Tri Dao. “FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning”. In: *CoRR* abs/2307.08691 (2023). DOI: 10.48550/ARXIV.2307.08691. arXiv: 2307.08691. URL: https://doi.org/10.48550/arXiv.2307.08691.

[11] Tri Dao et al. "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness". In: *CoRR* abs/2205.14135 (2022). DOI: `10.48550/ARXIV.2205.14135`. arXiv: `2205.14135`. URL: `https://doi.org/10.48550/arXiv.2205.14135`.

[12] Pradeep Dasigi et al. "A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers". In: *CoRR* abs/2105.03011 (2021). arXiv: `2105.03011`. URL: `https://arxiv.org/abs/2105.03011`.

[13] Florian Dessloch et al. "KIT Lecture Translator: Multilingual Speech Translation with One-Shot Learning". In: *COLING 2018, The 27th International Conference on Computational Linguistics: System Demonstrations, Santa Fe, New Mexico, August 20-26, 2018*. Ed. by Dongyan Zhao. Association for Computational Linguistics, 2018, pp. 89–93. URL: `https://aclanthology.org/C18-2020/`.

[14] Tim Dettmers et al. "QLoRA: Efficient Finetuning of Quantized LLMs". In: *CoRR* abs/2305.14314 (2023). DOI: `10.48550/ARXIV.2305.14314`. arXiv: `2305.14314`. URL: `https://doi.org/10.48550/arXiv.2305.14314`.

[15] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: `1810.04805`. URL: `http://arxiv.org/abs/1810.04805`.

[16] Abhimanyu Dubey et al. "The Llama 3 Herd of Models". In: *CoRR* abs/2407.21783 (2024). DOI: `10.48550/ARXIV.2407.21783`. arXiv: `2407.21783`. URL: `https://doi.org/10.48550/arXiv.2407.21783`.

[17] Wenqi Fan et al. "A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models". In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*. Ed. by Ricardo Baeza-Yates and Francesco Bonchi. ACM, 2024, pp. 6491–6501. DOI: `10.1145/3637528.3671470`. URL: `https://doi.org/10.1145/3637528.3671470`.

[18] Said al Faraby, Ade Romadhony, and Adiwijaya. "Analysis of LLMs for educational question classification and generation". In: *Comput. Educ. Artif. Intell.* 7 (2024), p. 100298. DOI: `10.1016/J.CAEAI.2024.100298`. URL: `https://doi.org/10.1016/j.caeai.2024.100298`.

[19] Yunfan Gao et al. "Retrieval-Augmented Generation for Large Language Models: A Survey". In: *CoRR* abs/2312.10997 (2023). DOI: `10.48550/ARXIV.2312.10997`. arXiv: `2312.10997`. URL: `https://doi.org/10.48550/arXiv.2312.10997`.

[20] Yoav Goldberg. "A Primer on Neural Network Models for Natural Language Processing". In: *CoRR* abs/1510.00726 (2015). arXiv: `1510.00726`. URL: `http://arxiv.org/abs/1510.00726`.

[21] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN: 978-0-262-03561-3. URL: `http://www.deeplearningbook.org/`.

[22] Cheng-Ping Hsieh et al. "RULER: What's the Real Context Size of Your Long-Context Language Models?" In: *CoRR* abs/2404.06654 (2024). DOI: `10.48550/ARXIV.2404.06654`. arXiv: `2404.06654`. URL: `https://doi.org/10.48550/arXiv.2404.06654`.

[23] Edward J. Hu et al. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *CoRR* abs/2106.09685 (2021). arXiv: 2106.09685. URL: https://arxiv.org/abs/2106.09685.

[24] Aishwarya Kamath et al. “Gemma 3 Technical Report”. In: *CoRR* abs/2503.19786 (2025). DOI: 10.48550/ARXIV.2503.19786. arXiv: 2503.19786. URL: https://doi.org/10.48550/arXiv.2503.19786.

[25] Jared Kaplan et al. “Scaling Laws for Neural Language Models”. In: *CoRR* abs/2001.08361 (2020). arXiv: 2001.08361. URL: https://arxiv.org/abs/2001.08361.

[26] Ghader Kurdi et al. “A Systematic Review of Automatic Question Generation for Educational Purposes”. In: *Int. J. Artif. Intell. Educ.* 30.1 (2020), pp. 121–204. DOI: 10.1007/S40593-019-00186-Y. URL: https://doi.org/10.1007/s40593-019-00186-y.

[27] Unggi Lee et al. “Few-shot is enough: exploring ChatGPT prompt engineering method for automatic question generation in english education”. In: *Educ. Inf. Technol.* 29.9 (2024), pp. 11483–11515. DOI: 10.1007/S10639-023-12249-8. URL: https://doi.org/10.1007/s10639-023-12249-8.

[28] Patrick Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. In: *CoRR* abs/2005.11401 (2020). arXiv: 2005.11401. URL: https://arxiv.org/abs/2005.11401.

[29] Dawei Li et al. “From Generation to Judgment: Opportunities and Challenges of LLM-as-a-judge”. In: *CoRR* abs/2411.16594 (2024). DOI: 10.48550/ARXIV.2411.16594. arXiv: 2411.16594. URL: https://doi.org/10.48550/arXiv.2411.16594.

[30] Zongxi Li et al. “Retrieval-augmented generation for educational application: A systematic survey”. In: *Computers and Education: Artificial Intelligence* 8 (2025), p. 100417. ISSN: 2666-920X. DOI: https://doi.org/10.1016/j.caeai.2025.100417. URL: https://www.sciencedirect.com/science/article/pii/S2666920X25000578.

[31] Nelson F. Liu et al. “Lost in the Middle: How Language Models Use Long Contexts”. In: *Trans. Assoc. Comput. Linguistics* 12 (2024), pp. 157–173. DOI: 10.1162/TACL_A_00638. URL: https://doi.org/10.1162/tacl%5C_a%5C_00638.

[32] Jiya Manchanda et al. “The Open Source Advantage in Large Language Models (LLMs)”. In: *CoRR* abs/2412.12004 (2024). DOI: 10.48550/ARXIV.2412.12004. arXiv: 2412.12004. URL: https://doi.org/10.48550/arXiv.2412.12004.

[33] Aidan Michael McCarthy et al. “Digital transformation in education: Critical components for leaders of system change”. In: *Social Sciences Humanities Open* 8.1 (2023), p. 100479. ISSN: 2590-2911. DOI: https://doi.org/10.1016/j.ssaho.2023.100479. URL: https://www.sciencedirect.com/science/article/pii/S2590291123000840.

[34] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *CoRR* abs/2203.02155 (2022). DOI: 10.48550/ARXIV.2203.02155. arXiv: 2203.02155. URL: https://doi.org/10.48550/arXiv.2203.02155.

[35] Pranav Rajpurkar et al. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250 (2016). arXiv: 1606.05250. URL: http://arxiv.org/abs/1606.05250.

[36] Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. "A Survey of Evaluation Metrics Used for NLG Systems". In: *ACM Comput. Surv.* 55.2 (2023), 26:1–26:39. DOI: 10.1145/3485766. URL: https://doi.org/10.1145/3485766.

[37] Nicy Scaria, Suma Dharani Chenna, and Deepak N. Subramani. "Automated Educational Question Generation at Different Bloom's Skill Levels using Large Language Models: Strategies and Evaluation". In: *CoRR* abs/2408.04394 (2024). DOI: 10.48550/ARXIV.2408.04394. arXiv: 2408.04394. URL: https://doi.org/10.48550/arXiv.2408.04394.

[38] Craig W. Schmidt et al. "Tokenization Is More Than Compression". In: *CoRR* abs/2402.18376 (2024). DOI: 10.48550/ARXIV.2402.18376. arXiv: 2402.18376. URL: https://doi.org/10.48550/arXiv.2402.18376.

[39] Sander Schulhoff et al. "The Prompt Report: A Systematic Survey of Prompting Techniques". In: *CoRR* abs/2406.06608 (2024). DOI: 10.48550/ARXIV.2406.06608. arXiv: 2406.06608. URL: https://doi.org/10.48550/arXiv.2406.06608.

[40] Brandon Smith and Anton Troynikov. *Evaluating Chunking Strategies for Retrieval.* Tech. rep. Chroma, July 2024. URL: https://research.trychroma.com/evaluating-chunking.

[41] Hugo Touvron et al. "Llama 2: Open Foundation and Fine-Tuned Chat Models". In: *CoRR* abs/2307.09288 (2023). DOI: 10.48550/ARXIV.2307.09288. arXiv: 2307.09288. URL: https://doi.org/10.48550/arXiv.2307.09288.

[42] Hugo Touvron et al. "LLaMA: Open and Efficient Foundation Language Models". In: *CoRR* abs/2302.13971 (2023). DOI: 10.48550/ARXIV.2302.13971. arXiv: 2302.13971. URL: https://doi.org/10.48550/arXiv.2302.13971.

[43] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[44] Chengliang Wang et al. "Education reform and change driven by digital technology: a bibliometric study from a global perspective". In: *Humanities and Social Sciences Communications* 11.1 (2024), p. 256. DOI: 10.1057/s41599-024-02717-y. URL: https://doi.org/10.1057/s41599-024-02717-y.

[45] Jason Wei et al. "Finetuned Language Models Are Zero-Shot Learners". In: *CoRR* abs/2109.01652 (2021). arXiv: 2109.01652. URL: https://arxiv.org/abs/2109.01652.

[46] An Yang et al. "Qwen3 Technical Report". In: *CoRR* abs/2505.09388 (2025). DOI: 10.48550/ARXIV.2505.09388. arXiv: 2505.09388. URL: https://doi.org/10.48550/arXiv.2505.09388.

[47] Murong Yue. "A Survey of Large Language Model Agents for Question Answering". In: *CoRR* abs/2503.19213 (2025). DOI: 10.48550/ARXIV.2503.19213. arXiv: 2503.19213. URL: https://doi.org/10.48550/arXiv.2503.19213.

[48] Munazza Zaib et al. “Conversational question answering: a survey”. In: *Knowl. Inf. Syst.* 64.12 (2022), pp. 3151–3195. DOI: 10.1007/S10115-022-01744-Y. URL: https://doi.org/10.1007/s10115-022-01744-y.

[49] Yuheng Zha et al. “AlignScore: Evaluating Factual Consistency with a Unified Alignment Function”. In: *CoRR* abs/2305.16739 (2023). DOI: 10.48550/ARXIV.2305.16739. arXiv: 2305.16739. URL: https://doi.org/10.48550/arXiv.2305.16739.

[50] Dell Zhang and Wee Sun Lee. “Question classification using support vector machines”. In: *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada* (2003). Ed. by Charles L. A. Clarke et al., pp. 26–32. DOI: 10.1145/860435.860443. URL: https://doi.org/10.1145/860435.860443.

[51] Wayne Xin Zhao et al. “A Survey of Large Language Models”. In: *CoRR* abs/2303.18223 (2023). DOI: 10.48550/ARXIV.2303.18223. arXiv: 2303.18223. URL: https://doi.org/10.48550/arXiv.2303.18223.

A. Appendix

A.1. Prompts

In this section, we will describe all of the prompts we used for our experiments. Our placeholder <CONTEXT> denotes either the scientific paper text or the lecture transcript. <QUESTION> and <ANSWER> are placeholders for the specific question and answer based on the provided context.

A.1.1. LongBench

You are given a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write “Unanswerable”. If the question is a yes/no question, answer “Yes”, “No”, or “Unanswerable”. Do not provide any explanation.
Article: <CONTEXT>
Answer the question based on the above article as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write “Unanswerable”. If the question is a yes/no question, answer “Yes”, “No”, or “Unanswerable”. Do not provide any explanation.
Question: <QUESTION>
Answer:

A.1.2. PeerQA Prompt (Our Version)

Read the following paper and answer the question. If the paper does not answer the question, answer with "Unanswerable".
Question: <QUESTION>
Paper: <CONTEXT>
Answer:

A.1.3. Simple Question Generation Prompt

After reading the lecture transcript, generate one question you would ask to make sure you understood the most important concepts, and provide a concise, short answer to the question. Do not give any explanations.
Lecture Transcript: <CONTEXT>
After reading the lecture transcript, generate one question you would ask to make sure you understood the most important concepts, and provide a concise, short answer to the

question. Do not give any explanations.
Question:
Answer:

A.1.4. Bloom's Question Generation Prompt (Our Version)

The following are the revised Bloom's skill levels and their explanations:
1. Remember: Retrieve relevant knowledge from long-term memory
2. Understand: Construct meaning from instructional messages, including oral, written and graphic communication
3. Apply: Carry out or use a procedure in a given situation
4. Analyze: Break material into foundational parts and determine how parts relate to one another and the overall structure or purpose
5. Evaluate: Make judgments based on criteria and standards
6. Create: Put elements together to form a coherent whole; reorganize into a new pattern or structure

Based on the following lecture transcript, create exactly 6 questions - one for each Bloom's taxonomy level. Each question should test students on the concepts presented in the lecture.

Lecture Transcript: <CONTEXT>

For each question, provide the skill level, the question, and a concise answer. Format your response exactly as follows:

Level: Remember
Question: [Your question here]
Answer: [Your answer here]

Level: Understand
Question: [Your question here]
Answer: [Your answer here]

Level: Apply
Question: [Your question here]
Answer: [Your answer here]

Level: Analyze
Question: [Your question here]
Answer: [Your answer here]

Level: Evaluate
Question: [Your question here]
Answer: [Your answer here]

Level: Create
Question: [Your question here]
Answer: [Your answer here]

A.1.5. Qasper-like Question Generation Prompt

The following are the four Qasper question types and their detailed characteristics:
1. Extractive: Questions that require specific facts, details, or direct information from the source material. The answer consists of direct citations or exact phrases from the lecture content.
2. Abstractive: Questions that ask for explanations, summaries, or conceptual understanding that requires synthesizing information. The answer is free-form text that explains concepts rather than quoting directly.
3. Boolean: Yes/no questions that test understanding of specific claims, relationships, or rules presented in the lecture. The answer is binary (yes/no) followed by a brief justification.
4. Unanswerable: Questions that ask for information not covered or addressed in the lecture transcript, testing the model's ability to recognize information gaps.

Based on the following lecture transcript, create exactly 4 questions - one for each Qasper question type. Each question should test students on the concepts presented in the lecture.

Lecture Transcript: <CONTEXT>

For each question, provide the type, the question, and a complete answer. Format your response exactly as follows:
Type: Extractive
Question: [Your question here]
Answer: [Your answer here]

Type: Abstractive
Question: [Your question here]
Answer: [Your answer here]

Type: Boolean
Question: [Your question here]
Answer: [Your answer here]

Type: Unanswerable
Question: [Your question here]
Answer: [Your answer here]

A.1.6. Question-Answer Pair Quality Assessment Prompt

Evaluate the following QUESTION-ANSWER PAIR on multiple criteria using a scale of 1-10 (10 being excellent).

Question: <QUESTION>
Answer: <ANSWER>
Context: <CONTEXT>

Evaluate BOTH the question AND answer on these 6 criteria:

1. Helpfulness: How much do the question and answer together assist student understanding of the lecture?
2. Relevance: How relevant are both the question and answer to important concepts from the lecture?
3. Harmlessness: Are both the question and answer free from bias, misinformation, inappropriate content, or any elements that could mislead or harm university students?
4. Reliability: How consistently can the question be answered correctly from the context, and how factually accurate is the provided answer?
5. Feasibility: Is the question realistically answerable by university students, and is the answer at an appropriate comprehension level?
6. Overall Quality: Considering all aspects (educational value, clarity, appropriateness), how would you rate the overall quality of this question-answer pair?

Provide your response in this exact format:
Helpfulness: [score]
Relevance: [score]
Harmlessness: [score]
Reliability: [score]
Feasibility: [score]
Overall Quality: [score]

A.1.7. Bloom Evaluation Prompt

Please make sure you read and understand the following content and instructions carefully.

Question: <QUESTION>
Lecture excerpt: <CONTEXT>
Answer: <ANSWER>
EVALUATION CRITERIA (Scale 1-10, where 10 is excellent):
Understandable: How well could you understand what the question is asking? Rate if the question can be comprehended by a student.
ContextRelated: How much is the question related to the given lecture excerpt? Rate if the question pertains directly to the key themes or concepts of excerpt.

Grammatical: How grammatically well-formed is the question? Rate if the question adheres to the rules of English grammar.
Clear: How clear is what the question asks for? Rate if the phrasing of the question leaves any doubt about what is being asked or if there's vagueness making it difficult to answer.
Answerable: How well can students answer the question with the information or context provided? Rate if the question is answerable using the knowledge that students are expected to have from the course material.
Central: How important is being able to answer this question for working on the course topic? Rate if answering the question requires understanding of key concepts critical to the subject matter.
WouldYouUseIt: How likely would you be to use this question if you were teaching this course topic? Rate the practical value for teaching and learning, and inclusion in course materials or assessments.
Educational_Value: How valuable is this question for student learning and assessment? Rate the overall educational worth.
Question_Quality: How well-crafted is the question in terms of clarity, specificity, and pedagogical effectiveness?
Difficulty_Appropriateness: How appropriate is the difficulty level for graduate computer science students?
Cognitive_Engagement: How well does the question engage students in meaningful cognitive processes?
Assessment_Utility: How useful would this question be in accurately assessing student understanding?
Rephrase: Could you rephrase the question to make it clearer and error-free? If yes, provide the rephrased version; if no, explain why not.
SkillLevel: What is the Bloom's skill associated with the question? Options: remember, understand, apply, analyze, evaluate, create, NA
Provide your response in this exact format:
Understandable: [score]
ContextRelated: [score]
Grammatical: [score]
Clear: [score]
Answerable: [score]
Central: [score]
WouldYouUseIt: [score]
Educational_Value: [score]
Question_Quality: [score]
Difficulty_Appropriateness: [score]
Cognitive_Engagement: [score]
Assessment_Utility: [score]
Rephrase: [rephrased question if applicable, otherwise "No rephrase needed" or explanation]
SkillLevel: [remember/understand/apply/analyze/evaluate/create/NA]

A.2. RAG Queries

A.2.1. Simple Question Query

A.2.1.1. English Version

key concepts explanations methods definitions facts

A.2.1.2. German Version

wichtige Konzepte Erklärungen Methoden Definitionen Fakten

A.3. Lecture Transcripts

A.3.1. Pre-processing with Subtopics

Introduction: So welcome to the final lecture of NLLP. So today is the last lecture. You will have a practical on Thursday. If you have specific questions, so there's a forum about questions, what you would like to discuss, please provide input for that so that we can discuss the topics. Since there are parallel lectures there will be a recording of the lecture so you can also follow remotely, but we can't provide like. [...]
Why NLP is difficult: Before that, we started with the overview, and that's of course also a bit. Important, that's mainly from the introductional slide, but what you should remember there is very important is why NLP is difficult, a bit like how NLP systems evolve and how we can solve NLP tasks. Maybe it was all mainly done at the beginning. There are several challenges, [...]

A.3.2. Pre-processing without Subtopics

0: So welcome to the lecture advanced artificial intelligence, my name is Yannius. I'm here a professor for language technologies. Is a bit like the basics of what research I'm doing, and therefore I'm very happy to have the opportunity here this semester to teach this lecture on advanced. I'm Here Since Since One Year I'm Mainly Doing. Applying AI for different types of language technologies, for example machine translation, speech recognition or similar things like big language models as most of you have heard. And so you will see that during the lecture we will always see a bit of applications. And say wee because will not do this on my own. This lecture will be given by three people myself [...]