

Arbabha, Darius and Mohammad, Saknini, and Shaji, Shinas

Creating exam questions is tedious for professors and often time-consuming, distracting them from their primary responsibilities of teaching and research. Developing a large language model (LLM) pipeline to generate exam questions from lecture notes aims to streamline this process and enhance efficiency and accuracy. This automated solution ensures a comprehensive coverage of course material and produces high-quality, consistent questions. It allows customization to suit different question formats and difficulty levels. Ultimately, this pipeline should enable educators to focus more on student engagement and the quality of instruction rather than stressing about exam questions.

The diagram illustrates a Retrieval-Augmented Generation (RAG) system for generating and evaluating questions from lecture content. The process is divided into three main stages: Preprocessing, Questions Generation, and Question Evaluation.

Preprocessing

Input: Lectures (PDFs) and Assignments (Python code).

Text extraction: Text extraction from PDFs via PyMuPdf then filtered by passing into an LLM that removes nonsense by prompting.

Image processing: Using a Visual Question Answering (VQA) model along side context from the lecture to generate a description of the images on the slides was used in order to minimize the loss of content.

Table extraction: Directly extracting text from tables has resulted in inaccuracies during question generation hence we opted to using table extraction and passing an html table rather than raw text.

Assignment Injection: The assignments have been manually inspected and added as a dependency for the corresponding lectures that provide the knowledge to solve them.

Questions Generation

System Prompt: You are a Professor. Your task is to setup questions for an upcoming NLP quiz/examination. The questions should be diverse in nature across the slides. Restrict the questions to the context information provided.

Questions Pool: The system generates questions based on prompts: "How many free text questions?", "How many MC questions?", and "How many coding questions?". These prompts lead to "Free Text Prompt", "MC Questions Prompt", and "Code Question Prompt" respectively, which then generate "MC Questions", "Free Text Questions", and "Coding Questions".

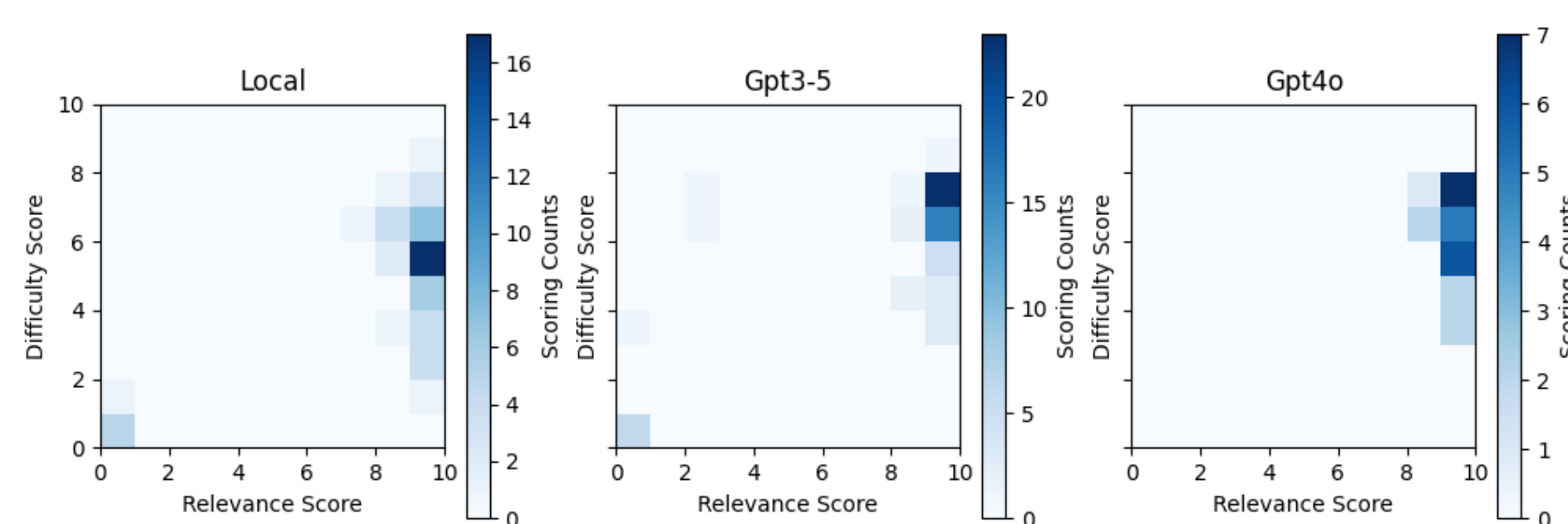
Question Evaluation

System Prompt: You are given the task of evaluating an examination question given the lecture content within <lecture> </lecture> and question within <question> </question> tags. Always provide a response in the following format using the appropriate tags: <reasoning>explain your evaluation in detail, including the section of the lecture that the question covers and your reasoning for the evaluation in markdown text and close with</reasoning> <relevance>an integer from 0 to 10...

Evaluation Result: The system evaluates questions based on "Question Prompt" and "Lecture Context". The results are categorized into "Difficulty Score", "Relevance Score", "Overall Coverage", and "LLM Reasoning LLM Answer".

The diagram also includes a section for **Model Selection**, listing various models: UNa-ThePitbull-21.4B-v2, Mistral-7B-Instruct-v0.3, GPT-4o, and GPT-3.5-turbo.

Heatmap of the self-evaluated difficulty and relevance scores of generated questions



Arbabha, Darius and Mohammad, Saknini, and Shaji, Shinas
Hochschule Bonn-Rhein-Sieg
Email: firstname.lastname@smail.inf.h-bris.de



Github Repository

Thanks to Prof. Dr. Jörn Hees and Tim Metzler for the materials and the insightful lecture **Natural Language Processing**.
We want to also thank Fraunhofer INT for allowing us to use an OpenAI API key for this project.

Group 1

- What is the importance of adding special tokens like [CLS], [PAD], [SEP], and [MASK] during the preprocessing of text data for transformer models?
- What is a Retrieval Augmented Generator (RAG) system, and how does it help in reducing the context size for document retrieval and generation tasks?
- What is the purpose of using Term Frequency – Inverse Document Frequency (TFIDF)?

Group 2

- Explain how one-hot encoding works and provide an example using a simple sentence.
- How do transformers solve the problem of parallelization in sequence-to-sequence models and why is this significant for NLP tasks?
- What is the primary purpose of training language models with human feedback?

Group 3

- How does prefix tuning differ from parameter-efficient fine-tuning methods like LoRa, in terms of their approach to updating parameters while maintaining model efficiency?
- What is the intuition behind the smoothing techniques in statistical language modeling, and how do they help with the sparsity issue of n-gram models?
- What is the role of the Byte Pair Encoding (BPE) token learner algorithm in text preprocessing?

Group 4

- What is model adaption and what is used for? Name at least 2 types of model adaption, explain them and provide an example for each of them.
- What does a positional encoder in the Transformer Encoder do? How does it effect the self-attention mechanism?
- How does TF-IDF differ from One Hot Encoding? What are the advantages of TF-IDF over One Hot Encoding?

Poll	Human	GPT3.5	GPT4o	Local
"Human" votes				