

Team Project HWS 2024

Web2Table: Using LLMs to Generate Tables about Arbitrary Topics from the Web



- **Prof. Dr. Christian Bizer**
- Professor for Information Systems V
- Research Interests:
 - Web-based Systems
 - Large-scale Data Integration
 - Data and Web Mining
- Room: B6, 26 - B1.15
- eMail: christian.bizer@uni-mannheim.de
- Consultation: Wednesday, 13:30-14:30



- **M. Sc. Wi-Inf. Alexander Brinkmann**
- Graduate Research Associate
- Research Interests:
 - Data Search using Deep Learning
 - Information Extraction using LLMs
- Room: B6, 26, C 1.04
- eMail: alexander.brinkmann@uni-mannheim.de



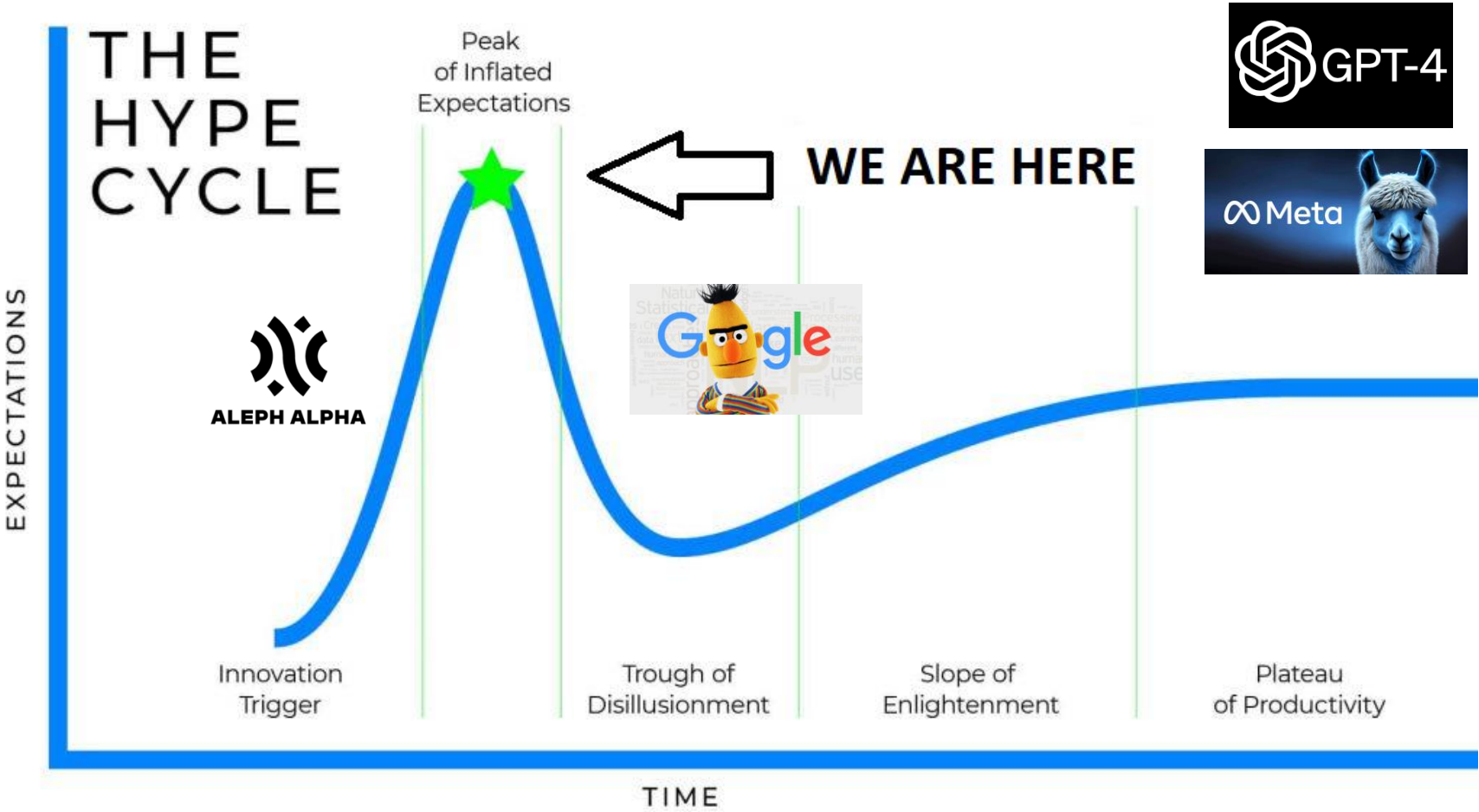
Agenda of Today's Kickoff Meeting

1. A round of introductions: You and Your Experience
2. Introduction to (Augmented) LLMs
3. Project Goals
4. Organization
5. Specific Subtasks
6. Schedule
7. Formal Requirements
8. Related Work

You and Your Experience

- A Short Round of Introductions
 - What are you studying? Which semester?
 - Which DWS courses did you already attend?
 - What are your programming and data wrangling skills?
 - What experience do you have with LLMs and prompt engineering?
- Participants
 1. Starke, Leonie
 2. Maimaiti, Yusufujiang
 3. Dhattrak, Ninad
 4. Ahmed, Zain
 5. Wu, Jui-Ling
 6. Hu, Rong
 7. Kallcishta, Sindi
 8. Nosova, Elizaveta
 9. Sattler, Ursula

Large Language Models



Disadvantages of LLMs

As at its core they are statistical models and can generate text based on the pre-training patterns they have learned, LLMs display certain disadvantages:

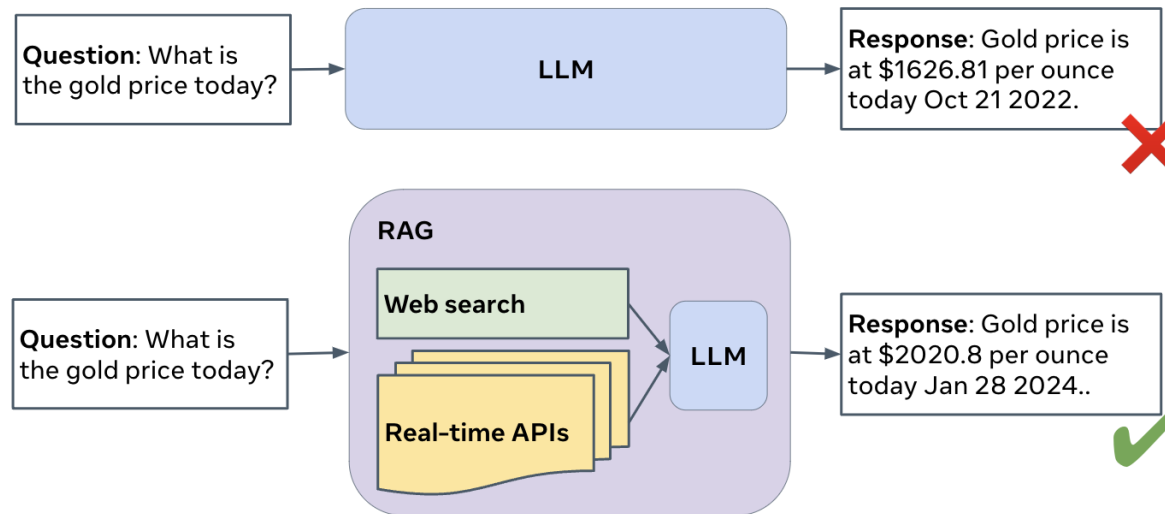
- They **have problems with advanced reasoning**, e.g. mathematical reasoning
- They may display **factual errors**, this problem is also referred as *hallucinations*
- LLMs **do not contain detailed information about many long-tail entities**, such as products, events, local businesses, or music recordings
- Knowledge stored in LLMs may be **outdated or incorrect**, as it depends on the training corpus

Borji, Ali. "A categorical archive of chatgpt failures." arXiv:2302.03494 (2023).

Bang, Yejin, et al. "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity." arXiv:2302.04023 (2023).

Augmented LLMs

- To overcome these disadvantages, LLMs can be *augmented* with *information* and *tools*
 - Pairing with an LLM a **python interpreter** to perform mathematical reasoning
 - LLMs can be combined with **visual language models** in order to control physical robots
 - The prompts of LLMs can augment with **retrieved documents, data from external APIs** to overcome non-factual and outdated information
- **Retrieval Augmented Question Answering**



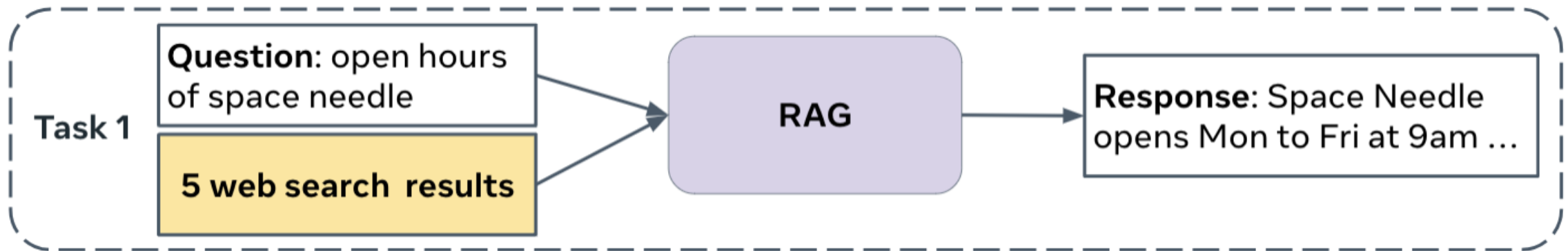
Mialon, et al.: **Augmented Language Models: a Survey**. arXiv:2302.07842 [cs.CL]

He, Hangfeng, Hongming Zhang, and Dan Roth. "Rethinking with retrieval: Faithful large language model inference." arXiv:2301.00303 (2022).

CRAG: Comprehensive RAG Benchmark

- A retrieval-augmented generative (RAG) question-answering (QA) system produces answers based on information retrieved from external sources or from the model's internal knowledge.
- The generated answer should provide useful information without introducing hallucinations or harmful content, such as profanity.
- **CRAG** evaluates retrieval-augmented generative (RAG) systems across five domains and eight question types, including both simple and complex questions, to assess their ability to handle time-sensitive facts, entity popularity, and reasoning challenges.

∞ Meta KDD Cup 2024
CRAG: Comprehensive RAG Benchmark



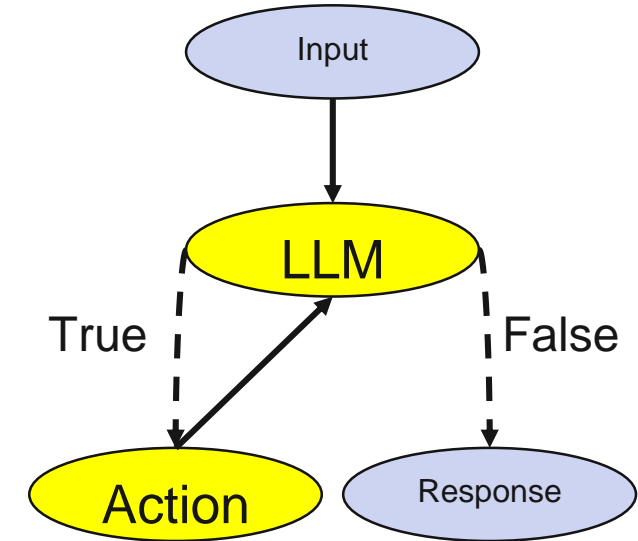
LLM Agents – The ReAct Architecture

Agents use an LLM to choose a sequence of actions to take.

ReAct is an architecture for building agents that combines „Reasoning“ and „Acting“.

1. The LLM thinks about **what step to take** based on the **input and previous observations**.
2. The LLM **chooses an action** from available tools e.g. web search, calculator or responds to the user.
3. The LLM generates arguments for the chosen tool e.g. a search query.
4. The **agent runtime (executor)** parses the chosen tool from the model output and calls it with the generated arguments.
5. The executor returns the results of the tool call back to the LLM as an **observation**.

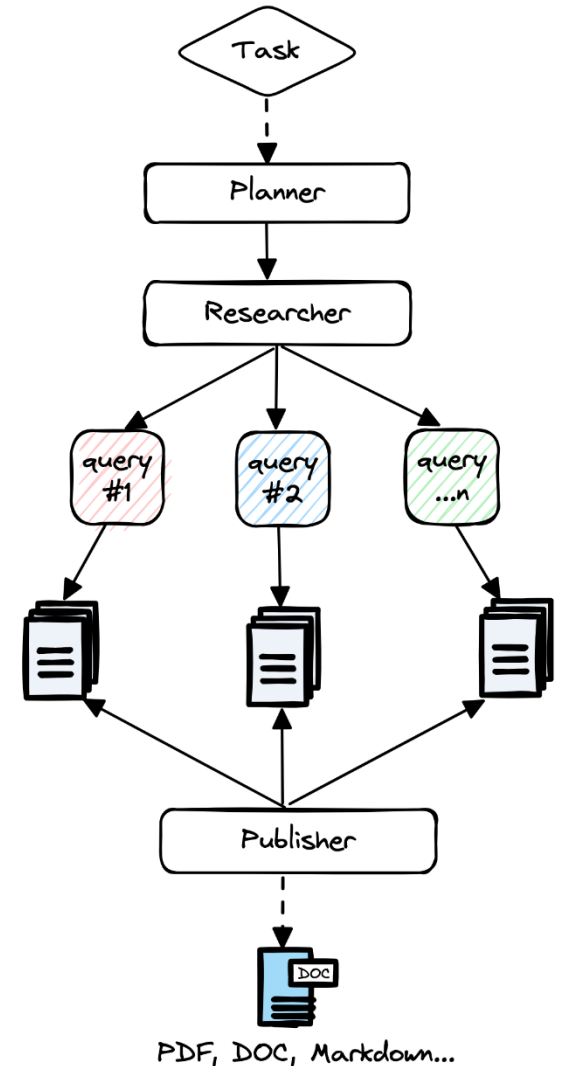
This process repeats until the agent chooses to respond.



Yao, et al: **ReAct: Synergizing Reasoning and Acting in Language Models**, arXiv:2210.03629 (2023)

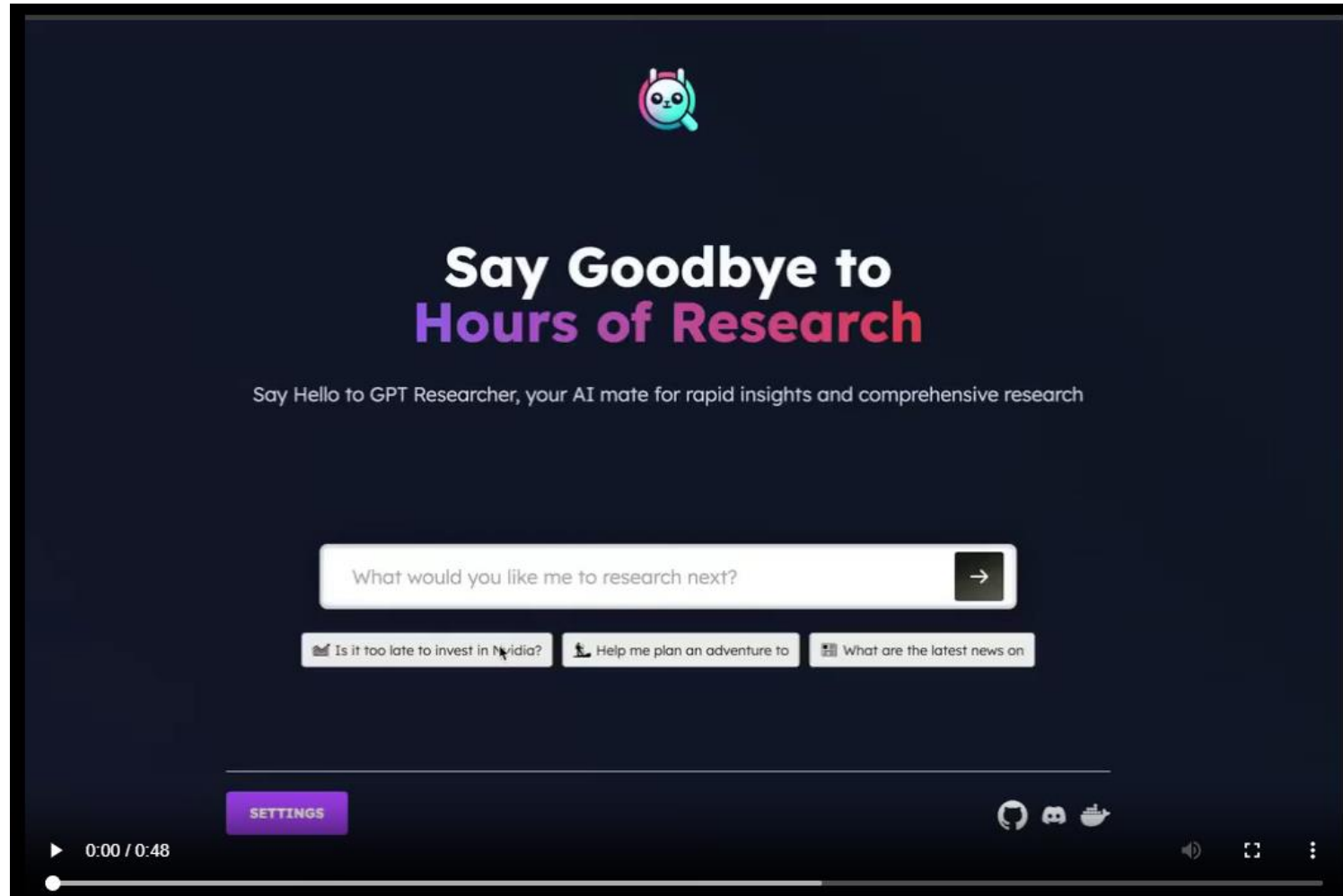
GPT-Researcher

- [GPT Researcher](#) is an autonomous agent designed for comprehensive online research on a variety of tasks.
- GPT Researcher runs a planner agent, researcher agents and a publisher agent.
 - The **planner** generates questions to research.
 - The **researchers** retrieve related information for each research question.
 - The **publisher** filters and aggregates all related information and creates a research report.



Shao, et al. **Assisting in Writing Wikipedia-like Articles From Scratch with Large Language Models**, arXiv:2402.14207 (2024)

Video Showcasing GPT Researcher

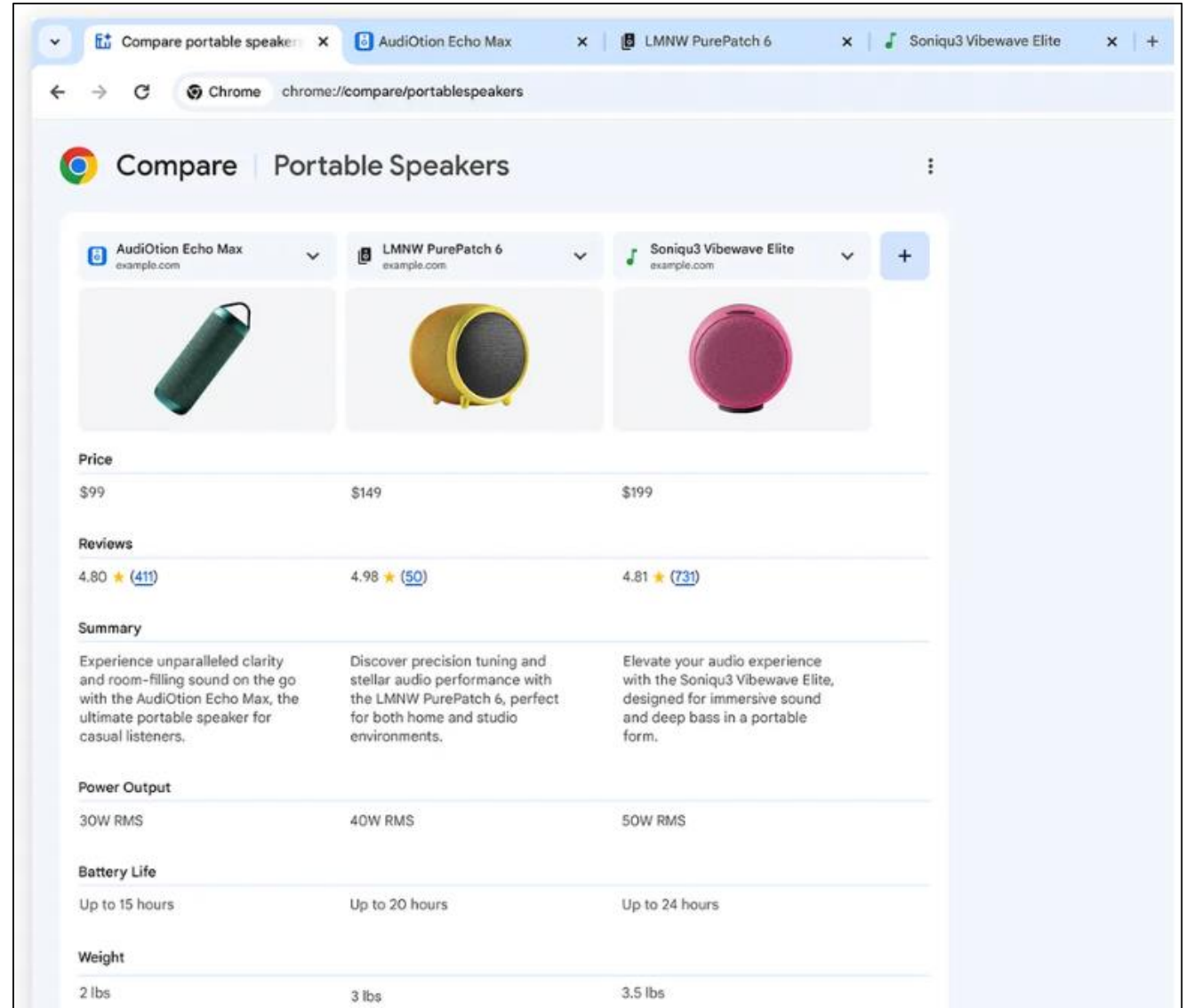


<https://gptr.dev/> or <https://github.com/user-attachments/assets/092e9e71-7e27-475d-8c4f-9ddddd28934a3>




Chrome: Tab Compare

US-version of Chrome web browser generates product comparison tables from open tabs

<https://blog.google/products/chrome/google-chrome-ai-features-august-2024-update/>



The screenshot shows the Chrome Tab Compare interface. At the top, there are four browser tabs: 'Compare portable speaker...', 'AudiOtion Echo Max', 'LMNW PurePatch 6', and 'Soniqu3 Vibewave Elite'. The address bar shows 'chrome://compare/portablespeakers'. The main content area is titled 'Compare Portable Speakers' and displays a comparison table for three products.

AudiOtion Echo Max example.com	LMNW PurePatch 6 example.com	Soniqu3 Vibewave Elite example.com
		
Price		
\$99	\$149	\$199
Reviews		
4.80 ★ (411)	4.98 ★ (50)	4.81 ★ (731)
Summary		
Experience unparalleled clarity and room-filling sound on the go with the AudiOtion Echo Max, the ultimate portable speaker for casual listeners.	Discover precision tuning and stellar audio performance with the LMNW PurePatch 6, perfect for both home and studio environments.	Elevate your audio experience with the Soniqu3 Vibewave Elite, designed for immersive sound and deep bass in a portable form.
Power Output		
30W RMS	40W RMS	50W RMS
Battery Life		
Up to 15 hours	Up to 20 hours	Up to 24 hours
Weight		
2 lbs	3 lbs	3.5 lbs

Web2Table: Using LLMs to Generate Tables from the Web

The Web contains information about arbitrary topics, including:

- products, events, local business, job postings

It is often desirable to combine data from multiple web sources into a clean and comprehensive table.

Steps that have to be performed by one or multiple LLM Agents:

1. Determine the schema of the table
2. Retrieve relevant web pages using a search engine
3. Extract data according to the schema from the retrieved web pages
4. Combine data from multiple web pages into a single table



“Generate a table containing Gymnasium schools in Stralsund in Mecklenburg-Vorpommern with their name, street, street number and a link to their website.”

Name	Street	Street Number	Website
Hansa-Gymnasium Stralsund	Fährwall	19	https://hansagymnasium-stralsund.de/
Schulzentrum Am Sund	Frankenhof	8	https://schulzentrum-am-sund.de/

Project Goals

1. Experiment and benchmark the table generation capabilities of LLMs.
 - How good are LLMs at each task of the table creation pipeline?
 - How good do LLMs derive the table schema from the query?
 - Which search tools are useful to retrieve web pages for LLMs?
 - How good are LLMs at table generation based on the retrieved web pages?
2. Create a benchmark by designing test cases for web page retrieval and table generation.
3. Evaluate the answers of GPT-4-mini and Llama-3.1-8B and conduct an error analysis to better understand the results.
4. Publish the benchmark and your analysis on the Web



Learning Targets

Improve your technical skills

- Improve your technical expertise concerning **Large Language Models**
- Learn how to design and set up **LLM Agents**
- Learn how to design **benchmarks** for testing AI models
- Improve your programming skills
- Work on the leading-edge AI topics

Improve your soft skills

- Work as part of a bigger team on a more complex project
- Organize yourself and assign tasks based on your skills
- Communicate and coordinate your work

Team Project Organization

Duration: 5 months (19 September 2024 – 23 January 2025)

Participants: 9 people

Type of work: Team and subgroup based

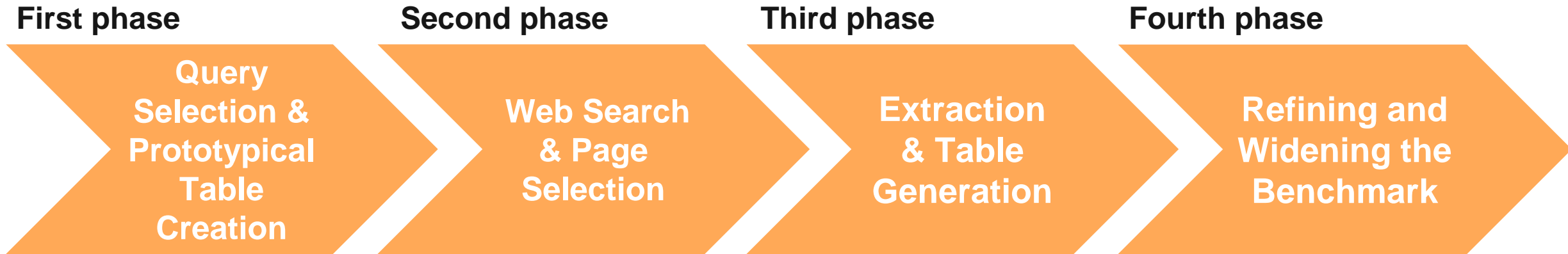
Milestones: 4 project phases

ECTS Points: 12

Evaluation

- Intermediate presentations
- Benchmark including documentation and code
- Individual contribution to the deliverables is graded

Team Project Organization: Project Phases



Phase 1: Query Selection & Prototypical Table Creation

Participants: all team members

Duration: 4 weeks

Sub-phases:

Phase 1.1: Topic Understanding and Initial Setup (1 week)

- Read papers about the topic of LLMs, get familiar with OpenAI models, complete the DeepLearning.ai Tutorials on [functions, tools and agents](#) and [AI Agents in Langgraph](#)
- **Sub-phase duration:** 19.09. to 26.09. (1 week)
- **At the end of this sub-phase:** Meet with Alex on the 26.09. to discuss baseline implementations.

Phase 1.2: Query Selection & Prototypical Table Creation (2 weeks)

Phase 1.3: Analysis of Results (1 week)

Intermediate Presentation (17.10.2024)

Getting familiar with the OpenAI API and OpenAI Models

- **First step:** Create OpenAI account <https://openai.com/>
 - To be able to use the models and also get a starting credit, you need to setup a paid account: <https://platform.openai.com/account/billing/overview> (later, we will cover some of your API costs)
 - You can also check how much you have spent in a day: <https://platform.openai.com/account/usage>
- **Second step:** Get familiar how the API works
 - A quick start tutorial by OpenAI: <https://platform.openai.com/docs/quickstart>
 - How are words tokenized in GPT models: <https://platform.openai.com/tokenizer>
 - Learn how to calculate your prompt token length so that you know beforehand how much a prompt will cost you: <https://platform.openai.com/docs/guides/gpt/managing-tokens> (Go to the Deep Dive: Counting tokens for chat API calls section)
 - To try some first prompts, you can use the Playground: <https://platform.openai.com/playground> (These prompts are also calculated in your costs!)
- Check **OpenAIs models**
 - <https://platform.openai.com/docs/models/overview>
 - For each model, its name, description and maximum token length is given.

LangChain Library

- Framework built for interacting with LLMs
- Offers **prompt templates** and **agent modules** which can be used for function calling
- Tutorials for getting started with LangChain:
 - **OpenAI code examples** for function calling: https://github.com/openai/openai-cookbook/blob/main/examples/How_to_call_functions_with_chat_models.ipynb
 - DeepLearning.ai tutorial **LangChain for LLM Application Development**: <https://www.deeplearning.ai/short-courses/langchain-for-llm-application-development/>
 - DeepLearning.ai tutorial on **Functions, Tools and Agents with LangChain**: <https://www.deeplearning.ai/short-courses/functions-tools-agents-langchain/>
 - DeepLearning.ai tutorial on **AI Agents in LangGraph**: <https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/>



Phase 1.2: Query Selection & Prototypical Table Creation

- Query formulation and table creation
 1. Select topics of your tables from the list below
 - products, events, local business, job postings
 2. Form four sub-teams with 2-3 people. Each sub-team should work on one entity type.
 3. Formulate 2 simple queries that ask for specific entities e.g. “gymnasium schools” and apply a filter e.g. “Stralsund in Mecklenburg-Vorpommern” to find 5 to 10 entities in the real world.
 4. Decide on the table schema with 3-5 attributes e.g. “name, street, street number and a link to their website” and add schema information to the query.
 5. Manually query a search engine of your choice and build the ground truth table for each query. Keep the HTML pages that you used for building the table.
- **Sub-phase duration:** 26.09. to 11.10. (2 week)
- **Deliverables:** Queries, query elements (open/closed schema) and manually created tables with verified attribute values, HTML pages, code for the evaluation and a first analysis.

Example Queries

Query 1:

- **Search Query:** “Generate a table containing shows of Taylor Swift’s eras tour in Germany with the date of the show, the city, the arena and the number of tickets sold.”
- **Topic:** Events
- **Restrictions:** Shows of Taylor Swift’s eras tour in Germany
- **Schema:**
 - **Open:** No attributes explicitly named
 - **Closed:** All (*date, city, arena, number of sold tickets*)/ some attributes explicitly named (*date, city*)

Query 2:

- **Search Query:** “Generate a table containing Samsung smart tvs released in 2023 with 4k display resolution. The table should contain their GTIN, model name, resolution and operating system.”
- **Topic:** Products
- **Restrictions:** Samsung smart tvs released in 2023 and with 4k display resolution
- **Schema:**
 - **Open:** No attributes explicitly named
 - **Closed:** All (*GTIN, model name, resolution and operating system*)/ some attributes explicitly named (*model name*)

Phase 1.3: Analysis of Results

Analyze the table created by the baseline implementations from the questions by comparing the created tables to the manually created ground truth tables

- Error analysis: Compare the attributes of the created schema to attributes of the ground truth schema, compare the created records to the ground truth entities and compare the extracted attribute values to the ground truth attribute values.
- **Sub-phase duration:** 11.10. to 17.10. (1 week)
- **Deliverables:**
 1. Provide executable code to create and evaluate the tables
 2. Report Accuracy score based on exact match for attributes used in the schema, records and extracted/normalized attribute values
 3. Sets of HTML pages and provenance information for the attribute values
 4. Error analysis highlighting different types of errors

Phase 2: Web Search and Page Selection

Participants: Three sub-groups split by web search tool + teams of two for web page selection

Duration: 17.10. to 14.11. (4 weeks)

Sub-phases:

Phase 2.1: Search engine and post-processing tool selection:

- Experiment with different search engines and post-processing tools
 - Web Search: [LangChain provides various web search tools](#).
 - Page Representation: Experiment with converting page content into various formats by using tools such as [Firecrawl](#) and [Spider](#).
- **Sub-phase duration:** 17.10. to 07.11. (3 weeks)
- **Deliverables:** Code and documentation for various web search and page representations implementations.

Phase 2.2: Analysis of Results

- Analyze the retrieved and parsed web pages: How relevant are the retrieved web pages?
- **Sub-phase duration:** 07.11. to 14.11. (1 week)
- **Deliverables:**
 - Evaluation and error analysis based on entity and attribute value overlap to ground truth tables.
 - Select a subset of interesting/useful web pages for your benchmark (e.g. 10 per query with high entity overlap and relevant attributes, see WDI)

Intermediate Presentation (14.11.2024)

Phase 3: Record Extraction and Table Generation

Participants: three subgroups of each 3 people

Duration: 14.11. to 06.12. (3 weeks)

Sub-phases:

Phase 3.1 Extraction schema creation, attribute value extraction/normalization and table generation

- Experiment with schema creation, attribute value extraction/normalization and table generation
- **Sub-phase duration:** 14.11. to 28.11. (2 weeks)
- **Deliverables:**
 - Executable code and documentation for all pipelines covering extraction schema creation, attribute value extraction/normalization and table generation

Phase 3.2: Analysis of Results

- Analyze the created extraction schemata, the extracted/normalized attribute values and the generated tables.
- **Sub-phase duration:** 28.11. to 06.12. (1 week)
- **Deliverables:**
 - Evaluation based on accuracy of the extraction schemata, the extracted/normalized attribute values and the generated tables. The generated tables should contain the correct schema, all relevant entities and the normalized attribute values. Evaluate the provenance of the extracted/normalized attribute values.
 - Error Analysis to understand strengths and weaknesses of complete pipelines and intermediate steps.

Intermediate Presentation (06.12.2024)

References for Phase 3

– Information Extraction

- Brinkmann, Shraga, Bizer: ExtractGPT: Exploring the Potential of Large Language Models for Product Attribute Value Extraction. arXiv preprint arXiv:2310.12537v3, 2023.
- Brinkmann, Baumann, and Bizer: Using LLMs for the Extraction and Normalization of Product Attribute Values. In Advances in Databases and Information Systems, 2024.

– Entity Matching

- Peeters, Bizer: Entity matching using large language models. arXiv preprint arXiv:2310.11244, 2023.
- Winte.r Data Integration Framework <https://github.com/wbsg-uni-mannheim/winter>

– Data Fusion

- Bleiholder, Naumann: Data Fusion. ACM Computing Surveys, 2008.
- Li, Gao, Meng, et al.: Survey on Truth Discovery. SIGKDD Explorations, 2016.
- Winte.r Data Integration Framework <https://github.com/wbsg-uni-mannheim/winter>

Phase 4: Refinement and Widening of Benchmark

Participants: Sub-teams of 2 per table topic

Duration: 06.12. to 23.01. (7 weeks)

Sub-Phases:

Phase 4.1 Refine and Widen the Benchmark

- Extend the initial 2 queries to 5 queries per sub-team.
- Engineer challenges based on your previous error analysis e.g. by asking for more challenging records or attributes or construct scenarios with conflicting and missing values.
- **Sub-phase duration:** 06.12. to 10.01. (5 weeks)

Phase 4.2 Final Error Analysis

- Evaluation based on accuracy of the extraction schemata, the extracted/normalized attribute values and the generated tables. The generated tables should contain the correct schema, all relevant entities and the normalized attribute values. Evaluate the provenance of the extracted/normalized attribute values.
- **Sub-phase duration:** 10.01. to 17.01. (1 week)

Phase 4.3 Finalize Documentation for Publication

- HTML page describing the benchmark and your experiments
- Well documented code for the code execution and benchmark creation
- **Sub-phase duration:** 17.01. to 23.01. (1 week)

Final Presentation (23.01.2025)

Schedule

Date	Session
Thursday, 19.09.2024	Kickoff meeting (today)
	Phase 1.1: Setup and Topic Understanding
	Phase 1.2: Table Topic Selection and Table Creation
	Phase 1.3: Analysis of Results
Thursday, 17.10.2024	1st Intermediate Presentation: Table Topic Selection, Table Creation and Baseline Evaluation
	Phase 2.1: Search engine and post-processing tool selection
	Phase 2.2: Error Analysis and evaluation
Thursday, 14.11.2024	2nd Intermediate Presentation: Search Engine Selection and Web Page Parsing
	Phase 3.1: Extraction schema creation, attribute value extraction and table generation
	Phase 3.2: Analysis of Results
Friday, 06.12.2024	3rd Intermediate Presentation: Record Extraction and Table Generation
	Phase 4.1: Refine and widen the benchmark
	Phase 4.2: Analysis of Results
	Phase 4.3: Finalizing the artefacts: Benchmark Documentation, Executable Code
Thursday, 23.01.2025	Final Deliverable Deadline: Benchmark including Documentation, Executable Code

Formal Requirements & Consultation

Deliverables

1. On the presentation dates provide us via e-mail with:
 - **Presentation slides**
 - **Task to member report:** excel sheet stating which team member conducted which subtask
 - **Code/Data:** link or zipped folder with your code and data

2. Final Deliverables
 - **Webpage** describing the benchmark and your experiments
 - **Repository** containing your code
 - **Test Driver** to run benchmark and calculate results

All deliverables must be sent to Alex & Chris!

Formal Requirements & Consultation

Final grade

- 20% for each phase
- 20% for final deliverables
- Late submission: -0.3 per day

Consultation

- Send anytime one e-mail per team or subgroup stating your questions to Alex, he will answer your question or assign you a slot for a meeting.
- In addition, regular progress meetings every two weeks. Need to agree on time slot.

Useful Software

- LangChain Tutorials
 - <https://python.langchain.com/v0.2/docs/tutorials/>
 - <https://langchain-ai.github.io/langgraph/tutorials/>
 - <https://www.deeplearning.ai/short-courses/functions-tools-agents-langchain/>
 - <https://www.deeplearning.ai/short-courses/ai-agents-in-langgraph/>
- Team Cooperation
 - GitHub/Lab for the code base
 - Project Management Tool of your choice
- Processing and GPUs
 - Teaching GPU-Server
 - Google Colab: <https://colab.research.google.com/>
 - BwUniCluster2.0: https://wiki.bwhpc.de/e/Category:BwUniCluster_2.0

Related Work: Large Language Models

- Zhao, Wayne Xin, et al. "**A survey of large language models.**" arXiv:2303.18223 [cs.CL] (2023).
- Mialon, et al.: **Augmented Language Models: a Survey.** arXiv:2302.07842 [cs.CL]
- Ouyang, Long, et al. "**Training language models to follow instructions with human feedback.**" Advances in Neural Information Processing Systems 35 (2022): 27730-27744.
- Borji, Ali. "**A categorical archive of chatgpt failures.**" arXiv:2302.03494 (2023).
- Bang, Yejin, et al. "**A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity.**" arXiv:2302.04023 (2023).
- Wei, Jason, et al. "**Emergent abilities of large language models.**" arXiv:2206.07682 (2022).
- Wei, Jason, et al. "**Chain-of-thought prompting elicits reasoning in large language models.**" Advances in Neural Information Processing Systems 35 (2022): 24824-24837.
- Wang, Lei, et al. "**Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models.**" arXiv preprint arXiv:2305.04091 (2023)
- Schick, Timo, et al. "**Toolformer: Language models can teach themselves to use tools.**" arXiv:2302.04761 (2023).
- He, Hangfeng, Hongming Zhang, and Dan Roth. "**Rethinking with retrieval: Faithful large language model inference.**" arXiv preprint arXiv:2301.00303 (2022).
- Yao, et al: ReAct: "**Synergizing Reasoning and Acting in Language Models**", arXiv preprint arXiv:2210.03629 (2023)

Questions?

