TaskTitle

Firstname Lastname
RPTU Kaiserslautern, Department of Computer Science

Note: *This report contains a project documentation and reflection on the portfolio task submitted for the lecture Engineering with Generative AI in WiSe 2024-25. This report is an original work and will be scrutinised for plagiarism and potential LLM use.*

1 Portfolio documentation

Compile a comprehensive documentation of your project, including all the project phases. You will need to explain every choice you made during the project and your thoughts about the results you get. You will introduce the results in suitable visualisation. Furthermore, you will need to explain which criteria you follow to build your prompts and how they affect the results.

Students write the entire documentation with sections, sub-sections, diagrams, etc in this section. Please write as comprehensively as possible. Head to the document 1_documentation.tex. You are free to use as many subsections as required. We will not provide a template for documentation.

START

1.1 Research and tool selection

1.1.1 Software and Platforms

Our multi agent system is developed using the AutoGen[1] framework from Microsoft. The framework allows fast and practical development of multi-agent LLM-based systems. This choice implicates the programming language we use, it is python. The version of the language corresponds to the one available by default on the Google Colab[2] platform, that we use as runtime environment. The python version is 3 and the minor version is not shown by the runtime information panel or configuration options. Besides that we use the following python libraries:

- arxiv (to get paper information from Arxiv[3])
- requests (to get PDF files from Arxiv)
- pymupdf (to parse PDF files)
- bibtexparser (to parse bibtex references)

[1] https://microsoft.github.io/autogen/stable/
[2] https://colab.research.google.com
[3] https://arxiv.org/

We also decided to use two model inference providers: Groq[4] and HuggingFace[5]. These choices, together with the choice of runtime environment are motivated by practicality: in our opinion it is more convenient to use free cloud-based infrastructure than to configure and maintain a local one.

AutoGen is well documented and practical framework, required by the exam task statement and Google Colab is a reliable platform whith free tier is sufficient for the task. Besides that, we have already had good experiences with these tools during exercises. Pymupdf is one of the common libraties for parsing PDF files and bibtexparser is also used for parsing bibtex references. The other two libraries were the best know choices for the respective tasks.

1.1.2 LLM Model

To select models for our system we used the following criteria:

- All models should be available without payment at least for non-commercial project (as required by exam task statement).
- Every model must be available at inference endpoints of either HuggingFace or Groq within free tiers.
- The model used for the processing of paper texts must have context window larger than 30 000 tokens (because the longest paper among chosen amounts for about 15 000 tokens and we wanted to have some slack to be sure, besides, 32 000 is one of the commonly used context lengths).
- The model for tool-equipped agents must be tuned or at least allow for function calling (for the exam task requires at least some agents to use tools and such model would allow tool usage with lesser amount of tinkering).
- The model for tool-equipped agents must be compatible with AutoGen framework (for this is the framework of choice, as described in 1.1.1.
- Optionally, the model used for the processing of paper texts should be adapted for summarization. Even more preferrable, if it's trained for summarization using materials like the scientific datasets used in [1]). This requirement is optional, because text-to-text generating models can be used for summarizations to some extent even without specialized training, for this a one of the possible text-to-text tasks. At the same time, the exam task does not state that the model must be specifically targeted for text summarization, but that it must be cabaple of it.

We started by looking for a model for paper processing. Using the model search interface of [2], we have set the following search parameters:

1. Tasks: Summarization or Text2Text generation
2. Other: HF Inference API

[4]https://groq.com
[5]http://huggingface.co

3. Language: we tried setting english language, but this has limited the search too much and excluded seemingly useful models that lack english language in model tags despite supporting it
4. Datasets: setting the ”tasks“ and ”other“ filters as described was limiting the search so much that we have omitted this filter
5. License: setting the ”tasks“ and ”other“ filters as described was limiting the search so much that we have omitted this filter
6. Libraries: since the filter does not have AutoGen framework, we left it empty

This has left us no more than 10 possible models. After examining the model cards as well as configuration files of the model, we did not find any with a sufficient context length. The largest specified context length (4096) had the model ”google/bigbird-pegasus-large-arxiv“ used in [3]. For some models neither the card nor the files has given any reliable information. This forced us to further relax the filtering by also using ”Text generation“ in Tasks filter. The new filter configuration showed 48 possible models. We have examined the model cards and configurations one by one until we found 3 promising candidates:

- deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B with 128K context window
- meta-llama/Llama-3.3-70B-Instruct with 128K context window, instruction tuned, also supports tool usage
- mistralai/Mistral-7B-Instruct-v0.3 with 32K context window, instruction tuned, supports tool usage

From these three we chose the latter, because it has a sufficient context length, supports tool usage and is smaller than the second one. The smaller size model is favourable in this case, because request processing on the same computation architecture takes less time. The chosen model is also instruction tuned, that, as we supposed, can enhance results in our case.

Unfortunately, it seems that the inference endpoint of HuggingFace is incompatible with reflect-on-tool-usage feature of AutoGen OpenAIChatCompletionClient: as soon as we were having the feature on, we were getting an error regarding bad response format. This is why we decided to choose Groq as the inference provider for tool equipped agents. It is important to mention that despite this decision, we still use HuggingFace for paper processing. This is due to the free tier limits of Groq that do not allow the required number of tokens to be processed.

On Groq, multiple models are viable candidates for the task, at least the following:

- gemma2-9b-it
- llama-3.1-8b-instant
- llama3-8b-8192
- llama3-70b-8192

Two smallest llama models both seemed to be good candidates for the promise of faster responses. However, to process more complex conversations a larger model could be better. Besides, llama3-70b-8192 was shown to work well for multi-agent orchestration with tool use. This way we chose the latter model.

Final choice:

- for processing of paper texts we use mistralai/Mistral-7B-Instruct-v0.3 through the inference endpoint of HuggingFace;
- for tool usage and decision making we use llama-3.1-8b-instant through Groq.

1.1.3 Research papers

We were interested in multi-agent LLM-based systems for end-to-end software development – such frameworks, that, given software requirements, emulate real world software development process and deliver a working application. Such software process may include not only coding, but also other software process activities, such as refinement of requirements, project management, creation of of interconnected files, quality assurance and other activities.

We searches on Google Scholar[6], Arxiv, Katalog+[7] and also DBLP[8]. We used the following search terms: multi agent, llm, software development. In the case where search engines allowed that, we searched in titles and abstracts. We have also limited the year with which the papers were dated with the timeframe of 2020 to 2024, where the search engines have allowed that. The searched resulted in a multitude of papers of which only a small amount falls into our area of interest.

[4], present in the results of multiple search databases, have caught our attention. This work, dated end 2024, reviews literature in LLM-based multi-agent systems for software engineering. Among other reviewed topics, it also write on multiple frameworks that relate to our interest. We used some of them for this work and also include work that was not reviewed in this article.

The total amount of candidate papers was higher than the required 6 and we have used the following criteria to chose:

- As stated above, all selected papers must be in the field of multi-agent llm-based frameworks for end-to-end software development
- No work should have used any of the frameworks that other selected papers have contributed as a component. This is motivated by our interest in different and comparable frameworks.
- Every work should include evaluation using at least one publicly available benchmark. Best case: all the papers use the same benchmarks to allow comparison of their efficiency.
- Selected papers must cover at least 2 software development process paradigms: waterfall and agile.
- All papers must be available on Arxiv to allow easy downloading by agents.

Application of this criteria has still left some freedom of choice and we have arbitrarily selected the following frameworks:

1. MetaGPT [5] (waterfall)

[6] https://scholar.google.com/
[7] https://hbz-rptu.primo.exlibrisgroup.com/discovery/search?vid=49HBZ_RTU:RPTU
[8] https://dblp.org

2. ChatDev [6] (waterfall)
3. CodePori [7] (waterfall)
4. AgileGen [8] (agile)
5. AgileCoder [9] (agile)
6. EvoMAC [10] (dynamic process)

At least one of the first two, MetaGPT and ChatDev, is used in other selected works as a comparison competitor, this has motivated these two choices. AgileGen and AgileCoder were the only frameworks for agile software process that we have found. We have included EvoMAC because it adopts a method similar to backpropagation from the realm of neural network training to update the development process so that the set of agents used and the topology of their communication may be changed multiple times during one run automatically.

Some of the papers are preprints, but include interesting contributions and since we are not writing this literature review for publication but only for examination and testing of our system, we consider this acceptable.

1.2 Deisgn

- summarizing chat text wont work, because the whole orchestration must be done with LLM - chunking chat text wont work either – neither RoundRobinGroupChat nor SelectorGroupChat allow that - handing off like in swarm wont work either (really?), because also in swarm agents use the shared chat environment - calling tools one after another from one agent isnt possible – it make seemingly parallel calls to all selected tools, but can't pass data returned from one tool to another

1.3 reflection

- comparing systems is very complex: two systems are equal if all the components are equal and also the outcome. components: models, prompts, agents, communication topology, tokenization, usage of additional techniques like RAG

2 Reflection

In 3-5 pages, 1500-2000 words

This section needs to be adjusted to align with the reflection requirements specified in the selected task.

Note: You should address all the questions from your selected task. Please list each question and provide your answers in the following enumeration.

For example:

1. What was the most interesting thing you learned while working on the portfolio? What aspects did you find interesting or surprising?

   Answer:

2.

All of the resources used by the student to complete the portfolio task should be organised in the references section. **Note that the Reference section does not count towards the number of pages of the report.** Example references are given below [11] [12] [13]. **If you are using a reference manager like Zotero, you can export your Zotero library as a .bib file and use it on Overleaf. As you cite the article/technology/library in your main text, the References section will automatically update accordingly.** Please include a full list of references found. If students are using Zotero for their research paper management, a bibTeX will help them during citation which automatically adds references to the report.

References

[1] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018.

[2] Hugging Face. Hugging face. Accessed: 13.03.2025.

[3] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021.

[4] Junda He, Christoph Treude, and David Lo. Llm-based multi-agent systems for software engineering: Literature review, vision and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 2024.

[5] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024.

[6] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development, 2024.

[7] Zeeshan Rasheed, Malik Abdul Sami, Kai-Kristian Kemell, Muhammad Waseem, Mika Saari, Kari Systä, and Pekka Abrahamsson. Codepori: Large-scale system for autonomous software development using multi-agent technology, 2024.

[8] Sai Zhang, Zhenchang Xing, Ronghui Guo, Fangzhou Xu, Lei Chen, Zhaoyuan Zhang, Xiaowang Zhang, Zhiyong Feng, and Zhiqiang Zhuang. Empowering agile-based generative software development through human-ai teamwork, 2024.

[9] Minh Huynh Nguyen, Thang Phan Chau, Phong X. Nguyen, and Nghi D. Q. Bui. Agilecoder: Dynamic collaborative agents for software development based on agile methodology, 2024.

[10] Yue Hu, Yuzhu Cai, Yaxin Du, Xinyu Zhu, Xiangrui Liu, Zijie Yu, Yuchen Hou, Shuo Tang, and Siheng Chen. Self-evolving multi-agent collaboration networks for software development, 2024.

[11] Albert Einstein. Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]. *Annalen der Physik*, 322(10):891–921, 1905.

[12] Donald Knuth. Knuth: Computers and typesetting.

[13] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.