LLMs on the Fly: Text-to-JSON for Custom API Calling

Miguel Escarda-Fernández[1], Iñigo López-Riobóo-Botana[1], Santiago Barro-Tojeiro[1], Lara Padrón-Cousillas[1], Sonia Gonzalez-Vázquez[1], Antonio Carreiro-Alonso[1,2] and Pablo Gómez-Area[1,2]

[1]Centro Tecnológico ITG, Cantón Grande 9, Planta 3, 15003, A Coruña, España
[2]FlyThings® Technologies - IoT for Solutions 4.0, Cantón Grande 9, Planta 3, 15003, A Coruña, España

Abstract

In the rapidly evolving landscape of Natural Language Processing (NLP), there is a growing demand for agile and intuitive tools due to the increasing model capabilities, primarily in the field of Large Language Models (LLMs). In recent months, we have seen great progress in the Natural Language Generation (NLG) landscape, with proliferation of generative AI applications leveraging LLMs for a vast number of tasks. The power of LLMs resides in their ability to generalize almost any NLP task to the problem of next token prediction, thus simplifying the traditional NLP pipelines consisting in intensive data labeling and domain-specific fine-tuning for a single task. Moreover, LLMs are enhanced (1) with external knowledge bases, which improve their reasoning and domain understanding and (2) with external tools, which improve their ability to perform actions.

We present a novel approach that harnesses the power of LLMs to transform natural language inputs into structured data representations, facilitating seamless interaction with custom APIs for real-time data visualization. We explore the integration of Flythings® Technologies API for Internet of Things (IoT) device solutions in the Industry 4.0 domain. This system demonstration presents a chat-based virtual assistant that allows users to query the status of monitored machines and devices. The core component of the application is a LLM that serves as a bridge between user queries and machine-readable JSON objects, which adhere to a predefined schema following the Flythings standard. Our LLM output facilitates the interaction with the Flythings API, leading to the generation of visualizations that illustrate IoT device status in real time.

Keywords

NLP, LLM, Fine-tuning, agents, assistants, visualization, API tools, IoT, Monitoring, Industry 4.0

1. Introduction

Undoubtedly, Large Language Models (LLMs) are here to stay. Their emergence has marked the beginning of an era in which natural language can be used to perform any Natural Language Processing (NLP) task following prompting techniques [1], which normally required large amount of labeled datasets for fine-tuning ad hoc models for a single task. LLMs, with their remarkable language understanding and generation capabilities, have the potential to dramatically ease the continuous iteration of common NLP workflows including, but not limited to, data labeling, data augmentation or fine-tuning. Moreover, production-ready systems using LLMs require less time than the traditional approaches [2], thanks to the generalization of almost any NLP task to the problem of Causal Language Modeling (CLM) [3, 4, 5, 6]. By providing such text interface, we communicate with machines in a natural way, allowing for intuitive interactions.

In this context, our system demonstration paper introduces a novel application that leverages the strengths of LLMs in industrial monitoring and Internet of Things (IoT) domains. We integrate our fine-tuned LLMs in the Flythings® Technologies platform[1]. Our chat-based application allows users to submit queries using natural language. Then, these are processed by our model, whose task is to extract the relevant information from the input query by identifying and mapping the Flythings API input fields, generating a well-formed JSON output following a specific schema. This enables us to easily interact with the Flythings API, sending the corresponding JSON objects to their services for real-time monitoring and visualization of IoT device details.

In Section 2, we conduct a comprehensive review of the existing research, establishing a solid foundation and context for our work. In Section 3, we present our pipeline and infrastructure, describing the design details and all the steps involved, including the data augmentation, fine-

[1]You have a brief description of the FlyThings® Technologies services in the Appendix A, check at https://itg.es/en/monitoring-iot-platform-flythings/ for additional information.

SEPLN-CEDI-PD 2024: Seminar of the Spanish Society for Natural Language Processing: Projects and System Demonstrations, June 19-20, 2024, A Coruña, Spain.

mescarda@itg.es (M. Escarda-Fernández); ibotana@itg.es (I. López-Riobóo-Botana); sbarro@itg.es (S. Barro-Tojeiro); lpadron@itg.es (L. Padrón-Cousillas); sgonzalez@itg.es (S. Gonzalez-Vázquez); acarreiro@itg.es (A. Carreiro-Alonso); pgomez@itg.es (P. Gómez-Area)
https://www.linkedin.com/in/%C3%AD%C3%B1igo-luis-l%C3%B3pez-riob%C3%B3o-botana-4a43001a2/ (I. López-Riobóo-Botana); https://www.linkedin.com/in/phd-sonia-gonz%C3%A1lez-v%C3%A1zquez-38b14a8b/ (S. Gonzalez-Vázquez)
0000-0002-9080-1535 (M. Escarda-Fernández); 0000-0002-7310-0702 (I. López-Riobóo-Botana); 0009-0006-2782-2567 (S. Gonzalez-Vázquez)

tuning and deployment of the optimized and production-ready LLM. In Section 4, we illustrate the practical examples carried out and the real world utility of our tool, presenting its limitations in Section 5. We conclude with Section 6 by summarizing our findings and outlining the future directions of our research.

2. Related Work

In recent months, we have seen a myriad of LLM research papers addressing the topic of context-aware LLMs through in-context learning. This capability enables them to generalize to almost any NLP task, commonly unseen during pre-training and fine-tuning stages [3, 5, 6]. This direction has led the research community to explore the integration of LLMs with external tools such as document stores [7] or APIs [8], enhancing their generalization capabilities even more. LLM agents [9] are a new concept arised from providing LLMs with (1) extensive up-to-date data pools beyond their fixed knowledge representations and (2) functions or tools to perform actions and automate processes [10, 11, 12, 13]. Such two-fold strategy reduces the need for regular retraining. For example, Gorilla [8] leverages a multitude of APIs and documentation through document retrievers, highlighting the effectiveness of this framework.

Moreover, the reasoning capabilities of LLMs are influenced by the prompt strategies followed [5, 14, 15], where how natural language instructions are written significantly affects the performance [16]. More complex prompting strategies like ReAct [9] became popular, combining reasoning and planning techniques by adding reasoning traces and task-specific actions to the prompt. These strategies benefit the integration of the LLM with external sources. In this new landscape, new benchmark frameworks were proposed [17, 18], which aim at designing reliable and robust evaluation methodologies.

The introduction of Generative Information Extraction (GIE) has further boosted the NLP field [19]. Recent studies [20] propose LLMs to generate structured information from natural language. Some closely-related tasks, like text-to-SQL [21, 22], involve the transformation of natural language into SQL language for querying external tools (i.e., databases). This generative approach proves to be effective even in scenarios involving complex schemas with millions of entities involved [23]. The ability of LLMs to manage these large schemas without dropping performance (effectively generating the target query following a specific format) is particularly significant for our research. We propose a generation step aiming at transforming natural language queries (sent to our virtual assistant) into structured JSON objects with the relevant parameters for the integration of the FlyThings® API.

3. Proposed Method

In this section, we present our methodology, covering all the steps involved in our pipeline. We describe our data preparation stage, including the seed data creation and data augmentation process. We also formulate our supervised fine-tuning (SFT) method for our information extraction task, as well as the inference optimizations taken into account for our LLM deployment. The overall process is depicted in Figure 1.

3.1. Seed Data

In the absence of pre-existing user data for our task, dependent on the FlyThings® technology, we started creating a dataset. We collected feedback from the Flythings team, who provided us with the initial examples of potential user inputs and expected outputs. In this way, we got a seed dataset consisting of 6 outputs, each of them with 3 different ways of expressing the input in accordance with the Flythings team. Given these pairs, we agreed on a specification, defining a JSON schema as the golden rule. Our pipeline starts with (1) a template-based method for generating new JSON outputs as described in Figure 1, randomly selecting one of the available options for each of the JSON fields, following the schema depicted in Figure 2. In this way, we got a pool of examples for the next data augmentation step.

3.2. Data Augmentation

Our seed dataset was scarce and limited in scope, lacking from input query diversity. Therefore, we followed a data augmentation approach. We created a custom pipeline for generating alternative input queries, given the reference (input, output) pairs from the seed data. For this task, we leveraged the Mixture of Experts (MoE) LLM *Mixtral-8x7B-Instruct-v0.1* model from Mistral AI [24].

We aimed at generating variant inputs for each JSON output from the previous pool depicted in Figure 1, so that we could increase the available (input, output) pairs. We used the original seed as reference within the instruction illustrated in Figure 3, generating 3 variations of the input for each target through few-shot in-context learning [6]. This process corresponds to the (2) data augmentation step depicted in Figure 1. We increased our dataset up to 355 curated samples for the following SFT stage.

3.3. Supervised Fine-Tuning

Before diving into the details of the fine-tuning process, it is important to understand why supervised fine-tuning was necessary in the first place. While zero-shot or few-shot (i.e., in-context) learning [25] can be effective for general NLP tasks, it entails challenges when the task

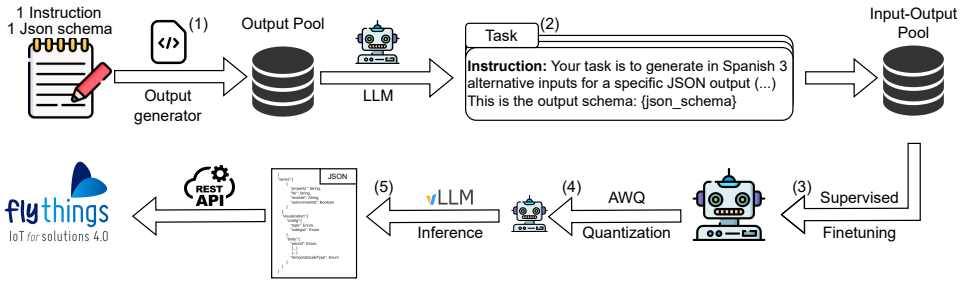

Figure 1: Our pipeline begins with the design of the JSON schema with the formatting rules, used as the specification for (1) a template-based method for the generation of random JSON output targets for our task. These outputs are fed into (2) a data augmentation phase utilizing a LLM to generate multiple inputs corresponding to each previously generated JSON output, so that we add diversity to how users convey queries. Subsequently, (3) the supervised fine-tuning task for our information extraction task, (4) the quantization stage for the model inference optimization and (5) the deployment phase culminating with the integration of the FlyThings® endpoint for the creation of a virtual assistant enhanced with visualizations.

Json Schema

```
{
  "series":[
    {
      "property": String,
      "foi": String,
      "module": String,
      "asIncremental": Boolean
    }
  ],
  "visualization":{
    "config":{
      "type": Enum,
      "subtype": Enum
    },
    "body":{
      "period": Enum,
      (...)
      (...)
      "temporalScaleType": Enum
    }
  }
}
```

Figure 2: Overview of the JSON schema used for output validation. Notice that, according to this specification, each JSON output will have two main parts: (1) the series field, which includes information about the specific Flythings IoT devices been queried and (2) the visualization properties, which include the required information for the visual representation of the series data in the virtual assistant.

Instruction: Your task is to generate 3 alternative inputs for a specific JSON output. ***{rules_to_follow}***
This is the output schema:
{"series": [{ "property": "tap 2", "foi": "greehouse water", "asIncremental": True }], "visualization": {"config": {"type": "chart", "subtype": "line"}, "body":{ "temporalScale": "DAILY", "temporalScaleType": "CHANGES" }}}

Input1: View the accumulated status changes for tap 2 of the greenhouse water device on a daily graph.
Input2: Observe the daily graph that displays the collective status alterations of tap 2 in the greenhouse watering device.
Input3: Examine the daily chart showing the aggregate changes in the status of greenhouse water device's tap 2.

Figure 3: The few-shot data augmentation task. We designed the following prompt: (1) the system instruction (displayed in black), including the rules (in bold curly brackets) with the seed pairs as reference guiding the generation with few-shot examples (omitted for clarity). Then, we present (2) one output from the pool as the target (highlighted in blue) and (3) we generate three new input queries (highlighted in green).

is very specific and requires a thorough generation process, limiting hallucinations [26]. In our case, we faced some issues with the in-context learning approach for classifying and extracting the corresponding fields for the Flythings® task. On the one hand, (1) zero-shot learning, which involves making direct predictions without any previous examples in the training distribution, had problems with detailed input queries requiring complex JSON outputs, in which the corresponding JSON schema in the instruction was not enough. These led to classification inaccuracies in the generation step. Similarly, (2) few-shot learning, which relies on providing the model with some examples of the task in the initial instruction, was limited and biased by the quality and expressiveness of the provided sequences at inference time. In short, these two methods neither captured the complexity nor the specificity of our domain, leading us to sub-optimal performance in terms of both accuracy and reliability.

Recognized these limitations, we transitioned to a fine-tuning approach to tailor the model for our specific needs. During the fine-tuning stage, we assessed multiple models up to 7 billion parameters, considering the trade-off between the model performance and our hardware limitations. We finally chose the instruction fine-tuned model *teknium/OpenHermes-2.5-Mistral-7B*[2] based on the *mistralai/Mistral-7B-Instruct-v0.1* model[3]. We leveraged the dataset from our previous data augmentation step

[2]https://huggingface.co/teknium/OpenHermes-2.5-Mistral-7B
[3]https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1

from Section 3.2, following the QLoRA [27] approach for efficient fine-tuning. Similar to LoRA (Low-Rank Adaptation of large language models) [28], which freezes the pre-trained model weights and adds trainable rank decomposition matrices to each transformer block (eliminating the need for full fine-tuning), QLoRA goes a step further by quantizing the weights of the frozen backbone LLM, adding the LoRA adapters with paged optimizers to manage memory spikes. This results in a more efficient memory management for fine-tuning [27].

3.4. Inference Optimization

After the supervised fine-tuning stage of our model, we had to determine the inference requirements under a production environment, considering (1) our hardware limitations and (2) the need for low latency supporting real-time queries. In this way, we explored the available options for reducing the computational requirements, while maintaining (or minimally decreasing) the LLM performance. We opted for the vLLM [29] library, specifically designed for fast and efficient serving of LLMs including, but not limited to, paged attention optimizations, continuous batching of incoming requests and optimized CUDA kernels. We compared the performance of different quantization techniques supported by vLLM, such as GPTQ [30] and AWQ [31]. We chose AWQ because it offered the best throughput while maintaining the performance[4]. We deployed our LLM service in the proprietary ITG clusters, using a RTX A6000 48 GB GDDR6 GPU.

4. Chatbot Experimentation

For our experimentation, we implemented a new virtual assistant view in the FlyThings® framework. The front-end of the chatbot is in charge of loading the user contexts, which is the list of their IoT devices available. With the environment all set, each input query is sent to the LLM service, which generates the corresponding JSON output following the schema described in Figure 2. We identify the closest IoT device information matching the extracted *device* and *property* (and optionally *module*, if present) JSON fields. Then, we follow these steps: (1) if there are no matches, the user is prompted to try again; (2) if there is exclusively one match, the next step is executed; (3) if there are more than one match, a radio button is displayed for the user to choose among them. Depending on the visualization format (graph, table, indicator and so on), a request to the observation API endpoints[5] is processed, including all the chart configuration. Finally, the visual widget is loaded, showing the results to the user. We include an example in Figure 4. We also provide a video demonstration[6] of the virtual assistant.

5. Limitations

In this paper we introduce the first version of the system as a proof-of-concept demo, still in its early stage of development. We focused on the data augmentation, fine-tuning and deployment stages mainly due to time constraints. We did not perform thorough evaluation and we acknowledge the importance of this process, but since the project is linked to a new market product by the Flythings® company, we aligned with the team requirements, which were more oriented to fast prototyping for a first usable version of the chat interface.

6. Conclusions and Future Work

In this paper we present a novel approach for querying the Flythings® framework. We described the system architecture and the NLP pipeline for the dataset preparation, LLM fine-tuning and inference optimization stages. Our approach is generalizable to any text-to-JSON or text-to-API task following the proposed pipeline. We handle user queries in natural language with a virtual assistant, considering visual feedback. Our next steps include refining the fine-tuned LLM using preference data from users interacting with the system. We will study in more detail both the helpfulness and the accuracy of our model outputs by means of thorough evaluation and benchmarking. We plan to explore Reinforcement Learning from Human Feedback (RLHF) [32] and Directed Preference Optimization (DPO) [33] for further alignment with human preferences. We also foresee future applications of Virtual Reality (VR), which would improve usability under real conditions and enhance user experience. We aim to broaden the current functionality beyond querying IoT devices, adding more complex Flythings® IoT operations, such as managing device actions, alerts or dashboards.

Acknowledgments

This ongoing R&D project is supported by the CEL.IA network initiative[7] through the CDTI (Centro para el Desarrollo Tecnológico Industrial) (grant CER-20211022) by the Ministerio de Ciencia e Innovación. This research is also possible thanks to the ITG-Flythings collaboration. We would like to express our gratitude to the Flythings

[4] The AWQ quantization method consistently outperforms GPTQ across different model scales in their evaluation benchmark. Check the original work for more details.

[5] https://deviot.flythings.io/api/apidocs/index.html#api-03-Request_Observations

[6] Demo (video) available at https://youtu.be/qHs47rcmpHU

[7] https://itg.es/cervera-celia/

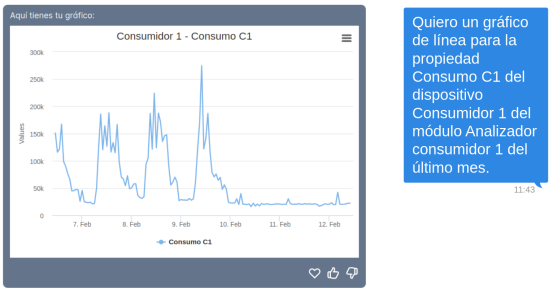

Figure 4: An end-to-end example of the FlyThings® virtual assistant, integrating the LLM service with the API services.

developers team, for their continuous support and feedback to enhance our LLM generation capabilities and integration within their systems.

References

[1] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al., Sparks of artificial general intelligence: Early experiments with gpt-4, arXiv preprint arXiv:2303.12712 (2023).

[2] A. Kulkarni, A. Shivananda, A. Kulkarni, D. Gudivada, LLMs for Enterprise and LLMOps, Apress, Berkeley, CA, 2023, pp. 117–154. URL: https://doi.org/10.1007/978-1-4842-9994-4_7. doi:10.1007/978-1-4842-9994-4_7.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language Models are Unsupervised Multitask Learners, 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

[4] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned Language Models Are Zero-Shot Learners, ArXiv abs/2109.01652 (2021). URL: https://api.semanticscholar.org/CorpusID:237416585.

[5] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large Language Models are Zero-Shot Reasoners, ArXiv abs/2205.11916 (2022). URL: https://api.semanticscholar.org/CorpusID:249017743.

[6] D. Dai, Y. Sun, L. Dong, Y. Hao, S. Ma, Z. Sui, F. Wei, Why Can GPT Learn In-Context? Language Models Implicitly Perform Gradient Descent as Meta-Optimizers (2023). arXiv:2212.10559.

[7] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.

[8] S. G. Patil, T. Zhang, X. Wang, J. E. Gonzalez, Gorilla: Large Language Model Connected with Massive APIs, arXiv preprint arXiv:2305.15334 (2023).

[9] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, arXiv preprint arXiv:2210.03629 (2022).

[10] A. Parisi, Y. Zhao, N. Fiedel, TALM: Tool Augmented Language Models, ArXiv abs/2205.12255 (2022). URL: https://api.semanticscholar.org/CorpusID:249017698.

[11] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, T. Scialom, Toolformer: Language models can teach themselves to use tools, arXiv preprint arXiv:2302.04761 (2023).

[12] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, J. Schulman, WebGPT: Browser-assisted question-answering with human feedback, CoRR abs/2112.09332 (2021). URL: https://arxiv.org/abs/2112.09332. arXiv:2112.09332.

[13] S. Yao, R. Rao, M. Hausknecht, K. Narasimhan, Keep CALM and explore: Language models for action generation in text-based games, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 8736–8754. URL: https://aclanthology.org/2020.emnlp-main.704. doi:10.18653/v1/2020.emnlp-main.704.

[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. hsin Chi, F. Xia, Q. Le, D. Zhou, Chain of Thought Prompting Elicits Reasoning in Large Language Models, ArXiv abs/2201.11903 (2022). URL: https://api.semanticscholar.org/CorpusID:246411621.

[15] W. Huang, P. Abbeel, D. Pathak, I. Mordatch, Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents, CoRR abs/2201.07207 (2022). URL: https://arxiv.org/abs/

2201.07207. arXiv:2201.07207.
[16] Anthropic, Long context prompting for claude 2.1, 2023. URL: https://www.anthropic.com/news/claude-2-1-prompting.
[17] Q. Xu, F. Hong, B. Li, C. Hu, Z. Chen, J. Zhang, On the Tool Manipulation Capability of Open-source Large Language Models, arXiv preprint arXiv:2305.16504 (2023).
[18] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, et al., Toolllm: Facilitating large language models to master 16000+ real-world apis, arXiv preprint arXiv:2307.16789 (2023).
[19] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, E. Chen, Large Language Models for Generative Information Extraction: A Survey, ArXiv abs/2312.17617 (2023). URL: https://api.semanticscholar.org/CorpusID:266690657.
[20] A. Dunn, J. Dagdelen, N. Walker, S. Lee, A. S. Rosen, G. Ceder, K. Persson, A. Jain, Structured information extraction from complex scientific text with fine-tuned large language models, 2022. arXiv:2212.05238.
[21] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Cao, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C. C. Chang, F. Huang, R. Cheng, Y. Li, Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs, 2023. arXiv:2305.03111.
[22] R. Srivastava, Defog SQLCoder, 2023. URL: https://github.com/defog-ai/sqlcoder.
[23] M. Josifoski, N. De Cao, M. Peyrard, F. Petroni, R. West, GenIE: Generative information extraction, in: M. Carpuat, M.-C. de Marneffe, I. V. Meza Ruiz (Eds.), Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Seattle, United States, 2022, pp. 4626–4643. URL: https://aclanthology.org/2022.naacl-main.342. doi:10.18653/v1/2022.naacl-main.342.
[24] Mistral AI, Mixtral of experts, 2023. https://mistral.ai/news/mixtral-of-experts/ and https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1.
[25] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, 2020. arXiv:2005.14165.
[26] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, T. Liu, A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions, ArXiv abs/2311.05232 (2023). URL: https://api.semanticscholar.org/CorpusID:265067168.
[27] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, QLoRA: Efficient Finetuning of Quantized LLMs, ArXiv abs/2305.14314 (2023). URL: https://api.semanticscholar.org/CorpusID:258841328.
[28] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-Rank Adaptation of Large Language Models, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9.
[29] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, I. Stoica, Efficient Memory Management for Large Language Model Serving with PagedAttention, in: Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles, 2023.
[30] E. Frantar, S. Ashkboos, T. Hoefler, D. Alistarh, GPTQ: Accurate Post-training Compression for Generative Pretrained Transformers, arXiv preprint arXiv:2210.17323 (2022).
[31] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, S. Han, AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration, arXiv (2023).
[32] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe, Training language models to follow instructions with human feedback, 2022. arXiv:2203.02155.
[33] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, C. Finn, Direct Preference Optimization: Your Language Model is Secretly a Reward Model, 2023. arXiv:2305.18290.

A. Flythings

The FlyThings® platform is an all-in-one tool for IoT device management for many different productive sectors. It is designed for the analysis and forecasting of data records of IoT devices, considering any of the data types available at scale. FlyThings® handles a wide variety of sensors, systems and applications for specific use cases including, but not limited to, smart industries or intelligent energy. FlyThings® helps in the decision making process, yielding better results for enterprises, with ad hoc offerings including modular Big Data as a Service (BDaaS) with standard APIs for data management and visualization. Check https://itg.es/en/monitoring-iot-platform-flythings/ for more details.