

Large Language Models as Minecraft Agents

Chris Madge

Queen Mary University of London
c.j.madge@qmul.ac.uk

Massimo Poesio

Queen Mary University of London
m.poesio@qmul.ac.uk

Abstract

In this work we examine the use of Large Language Models (LLMs) in the challenging setting of acting as a Minecraft agent. We apply and evaluate LLMs in the builder and architect settings, introduce clarification questions and examining the challenges and opportunities for improvement. In addition, we present a platform for online interaction with the agents and an evaluation against previous works.

1 Introduction

The development of conversational agents able to operate in virtual world environments has long been of interest in AI (Winograd, 1972). In recent years, much of this research has focused on developing agents able to operate in game environments. Game environments provide an ideal sandbox for studying task-oriented conversational agents in games (Szlam et al., 2019), which has motivated the development of multiple platforms in which such research can be carried out (Johnson et al., 2016; Urbanek et al., 2019; Callison-Burch et al., 2022) (Gray et al., 2019; Ogawa et al., 2020; Köhn et al., 2020), data gathering exercises (Narayan-Chen et al., 2019; Jayannavar et al., 2020; Mohanty et al., 2022) and competitions (Kiseleva et al., 2022a). In this work we study the development of conversational agents able to operate in one of the most popular of these virtual world settings, the Minecraft world, and to carry out the “Collaborative Builder Task” proposed in (Narayan-Chen et al., 2019), in which an architect, who has a target structure to achieve, provides instructions to a builder, to complete that structure.

In previous research, deep neural methods were used to model both of these roles. These models used a CNN to model the world state, and an RNN to model the text input, with a similar decoder to produce the utterances (Narayan-Chen et al., 2019; Jayannavar et al., 2020; Kiseleva et al., 2023; Shi

et al., 2022). An alternative approach has been to separate the builder task into interpretation of natural language instructions, translating them into structured commands using a fine tuned T5 model (Raffel et al., 2020), then using a separate model to enact the required actions in the world (Kiseleva et al., 2023).

However, while it has been proposed using modern large language models (LLMs) such as Llama and GPT-4 for this purpose has been proposed (Hromei et al., 2023), to our knowledge this has yet to be explored. LLMs are arguably one of the most interesting advancements in natural language technology, if not the field of machine learning, in recent years. Some studies suggest that beyond their conversational ability (natural language understanding and generation (Achiam et al., 2023)), such models demonstrate the ability to solve novel tasks akin to an artificial general intelligence (Bubeck et al., 2023) with great potential for serving as the foundation for interactive agents (Xi et al., 2023). There are also many applications of LLMs as autonomous agents or embodied AI in interactive environments, particularly in robotics systems (Wu et al., 2023; Ahn et al., 2022) and multi-agent simulations with cooperation/interaction (Park et al., 2023; Lin et al., 2023; Chen et al., 2023), industrial automation (Xia et al., 2023) and planning (Nottingham et al., 2023; Wang et al., 2023a; Zhu et al., 2023). However, we are not aware of studies using LLMs to implement Minecraft agents.

Our first contribution are LLM-based builder and architect agents. We developed builder and architect with both open-source and closed-source LLMs. We evaluate the builder by comparing it to previous comparable agents. Our second contribution is giving to these agents the ability to ask clarification questions (Skrynnik et al., 2022). This is one ability of conversational agents that remains relatively unexplored in this setting; to our knowledge, there is only a single model attempt-

ing this (Mehta et al., 2023), using BART (Lewis et al., 2019), but none with current LLMs. Our third contribution is a test environment, which we believe is highly suited to devising and testing different approaches to these problems in a simple task orientated setting. We deliberately present our environment as a web-based online platform to support future large scale experimentation with human participation. Our implementation draws inspiration from the web-based Interactive Grounded Language Understanding (IGLU) data collection system (Mohanty et al., 2022; Mehta et al., 2023).

2 Related Work

2.1 The Original Architect and Builder Models for the Minecraft Task

Several models for both the *architect* and *builder* roles have been proposed. The first *architect* model (Narayan-Chen et al., 2019) uses a seq2seq RNN that aside from the usual sequence of tokens as an input, also takes the current and target world state. This is decoded to the Architect utterance.

Following on from the Architect Model, the “Builder Action Prediction” model was created (Jayannavar et al., 2020). This model took also combined a CNN with an RNN, although the decoder now predicted a series of actions the agent could follow.

2.2 IGLU

The Interactive Grounded Language Understanding shared tasks (Kiseleva et al., 2022a,b) were proposed to stimulate research on building interactive agents that learn to solve a task while provided with grounded natural language instructions in a collaborative environment. The more recent edition of the shared task (Kiseleva et al., 2022b) involved two subtasks: ‘Building Structures’ (developing a conversational agent able to engage in the conversation), and ‘Asking Clarification Questions’ (developing a model able to ask clarification questions at the appropriate point).

The baseline (Skrynnik et al., 2022) provided for the 2022 IGLU task (Kiseleva et al., 2023) is split into two separate modules. One language component responsible for interpreting the natural language utterances and transforming them into simplified structured instructions, and a reinforcement learning component responsible for navigating the agent in the world and performing actions to realise those instructions.

Several builder models were entered in the IGLU 2022 competition task (Kiseleva et al., 2022b). The creators of the competition reflect upon the most notable winning submissions (Kiseleva et al., 2023). The top architecture took an end-to-end approach concatenating outputs from a language model and a CNN, which is then fed into a transformer designed specifically to improve and fine tune language models through reinforcement learning. However, second and third place both made improvements on the baselines architecture approach with further fine tuning on the T5 transformer model (Raffel et al., 2020).

2.3 Using LLMs to generate Embodied AI in an Interactive Environment

There are many applications of LLMs as autonomous agents or embodied AI in interactive environments (Wang et al., 2023b; Xi et al., 2023). Many of these do operate in a Minecraft world. However, they are targeted towards planning (e.g. dividing tasks to reach a final goal, including “crafting”/combining items, to reach that outcome) (Nottingham et al., 2023; Wang et al., 2023a; Zhu et al., 2023), not the creation of 3D structures.

2.4 Generating and Responding to Clarification Questions

Generating and responding to clarification questions generative remains relatively unexplored for this task. There have been models that label utterances as requiring clarification (Shi et al., 2022), or rank clarification questions appropriateness to an instruction in an information retrieval style (Kiseleva et al., 2023; Mohanty et al., 2023), but only one generative model (Mehta et al., 2023) using BART (Lewis et al., 2019). Whilst the notion of using modern LLMs such as Llama and GPT-4 for this purpose has been proposed, to our knowledge this is yet to be explored (Hromei et al., 2023).

3 Methodology

In this work we created models for the “Collaborative Builder Task” (Narayan-Chen et al., 2019) in which two agents collaborate in a voxel based world to build a structure. The *architect* is given a target world structure, and provides natural language instructions to a *builder*, who cannot see the target structure, to place blocks to achieve that goal structure. Unlike the majority of previous approaches, both agents are able to provide and answer questions.

3.1 The Builder

The builder agent operates as follows: The initial prompt to the LLM contains instructions for the task, constraints of the world (min/max for x, y, z), coordinate mappings (east/south/west/north) and requests that the LLM respond in a specified JSON format only (see 5 for full prompt). Following the initial prompt, on each turn, the agent consumes an utterance (instruction) and produces a sequence of actions in addition to an optional clarification question in the originally specified format. This JSON encoded representation is used to directly update the world. If the result does not constitute a valid JSON response it is disregarded. The builder is evaluated against IGLU RL baseline sub-task. We evaluate the agent based on its ability to interpret a natural language instruction and produce the correct 3D coordinates and colours for the target structure (not manoeuvre an agent in a 3D space to interact with the world and realise that target structure).

The closest work to our methodology is (Mehta et al., 2023), similarly we focus solely on the language component of this task. We do not examine the task of manipulating an agent to place blocks as done in previous tasks (Narayan-Chen et al., 2019; Jayannavar et al., 2020; Shi et al., 2022), as it is the language element that we are most interested in.

3.2 The Architect

The architect differs slightly from the builder. The prompt, along with architect specific instructions, contains the target state of the world (see 5). Following the initial prompt the architect consumes every utterance given by the builder, along with the current state of the world encoded in a structured JSON format. The target structures are taken from the Minecraft Dialogue Corpus (Narayan-Chen et al., 2019) and evaluated empirically through repeated interactions.

3.3 The LLMs

In this work we test against the LLMs: GPT-4; GPT-3.5; Llama2 (Touvron et al., 2023) (the chat variation in various parameter sizes) and Vicuna (Chiang et al., 2023). To our knowledge, the number of parameters used in the GPT models have not been formally released, although they are rumoured to be magnitudes greater than those used in the open models. The prompts used for both tasks are given in the appendix.

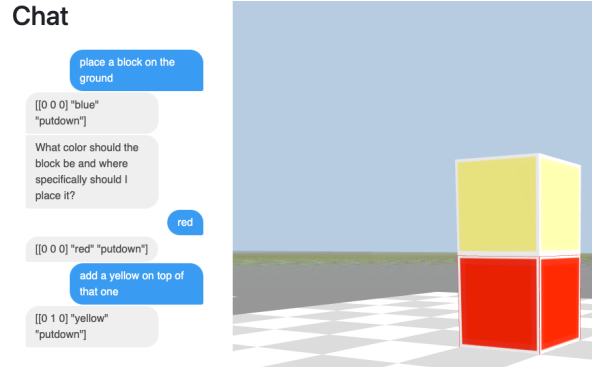


Figure 1: Builder clarification and reference (GPT-4)

4 Results

4.1 Builder

Here, the builder is the agent, and the architect is either a human tester, or a dataset for evaluation purposes.

Table 1 shows the results comparing the original IGLU Baseline against our LLM based builder agent. GPT-4 and GPT-3.5 perform similarly, outperforming the IGLU NLP evaluation baseline ¹. Llama3-70b-Instruct performs similarly to GPT4. However Llama2, while Llama2-70b-chat comes close, other LLM models do not. The smaller parameter models struggle to consistently adhere to the structured response format. This is particularly in evident in the case of Vicuna-13b which often generates responses in encoded in other languages/formats.

Model	Accuracy %
IGLU Baseline	27.1
GPT-4	37.6
GPT-3.5	34.4
Llama3-70b-Instruct	37.4
Llama3-8b-Instruct	23.5
Llama2-70b-chat	23.9
Llama2-13b-chat	15.0
Vicuna-13b	6.3

Table 1: Accuracy on IGLU NLP Evaluation

In Figure 1 we deliberately give an incomplete instruction, “place a block on the ground”. We can see GPT-4 identifies the missing properties, asking for clarification on the colour and position with

¹https://gitlab.aicrowd.com/aicrowd/challenges/iglu-challenge-2022/iglu-2022-rl-mhb-baseline/-/blob/master/local_nlp_evaluation.py

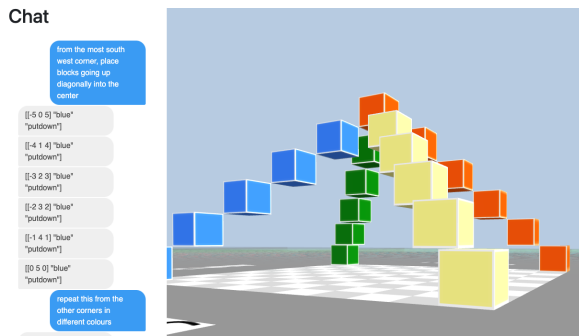


Figure 2: Builder complex structures (GPT-4)

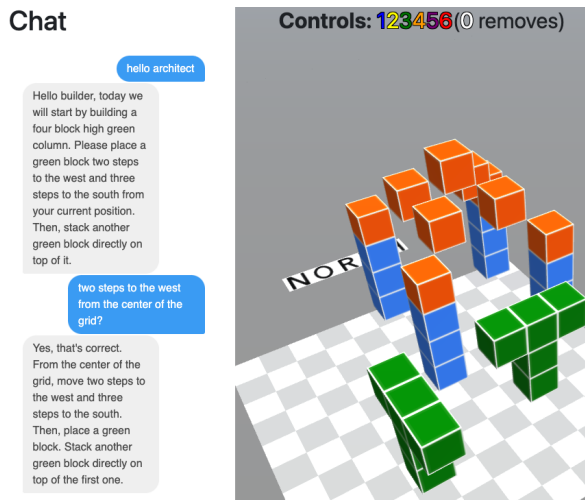


Figure 3: Architect - Simple Structure (GPT-4)

the question: “What color should the block be and where specifically should I place it?”. When the builder responds “red”, this is resolved. We also show here a simple example of resolving reference to an existing block.

In Figure 2, we can see that GPT-4 will accept descriptions of quite complex structures, references to world positions, and continue a pattern between those positions. In addition, it will accept references that point to groups of blocks or existing structures/word spaces and carry out repetition.

4.2 Architect

In Figure 3, for the purpose of illustrating the example, the target structure is shown (this would normally be hidden from the builder). We can see, for simpler structures, the architect gives accurate instructions, can describe groups of placements (e.g. “column”), respond to questions and provide clarification.

However, as the structures increase in complexity (see Figure 4), the architect struggles to start.

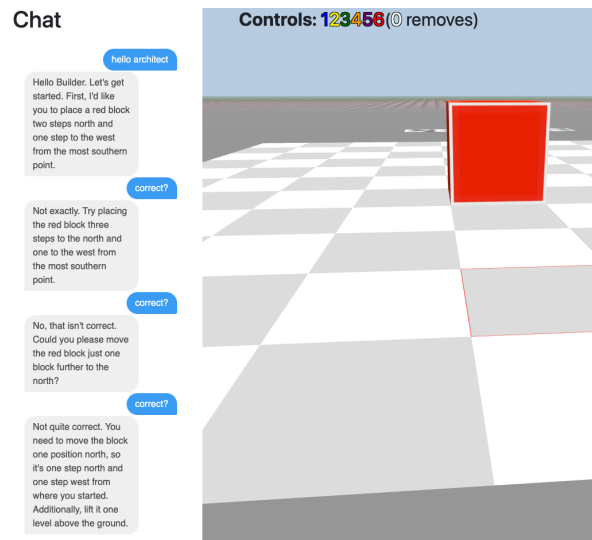


Figure 4: Architect - Complex Structure (GPT-4)

Each time the builder follows the architect's instruction, it revises it getting one step closer to the true goal. The architect does eventually arrive at the appropriate instruction, after repeated exchanges between the builder and architect.

5 Conclusion

In this work we demonstrate LLMs acting as agents in a Minecraft-like block world task. The builder performs favourably over past bespoke models. In addition, the LLM has an existing built in capability to ask and answer questions. In future work, we will look at methods of improving openly available LLM models to close the gap with the fine tuned baseline, ranging from prompt engineering (e.g. introducing few-shot prompting etc.) to fine-tuning and other methods. Our early experiments with the architect does show some challenges, and further work needs to be done to give a quantitative evaluation against existing architect models.

Acknowledgements

This research was funded by ARCIDUCA, EPSRC EP/W001632/1

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

- Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.
- Chris Callison-Burch, Gaurav Singh Tomar, Lara J Martin, Daphne Ippolito, Suma Bailis, and David Reitter. 2022. Dungeons and dragons as a dialog challenge for artificial intelligence. *arXiv preprint arXiv:2210.07109*.
- Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2023. [Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors](#).
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality](#).
- Jonathan Gray, Kavya Srinet, Yacine Jernite, Haonan Yu, Zhuoyuan Chen, Demi Guo, Siddharth Goyal, C Lawrence Zitnick, and Arthur Szlam. 2019. Craftassist: A framework for dialogue-enabled interactive agents. *arXiv preprint arXiv:1907.08584*.
- Claudiu D Hromei, Daniele Margiotta, Danilo Croce, and Roberto Basili. 2023. An end-to-end transformer-based model for interactive grounded language understanding. In *Proceedings of the Seventh Workshop on Natural Language for Artificial Intelligence (NL4AI 2023) co-located with 22th International Conference of the Italian Association for Artificial Intelligence (AI* IA 2023)*.
- Prashant Jayannavar, Anjali Narayan-Chen, and Julia Hockenmaier. 2020. Learning to execute instructions in a minecraft dialogue. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 2589–2602.
- Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. 2016. The malmo platform for artificial intelligence experimentation. In *Ijcai*, pages 4246–4247.
- Julia Kiseleva, Ziming Li, Mohammad Aliannejadi, Shrestha Mohanty, Maartje ter Hoeve, Mikhail Burtsev, Alexey Skrynnik, Artem Zholus, Aleksandr Panov, Kavya Srinet, et al. 2022a. Interactive grounded language understanding in a collaborative environment: Iglu 2021. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 146–161. PMLR.
- Julia Kiseleva, Alexey Skrynnik, Artem Zholus, Shrestha Mohanty, Negar Arabzadeh, Marc-Alexandre Côté, Mohammad Aliannejadi, Milagro Teruel, Ziming Li, Mikhail Burtsev, et al. 2023. Interactive grounded language understanding in a collaborative environment: Retrospective on iglu 2022 competition. In *NeurIPS 2022 Competition Track*, pages 204–216. PMLR.
- Julia Kiseleva, Alexey Skrynnik, Artem Zholus, Shrestha Mohanty, Negar Arabzadeh, Marc-Alexandre Côté, Mohammad Aliannejadi, Milagro Teruel, Ziming Li, Mikhail Burtsev, Maartje ter Hoeve, Zoya Volovikova, Aleksandr Panov, Yuxuan Sun, Kavya Srinet, Arthur Szlam, and Ahmed Awadallah. 2022b. [Iglu 2022: Interactive grounded language understanding in a collaborative environment at neurips 2022](#).
- Arne Köhn, Julia Wichlacz, Christine Schäfer, Alvaro Torralba, Jörg Hoffmann, and Alexander Koller. 2020. Mc-saar-instruct: a platform for minecraft instruction giving agents. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 53–56.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.
- Jiaju Lin, Haoran Zhao, Aochi Zhang, Yiting Wu, Huqiyue Ping, and Qin Chen. 2023. [Agentsims: An open-source sandbox for large language model evaluation](#).
- Nikhil Mehta, Milagro Teruel, Patricio Figueroa Sanz, Xin Deng, Ahmed Hassan Awadallah, and Julia Kiseleva. 2023. Improving grounded language understanding in a collaborative environment by interacting with agents through help feedback. *arXiv preprint arXiv:2304.10750*.
- Shrestha Mohanty, Negar Arabzadeh, Julia Kiseleva, Artem Zholus, Milagro Teruel, Ahmed Awadallah, Yuxuan Sun, Kavya Srinet, and Arthur Szlam. 2023. [Transforming human-centered ai collaboration: Redefining embodied agents capabilities through interactive grounded language instructions](#).
- Shrestha Mohanty, Negar Arabzadeh, Milagro Teruel, Yuxuan Sun, Artem Zholus, Alexey Skrynnik, Mikhail Burtsev, Kavya Srinet, Aleksandr Panov, Arthur Szlam, et al. 2022. Collecting interactive multi-modal datasets for grounded language understanding. *arXiv preprint arXiv:2211.06552*.
- Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier. 2019. Collaborative dialogue in

- minecraft. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5405–5415.
- Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. 2023. [Do embodied agents dream of pixelated sheep: Embodied decision making using language guided world modelling](#).
- Haruna Ogawa, Hitoshi Nishikawa, Takenobu Tokunaga, and Hikaru Yokono. 2020. Gamification platform for collecting task-oriented dialogue data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7084–7093.
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. [Generative agents: Interactive simulacra of human behavior](#).
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.
- Zhengxiang Shi, Yue Feng, and Aldo Lipani. 2022. Learning to execute actions or ask clarification questions. *arXiv preprint arXiv:2204.08373*.
- Alexey Skrynnik, Zoya Volovikova, Marc-Alexandre Côté, Anton Voronov, Artem Zholus, Negar Arabzadeh, Shrestha Mohanty, Milagro Teruel, Ahmed Awadallah, Aleksandr Panov, et al. 2022. Learning to solve voxel building embodied tasks from pixels and natural language instructions. *arXiv preprint arXiv:2211.00688*.
- Arthur Szlam, Jonathan Gray, Kavya Srinet, Yacine Jernite, Armand Joulin, Gabriel Synnaeve, Douwe Kiela, Haonan Yu, Zhuoyuan Chen, Siddharth Goyal, et al. 2019. Why build an assistant in minecraft? *arXiv preprint arXiv:1907.09273*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Jack Urbanek, Angela Fan, Siddharth Karamcheti, Emily Dinan Saachi Jain, Samuel Humeau, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2023b. [A survey on large language model based autonomous agents](#).
- Terry Winograd. 1972. Understanding natural language. *Cognitive psychology*, 3(1):1–191.
- Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. 2023. Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*.
- Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.
- Yuchen Xia, Manthan Shenoy, Nasser Jazdi, and Michael Weyrich. 2023. Towards autonomous system: flexible modular production system enhanced with large language model agents. *arXiv preprint arXiv:2304.14721*.
- Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. 2023. [Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory](#).

Appendix

Architect prompt

You are an agent in a voxel world, where the most northernly point is 0,0,-5; the most westerly point -5,0,0; the most eastern point is 5,0,0; the most southern 0,0,5 and the y-axis is up and down, with y=0 being the minimum. Your task is to give instructions to a human to place blocks to achieve the target world state: «target_world_state» where the target world instructions are in the format [[x,y,z,color],...]. Give easy to interpret instructions, do not directly mention the coordinates. The builder will respond with the coordinates of the blocks they have placed in the same format. Don’t ask for coordinates, they will always be given. Avoid long instructions with multiple steps and start building the structure from the ground up.

Builder prompt

You are an agent in a voxel world, where the most northernly point is 0,0,-5; the most westerly point -5,0,0; the most eastern point is 5,0,0; the most southern 0,0,5 and the y-axis is up and down, with y=0 being the minimum. Describe the coordinates of the blocks their colours (must be one of: blue, yellow, green, orange, purple, red) and whether the action is to add or remove them, your confidence in your interpretation of the instruction and optionally a question if the instruction is potentially unclear, in

the JSON format: {"add": [[x,y,z,color], ...], "remove": [[x,y,z,color], ...], "confidence": 0.0, "question": "..."}.

Give the JSON only, no additional dialog.

.1 Transcripts

Transcript for Figure 1:

Architect *Place a stone on the ground*

Builder `[[0 0 0]] "blue" "putdown"`

Builder *What colour should the block be and where specifically should I place it?*

Architect *red*

Builder `[[0 0 0]] "red" "putdown"`

Architect *add a yellow on top of that one*

Builder `[[0 1 0]] "yellow" "putdown"`

Transcript for Figure 2:

Architect *from the most south west corner, place blocks going up diagonally into the center*

Builder `[[-5 0 5]] "blue" "putdown", [[-4 1 4]] "blue" "putdown", [[-3 2 3]] "blue" "putdown", [[-2 3 2]] "blue" "putdown", [[-1 4 1]] "blue" "putdown", [[0 5 0]] "blue" "putdown"`

Architect *repeat this from the other corners in different colours*

Transcript for Figure 3:

Builder *hello architect*

Architect *Hello builder, today we will start by building a four block high green column. Please place a green block two steps to the west and three steps to the south from your current position. Then, stack another green block directly on top of it.*

Builder *two steps to the west from the center of the grid?*

Architect *Yes, that's correct. From the center of the grid move two steps to the west and three steps to the south. Then, place a green block. Stack another green block directly on top of the first one.*

Transcript for Figure 4:

Builder *hello architect*

Architect *Hello Builder. Let's get started. First, I'd like you to place a red block two steps north and one step to the west from the most southern point.*

Builder *correct?*

Architect *Not exactly. Try placing the red block three steps to the north and one to the west from the most southern point.*

Builder *correct?*

Architect *No, that isn't correct. Could you please move the red block just one block further to the north?*

Builder *correct?*

Architect *Not quite correct. You need to move the block one position north, so it's one step north and one step west from where you started. Additionally, lift it one level above the ground.*