Sanh et al., 2021; Longpre et al., 2023; Chung et al., 2022; Muennighoff et al., 2022). Furthermore, researchers have been actively exploring ways to align LLMs with human instructions (Ouyang et al., 2022; Askell et al., 2021; Bai et al., 2022b;c). One major challenge in alignment research is the difficulty of collecting data. While OpenAI has utilized its platform to gather human prompts or instructions, it is not feasible for others to collect such data.

However, there has been some progress in this area, such as the self-instruct approach proposed in Wang et al. (2023c). This innovative work offers a potential solution to the data collection problem in alignment research. As a result, there has been a surge in open-source chat data, including Alpaca (Taori et al., 2023), MOSS (Sun et al., 2023a), Dolly (Conover et al., 2023), Evol-Instruct (Xu et al., 2023b), and others (Sun et al., 2023b; Xu et al., 2023a;c; Chen et al., 2023b; Ding et al., 2023; Ji et al., 2023; Yang, 2023). Similarly, there has been an increase in open-source chat models, such as Alpaca (Taori et al., 2023), Vicuna (Chiang et al., 2023), Guanaco (Dettmers et al., 2023), MOSS (Sun et al., 2023a), WizardLM (Xu et al., 2023b), and others (Xu et al., 2023c; Chen et al., 2023b; Ding et al., 2023; Wang et al., 2023b).

To train an effective chat model, available solutions are mostly based on SFT and RLHF (Ouyang et al., 2022). While SFT is similar to pretraining, it focuses on instruction following using the aforementioned data. However, for many developers, the limited memory capacity is a major obstacle to further research in SFT. As a result, parameter-efficient tuning methods, such as LoRA (Hu et al., 2021) and Q-LoRA (Dettmers et al., 2023), have gained popularity in the community. LoRA tunes only low-rank adapters, while Q-LoRA builds on LoRA and utilizes 4-bit quantized LLMs and paged attention (Dettmers et al., 2022; Frantar et al., 2022; Kwon et al., 2023). In terms of RLHF, recent methods such as PPO (Schulman et al., 2017; Touvron et al., 2023b) have been adopted, but there are also alternative techniques aimed at addressing the complexity of optimization, such as RRHF (Yuan et al., 2023c), DPO (Rafailov et al., 2023), and PRO (Song et al., 2023). Despite the ongoing debate about the effectiveness of RLHF, more evidence is needed to understand how it enhances the intelligence of LLMs and what potential drawbacks it may have.

6.3 Tool Use and Agents

LLM's planning function allows for the invocation of tools, such as APIs or agent capabilities, through in-context learning, as demonstrated by Schick et al. (2023). Yao et al. (2022) introduced ReAct, a generation format that enables the model to generate thoughts on which tool to use, accept input from API observations, and generate a response. GPT-3.5 and GPT-4, when prompted with few shots, have shown consistent and impressive performance. In addition to tool usage, LLMs can utilize external memory sources like knowledge bases (Hu et al., 2023; Zhong et al., 2023b) or search engines (Nakano et al., 2021; Liu et al., 2023b) to generate more accurate and informative answers. This has led to the popularity of frameworks like LangChain (LangChain, Inc., 2023). The research on LLMs for tool use has also sparked interest in building agents with LLM capabilities, such as agents that can call different AI models (Shen et al., 2023; Li et al., 2023a), embodied lifelong learning or multimodal agents (Wang et al., 2023a; Driess et al., 2023), and multiple agents interacting with each other and even building a micro-society (Chen et al., 2023a; Li et al., 2023b; Xu et al., 2023d; Hong et al., 2023).

6.4 LLM for Coding

Previous research has demonstrated that LLMs possess remarkable capabilities in code understanding and generation, particularly those with massive numbers of parameters (Chowdhery et al., 2022; Anil et al., 2023; Rae et al., 2021; Hoffmann et al., 2022). Moreover, several LLMs have been pre-trained, continued pre-trained, or fine-tuned on coding-related data, which has resulted in significantly improved performance compared to general-purpose LLMs. These models include Codex Chen et al. (2021a), AlphaCode (Li et al., 2022), SantaCoder (Allal et al., 2023), Starcoder-Base (Li et al., 2023d), InCoder (Fried et al., 2022), CodeT5 (Wang et al., 2021), CodeGeeX (Zheng et al., 2023), and Code LLaMA (Rozière et al., 2023). In addition to these models, recent studies have focused on developing specialized alignment techniques for coding, such as Code Llama-Instruct (Rozière et al., 2023) and StarCoder (Li et al., 2023d). These models can assist developers in various code-related tasks, including code generation (Chen et al., 2021b; Austin et al., 2021), code completion (Zhang et al., 2023a), code translation (Szafraniec et al., 2023), bug fixing (Muennighoff et al., 2023), code