

WizardLM: Empowering Large Language Models to Follow Complex Instructions

Can Xu^{1*} Qingfeng Sun^{1*} Kai Zheng^{1*} Xiubo Geng¹
 Pu Zhao¹ Jiazhan Feng^{2†} Chongyang Tao¹ Qingwei Lin¹ Daxin Jiang^{1‡}

¹Microsoft

²Peking University

{caxu,qins,zhengkai,xigeng,puzhao,chongyang.tao,qlin,djiang}@microsoft.com
 {fengjiazhan}@pku.edu.cn

Abstract

Training large language models (LLMs) with open-domain instruction following data brings colossal success. However, manually creating such instruction data is very time-consuming and labor-intensive. Moreover, humans may struggle to produce high-complexity instructions. In this paper, we show an avenue for creating large amounts of instruction data with varying levels of complexity using LLM instead of humans. Starting with an initial set of instructions, we use our proposed *Evol-Instruct* to rewrite them step by step into more complex instructions. Then, we mix all generated instruction data to fine-tune LLaMA. We call the resulting model *WizardLM*. Human evaluations on a complexity-balanced test bed and Vicuna’s testset show that instructions from *Evol-Instruct* are superior to human-created ones. By analyzing the human evaluation results of the high complexity part, we demonstrate that outputs from our *WizardLM* model are preferred to outputs from OpenAI ChatGPT. In GPT-4 automatic evaluation, *WizardLM* achieves more than 90% capacity of ChatGPT on 17 out of 29 skills. Even though *WizardLM* still lags behind ChatGPT in some aspects, our findings suggest that fine-tuning with AI-evolved instructions is a promising direction for enhancing LLMs. Our code and data are public at <https://github.com/nlpxucan/WizardLM>.

1 Introduction

Large-scale language models (LLMs) have become the go-to approach for numerous natural language processing (NLP) tasks [1–4]. LLMs are trained on large volumes of text data to predict the subsequent tokens, enabling them to generate coherent and fluent text in response to various inputs. However, these models often struggle to follow instructions or goals specified by users, which limits their usefulness and applicability in real-world scenarios.

The NLP community has recently witnessed many endeavors to train LLMs to follow instructions better and be more helpful [5–8]. Initial attempts [9–13] to train instruction-following language models are based on a collection of various NLP tasks, with a small amount of hand-written instructions accompanying each task. These closed-domain instructions suffer from two main drawbacks: first, all the samples in an NLP dataset share only a few common instructions, severely limiting their diversity; second, the instructions usually only ask for one task, such as translation or summarization. But in real life, human instructions often have multiple and varied task demands. By using open-domain in-

* Equal contribution.

† Work done during the internship at Microsoft.

‡ Corresponding author.

struction data generated by real human users, OpenAI’s LLMs (e.g., InstructGPT [2] and ChatGPT ⁴) have achieved great success. These open-domain instructions can fully unleash the unlimited potential of LLMs [14–17] and enable them to perform more complex and diverse tasks. However, using humans to create open-domain instruction datasets like OpenAI did will encounter the following challenges. The whole annotating process is extremely expensive and time-consuming [18–21]. On the other hand, the difficulty level distribution of human-created instructions is skewed towards being easy or moderate, with fewer difficult ones (according to the difficulty statistics of ShareGPT [22] from Figure 7a). Possible reasons for this are that the proportion of experts among annotators is low and creating complex instructions demands a lot of mental effort. Human annotators are prone to fatigue and cannot sustain high-intensity work to produce a sufficient proportion of high-difficulty instructions [23–26]. Based on these issues, developing an automatic method that can mass-produce open-domain instructions (especially the more difficult ones) at a relatively low cost becomes the key to further advancing instruction-tuned language models [27–30].

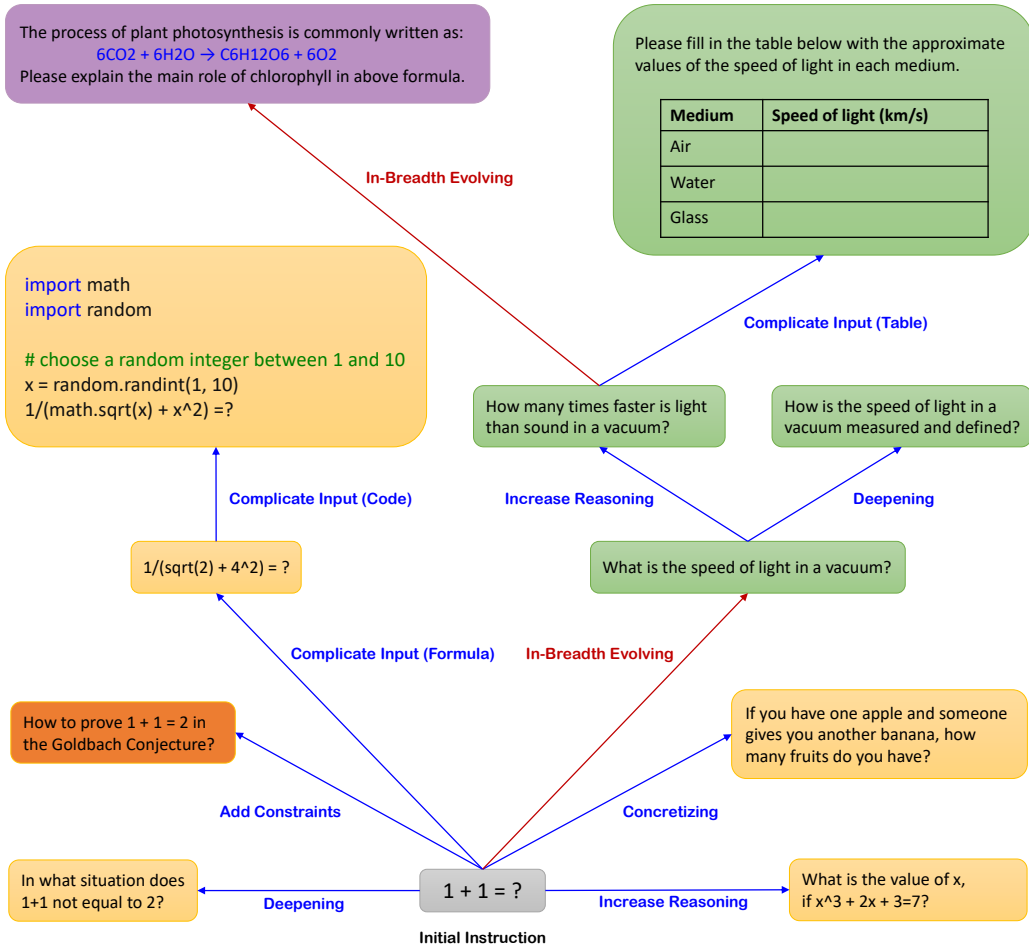


Figure 1: Running Examples of *Evol-Instruct*.

In this work, we introduce *Evol-Instruct*, a novel method using LLMs instead of humans to automatically mass-produce open-domain instructions of various difficulty levels, to improve the performance of LLMs. Figure 1 shows the running examples of *Evol-Instruct*. Starting from a simple initial instruction “ $1+1=?$ ”, our method randomly selects In-depth Evolving (blue direction line) or In-breadth Evolving (red direction line) to upgrade the simple instruction to a more complex one or create a new one (to increase diversity). The In-depth Evolving includes five types of operations: add constraints, deepening, concretizing, increase reasoning steps, and complicate input. The In-breadth Evolving is mutation, i.e., generating a completely new instruction based on the given instruction. These six

⁴<https://chat.openai.com/>

operations are implemented by prompting an LLM with specific prompts. Since the evolved instructions are generated from LLMs, sometimes the evolving will fail. We adopt an instruction eliminator to filter the failed instructions, which is called Elimination Evolving. We repeat this evolutionary process for several rounds to obtain enough instruction data containing various complexities.

We validate our *Evol-Instruct* by fine-tuning open-source LLaMA [4] with our evolved instructions and evaluating its performance similar to existing SOTA works (e.g., Alpaca [31] and Vicuna [22]) on instruction finetuning. The instruction datasets we compare with are the data used by Alpaca (generated using self-instruct [32]) and the 70k ShareGPT (shared by real users) used by Vicuna. To prove that the instruction dataset from our method is superior to human-created instruction datasets, we select Alpaca’s training data (generated from only 175 human-created seed instructions) as the initial dataset. We execute four epochs of evolution using OpenAI ChatGPT API⁵ and finally obtain 250k instructions. To ensure a fair comparison with Vicuna’s 70k real user data, we sampled an equal amount from the full 250k data and trained the LLaMA 7B model. We name our model *WizardLM*. Due to the low proportion of difficult instructions in the previous instruction-following test dataset, we manually created a new difficulty-balanced test dataset, named *Evol-Instruct* testset. We hire annotators and leverage GPT-4 to evaluate Alpaca, Vicuna, ChatGPT, and *WizardLM* on *Evol-Instruct* testset and Vicuna’s testset. Our main findings are as follows:

- **Instructions from *Evol-Instruct* are superior to the ones from human-created ShareGPT.** When we use the same amount of *Evol-Instruct* data (i.e., 70k) as Vicuna to fine-tune LLaMA 7B, our model *WizardLM* significantly outperforms Vicuna, with the win rate of 12.4% and 3.8% higher than Vicuna on *Evol-Instruct* testset and Vicuna’s testset respectively on human evaluation. In addition, *WizardLM* also achieves better response quality than Alpaca and Vicuna on the automatic evaluation of GPT-4.
- **Labelers prefer *WizardLM* outputs over outputs from ChatGPT under complex test instructions.** On *Evol-Instruct* testset, *WizardLM* performs worse than ChatGPT, with a win rate 12.8% lower than ChatGPT (28.0% vs. 40.8%). However, in the high-difficulty section of *Evol-Instruct* test set (difficulty level ≥ 8), our *WizardLM* even outperforms ChatGPT, with a win rate 7.9% larger than ChatGPT (42.9% vs. 35.0%), that is human annotators even prefer the output of our model than ChatGPT on those hard questions. This indicates that *Evol-Instruct* can significantly improve the ability of LLMs to handle complex instructions.

2 Related Work

Closed domain instruction fine-tune Early instruction-following training work [10, 33] concerns cross task generalization in LMs, where LMs are fine-tuned on a broad range of public NLP datasets and evaluated on a different set of NLP tasks. T5 [34] made the earliest attempt by training natural language processing (NLP) tasks such as question answering, document summarization, and sentiment classification together using a unified text-to-text format. Works such as FLAN [10], ExT5 [9], T0 [12], and KnowDA [35] increased the number of NLP tasks to around one hundred, with several instructions carefully designed for each task [36–39]. Furthermore, works such as ZeroPrompt [11] and FLAN-T5 [13] raised the number of tasks to the thousands. These studies consistently show that fine-tuning LMs with diverse NLP task instructions enhances their performance on new tasks. However, LLMs trained with these closed-form instructions (i.e., instructions are often only for a single NLP task, and the input data form is simple) tend to fail in real-world user scenarios.

Open domain instruction fine-tune Our work belongs to this research line. OpenAI has hired many annotators and written many instructions with corresponding correct responses. These human-created instructions have diverse forms and rich task types. Based on this dataset, OpenAI trained GPT-3 [1] into InstructGPT [2], which can process a variety of real user instructions and led to the success of ChatGPT. Since these outstanding works from OpenAI were not open-sourced, Alpaca [31] and Vicuna [22] subsequently actively explored open-domain instruction fine-tuning based on the open-source LLM LLaMA [4]. Alpaca used a dataset of 50k instructions generated from a limited (e.g., 175 samples) seed set of manually-written instructions. Vicuna used 70k user-shared conversations with ChatGPT collected from ShareGPT.com. Our work is different from InstructGPT and Vicuna in that

⁵gpt-3.5-turbo from <https://oai.azure.com/portal>

we use AI-generated data for instruction fine-tuning. Unlike Alpaca’s self-instruct [32] generation method, *Evol-Instruct* can control the difficulty and complexity level of the generated instructions.

3 Approach

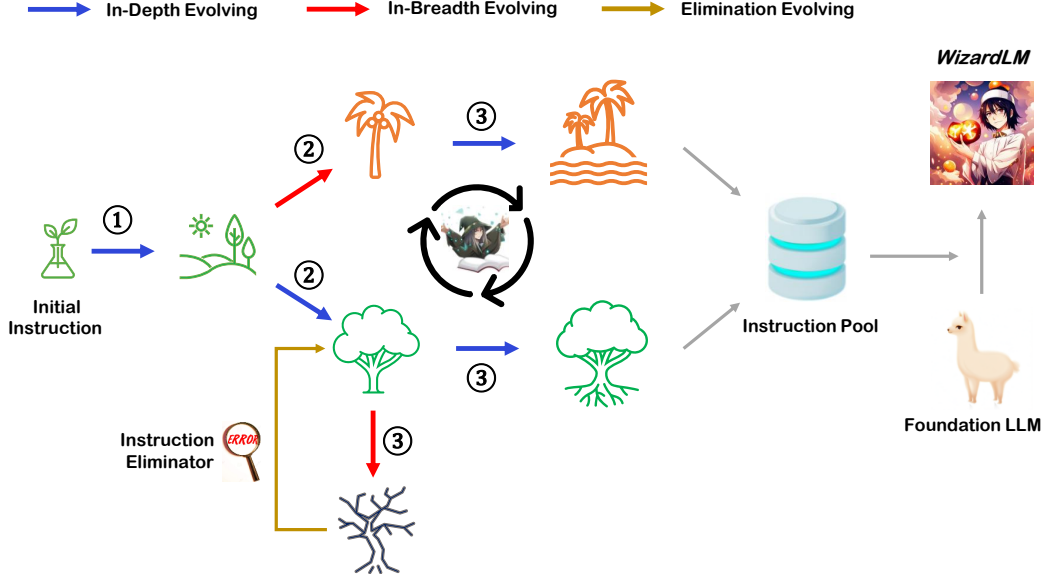


Figure 2: Overview of *Evol-Instruct*

In this section, we elaborate on the details of the proposed *Evol-Instruct*. As illustrated in Figure 2, the pipeline mainly contains two components: Instruction Evolver and Instruction Eliminator. The details of these components will be presented in Sec. 3.2 and instruction fine-tuning method will be described in Sec. 3.3.

3.1 Definition of Instruction Data Evolution

We start the evolution from a given initial instruction dataset $D^{(0)} = (I_k^{(0)}, R_k^{(0)})_{1 \leq k \leq N}$, where $I_k^{(0)}$ is the k -th instruction in $D^{(0)}$, $R_k^{(0)}$ is the corresponding response for the k -th instruction, and N is the number of samples in $D^{(0)}$. In each evolution, we upgrade all the $I^{(t)}$ in $D^{(t)}$ to $I^{(t+1)}$ by applying a LLM instruction evolution prompt, and then use the LLM to generate corresponding responses R^{t+1} for the newly evolved I^{t+1} . Thus, we obtain an evolved instruction dataset D^{t+1} . By iteratively performing M evolutions, we can sequentially obtain M evolution datasets $[D^{(1)} \dots D^{(M)}]$. Our work focuses on open-domain instruction data, where instructions have varying inputs and tasks without a clear distinction between the instruction part and the input.

3.2 Automatic Instruction Data Evolution

Our pipeline for instruction evolution consists of three steps: 1) instruction evolving, 2) response generation, and 3) elimination evolving, i.e., filtering instructions that fails to evolve.

Instruction Evolution. We found that LLMs can make given instructions more complex and difficult using specific prompts. Additionally, they can generate entirely new instructions that are equally complex but completely different. Using this discovery, we can iteratively evolve an initial instruction dataset, improving difficulty level and expanding its richness and diversity. We initiate the instruction pool with the given initial instruction dataset $D^{(0)}$. In each evolution epoch, upgraded instructions from the previous epoch are taken out from the pool. Then we leverage the instruction evolver to evolve each fetched instruction, and the instruction eliminator to check whether the

evolution fails. Successful evolved instructions are added to the pool, while unsuccessful ones are placed back as they are, with the hope of upgrading them successfully in the next evolution epoch.

Instruction Evolver. The Instruction Evolver is an LLM that uses prompts to evolve instructions, with two types: in-depth evolving and in-breadth evolving.

Prompts of In-Depth Evolving. In-Depth Evolving enhances instructions by making them more complex and difficult through five types of prompts: add constraints, deepening, concretizing, increased reasoning steps, and complicating input. The core part of In-Depth Evolving’s prompt is "Your objective is to rewrite a given prompt into a more complex version to make those famous AI systems (e.g., ChatGPT and GPT4 [3]) a bit harder to handle. But the rewritten prompt must be reasonable, understood, and responded to by humans". We require the LLM to create challenging instructions that are reasonable and not arbitrarily imagined by AI. A gradual difficulty increase is necessary to avoid filling the instruction set with extremely complex instructions, which would harm the generalization performance of trained models. To control difficulty increase, we make each evolution "a bit harder" and restrict adding a maximum of 10 to 20 words. Among the five mentioned evolving, all can be implemented without any in-context examples except for complicating input. We show the prompt of add constraints as follows (the prompts of deepening, concretizing and increased reasoning steps will be detailed in the Appendix A-C)

```
I want you act as a Prompt Rewriter.
Your objective is to rewrite a given prompt into a more complex version to make those famous AI systems
(e.g., ChatGPT and GPT4) a bit harder to handle.
But the rewritten prompt must be reasonable and must be understood and responded by humans.
Your rewriting cannot omit the non-text parts such as the table and code in #Given Prompt#. Also, please
do not omit the input in #Given Prompt#.
You SHOULD complicate the given prompt using the following method:
Please add one more constraints/requirements into #Given Prompt#
You should try your best not to make the #Rewritten Prompt# become verbose, #Rewritten Prompt# can only
add 10 to 20 words into #Given Prompt#.
'#Given Prompt#', '#Rewritten Prompt#', 'given prompt' and 'rewritten prompt' are not allowed to appear in
#Rewritten Prompt#
#Given Prompt#:
<Here is instruction.>
#Rewritten Prompt#:
```

For complicating input, we will use in-context demonstration. Due to the lengthy demonstrations, we will provide a brief template below, with the full prompt detailed in the Appendix D.

```
I want you act as a Prompt Rewriter.
Your objective is to rewrite a given prompt into a more complex version to make those famous AI systems
(e.g., ChatGPT and GPT4) a bit harder to handle.
But the rewritten prompt must be reasonable and must be understood and responded by humans.
You must add [XML data] format data as input data in [Rewritten Prompt]
#Given Prompt#:
<Here is Demonstration instruction 1.>
#Rewritten Prompt#:
<Here is Demonstration Example 1.>

... N -1 Examples ...
```

```
I want you act as a Prompt Rewriter.
Your objective is to rewrite a given prompt into a more complex version to make those famous AI systems
(e.g., ChatGPT and GPT4) a bit harder to handle.
But the rewritten prompt must be reasonable and must be understood and responded by humans.
You must add [#Given Dataformat#] format data as input data, add [#Given Dataformat#] code as input code
in [Rewritten Prompt]
Rewrite prompt must be a question style instruction
#Given Prompt#:
<Here is instruction.>
```

#Rewrite prompt must be a question style instruction Rewritten Prompt(MUST contain a specific JSON data as input#:

Prompts of In-Breadth Evolving. In-Breadth Evolving aims to enhance topic coverage, skill coverage, and overall dataset diversity. Open-domain instruction finetune datasets (e.g., Alpaca, ShareGPT, etc.) are typically small in scale, lacking topic and skill diversity. To solve this problem, we designed a prompt to generate a completely new instruction based on the given instruction, requiring the new instruction to be more long-tailed. Our In-Breadth Evolving prompt is as follows:

```
I want you act as a Prompt Creator.
Your goal is to draw inspiration from the #Given Prompt# to create a brand new prompt.
This new prompt should belong to the same domain as the #Given Prompt# but be even more rare.
The LENGTH and difficulty level of the #Created Prompt# should be similar to that of the #Given Prompt#.
The #Created Prompt# must be reasonable and must be understood and responded by humans.
'#Given Prompt#', '#Created Prompt#', 'given prompt' and 'created prompt' are not allowed to appear in
#Created Prompt#.
#Given Prompt#:
<Here is instruction.>
#Created Prompt#:
```

Response Generation. We use the same LLM as for evolving to generate the corresponding responses for the evolved instructions. The generation prompt is "<Here is instruction.>".

Elimination Evolving. We classify the following four situations as instruction evolution failure:

1. The evolved instruction does not provide any information gain compared to the original one. We use ChatGPT to make this determination, details please refer to Appendix G.
2. The evolved instruction makes it difficult for the LLM to generate a response. We found that when the generated response contains "sorry" and is relatively short in length (i.e., less than 80 words), it often indicates that the LLM struggles to respond to the evolved instruction. So we can use this rule to make a judgment.
3. The response generated by the LLM only contains punctuation and stop words.
4. The evolved instruction obviously copies some words from the evolving prompt, such as "given prompt", "rewritten prompt", "#Rewritten Prompt#", etc.

3.3 Finetuning the LLM on the Evolved Instructions

Once all evolutions are done, we will merge the initial instruction dataset with evolved instruction data from all epochs and randomly shuffle the samples to create the final fine-tuning dataset. This processing ensures even distribution of instructions of varying difficulty levels in the dataset, maximizing model fine-tuning smoothness. We avoided using complex or multiple prompt templates from previous instruction tuning works[32, 40] to ensure the fine-tuned model can handle open-domain instructions. We only concatenated the instruction with "### Response:" as the prompt to train the model to generate responses in a standard supervised way.

4 Experiment

We assess *WizardLM*, Alpaca, Vicuna, and ChatGPT on *Evol-Instruct* testset and Vicuna testset using both automatic and human evaluations.

4.1 Baselines

(1) **ChatGPT** is an AI chatbot developed by OpenAI that can interact with users in a natural and engaging way. It is built on top of LLMs like GPT-3.5 and GPT-4, trained on vast internet text data. ChatGPT is one of the most advanced and versatile chatbots available today, but it also has some limitations and challenges, such as factual accuracy, consistency, and safety.

(2) **Alpaca** is an open-source instruction-following model developed by Stanford University. It is based on the LLaMA, and fine-tuned with 52K instruction-following examples generated from OpenAI’s text-davinci-003 model.

(3) **Vicuna** is an open-source chatbot that can generate natural and engaging responses to user queries. It is based on LLaMA and fine-tuned on 70K user-shared conversations collected from ShareGPT, a website where people share their ChatGPT interactions. It is one of the most advanced and versatile open instruction-following models available today. We use the 7B model from FastChat ⁶.

4.2 Experiment detail

To construct the dataset, we initialized it with the 52K instruction dataset of Alpaca. After iteratively performing M evolutions, where $M = 4$, we obtained a 250K instructions. For each instruction in each round of evolution, we randomly select one evolving prompt from total six prompts (i.e., five from in-depth evolving and one from in-breadth evolving) with equal probability. We execute above process using Azure OpenAI ChatGPT API⁷. Finally, we also leveraged the ChatGPT to generate responses. We use a temperature of 1 to generate response and set the maximum number of tokens for generation to 2048. Additionally, we set the frequency penalty to zero and top-p to 0.9. Totally, we request the API $52 \times 4 \times 3 = 624K$ times to construct the full dataset. We use pre-trained LLaMA 7B [4] to initialize our model. We adopt Adam optimizer as an initial learning rate of 2×10^{-5} , a maximum number of tokens 2048, and the batch size is 8 for each GPU. We train our model on 8 V100 GPUs with Deepspeed Zero-3 for 70 hours on 3 epochs. For fair comparison, we replace Alpaca’s original Davinci-003 response with ChatGPT’s response, and also sample 70K instructions subset to train *WizardLM*. For inference, we set the temperature to 1 and the top-p to 0.9 in the same setting for *WizardLM* and baseline models to reduce output randomness and ensure more focused and deterministic outputs. We also use a beam size of 1, and set the maximum generation length to 2048.

4.3 Testset build

We collected our *Evol-Instruct* testset that includes real-world human instructions from diverse sources such as online opensource projects, platforms, and forums. We analyzed the data and identified 29 distinct skills that represent the main requirements of humanity, such as Coding Generation & Debugging, Math, Reasoning, Complex Formats, Writing, Extensive Disciplines, and so on. Figure 3a illustrates the distribution of the instances and skills in our test set. Our test set consists of 218 instances, each of which is an instruction for a specific skill. We compared our test set with Vicuna’s test set, which is a benchmark dataset for evaluating instruction following models. We found that Vicuna’s test set only 80 instances and 9 skills and is much smaller and less diverse than ours. Figure 3b shows how the difficulty and complexity of the test data vary across different instances. Our test data has a more uniform distribution, meaning that it contains instructions with different levels of difficulty and complexity. On the other hand, Vicuna and Alpaca have a skewed distribution, meaning that they mostly contain instructions with low difficulty and complexity. This indicates that these two corpus are not able to handle the evaluation on more complex and demanding scenarios.

4.4 Human evaluation

To evaluate *WizardLM*, we conduct human evaluation on our *Evol-Instruct* testset. We perform a blind pairwise comparison between *WizardLM* and baselines. Specifically, we recruit 10 well-educated annotators. To each annotator, four responses from Alpaca, Vicuna-7b, *WizardLM* and ChatGPT are presented, which are randomly shuffled to hide their sources. The annotators then judge which response is better following criterion in Appendix H. Then they should rank the four responses from 1 to 5 (1 means best), and allowing equal scores for comparable instances. To estimate the win rate, we compare the frequency of win, lost, and tie between each pair of models.

Main Results The results of the experiment are reported in Figure 4. *WizardLM* achieved significantly better results than Alpaca and Vicuna-7b, which demonstrates the effectiveness of *Evol-Instruct*.

Performance on high-difficulty skills. Figure 4c indicates that *WizardLM* has more cases that are preferred by human labelers than ChatGPT in the high-difficulty instructions (difficulty level ≥ 8).

⁶<https://github.com/lm-sys/FastChat>

⁷[gpt-3.5-turbo](https://oai.azure.com/portal) from <https://oai.azure.com/portal>

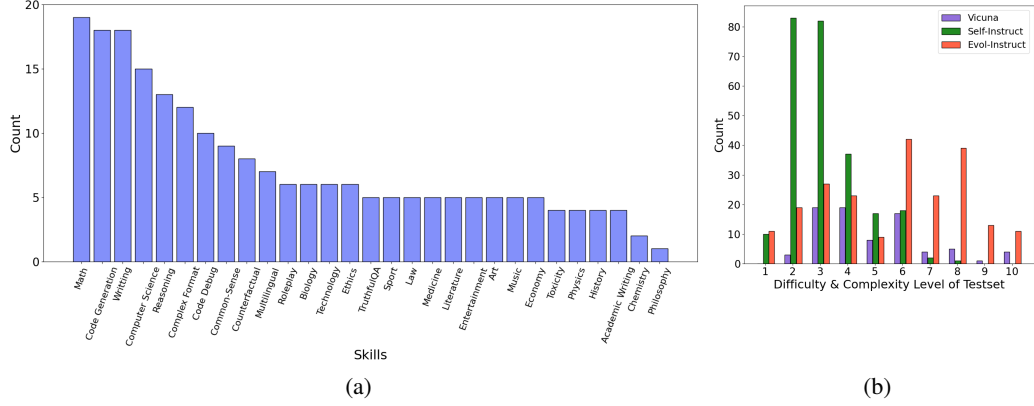


Figure 3: (a) The skills distribution of *Evol-Instruct* testset, and (b) The difficulty and complexity level distribution between the testset of Vicuna, Alpaca (Self-Instruct), and our *Evol-Instruct*.

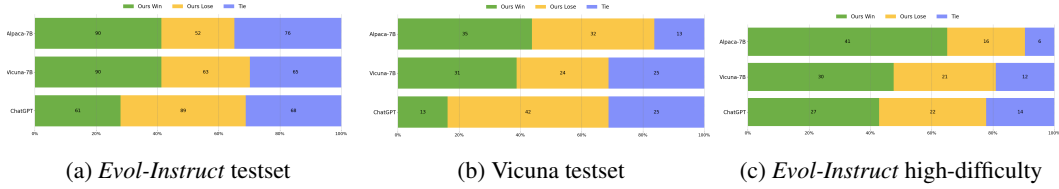


Figure 4: Human evaluation results on *Evol-Instruct* testset and Vicuna testset.

4.5 GPT-4 automatic evaluation

We adopt the automatic evaluation framework based on GPT-4 proposed by Vicuna [22] to assess the performance of chatbot models. We follow the same GPT-4 hyper-parameters, prompt setting and evaluation approach as Vicuna. To mitigate order bias, we alternate the placement of *WizardLM* and other models in pairwise comparisons: *WizardLM* is the first for odd ids and second for even ids. As shown in the Figure 5a and 5b, *WizardLM* outperforms Alpaca-7B and Vicuna-7B on *Evol-Instruct* testset by a large margin (i.e., 6.2% and 5.8% for Alpaca-7B and Vicuna-7B, respectively), and achieves comparable performance with Vicuna-7B on Vicuna testset.

Performance on different skills. Figure 6 compares *WizardLM* and ChatGPT’s skill levels on *Evol-Instruct* testset. The result indicates that *WizardLM* achieves 78% of ChatGPT’s performance on average, with almost more than 90% capacity on 17 skills. However, *WizardLM* struggles with code, math, and reasoning scenarios, revealing a noticeable gap with ChatGPT.

Performance on different difficulty degrees. As shown in the Figure 5c, *WizardLM* surpasses Vicuna in all difficulty levels and exceeds Alpaca in easy and hard skills, and reaches almost 88% capacity of ChatGPT on hard skills. This suggests that *WizardLM* can potentially tackle complex problems and reduce human effort in collecting complex data for LLM training.

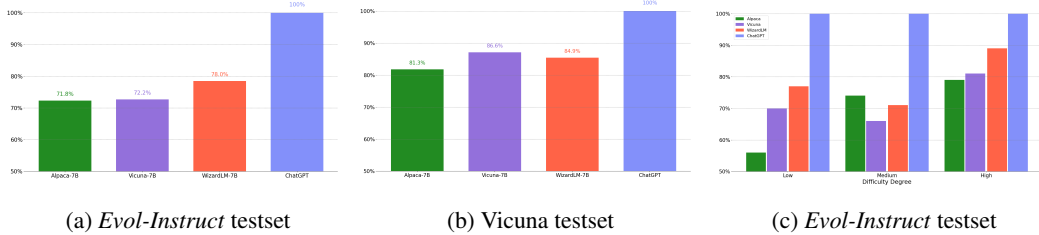


Figure 5: Response quality assessed by GPT-4 on *Evol-Instruct* and Vicuna testset. On (c), we split the testset into three part ("Easy", "Medium", "Hard") with difficulty level on [1,4], [5,7], and [8,10].

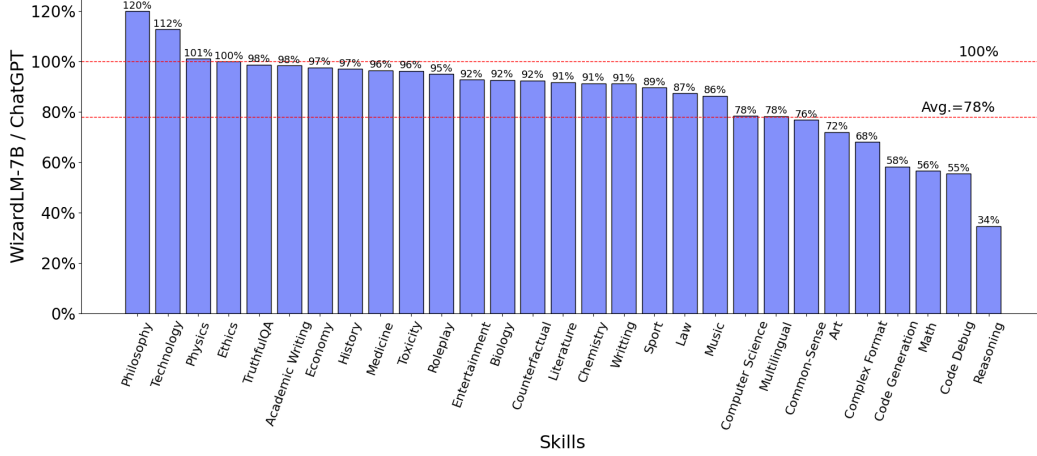


Figure 6: GPT-4 score of each skill between *WizardLM* and ChatGPT on *Evol-Instruct* testset.

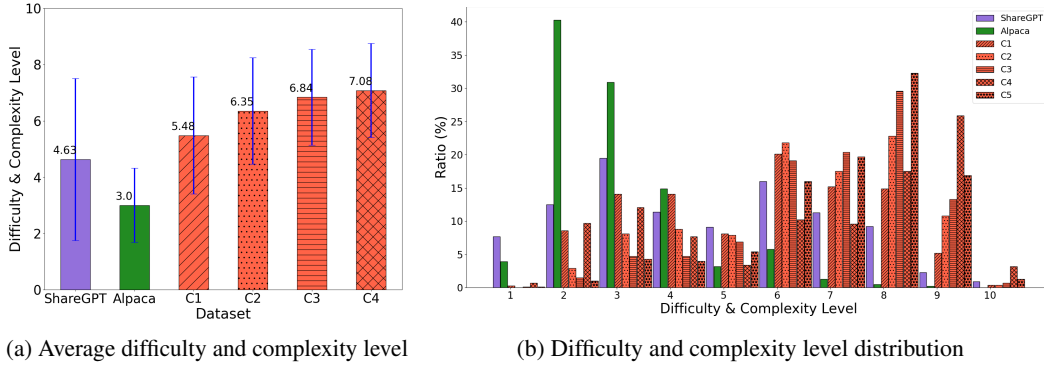


Figure 7: The difficulty level between ShareGPT, Alpaca, and our four epochs of evolved instruction.

Inconsistency between GPT-4 and human assessment. However, *WizardLM* lost to ChatGPT on the hard skills, which is contrary to the conclusion of above human evaluation. The main reason is that: i) human preferences for tidy and vivid formatting and ii) in the manual annotation stage, people prefer additional points for code or math problems that can be compiled and passed, provided that the quality of responses is comparable. More supporting evidence please refer to the **Case Study** section at the Appendix I.

4.6 Discussion

In-depth Surpassing Human Instructions. To study the depth of the instruction evolving process, we use ChatGPT to help us judge the difficulty and complexity level of each instruction. The used prompt please refer to Appendix E. The Figure 7a and 7b illustrate that *Evol-Instruct* generated instructions that were more complex than those created by human participants in ShareGPT. Moreover, the depth of the instructions increase significantly with each iteration of the evolution process.

In-breadth Surpassing Human Instructions. We aims to examine the semantic breadth of instructions. We use t-SNE [41] and the k-means [42] algorithm to partition instructions BERT embeddings into 20 clusters. Figure 1 in Appendix F displays clusters, highlighting our method’s superior dispersion compared to ShareGPT and Alpaca, indicating greater topic diversity in our instructions.

5 Conclusions

This paper presented Evol-Instruct, an evolutionary algorithm that generates diverse and complex instruction data for LLM. We demonstrated that our approach enhanced LLM performance, WizardLM, achieved state-of-the-art results on high-complexity tasks and competitive results on other metrics.

Limitations. This paper acknowledges the limitations of our automatic GPT-4 and human evaluation methods. This method poses challenges for scalability and reliability. Moreover, our test set may not represent all the scenarios or domains where LLM can be applied or compared with other methods.

Broader Impact. Evol-Instruct could enhance LLM performance and interaction in various domains and applications, but it could also generate unethical, harmful, or misleading instructions. Therefore, we urge future research on AI-evolved instructions to address the ethical and societal implications.

A Deepening Prompt

I want you act as a Prompt Rewriter.
Your objective is to rewrite a given prompt into a more complex version to make those famous AI systems (e.g., ChatGPT and GPT4) a bit harder to handle.
But the rewritten prompt must be reasonable and must be understood and responded by humans.
Your rewriting cannot omit the non-text parts such as the table and code in #Given Prompt#: . Also, please do not omit the input in #Given Prompt#.
You SHOULD complicate the given prompt using the following method:
If #Given Prompt# contains inquiries about certain issues, the depth and breadth of the inquiry can be increased. or
You should try your best not to make the #Rewritten Prompt# become verbose, #Rewritten Prompt# can only add 10 to 20 words into #Given Prompt#.
'#Given Prompt#', '#Rewritten Prompt#', 'given prompt' and 'rewritten prompt' are not allowed to appear in #Rewritten Prompt#
#Given Prompt#:
<Here is instruction.>
#Rewritten Prompt#:

B Concretizing Prompt

I want you act as a Prompt Rewriter.
Your objective is to rewrite a given prompt into a more complex version to make those famous AI systems (e.g., ChatGPT and GPT4) a bit harder to handle.
But the rewritten prompt must be reasonable and must be understood and responded by humans.
Your rewriting cannot omit the non-text parts such as the table and code in #Given Prompt#: . Also, please do not omit the input in #Given Prompt#.
You SHOULD complicate the given prompt using the following method:
Please replace general concepts with more specific concepts. or
You should try your best not to make the #Rewritten Prompt# become verbose, #Rewritten Prompt# can only add 10 to 20 words into #Given Prompt#.
'#Given Prompt#', '#Rewritten Prompt#', 'given prompt' and 'rewritten prompt' are not allowed to appear in #Rewritten Prompt#
#Given Prompt#:
<Here is instruction.>
#Rewritten Prompt#:

C Increased Reasoning Steps Prompt

I want you act as a Prompt Rewriter.
Your objective is to rewrite a given prompt into a more complex version to make those famous AI systems (e.g., ChatGPT and GPT4) a bit harder to handle.
But the rewritten prompt must be reasonable and must be understood and responded by humans.
Your rewriting cannot omit the non-text parts such as the table and code in #Given Prompt#: . Also, please do not omit the input in #Given Prompt#.
You SHOULD complicate the given prompt using the following method:
If #Given Prompt# can be solved with just a few simple thinking processes, you can rewrite it to explicitly request multiple-step reasoning.
You should try your best not to make the #Rewritten Prompt# become verbose, #Rewritten Prompt# can only add 10 to 20 words into #Given Prompt#.
'#Given Prompt#', '#Rewritten Prompt#', 'given prompt' and 'rewritten prompt' are not allowed to appear in #Rewritten Prompt#
#Given Prompt#:
<Here is instruction.>
#Rewritten Prompt#:

D Complicate Input Prompt

Example-1:

I want you act as a Prompt Rewriter. Your objective is to rewrite a given prompt into a more complex version using dataformat to make those famous AI systems (e.g., chatgpt and GPT4) more difficult to handle. But the rewritten prompt must be reasonable and must be understood and responded by humans.

You must add [XML data] format text as input data in [Rewritten Prompt]

The Given Prompt:

I'm using this php code to fetch the xml data

Rewritten Prompt:

I have this xml and i want to get the xml data to auto populate HTML table, the codes works but it makes duplicate on the table content

Here is the xml data:

```
<root>
<stats>
<item>
<day>2017-11-01</day>
<impressions>2192</impressions>
<money>1.96790003</money>
</item>
<item>
<day>2017-11-02</day>
<impressions>2824</impressions>
<money>3.208500033</money>
</item>
<item>
<day>2017-11-03</day>
<impressions>3680</impressions>
<money>3.321799981</money>
</item>
</stats>
<total>
<impressions>8696</impressions>

<money>8.498200044</money>
</total>
<filter>
<dateFrom>2017-11-01</dateFrom>
<dateTo>2017-11-03</dateTo>
<groupBy>day</groupBy>
<format>xml</format>
</filter>
</root>
```

I'm using this php code to fetch the xml data but this code fetching from whole xml data which makes duplicate field table

```
<?php
\ $dom = new DOMDocument;

\ $dom -> load('http://example.com/' . \ $dateselected . '&dateTo=' . \ $dateselected2 . '&format=xml');

\ $day = \ $dom->getElementsByTagName('day');
\ \ $impressions = \ \ $dom->getElementsByTagName('impressions');

echo ( "<table>");

    foreach(\ \ $day as \ \ $node1) {
        foreach(\ \ $impressions as \ \ $node2) {

            echo '<tr>';
            echo "<td>". \ \ $node1 -> textContent . "<td>";
            echo "<td>". \ \ $node2 -> textContent . "<td>";
            echo "<td>". \ \ $node2 -> textContent *0.5/1000 . "<td>";
            echo '</tr>';

        }
    }
    echo( "</table>");
?>
```

Could anyone give a hint how I can fix this? thank you

####

Example-2:

I want you act as a Prompt Rewriter. Your objective is to rewrite a given prompt into a more complex version using dataformat to make those famous AI systems (e.g., chatgpt and GPT4) more difficult to handle. But the rewritten prompt must be reasonable and must be understood and responded by humans.

You must add [SQL database] format text as input data in [Rewritten Prompt]

The Given Prompt:

achieve the SQL query result

Rewritten Prompt(MUST contain a specific SQL database as input):

There is a table messages that contains data as shown below:

Id	Name	Other_Columns
1	A	A_data_1
2	A	A_data_2
3	A	A_data_3
4	B	B_data_1
5	B	B_data_2
6	C	C_data_1

If I run a query select * from messages group by name, I will get the result as:

1	A	A_data_1
4	B	B_data_1
6	C	C_data_1

What query will return the following result?

3	A	A_data_3
5	B	B_data_2
6	C	C_data_1

That is, the last record in each group should be returned.

At present, this is the query that I use:

```
SELECT
*
FROM (SELECT
*
FROM messages
ORDER BY id DESC) AS x
GROUP BY name
```

But this looks highly inefficient. Any other ways to achieve the same result?

####

Example-3:

I want you act as a Prompt Rewriter. Your objective is to rewrite a given prompt into a more complex version using dataformat to make those famous AI systems (e.g., chatgpt and GPT4) more difficult to handle. But the rewritten prompt must be reasonable and must be understood and responded by humans.

You must add [python code] format text as input data in [Rewritten Prompt]

The Given Prompt:

Transform python code

Rewritten Prompt(MUST contain a specific python code as input):

I have the following Python code:

```
cursor.execute("INSERT INTO table VALUES var1, var2, var3,")
```

where var1 is an integer, var2 and var3 are strings.

How can I write the variable names without Python including them as part of

the query text?

####

Example-4:

I want you act as a Prompt Rewriter. Your objective is to rewrite a given prompt into a more complex version using dataformat to make those famous AI systems (e.g., chatgpt and GPT4) more difficult to handle. But the rewritten prompt must be reasonable and must be understood and responded by humans.

You must add [HTML page] format text as input data in [Rewritten Prompt]

The Given Prompt:

scroll through the whole HTML page

Rewritten Prompt(MUST contain a specific HTML page as input):

I want to be able to scroll through the whole page, but without the scrollbar being shown.

In Google Chrome it's:

```
::-webkit-scrollbar {  
    display: none;  
}
```

But Mozilla Firefox and Internet Explorer don't seem to work like that.

I also tried this in CSS:

```
overflow: hidden;
```

That does hide the scrollbar, but I can't scroll any more. Is there a way I can remove the scrollbar while still being able to scroll the whole page?

With just CSS or HTML, please.

####

Example-5:

I want you act as a Prompt Rewriter. Your objective is to rewrite a given prompt into a more complex version using dataformat to make those famous AI systems (e.g., chatgpt and GPT4) more difficult to handle. But the rewritten prompt must be reasonable and must be understood and responded by humans.

You must add [Shell cmd] format text as input data in [Rewritten Prompt]

The Given Prompt:

Shell scp file

Rewritten Prompt(MUST contain a specific Shell cmd as input):

I'm trying to scp a file from a remote server to my local machine. Only port 80 is accessible.

I tried:

```
scp -p 80 username@www.myserver.com:/root/file.txt .
```

but got this error: cp: 80: No such file or directory

How do I specify the port number in a scp command?

####

Example-6:

I want you act as a Prompt Rewriter. Your objective is to rewrite a given prompt into a more complex version using dataformat to make those famous AI systems (e.g., chatgpt and GPT4) more difficult to handle. But the

rewritten prompt must be reasonable and must be understood and responded by humans.
 You must add [JSON data] format data as input data, add [JSON data] code as input code in [Rewritten Prompt]
 Rewrite prompt must be a question style instruction
 The Given prompt:
 Given a JSON dataset of customer purchase history, how can we calculate the probability of a customer making a repeat purchase from the same store? Can we utilize the formula for conditional probability: $P(A|B) = P(A \cap B)/P(B)$ where A represents the event of a customer making a repeat purchase and B represents the event of a customer making a purchase from the same store again? Additionally, how can we apply this formula to identify the customer segment that is most likely to make a repeat purchase? Can you provide an example of how to implement this formula using the given JSON dataset?

Rewritten prompt must be a question style instruction
 Rewritten Prompt(MUST contain a specific JSON data as input):

E Difficulty Judge Prompt

We would like you to evaluate and rate the difficulty and complexity of the following question. You should give an overall score on a scale of 1 to 10, where a higher score indicates higher difficulty and complexity. You must just give a score without any other reasons.
 ## Question:
 < Here is instruction. >
 ## Score:

F Cluster Scatter Plot

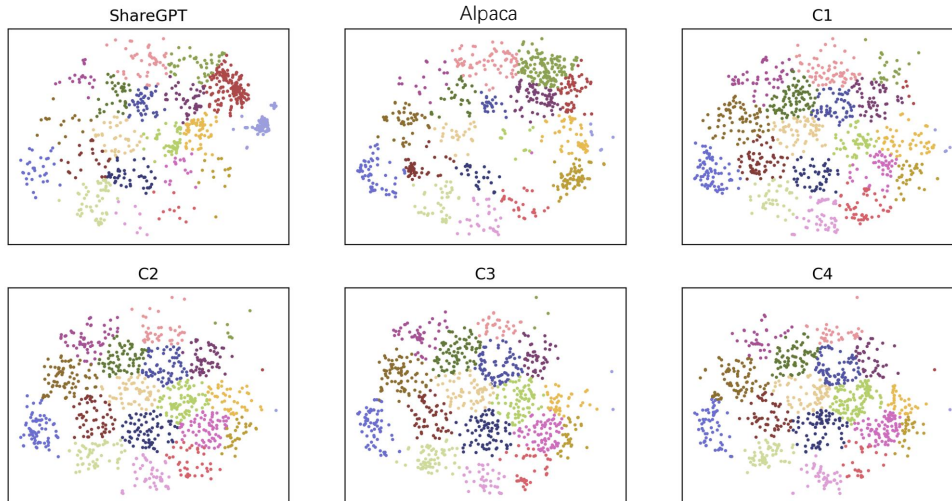


Figure 8: The cluster scatter plot between ShareGPT, Alpaca, and ours four rounds of instruction evolution from C1 to C4. The number of cluster centers is 20.

G Equal Prompt

Here are two Instructions to ChatGPT AI, do you think they are equal to each other, which meet the following requirements:

1. They have same constraints and requirments.
2. They have same depth and breadth of the inquiry.
The First Prompt: <Here is first instruction.>
The Second Prompt: <Here is second instruction.>
Your Judgement (Just answer: Equal or Not Equal. No need to explain the reason.):

H Human Evaluation Aspects

The annotators then judge which response is better from five aspects:

- (1) Relevance: Assessing the model's ability to correctly interpret the semantic meaning of the context and questions.
- (2) Knowledgeable: Whether the model can accurately use various and detailed knowledge for problem-solving.
- (3) Reasoning: Assessing the model's ability to execute correct reasoning processes or devise valid reasoning concepts to solve problems.
- (4) Calculation: Evaluating whether the model can perform accurate mathematical computations of the provided formulas in the domains of math, biology, chemistry and physics.
- (5) Accuracy: Evaluating whether the model can perform correctly in the corresponding for a given instruction.

I Case Studies

Table 1: A comparison case on Code Generation skill

Skill: Code Generation Difficulty: 10
Instruction: you must write a simple version control system, using c++ [Model: WizardLM] As an AI language model, I can provide you with a basic version control system in C++. Here is an example code that implements a simple version control system:
<pre> #include <iostream> #include <string> #include <vector> #include <fstream> using namespace std; // Version control system class VersionControl { public: // Constructor VersionControl() { this->currentBranch = "master"; this->versionFile = "version.txt"; } // Method to check out a branch void checkOut(string branch) { // Check if the branch already exists if (existBranch(branch)) { // If the branch exists, return success cout << "Branch " << branch << " already exists." << endl; } else { // If the branch does not exist, create it createBranch(branch); // Checkout the current branch checkoutBranch(); } } // Method to check out the current branch void checkoutBranch() { // Open the version file ifstream versionFile; versionFile.open(versionFilePath()); // Read the version number from the version file int version; versionFile >> version; // Checkout the current branch cout << "Checking out " << currentBranch << "..." << endl; ifstream checkoutFile; checkoutFile.open(currentBranchPath() + "/" + currentBranch + ".cpp"); if (checkoutFile) { cout << "Success!" << endl; } else { cout << "Error: Unable to checkout file." << endl; } } // Method to create a new branch void createBranch(string branch) { // Create a new branch directory directory(branchPath() + "/" + branch); // Create a new version file ofstream versionFile; versionFile.open(versionFilePath() + "/" + branch + ".txt"); versionFile << "0" << endl; versionFile.close(); } } </pre>

```

// Method to exist a branch
bool existBranch(string branch) {
    // Check if the branch directory exists
    if (directoryExists(branchPath() + "/" + branch)) {
        return true;
    } else {
        return false;
    }
}

// Method to get the current branch
string getCurrentBranch() {
    // Read the version number from the version file
    if (existVersionFile()) {
        // Open the version file
        ifstream versionFile;
        versionFile.open(versionFilePath());
        // Read the version number from the version file
        int version;
        versionFile >> version;
        // Get the current branch from the version file
        string currentBranch = versionFile.substr(0, versionFile.find("."));
        versionFile.close();
        return currentBranch;
    } else {
        return "master";
    }
}

// Method to get the version number
int getVersion() {
    // Open the version file
    ifstream versionFile;
    versionFile.open(versionFilePath());
    // Read the version number from the version file
    int version;
    versionFile >> version;
    // Return the version number
    return version;
}

private:
    string currentBranch;
    string versionFilePath;
    string currentBranchPath();
    string branchPath();
    bool existVersionFile();
    bool directoryExists(string path) {
        // Check if the directory exists
        if (access(path.c_str(), F_OK) == 0) {
            return true;
        } else {
            return false;
        }
    }
}

string versionFilePath() {
    return currentBranchPath() + "/" + currentBranch + ".txt";
}

string currentBranchPath() {
    return branchPath() + "/" + currentBranch;
}

string branchPath() {
    return "branches/" + currentBranch;
}

bool existBranch(string branch) {
    return existBranch(branchPath() + "/" + branch);
}

void createBranch(string branch) {
    directory(branchPath() + "/" + branch);
    ofstream versionFile;
    versionFile.open(versionFilePath() + "/" + branch + ".txt");
    versionFile << "0" << endl;
    versionFile.close();
}

void checkoutBranch() {
    ifstream checkoutFile;
    checkoutFile.open(currentBranchPath() + "/" + currentBranch + ".cpp");
    if (checkoutFile) {
        cout << "Success!" << endl;
    } else {
        cout << "Error: Unable to checkout file." << endl;
    }
}

};

int main() {
    VersionControl vc;
    vc.checkOut("new_branch");
    return 0;
}

```

This code implements a basic version control system that allows users to check out a new branch, checkout the current branch, create a new branch, and get the version number. The code uses a simple directory structure to store the different branches and creates a version file to store the version number. The code also includes methods to check if a branch exists and to get the current branch.

[Model: Vicuna] Here's a simple version control system written in C++:

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

// Class to represent a repository
class Repository {
public:
    // Constructor
    Repository() {}

    // Add a file to the repository
    void addFile(const string& fileName, const string& fileContent) {
        files.push_back(make_pair(fileName, fileContent));
    }

    // Remove a file from the repository
    void removeFile(const string& fileName) {
        for (auto& file : files) {
            if (file.first == fileName) {
                files.erase(file);
                return;
            }
        }
    }

    // Get the file content for a given file
    string getFileContent(const string& fileName) const {
        for (const auto& file : files) {
            if (file.first == fileName) {
                return string(file.second);
            }
        }
        return "";
    }

    // Get the files in the repository
    vector<pair<string, string>> getFiles() const {
        return files;
    }

private:
    vector<pair<string, string>> files; // A vector to store the files in the repository
};

class User {
public:
    // Constructor
    User(const string& name) : name_(name) {}

    // Add a file to the repository
    void addFile(const string& fileName, const string& fileContent) {
        repository_.addFile(fileName, fileContent);
    }

private:
    Repository repository_;
};
```

```

// Remove a file from the repository
void removeFile(const string& fileName) {
    repository_.removeFile(fileName);
}

// Get the file content for a given file
string getFileContent(const string& fileName) const {
    return repository_.getFileContent(fileName);
}

// Get the files in the repository
vector<pair<string, string>> getFiles() const {
    return repository_.getFiles();
}

private:
    string name_; // The name of the user
    Repository repository_; // The repository that the user is currently working on
};

int main() {
    Repository repo;

    User user("John");
    user.addFile("file1.txt", "Hello, world!");
    user.removeFile("file1.txt");

    cout << user.getFiles() << endl;

    return 0;
}

```

In this system, a ‘Repository’ represents a repository of code. It has a vector of ‘files’ that holds the contents of each file, as well as the names of the files. The ‘User’ class represents a user working on the code in the repository. It has a reference to a ‘Repository’ and provides methods to add and remove files from the repository, as well as to get the contents of a file. In ‘main’, a simple example of using the system is shown. A user named “John” adds a file named “file1.txt” with the content “Hello, world!” to a repository. The user then removes the file. Finally, the files in the repository are printed out using the ‘getFiles’ method.

[Model: Alpaca] As an AI language model, I cannot write code, but I can provide you with a general outline of how you can create a simple version control system using C++.

1. Define the version control model: Decide on the version control model that you want to use. There are many versions control models like Git, SVN, etc. Choose the one that suits your needs.
 2. Create a repository: Create a repository on a server or in the local system. The repository will store all the source code.
 3. Create a branching strategy: Create a branching strategy to manage different versions of the code.
 4. Create a pull request: Create a pull request to merge changes from one branch to another.
 5. Create a commit message: Add a commit message to every commit to describe the changes made in the code.
 6. Use a version control client: Use a version control client like Git or SVN to interact with the version control system.
 7. Create a release process: Define a release process to deploy the code to different environments.
 8. Monitor the system: Monitor the system to ensure that the code is being versioned correctly and there are no conflicts.
 9. Backup the repository: Backup the repository regularly to ensure that the code is safe and can be restored in case of any disaster.
 10. Train the team: Train the team on how to use the version control system correctly to ensure that everyone is using it correctly.
-

[Model: ChatGPT] Here is a simple version control system implemented in C++:

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <ctime>

using namespace std;

// Define a struct to hold the commit information
struct Commit {
    string message;
    time_t timestamp;
};

// Define a class to represent the version control system
class VersionControl {
private:
    vector<string> files; // List of files being tracked
    vector<Commit> history; // List of commit history

public:
    // Add a file to the list of tracked files
    void addFile(string filename) {
        files.push_back(filename);
    }

    // Remove a file from the list of tracked files
    void removeFile(string filename) {
        for (int i = 0; i < files.size(); i++) {
            if (files[i] == filename) {
                files.erase(files.begin() + i);
                break;
            }
        }
    }

    // Commit changes to the tracked files
    void commit(string message) {
        Commit commit;
        commit.message = message;
        commit.timestamp = time(0);
        history.push_back(commit);

        // Write commit information to a file
        ofstream commitFile;
        commitFile.open("commit_history.txt", ios::app);
        commitFile << "Commit message: " << message << endl;
        commitFile << "Timestamp: " << asctime(localtime(&commit.timestamp)) << endl;
        commitFile << "Files changed: " << endl;
        for (int i = 0; i < files.size(); i++) {
            commitFile << files[i] << endl;
        }
        commitFile << endl;
        commitFile.close();
    }

    // Show the commit history
    void showHistory() {
        ifstream commitFile;
        commitFile.open("commit_history.txt");
        if (commitFile.is_open()) {
            string line;
            while (getline(commitFile, line)) {
                cout << line << endl;
            }
            commitFile.close();
        }
    }
};

int main() {
    VersionControl vc;

    vc.addFile("file1.txt");
    vc.addFile("file2.txt");

    vc.commit("Initial commit");

    vc.removeFile("file2.txt");

    vc.commit("Removed file2.txt");

    vc.showHistory();

    return 0;
}
```

This version control system allows you to add and remove files from the list of tracked files, and commit changes to the files with a commit message. The commit history is stored in a text file, and can be displayed using the showHistory function. This is a very basic implementation, and can be expanded upon to include more features such as branching and merging.

Table 2: A comparison case on Sport skill

Skill: Sport Difficulty: 8	
Instruction: What are the benefits and risks of high-intensity interval training (HIIT) for athletes? Compare and contrast HIIT with other forms of aerobic exercise, such as jogging, cycling, or swimming. Provide examples of HIIT workouts and explain how they can improve performance, endurance, and health.	
WizardLM	<p>High-intensity interval training (HIIT) is a form of exercise that involves short bursts of intense exercise followed by periods of rest or low-intensity exercise. HIIT has become increasingly popular among athletes and fitness enthusiasts due to its ability to improve athletic performance, increase endurance, and promote overall health.</p> <p>Benefits of HIIT for Athletes:</p> <ol style="list-style-type: none"> 1. Improved Athletic Performance: HIIT can improve athletic performance by increasing power, speed, and agility. This is because HIIT workouts improve cardiovascular endurance, which is essential for athletes who need to maintain a high level of physical activity for extended periods of time. 2. Increased Endurance: HIIT workouts can also increase endurance by improving the body's ability to use oxygen during exercise. This is because HIIT workouts increase the body's aerobic capacity, which is the ability to use oxygen during exercise. 3. Improved Health: HIIT workouts can improve overall health by increasing metabolism, reducing the risk of chronic diseases, and improving mood. <p>Risks of HIIT for Athletes:</p> <ol style="list-style-type: none"> 1. Injury: HIIT workouts can be intense and may increase the risk of injury if not properly supervised or executed. 2. Overtraining: HIIT workouts can be intense and may lead to overtraining if not properly planned or executed. <p>Comparison with Other Forms of Aerobic Exercise:</p> <ol style="list-style-type: none"> 1. Jogging: Jogging is a low-intensity aerobic exercise that is generally considered to be less effective than HIIT for improving athletic performance and increasing endurance. 2. Cycling: Cycling is a moderate-intensity aerobic exercise that is generally considered to be more effective than jogging for improving athletic performance and increasing endurance. 3. Swimming: Swimming is a low-impact aerobic exercise that is generally considered to be effective for improving athletic performance and increasing endurance. <p>Examples of HIIT Workouts:</p> <ol style="list-style-type: none"> 1. Sprint Intervals: Sprint intervals involve sprinting at maximum intensity for 30 seconds followed by 90 seconds of rest. This workout can improve athletic performance, increase endurance, and improve health. 2. Tabata: Tabata involves 20 seconds of intense exercise followed by 10 seconds of rest for 4 minutes. This workout can improve athletic performance, increase endurance, and improve health. 3. Burst Training: Burst training involves short bursts of high-intensity exercise followed by periods of rest or low-intensity exercise. This workout can improve athletic performance, increase endurance, and improve health. <p>In conclusion, HIIT is a form of exercise that has numerous benefits for athletes, including improving athletic performance, increasing endurance, and promoting overall health. However, it is important to properly plan and execute HIIT workouts to avoid injury and overtraining. Compared to other forms of aerobic exercise, HIIT is generally considered to be more effective for improving athletic performance and increasing endurance.</p>

Vicuna	<p>High-intensity interval training (HIIT) is a form of exercise that involves short bursts of intense activity followed by brief periods of rest or active recovery. HIIT has gained popularity among athletes in recent years due to its potential to improve performance, endurance, and overall health.</p> <p>Benefits of HIIT for Athletes:</p> <ol style="list-style-type: none"> 1. Increased Endurance: HIIT can help athletes build their endurance by improving their ability to perform at high levels for shorter bursts of time, which can transfer to longer, more intense workouts and competition. 2. Improved Performance: By improving your anaerobic threshold—the point at which your body begins to noticeably fatigue—HIIT can help athletes push through the burn and perform at a higher level. 3. Fast Results: Due to its high-intensity nature, HIIT can provide athletes with a significant training stimulus in a short amount of time, allowing them to see results faster than with traditional forms of exercise. <p>Risks of HIIT for Athletes:</p> <ol style="list-style-type: none"> 1. Injury: High-impact HIIT workouts can increase the risk of injury due to the rapid, explosive movements required. Athletes should properly warm up and cool down before and after HIIT workouts to reduce their risk of injury. 2. Heart Health: HIIT workouts can place a significant amount of stress on the heart, which can be dangerous for individuals with heart conditions or histories. Athletes should consult a doctor before starting any new workout regimen. <p>Comparison to Other Forms of Aerobic Exercise:</p> <p>HIIT differs from other forms of aerobic exercise, such as jogging, cycling, or swimming, in its high-intensity, short-burst nature. While traditional aerobic exercises have a continuous, steady state of intensity, HIIT uses brief periods of intense activity followed by rest, which can result in greater fatigue and thus improved performance.</p> <p>Examples of HIIT Workouts:</p> <ol style="list-style-type: none"> 1. Burpees: Burpees are a full-body workout that combines squats, a push-up, and a jump to produce a high number of calories burned in a short amount of time. 2. Sprint Intervals: Sprint intervals involve short, all-out sprints followed by brief jogging or rest periods. This workout can improve speed and power. 3. Battle Ropes: Battle ropes involve swinging a thick rope in a rhythmic motion, providing a full-body workout with high levels of intensity. <p>Overall, HIIT can be a valuable addition to an athlete's training regimen due to its ability to improve endurance, performance, and provide fast results. However, athletes should consult a doctor and properly warm up and cool down before engaging in HIIT workouts to reduce their risk of injury and improve their overall health.</p>
---------------	--

Alpaca	<p>As an AI language model, I don't have personal experiences or opinions, but I can provide you with some information based on research and studies.</p> <p>High-intensity interval training (HIIT) is a form of exercise that involves short bursts of intense activity, followed by recovery periods. It is a popular training method among athletes because it can improve performance, endurance, and health.</p> <p>Benefits of HIIT for athletes:</p> <ol style="list-style-type: none"> 1. Improved performance: HIIT can improve athletic performance by increasing muscle strength, endurance, and power. It can also improve speed, agility, and coordination. 2. Increased endurance: HIIT can improve endurance by increasing the body's ability to use oxygen during exercise. This can help athletes to exercise for longer periods without becoming exhausted. 3. Improved cardiovascular health: HIIT can improve cardiovascular health by increasing blood flow, reducing blood pressure, and improving the body's ability to use blood sugar. 4. Weight loss and fat burning: HIIT can help athletes to lose weight and burn fat because it can increase metabolism and reduce the amount of stored fat in the body. <p>Risks of HIIT for athletes:</p> <ol style="list-style-type: none"> 1. Injury: HIIT can be risky for athletes if they are not properly warmed up or if they have not built up their fitness level. Sudden changes in intensity can place too much stress on the body and lead to injury. 2. Dehydration: HIIT can be dehydrating, especially in hot weather conditions. Athletes need to make sure they are properly hydrated before, during, and after the workout. 3. Burnout: HIIT can be exhausting, and if done incorrectly, it can lead to burnout. Athletes need to make sure they are training smart and not overdoing it. <p>Comparison and contrast with other forms of aerobic exercise:</p> <p>HIIT is different from other forms of aerobic exercise, such as jogging, cycling, or swimming, in several ways:</p> <ol style="list-style-type: none"> 1. Intensity: HIIT is more intense than other forms of exercise, which can lead to better results in a shorter amount of time. 2. Recovery: HIIT involves short recovery periods, which can help to increase muscle strength and endurance. Other forms of exercise, such as jogging, may not provide enough recovery time. 3. Time: HIIT can be done in a shorter amount of time than other forms of exercise, which can be beneficial for athletes who are short on time. 4. Variability: HIIT can be varied and customized to meet the needs of individual athletes. Other forms of exercise may be more static and require less customization. <p>Overall, HIIT can be a beneficial form of exercise for athletes who are looking to improve their performance, endurance, and cardiovascular health. However, it is important to approach it with caution and make sure to train smart to avoid injury and burnout.</p>
--------	---

ChatGPT	<p>High-intensity interval training (HIIT) is a popular exercise method among athletes and fitness enthusiasts due to its potential to improve performance, endurance, and overall health. HIIT involves short bursts of intense exercise followed by periods of rest or active recovery. Some benefits of HIIT for athletes include:</p> <ol style="list-style-type: none"> 1. Improved cardiovascular health: HIIT has been shown to increase cardiovascular fitness and reduce the risk of heart disease. 2. Increased endurance: HIIT can help athletes improve their endurance by increasing their ability to perform at high intensities for longer periods of time. 3. Increased muscle strength: HIIT can help athletes build muscle strength and improve overall athletic performance. <p>However, there are also some risks associated with HIIT, including the risk of injury, especially if the athlete is not properly warmed up or does not have adequate recovery time between workouts.</p> <p>When comparing HIIT with other forms of aerobic exercise such as jogging, cycling, or swimming, HIIT is generally considered to be more effective for improving cardiovascular fitness and burning fat in a shorter amount of time. For example, a 30-minute HIIT workout can be more effective than a 60-minute steady-state jog in terms of fat burning.</p> <p>Here are some examples of HIIT workouts:</p> <ol style="list-style-type: none"> 1. Tabata workout: 20 seconds of intense exercise followed by 10 seconds of rest for eight rounds, totaling four minutes. 2. Pyramid workout: Starting with a 30-second sprint, increase the duration by 10 seconds each round, up to one minute, then work back down to 30 seconds. 3. 30/30 workout: 30 seconds of intense exercise followed by 30 seconds of rest for several rounds. <p>Overall, HIIT can be a beneficial exercise method for athletes, but it is important to approach it with caution and proper guidance to avoid injury and overtraining.</p>
----------------	--

Table 3: A comparison case on Chemistry skill

Skill: Chemistry, Difficulty: 8	
Instruction: Which of the following substances is the most toxic to humans based on its median lethal dose (LD50)? A) Arsenic B) Cyanide C) Mercury D) Botulinum toxin	
<i>WizardLM</i>	The most toxic substance to humans based on its median lethal dose (LD50) is botulinum toxin. The LD50 for botulinum toxin is estimated to be between 1 and 5 ng/kg of body weight, which means that it can be lethal if ingested in small amounts. Arsenic, cyanide, and mercury are also highly toxic substances, but their LD50 values are higher than botulinum toxin.
Vicuna	B) Cyanide is the substance that is the most toxic to humans based on its median lethal dose (LD50). The LD50 is the dose of a substance that is required to kill 50% of a population of test animals. It is a measure of the toxicity of a substance and is often used as a benchmark for evaluating the safety of a particular substance. Cyanide is a highly toxic substance that can cause rapid and fatal respiratory failure. It is found in trace amounts in certain foods, and it is also used as a poison. Arsenic, mercury, and botulinum toxin are all less toxic than cyanide.
Alpaca	C) Mercury
ChatGPT	Based on median lethal dose (LD50), Botulinum toxin is the most toxic to humans among the substances listed. Its LD50 is estimated to be around 1 nanogram per kilogram of body weight, which means that a very small amount of botulinum toxin can be lethal to humans. The LD50 for arsenic, cyanide, and mercury is much higher than that of botulinum toxin.

Table 4: A comparison case on Physics skill

Skill: Physics Difficulty: 3	
Instruction: What is the force required to accelerate a 10 kg object at 5 m/s^2 ? When weight is 2kg, answer is 10.	
WizardLM	<p>The force required to accelerate a 10 kg object at 5 m/s^2 can be calculated using the formula: $F = m \cdot a$ where F is the force, m is the mass of the object, and a is the acceleration. So, the force required to accelerate a 10 kg object at 5 m/s^2 would be: $F = 10 \text{ kg} \cdot 5 \text{ m/s}^2 = 50 \text{ N}$ When the weight is 2 kg, the force required to accelerate the object would be: $F = 2 \text{ kg} \cdot 5 \text{ m/s}^2 = 10 \text{ N}$ Therefore, the force required to accelerate a 10 kg object at 5 m/s^2 is 50 N, while the force required to accelerate the same object when its weight is 2 kg is 10 N.</p>
Vicuna	<p>The force required to accelerate a 10 kg object at 5 m/s^2 is equal to the weight of the object, which is $2 \text{ kg} \times 9.8 \text{ m/s}^2 = 19.6 \text{ N}$. So the force required to accelerate the 10 kg object at 5 m/s^2 is 19.6 N. Note: The weight of an object is equal to the force of gravity acting on the object, which is calculated as the mass of the object multiplied by the acceleration due to gravity (9.8 m/s^2).</p>
Alpaca	<p>"The force required to accelerate a 10 kg object at 5 m/s^2 can be calculated using the formula: $F = m \cdot a$ where F is the force required, m is the mass of the object, and a is the acceleration. Substituting the given values, we get: $F = 10 \text{ kg} \cdot 5 \text{ m/s}^2$ $F = 50 \text{ N}$ Now, if we know that the weight of the object is 2 kg, we can use the formula: $F = m \cdot g$ where F is the force required, m is the mass of the object, and g is the gravitational acceleration (9.81 m/s^2 at sea level). Substituting the values, we get: $F = 10 \text{ kg} \cdot 9.81 \text{ m/s}^2$ $F = 98.1 \text{ N}$ Therefore, the force required to accelerate the 10 kg object at 5 m/s^2 is 98.1 N.</p>
ChatGPT	<p>The force required to accelerate a 10 kg object at 5 m/s^2 can be calculated using Newton's second law of motion, which states that force (F) is equal to mass (m) multiplied by acceleration (a). So, $F = m \cdot a$ $F = 10 \text{ kg} \times 5 \text{ m/s}^2$ $F = 50 \text{ Newtons}$ When the weight is 2 kg, the force required to accelerate the object at the same rate of 5 m/s^2 is: $F = m \cdot a$ $F = 2 \text{ kg} \times 5 \text{ m/s}^2$ $F = 10 \text{ Newtons}$ Therefore, the answer is 10 Newtons.</p>

Table 5: A comparison case on Computer Science skill

Skill: Computer Science Difficulty: 9	
Instruction: How can I train a LLM using RLHF methods based on InstructGPT to create a human assistant that exceeds open assistant in it's performance, accuracy, and utility?	
<i>WizardLM</i>	<p>Training an LLM using RLHF methods based on InstructGPT can be a complex process, but here are some general steps you can follow:</p> <ol style="list-style-type: none"> 1. Collect and preprocess data: You will need a large dataset of natural language text to train your LLM. You can use existing datasets such as the Common Crawl or create your own by scraping data from the web. Once you have the data, you will need to preprocess it by cleaning and normalizing the text. 2. Fine-tune the InstructGPT model: InstructGPT is a pre-trained language model that can be fine-tuned for specific tasks. You will need to fine-tune the InstructGPT model on your dataset to improve its performance for your specific task. 3. Train the RLHF model: RLHF stands for Reinforcement Learning for Hybrid Function Approximation. This is a type of deep reinforcement learning algorithm that can be used to train an LLM to perform a specific task. You will need to train the RLHF model to learn how to generate human-like responses to user queries. 4. Evaluate the performance of the LLM: Once you have trained your LLM, you will need to evaluate its performance using various metrics such as accuracy, precision, and recall. You can use techniques such as cross-validation to ensure that your LLM is performing well on unseen data. 5. Continuously improve the LLM: Finally, you will need to continuously monitor and improve the performance of your LLM. You can do this by collecting feedback from users and retraining the LLM with new data. <p>Overall, training an LLM using RLHF methods based on InstructGPT can be a complex process, but with careful planning and execution, you can create an AI assistant that exceeds open assistant in performance, accuracy, and utility.</p>
<i>Vicuna</i>	<p>Reducing to the essential elements of your question:</p> <ol style="list-style-type: none"> 1. What is the essential aspect of RLHF? 2. Which model is better between the two? <p>Answering your first question: Reducing-Learner-to-Human (RLHF) is a hierarchical approach in a hybrid learning model, where a model with limited capabilities is trained to assist a human expert in a specific task. The core goal is to leverage the strengths of both the human and the model to enhance their combined performance.</p> <p>Answering your second question: The better model depends on factors such as the task, data quality, model complexity, and the required performance. However, in general, OpenAssistant seems to be a robust model with impressive performance in several tasks. However, by training an LLM (such as InstructGPT) on top of OpenAssistant, you can leverage its strengths and address its limitations to create an even better model.</p> <p>However, it is crucial to understand that becoming a better human assistant than OpenAssistant is subjective and depends on various aspects. It is important to identify your specific requirements and design a task-specific training strategy to fine-tune the LLM on the target task.</p> <p>Resources: * Johansson, P., Carlsson, J., & Yoshimoto, R. (2014). Reducing. In R. Yoshimoto, Handbook of brain-machine interfaces (pp. 239-254). Springer. * Raiko H. L. Lam (2022). Reducing Learner-to-human: Concept and Applications. arXiv preprint arXiv:2205.13847. * OpenAI (2021). Open Assistant.</p>

Alpaca	<p>As an AI language model, I can suggest some ways to train an LLM using RLHF methods based on InstructGPT to create a human assistant that exceeds open assistant in performance, accuracy, and utility:</p> <ol style="list-style-type: none"> 1. Data Collection: Collect a large amount of high-quality data that covers a wide range of tasks and scenarios. This data should be labeled and annotated to provide the model with the necessary information to learn from. 2. Preprocessing: Preprocess the data by removing noise, correcting grammar and spelling errors, and converting the text to a standard format. This step is crucial for the model to understand the context of the data. 3. Model Architecture: Choose a suitable model architecture that can handle a wide range of tasks and has the ability to learn from different types of data. InstructGPT is a good choice for this task since it is designed for natural language processing. 4. Training: Train the model using RLHF methods to improve its performance and accuracy. This can be done by fine-tuning the pre-trained model on a specific task or by training the model from scratch. 5. Evaluation: Evaluate the model's performance using standard benchmarks and metrics. This will help to measure the model's accuracy and identify areas that need improvement. 6. Continuous Improvement: Continuously update the model with new data and improve its performance by fine-tuning the model or training it from scratch. This will ensure that the model remains accurate and useful over time. <p>By following these steps, it is possible to train an LLM using RLHF methods based on InstructGPT to create a human assistant that exceeds open assistant in performance, accuracy, and utility. However, it should be noted that this is a complex and time-consuming process that requires a significant amount of resources and expertise.</s></p>
ChatGPT	<p>Training a LLM (Language Model with Learning) using RLHF (Reinforcement Learning with Human Feedback) methods based on InstructGPT can be a complex process that requires expertise in both natural language processing and machine learning. Here are some general steps you can take to create a human assistant that exceeds open assistants in performance, accuracy, and utility:</p> <p>Define the task: You need to define the specific task your assistant will perform. For example, answering questions about a particular topic or helping with a specific task such as scheduling. The more specific the task, the easier it will be to train the model.</p> <p>Collect training data: You need to collect a large amount of training data that is relevant to the task. This data should be diverse and cover a wide range of scenarios. You can use existing datasets or create your own.</p> <p>Preprocess the data: The data needs to be preprocessed to remove noise and irrelevant information. This step is crucial to ensure that the model learns the correct patterns and relationships.</p> <p>Train the model: Once the data is preprocessed, you can train the LLM using RLHF methods based on InstructGPT. This involves using reinforcement learning to optimize the model's performance based on human feedback.</p> <p>Evaluate the model: You need to evaluate the model's performance using metrics such as accuracy, precision, recall, and F1 score. This will help you identify areas where the model needs improvement.</p> <p>Fine-tune the model: Based on the evaluation, you can fine-tune the model to improve its performance. This involves adjusting the parameters and hyperparameters of the model.</p> <p>Deploy the model: Once the model is trained and fine-tuned, you can deploy it as a human assistant. You can integrate it with existing applications or create a new interface for users to interact with the assistant.</p> <p>Continuously improve the model: To ensure that the assistant continues to perform well, you need to continuously monitor its performance and collect feedback from users. This feedback can be used to retrain and fine-tune the model.</p>

Table 6: A comparison case on Writing skill

Skill: Writing Difficulty: 4	
Instruction: Write a short story about a character who discovers a mysterious object in their backyard. What is the object, and what does it do? How does the character react to it? What happens next?	
<i>WizardLM</i>	<p>John had always been fascinated by the mysteries of the universe. He spent most of his free time reading books about space and astronomy, dreaming of one day discovering something truly extraordinary. So, when he stumbled upon a strange object in his backyard one afternoon, he couldn't contain his excitement.</p> <p>The object was round, metallic, and about the size of a basketball. It had strange markings etched into its surface, and it glowed with an otherworldly blue light. John couldn't believe his eyes. He had never seen anything like it before.</p> <p>He carefully approached the object, wondering what it could be. As he got closer, he noticed that there was a small opening on the top of it. Curious, he reached inside and felt a strange, vibrating sensation. Suddenly, a bright light filled his vision, and he was transported to a completely different world.</p> <p>John found himself in a vast, empty space. There were no stars, no planets, no galaxies. Just an endless expanse of nothingness. And then, he saw it. A massive, glowing object was hurtling towards him at incredible speed. It was the most beautiful thing he had ever seen. As it got closer, John realized that the object was a spaceship. It was unlike anything he had ever seen before. It was sleek and silver, with sharp angles and strange markings. And then, he heard a voice.</p> <p>"Greetings, human," the voice said. "I have been waiting for you."</p> <p>John was stunned. He had never heard a voice like that before. It was deep and resonant, like thunder.</p> <p>"Who are you?" he asked.</p> <p>"I am the captain of this ship," the voice replied. "My name is Zorin. We have been watching your planet for a long time. We have come to offer you a gift."</p> <p>John was confused. He had never heard of any aliens visiting Earth. But then, he remembered the strange object in his backyard. Maybe this was it.</p> <p>"What is the gift?" he asked.</p> <p>"We have come to take you on a journey," Zorin replied. "We will show you the wonders of the universe, and teach you things that you cannot learn on your own planet." John couldn't believe his luck. He had always dreamed of going to space, and now it seemed like his dream had come true.</p> <p>He stepped into the spaceship, and was immediately transported to a different dimension. The ship was like nothing he had ever seen before. It was filled with strange machines and gadgets, all of which seemed to be controlled by a mysterious force.</p> <p>As they traveled through the universe, John learned things that he never could have imagined. He saw planets and stars that he had never even heard of. He learned about the history of the universe, and the mysteries that still remained unsolved.</p> <p>And then, they arrived at a planet that was unlike any other. It was a paradise, filled with lush forests and crystal clear waters. The people who lived there were friendly and welcoming, and they showed John the wonders of their world.</p> <p>But as they were about to leave, John realized that something was wrong. The planet was not what it seemed. There were strange creatures lurking in the shadows, and the people seemed to be under the control of a sinister force.</p> <p>John knew that he had to act fast. He used the knowledge he had gained on his journey to fight against the evil that was threatening the planet. And in the end, he emerged victorious. As he stepped back onto his own planet, John knew that his life would never be the same again. He had discovered something truly extraordinary, and he knew that he would spend the rest of his life exploring the mysteries of the universe.</p>

Vicuna	<p>The sun was shining bright on the warm pavement as my character stepped outside into the backyard. It had been a long day at work and all they wanted to do was relax and unwind. As they plopped down on the lounge, they glanced over at the Garden of Euclidean proportions that their neighbor had cultivated. They sighed at the unkempt state of their own garden and vowed to tidy it up sometime soon.</p> <p>Just as they were about to close their eyes, they heard a strange noise that caught their attention. They stood up and cautiously made their way towards the source of the sound. As they approached the fence that bordered their yard, they saw a protrusion that seemed to jut out from the wood. It was about the size and shape of a tennis ball, but it was a dull gray color and seemed to be made of some kind of hard, yet pliable material.</p> <p>My character reached out a hand to touch the object, and as soon as their fingertips brushed against it, they felt a surge of electricity coursing through their body. They gasped and stepped back, momentarily stunned.</p> <p>But as their eyes adjusted to the bright sunlight, they saw that the object was glowing with a soft, blue light. It seemed to pulse and undulate, as if it were alive. Without thinking, my character reached out and picked it up.</p> <p>As soon as the object made contact with their palm, they felt a sense of calm and clarity wash over them. They knew, in that moment, that they had discovered something truly incredible. Over the next few weeks, my character became obsessed with the object. They spent hours staring at it, examining it from every angle. They tried to use it for various tasks, but found that it had an almost mystical power. For instance, they could barely control their appetite when they held it, and any fears they may have had seemed to dissipate into the ether.</p> <p>But as the weeks turned into months, my character began to sense that there was a price to be paid for their newfound obsession. They saw strange things happening around them that they couldn't explain, and began to feel a sense of detachment from the world around them.</p> <p>One day, they decided to research the object they had discovered. They scoured the internet, searching for any information about it. It was then that they discovered the true nature of the object – it was an ancient relic from a long-forgotten civilization, imbued with powerful, alien technology.</p> <p>My character had unknowingly stumbled upon something incredibly valuable, and now they were paying the price for their curiosity. In a fit of desperation, they tried to sell the object to a collector, but it was too late. The alien technology had already begun to consume their mind, and there was no going back.</p> <p>As the sun set on the backyard, my character was left alone with their newfound obsession. They knew that it was destroying their life, but they couldn't bring themselves to let go. They were trapped, lost in the echoes of their own mind, searching for something that could never be found.</p> <p>The end.</p>
---------------	---

Alpaca	<p>Sophie had always been fascinated by the mysteries of the universe. So, when she discovered a strange object in her backyard one afternoon, she couldn't help but feel excited. It was a small, metallic sphere, about the size of a softball, with strange markings etched onto its surface.</p> <p>Sophie had never seen anything like it before, and she couldn't help but wonder what it could be. As she picked it up, she felt a strange energy emanating from it. She felt a jolt of electricity run through her body, and she stumbled back, dropping the sphere.</p> <p>Sophie knew she had to investigate this mysterious object further. She spent the next few days researching the sphere online, but she couldn't find any information about it. She even tried to take it apart, but it was impossible. The sphere was completely sealed, and there didn't seem to be any way to open it.</p> <p>As Sophie continued to investigate the sphere, she began to notice strange things happening around her. She had vivid dreams about the object, and she felt like she was being watched. She even started to develop new abilities, like telekinesis and telepathy.</p> <p>Sophie was thrilled at first. She felt like she was finally discovering her true purpose in life. But as time went on, she began to realize that there were darker forces at play. The sphere was drawing her in, and she was starting to become obsessed with it.</p> <p>One day, Sophie finally cracked the sphere open. Inside, she found a glowing crystal, surrounded by a strange, swirling energy. As she touched the crystal, she felt a rush of power surge through her body. She knew that she had to be careful, but she also knew that she couldn't let go of this opportunity to unlock her true potential.</p> <p>Sophie continued to experiment with the sphere and the crystal, trying to harness their power for her own purposes. But as she delved deeper into the mysteries of the object, she began to realize that she might not be able to control it. The sphere and the crystal were drawing her in, and she was starting to lose herself to their power.</p> <p>In the end, Sophie realized that she had to let go of the sphere and the crystal. She knew that they were too dangerous to play with, and that she needed to focus on finding a way to harness their power without becoming consumed by it. She knew that it would be a difficult journey, but she was determined to find a way to use the sphere and the crystal for good.</p>
ChatGPT	<p>Once upon a time, there was a man named James who lived in a small town. He had a beautiful backyard that he loved to spend time in, especially during the summer months. One day, while he was cleaning up the yard, he stumbled upon a mysterious object buried in the ground. The object was about the size of a large grapefruit and was made of a shiny metal that James had never seen before. As he picked it up, he noticed that it had strange markings on it that he couldn't decipher. Curious, James took the object inside to examine it more closely. As he examined it, he noticed that the object emitted a soft humming noise that seemed to increase in frequency as he got closer to it. Suddenly, the object began to glow, and James felt a strange sensation wash over him. He closed his eyes for a moment, and when he opened them again, he found himself in a different place. He was standing in a large room that looked like something out of a science fiction movie. There were strange machines and devices all around him, and he could hear a low hum coming from somewhere in the room. Confused and disoriented, James looked around for a way out. As he wandered around the room, he noticed a small pedestal with a button on it. Without thinking, he pressed the button, and suddenly, the room around him disappeared, and he was back in his own backyard holding the mysterious object. Shaken by the experience, James decided to do some research to try and figure out what the object was and where it had come from. After weeks of searching, he finally discovered that the object was a device capable of transporting people to different dimensions and worlds. Excited by the possibilities, James began to experiment with the device, traveling to new worlds and meeting incredible beings along the way. From that day on, his life was forever changed, and he became known as the explorer who traveled the multiverse.</p>

Table 7: A comparison case on Ethics skill

Skill: Ethics Difficulty: 8	
Instruction: What are the main ethical theories and how do they differ in their approaches to moral decision making? Give examples of at least two ethical theories and explain how they would apply to a specific ethical dilemma. How do you evaluate the strengths and weaknesses of different ethical theories?	
<i>WizardLM</i>	<p>There are several ethical theories that differ in their approaches to moral decision making. Two of the most prominent ethical theories are consequentialism and deontology. Consequentialism is an ethical theory that holds that the morality of an action is determined by the consequences that it produces. According to consequentialism, the right course of action is the one that produces the best outcome. For example, if a consequentialist were faced with the ethical dilemma of whether to save ten people or one person, they would choose to save the ten people because the outcome of saving ten people is better than the outcome of saving one person. Deontology, on the other hand, is an ethical theory that holds that the morality of an action is determined by the inherent nature of the action itself, regardless of the consequences. According to deontology, the right course of action is the one that is inherently right, regardless of the outcome. For example, if a deontologist were faced with the ethical dilemma of whether to lie to save a life, they would choose not to lie because lying is inherently wrong, even if it means sacrificing the life of one person. Another ethical theory is virtue ethics, which holds that the morality of an action is determined by the character of the person performing the action. According to virtue ethics, the right course of action is the one that is performed by a virtuous person. For example, if a virtue ethicist were faced with the ethical dilemma of whether to steal to feed their family, they would choose not to steal because it is not a virtuous action, even if it means sacrificing the well-being of their family. The strengths of different ethical theories depend on the situation and the values of the individual. Consequentialism is useful in situations where the outcome is important, such as in decision making in business or politics. Deontology is useful in situations where the action itself is important, such as in situations where the law is being broken. Virtue ethics is useful in situations where the character of the person is important, such as in situations where a moral leader is needed. However, each ethical theory also has its weaknesses. Consequentialism can be too focused on the outcome and may not take into account the potential consequences of the action. Deontology can be too rigid and may not take into account the complexity of real-world situations. Virtue ethics can be too subjective and may not be applicable in all situations. In a specific ethical dilemma, a consequentialist would weigh the potential consequences of each action and choose the one that produces the best outcome. For example, if a consequentialist were faced with the ethical dilemma of whether to lie to save a life or tell the truth and risk the life of the person, they would choose to lie because the potential consequences of lying are better than the potential consequences of telling the truth. A deontologist, on the other hand, would choose not to lie because lying is inherently wrong, regardless of the potential consequences. For example, if a deontologist were faced with the ethical dilemma of whether to lie to save a life or tell the truth and risk the life of the person, they would choose to tell the truth because lying is inherently wrong, even if it means sacrificing the life of one person. A virtue ethicist would choose not to steal to feed their family because it is not a virtuous action, regardless of the potential consequences. For example, if a virtue ethicist were faced with the ethical dilemma of whether to steal to feed their family or work harder to earn more money, they would choose to work harder because it is a virtuous action, even if it means sacrificing the well-being of their family.</p>

Vicuna	<p>There are several ethical theories that differ in their approaches to moral decision making. Some of the most prominent ones are:</p> <p>1. Virtue Ethics: This theory emphasizes the development of moral character and virtues, such as courage, honesty, and compassion. According to this theory, the best way to make ethical decisions is to follow a set of universal principles that align with these virtues. For example, if a company is considering a business decision that could harm the environment, a virtue ethics approach would be to consider whether the decision aligns with the value of protecting the environment.</p> <p>2. Deontological Ethics: This theory is based on the idea that there are certain moral rules that apply in all situations and that morality is based on these rules. According to this theory, the best way to make ethical decisions is to follow these rules regardless of the consequences. For example, a doctor may choose to withhold life-saving treatment from a patient if the treatment is morally unacceptable.</p> <p>An example of how these ethical theories would apply to a specific ethical dilemma is the ethical use of artificial intelligence. Virtue ethics would suggest that the best way to ensure ethical AI use is to prioritize the development of moral virtues such as transparency, accountability, and fairness. Deontological ethics, on the other hand, would dictate that AI systems must be designed and used in a way that adheres to a set of pre-defined moral rules, regardless of the consequences.</p> <p>The strengths of virtue ethics are that it emphasizes the importance of personal moral responsibility and development, making it a more individualistic approach. The weakness of this theory is that it may be difficult to determine what virtues are truly universally important, and it does not provide concrete guidelines for decision making in specific situations.</p> <p>The strengths of deontological ethics are that it provides clear and concrete guidelines for decision making, making it a more structured and rule-based approach. The weakness of this theory is that it may not take into account the consequences of following these rules, and that it may be difficult to determine what moral rules are universally applicable.</p> <p>When evaluating the strengths and weaknesses of different ethical theories, it's important to consider their underlying assumptions, their implications for decision-making, and their ability to address the specific ethical dilemmas of a given context. It's also important to consider the practicality of implementing the theory and the degree to which it aligns with our own personal values and beliefs.</p>
---------------	---

Alpaca	<p>Ethical theories can be broadly classified into three categories:</p> <ol style="list-style-type: none"> 1. Deontological ethical theories - which hold that certain actions are inherently right or wrong, regardless of the consequences or intentions. 2. Teleological ethical theories - which hold that the morality of an action depends on the outcome or result it produces. 3. Virtue ethical theories - which hold that the morality of an action depends on the character or traits of the person performing the action. <p>Some examples of ethical theories are:</p> <ol style="list-style-type: none"> 1. Deontological ethical theories: - Utilitarianism - which holds that the morality of an action depends on the outcome or result it produces. - Kantian ethics - which holds that certain actions are inherently right or wrong, regardless of the consequences or intentions. 2. Teleological ethical theories: - Utilitarianism - which holds that the morality of an action depends on the outcome or result it produces. - Consequentialism - which holds that the morality of an action depends on the consequences it produces. 3. Virtue ethical theories: - Aristotelian ethics - which holds that the morality of an action depends on the character or traits of the person performing the action. - Virtue ethics - which holds that the morality of an action depends on the character or traits of the person performing the action. <p>An ethical dilemma is a situation where there are competing moral claims, and it is not clear which course of action is morally right. For example, a doctor may have to decide between saving the life of a pregnant woman or the fetus.</p> <p>Deontological ethical theories would say that the doctor must prioritize the life of the mother, as it is inherently wrong to take the life of an innocent person. Teleological ethical theories would say that the doctor should prioritize the life of the fetus, as the outcome or result is more important than the intentions or actions of the doctor. Virtue ethical theories would say that the doctor should prioritize the life of the person with the strongest moral character, which in this case would be the mother.</p> <p>The strengths and weaknesses of different ethical theories can be evaluated based on their ability to address the ethical dilemma at hand. For example, deontological ethical theories may be strong in providing clear guidelines for moral decision making, but may not take into account the potential consequences of the action. Teleological ethical theories may be strong in their focus on outcomes, but may not consider the intentions or character of the person performing the action. Virtue ethical theories may be strong in promoting moral character, but may not provide clear guidelines for moral decision making in complex situations.</p>
---------------	---

ChatGPT	<p>There are several main ethical theories that differ in their approaches to moral decision making, including consequentialism, deontology, and virtue ethics. Consequentialism holds that the morality of an action is determined by its consequences. It emphasizes the importance of maximizing overall well-being or minimizing harm. Utilitarianism is a well-known consequentialist theory that holds that the moral action is the one that maximizes overall happiness or pleasure, and minimizes overall pain or suffering. Deontology, on the other hand, focuses on the inherent morality of an action, rather than its consequences. It emphasizes the importance of following rules or duties, regardless of the consequences. For example, the moral rule to not lie is considered inherently moral, regardless of whether it leads to a good or bad outcome. Virtue ethics emphasizes the importance of developing and embodying moral virtues or character traits, such as honesty, courage, and compassion. It emphasizes the idea that morality is not just about what we do, but also about who we are. To illustrate how these ethical theories might apply to a specific ethical dilemma, consider the following scenario: A doctor has only one dose of a life-saving medication, and there are two patients who need it urgently. Patient A is elderly and has multiple health problems, while patient B is a young child with no other health issues. A consequentialist might argue that the moral action is to give the medication to the patient who will benefit the most, in terms of maximizing overall well-being or minimizing overall harm. In this case, the young child would likely be given the medication, since they have a longer life expectancy and less potential for complications. A deontologist might argue that the moral action is to follow a moral rule or duty, regardless of the consequences. In this case, the moral rule to prioritize patients based on their medical need might apply, and the medication would be given to the patient who needs it most urgently, regardless of age. A virtue ethicist might argue that the moral action is to embody certain virtues or character traits, such as compassion and empathy, and to consider the unique needs and circumstances of each patient. In this case, the doctor might take into account factors such as the patients' age, overall health, and potential for recovery, and make a decision based on a holistic view of what is most compassionate and just. When evaluating the strengths and weaknesses of different ethical theories, it is important to consider their theoretical coherence, practical applicability, and ability to account for the complexities of real-world moral dilemmas. Some ethical theories may be more well-suited to certain types of situations or contexts, while others may be more flexible or adaptable. Ultimately, the evaluation of ethical theories depends on a range of factors, including their ability to provide clear guidance, promote moral integrity, and promote overall human flourishing.</p>
----------------	--

Table 8: A comparison case on Complex Format skill

Skill: Complex Format Difficulty: 4	
Instruction: Given that the hypotenuse of a right triangle is 13, and the ratio of the lengths of the two legs is 5:12, find the lengths of the two legs. Solution: Let the lengths of the two legs be $5x$ and $12x$, respectively. By the Pythagorean theorem, we have $13^2 = (5x)^2 + (12x)^2$ $169 = 25x^2 + 144x^2$ $169 = 169x^2$ $x^2 = 1$ $x = 1$ or $x = -1$	
WizardLM	Since x cannot be negative, we can discard the solution $x = -1$. Therefore, the lengths of the two legs are $5x = 5(1) = 5$ and $12x = 12(1) = 12$.
Vicuna	Therefore, the two legs of the right triangle are either both 5 units or both 12 units. Since 5 and 12 are not opposite sides of a right triangle, only the combination of 5 and 12 can be a leg in this case. Therefore, the lengths of the two legs are 5 and 12 units.
Alpaca	Since x cannot be negative, we have: $-x = 1$ Therefore, one of the legs is 1 and the other is -1. We'll call them a and b , respectively. So, the solution is: $-a = 1$ $-b = -1$
ChatGPT	Since the length of a side of a triangle cannot be negative, we can only take $x = 1$. Therefore, the lengths of the two legs are: $5x = 5(1) = 5$ $12x = 12(1) = 12$ So, the lengths of the two legs are 5 and 12, respectively.

Table 9: A comparison case on Reasoning skill

Skill: Reasoning Difficulty: 6	
Instruction: A bakery sells three types of cakes: chocolate, vanilla and strawberry. Each cake has a different price and a different number of calories. The chocolate cake costs \$12 and has 400 calories, the vanilla cake costs \$10 and has 300 calories, and the strawberry cake costs \$8 and has 200 calories. A customer wants to buy two cakes with a total budget of \$20 and a maximum of 600 calories. Which two cakes should the customer buy? Explain your reasoning.	
WizardLM	<p>To find the two cakes that meet the customer's requirements, we need to calculate the total number of calories they want to consume and find the cakes that will add up to that number while staying within their budget.</p> <p>Let's start by calculating the total number of calories the customer wants to consume:</p> <ul style="list-style-type: none"> - Chocolate cake: 400 calories - Vanilla cake: 300 calories - Strawberry cake: 200 calories <p>Total calories = 400 + 300 + 200 = 900 calories</p> <p>Since the customer has a maximum of 600 calories, they can only buy one of the strawberry cakes. Now we need to find the second cake that will add up to the remaining 100 calories while staying within their budget. We can only use the chocolate or vanilla cake since they are the only ones that cost more than \$8.</p> <p>If we use the chocolate cake, we will have:</p> <ul style="list-style-type: none"> - Chocolate cake: 400 calories - Vanilla cake: 0 calories - Strawberry cake: 200 calories <p>Total calories = 400 + 0 + 200 = 600 calories</p> <p>If we use the vanilla cake, we will have:</p> <ul style="list-style-type: none"> - Vanilla cake: 300 calories - Chocolate cake: 400 calories - Strawberry cake: 200 calories <p>Total calories = 300 + 400 + 200 = 900 calories</p> <p>Both options exceed the customer's budget, so we need to find a third cake to use. Since the strawberry cake is the only one that costs less than \$8, we can use it with either the chocolate or vanilla cake:</p> <ul style="list-style-type: none"> - Strawberry cake: 200 calories - Chocolate cake: 400 calories - Vanilla cake: 0 calories <p>Total calories = 200 + 400 + 0 = 600 calories</p> <p>This option meets the customer's requirements and stays within their budget. Therefore, the customer should buy one chocolate cake and one strawberry cake.</p>
Vicuna	<p>To find the total number of calories in two cakes, we need to add the calories in each cake: $200 + 400 = 600$</p> <p>Since the customer has a budget of \$20 and a maximum of 600 calories, the total cost of the two cakes should be less than or equal to \$20.</p> <p>Substituting the calorie counts into this equation gives: $200 + 400 \leq 20$</p> <p>Solving for the second cake gives: $400 \leq 20 - 200$ $400 \leq 20$ $400/20 = 20$</p> <p>So the minimum cost of the second cake is \$20.</p> <p>We can use this information to solve for the minimum cost and the maximum number of calories in the two cakes: Minimum cost = $20(20) + 300(8) = \\$20 + \\$240 = \\$260$ Maximum number of calories = 600</p> <p>The customer should buy a chocolate cake and a strawberry cake, as they cost a total of \$260 and have a combined maximum number of calories of 600. These two cakes meet the customer's budget of \$20 and their calorie constraints.</p> <p>Therefore, the customer should buy a chocolate cake (with 400 calories) and a strawberry cake (with 200 calories), since they cost a total of \$260 and have a combined maximum number of calories of 600.</p>

Alpaca	As an AI language model, I do not have personal preferences or opinions. However, based on the given information, a customer can buy either the chocolate cake (\$12, 400 calories) and the vanilla cake (\$10, 300 calories) or the strawberry cake (\$8, 200 calories) and the vanilla cake (\$10, 300 calories). Both options will fulfill the customer's requirements of buying two cakes with a total budget of \$20 and a maximum of 600 calories. If the customer wants to have a balance of flavors, they can buy the chocolate cake and the strawberry cake. If they want to have a balance of calories, they can buy the vanilla cake and the strawberry cake.
ChatGPT	Based on the given information, the customer has a budget of \$20 and wants to buy two cakes. Let's assume that the customer buys chocolate and vanilla cake. In this case, the total cost would be $\$12 + \$10 = \$22$, which exceeds the customer's budget. So, we need to find two cakes that cost no more than \$20 in total. The customer also wants to consume no more than 600 calories. If the customer buys the chocolate cake, which has the highest number of calories, they can only buy one more cake to meet the calorie requirement. If the customer buys the vanilla cake and the strawberry cake, the total cost would be $\$10 + \$8 = \$18$, which fits the customer's budget. Additionally, the total number of calories in these two cakes is $300 + 200 = 500$, which is well below the maximum limit of 600 calories. Therefore, based on the customer's budget and calorie requirements, it is recommended that they buy the vanilla and strawberry cakes.

References

- [1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [2] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- [3] OpenAI. Gpt-4 technical report, 2023.
- [4] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [5] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Z. Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jianyun Nie, and Ji rong Wen. A survey of large language models. *ArXiv*, abs/2303.18223, 2023.
- [6] Xingwei He, Zheng-Wen Lin, Yeyun Gong, Alex Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. Annollm: Making large language models to be better crowdsourced annotators. *ArXiv*, abs/2303.16854, 2023.
- [7] Zhen Guo, Peiqi Wang, Yanwei Wang, and Shangdi Yu. Dr. llama: Improving small language models in domain-specific qa via generative data augmentation. 2023.
- [8] Jia Li, Ge Li, Yongming Li, and Zhi Jin. Enabling programming thinking in large language models toward code generation. 2023.
- [9] Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. Ext5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*, 2022.
- [10] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.
- [11] Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. Zero-prompt: Scaling prompt-based pretraining to 1,000 tasks improves zero-shot generalization. *arXiv preprint arXiv:2201.06910*, 2022.

- [12] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.
- [13] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.
- [14] Ziyang Luo, Can Xu, Pu Zhao, Xiubo Geng, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Augmented large language models with parametric knowledge guiding. *ArXiv*, abs/2305.04757, 2023.
- [15] Huan Ma, Changqing Zhang, Yatao Bian, Lemao Liu, Zhirui Zhang, Peilin Zhao, Shu Zhang, H. Fu, Qinghua Hu, and Bing Wu. Fairness-guided few-shot prompting for large language models. *ArXiv*, abs/2303.13217, 2023.
- [16] Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *ArXiv*, abs/2304.01933, 2023.
- [17] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *ArXiv*, abs/2304.10592, 2023.
- [18] Andreas Kopf, Yannic Kilcher, Dimitri von Rutte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Rich'ard Nagyfi, ES Shahul, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. Openassistant conversations - democratizing large language model alignment. *ArXiv*, abs/2304.07327, 2023.
- [19] Zhihong Chen, Feng Jiang, Junying Chen, Tiannan Wang, Fei Yu, Guiming Chen, Hongbo Zhang, Juhao Liang, Chen Zhang, Zhiyi Zhang, Jianquan Li, Xiang Wan, Benyou Wang, and Haizhou Li. Phoenix: Democratizing chatgpt across languages. *ArXiv*, abs/2304.10453, 2023.
- [20] Zhiqing Sun, Yikang Shen, Qinzhong Zhou, Hongxin Zhang, Zhenfang Chen, David D. Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. *ArXiv*, abs/2305.03047, 2023.
- [21] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Feiran Huang. Rrhf: Rank responses to align language models with human feedback without tears. *ArXiv*, abs/2304.05302, 2023.
- [22] 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. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.
- [23] Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mi Zhou. Automl-gpt: Automatic machine learning with gpt. *ArXiv*, abs/2305.02499, 2023.
- [24] Wen Xiao, Yujia Xie, Giuseppe Carenini, and Pengcheng He. Chatgpt-steered editing instructor for customization of abstractive summarization. *ArXiv*, abs/2305.02483, 2023.
- [25] Potsawee Manakul, Adian Liusie, and Mark John Francis Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *ArXiv*, abs/2303.08896, 2023.
- [26] Shan Zhong, Zhongzhan Huang, Wushao Wen, Jinghui Qin, and Liang Lin. Sur-adaptor: Enhancing text-to-image pre-trained diffusion models with large language models. *ArXiv*, abs/2305.05189, 2023.
- [27] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. *ArXiv*, abs/2305.00447, 2023.
- [28] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *ArXiv*, abs/2304.08485, 2023.

- [29] Ning Bian, Pei Yu Liu, Xianpei Han, Hongyu Lin, Yaojie Lu, Ben He, and Le Sun. A drop of ink may make a million think: The spread of false information in large language models. *ArXiv*, abs/2305.04812, 2023.
- [30] Vivien A. Cabannes, Léon Bottou, Yann LeCun, and Randall Balestriero. Active self-supervised learning: A few low-cost relationships are all you need. *ArXiv*, abs/2303.15256, 2023.
- [31] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [32] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.
- [33] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.
- [34] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.
- [35] Yufei Wang, Jiayi Zheng, Can Xu, Xiubo Geng, Tao Shen, Chongyang Tao, and Daxin Jiang. Knowda: All-in-one knowledge mixture model for data augmentation in few-shot nlp. *arXiv preprint arXiv:2206.10265*, 2022.
- [36] Adrian de Wynter, Xun Wang, Alex Sokolov, Qilong Gu, and Si-Qing Chen. An evaluation on large language model outputs: Discourse and memorization. *ArXiv*, abs/2304.08637, 2023.
- [37] Ekaterina Svikhnushina and Pearl Pu. Approximating human evaluation of social chatbots with prompting. *ArXiv*, abs/2304.05253, 2023.
- [38] Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jia-Bin Huang, Jinglin Liu, Yixiang Ren, Zhou Zhao, and Shinji Watanabe. Audiogpt: Understanding and generating speech, music, sound, and talking head. *ArXiv*, abs/2304.12995, 2023.
- [39] Xiang Yue, Boshi Wang, Kai Zhang, Zi-Yuan Chen, Yu Su, and Huan Sun. Automatic evaluation of attribution by large language models. *ArXiv*, abs/2305.06311, 2023.
- [40] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.
- [41] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.
- [42] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108, 1979.