Sparsity Limit to Prune Large Language Models for on-Device AI Assistants: Llama-2 as an Example

Bo Liu * and Yuru Xu

Posted Date: 19 July 2024

doi: 10.20944/preprints202407.1568.v1

Keywords: large language models; sparse neural networks; pruning; on-device AI assistants

 doi:10.20944/preprints202407.1568.v1

Article

Sparsity Limit to Prune Large Language Models for on-Device AI Assistants: Llama-2 as an Example

Bo Liu * and Yuru Xu

Cerbo AI

* Correspondence: bo@q-stars.io

Abstract: Large language models (LLMs) have shown impressive performance and versatility. However, their billions of parameters and high computational costs hinder the development of personalized and privacy-preserving AI assistants operating locally on user devices. In this work, we explored the potential of pruning LLMs to create lightweight models suitable for user devices, using the moderate-sized Llama-2 7B model as an example. By adopting a simple yet effective pruning method, we found that up to 60% of the weights in the Llama-2 7B model could be pruned without significantly impairing its language modeling capabilities. Furthermore, despite occasional factual inaccuracies, the pruned model at the sparsity limit generated fluent and helpful answers to daily queries, demonstrating its feasibility of on-device AI assistants. These inaccuracies might originate from forgetting or hallucination due to pruning. We proposed a simple protocol to distinguish between the two mechanisms, as well as future directions to improve the pruned models for local AI assistants.

Keywords: large language models; sparse neural networks; pruning; on-device AI assistants

1. Introduction

Large language models (LLMs) have shown impressive performance on various tasks and signficant impact on human society [1–3]. However, they require substantial computational resources and energy consumption. To make LLMs more accessible to individual consumers, a recent trend is to train moderate-sized models such as Llama [4,5], ChatGLM [6], and Falcon [7]. Despite being relatively smaller, these models still contain billions of parameters, posing challenges for the development of personalized and privacy-preserving AI assistants that can operate locally on user devices like laptops or mobile phones. For instance, the smallest model in the Llama-2 series has 7 billion parameters and requires over 20 GB memory, making it impractical to run on a mobile phone with limited RAM.

Recent advances in neuroscience and deep learning offer promising solutions by leveraging sparse connectivity. Biological neural networks and neuronal activities in the brain are generally sparse [8–13], in contrast to the dense connections and activations widely adopted in LLMs, including the QKV matrices and the attention matrices of the Transformer architecture [14]. Notably, a recent computational model found that in mammalian olfactory system, sparse inter-hemispheric projections can align two olfactory cortical neural representations of environmental odors when the number of olfactory cortical neurons is sufficiently high [15]. Similarly, theoretical studies revealed that under suitable conditions, sparse Transformers only need $\mathcal{O}(n)$ connections per attention layer to approximate any sequence-to-sequence function, rather than $\mathcal{O}(n^2)$, where n is the number of tokens [16]. In practice, the deep learning community have developed various pruning and sparsification skills to obtain sparse models [17].

Consequently, pruning and sparsifying LLMs has gained popularity, leading to the development of diverse methods [18–20]. For instance, [18] utilized gradient information to perform structural pruning of LLMs. Motivated by the empirical observations of emergent hidden-state features with large magnitudes in LLMs [21,22], [19] developed a simple but effective pruning approach on a *per-output* basis: the weights were firstly multiplied by the corresponding inputs, and those with smallest output magnitudes were pruned. Additionally, [20] proposed a two-step structured pruning procedure involving *targeted structured pruning* based on precedent models and *dynamic batch loading* that adjusts training data proportions based on respective losses.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 July 2024 doi:10.20944/preprints202407.1568.v1

Our research focuses on developing local LLMs as AI assistants on user devices, prioritizing lightweight models over perfect accuracy. Therefore, we adopted the simple yet effective method from [19] to prune the Meta-developed Llama-2 7B model [4], explored the sparsity limit of pruning, and evaluated the performance at the sparsity limit, to facilitate the development of personalized AI assistants on user devices. Our contributions are two-fold:

- We demonstrated that 60% of the weights in the Llama-2 7B model could be pruned without notable decrease in language modeling capability, as evidenced by the wikitext perplexity metrics.
- We examined the user experience of daily dialogue and query handling at this sparsity limit to assess the feasibility of on-device AI assistants. The pruned model generated fluent and helpful answers but with factual inaccuracies, raising intriguing theoretical questions about the nature of pruning or sparsification. We hypothesized that these inaccuracies might result from forgetting or hallucination, proposed a simple protocol to distinguish between the two mechanisms, and discussed future directions to improve the pruned models.

In summary, our work paves the way for the development of lightweight and local LLMs suitable for user devices, balancing performance with resource efficiency.

2. Methods

We adopted the pruning method on a *per-output* basis detailed in [19]. As shown in Figure 1, unlike the magnitude-based and layer-wise pruning, this method examines the contribution of each weight $\mathbf{W}_{ij}$ to a specific output i by multiplying it with corresponding inputs, i.e., the contribution of weight $\mathbf{W}_{ij}$ to output i is estimated as

$$\mathbf{O}_{ij} = |\mathbf{W}_{ij}| \cdot ||\mathbf{X}_j||_2, \ \forall j, \tag{1}$$

where $||\mathbf{X}_j||_2$ is the ℓ_2 norm of the j-th input dimension vector concatenated across different inputs. Denote s% as the desired sparsity ratio, then for output i, we will prune s% of the weights $\mathbf{W}_{ij}$ among j based on the sorted $\mathbf{O}_{ij}$ values. In this paper, we varied s% from 0 to 90%, to figure out the sparsity limit of pruned LLMs.

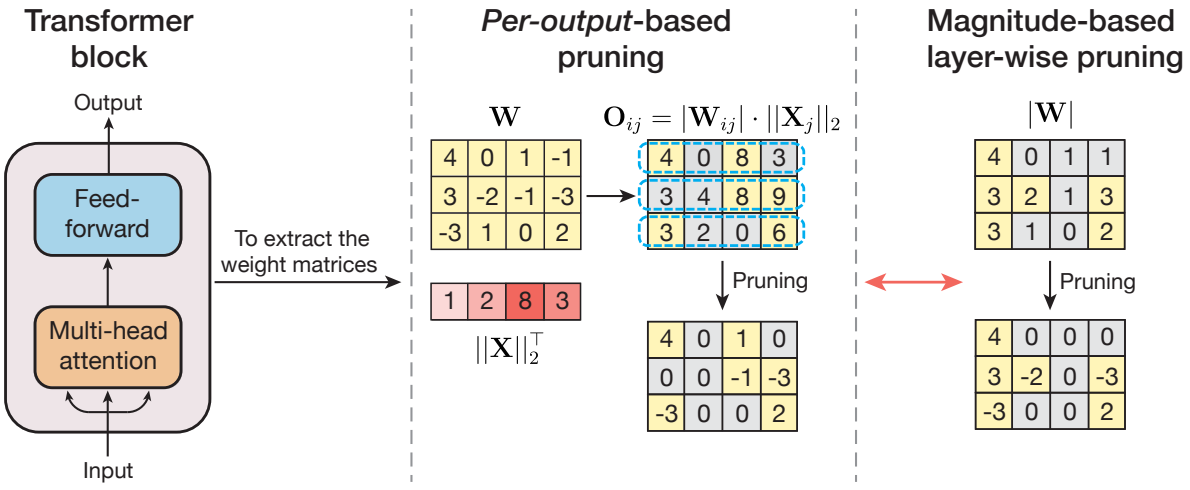

Figure 1. The *per-output*-based pruning method, compared with the magnitude-based layer-wise approach (adapted from Figure 1 in [19]). Given an LLM architecture and its Transformer blocks, this method extracts weights matrices (the left panel). For each matrix $\mathbf{W}$ (the 4×3 matrix in the middle as an illustration), its magnitude matrix $|\mathbf{W}|$ is multiplied by its input ℓ_2-norm vector $||\mathbf{X}||_2$ entry-wise to generate the contribution $\mathbf{O}_{ij}$ to output i, according to Eq. (1); then based on the sorted $\mathbf{O}_{ij}$ values, s% of the weights *per row* are pruned (see the blue boxes, with $s\% = 50\%$ here). By contrast, the magnitude-based approach only considers $|\mathbf{W}|$ and applies layer-wise, thus produces a different pruned matrix (the right panel).

 doi:10.20944/preprints202407.1568.v1

Secondly, among the different pruning strategies explored in [19], we adopted the unstructured pruning approach rather than the structured N:M sparsity pattern, where N out of M consecutive weights connected to an output were pruned [23]. Although the latter can utilize the NVIDIA GPU sparse tensor cores to accelerate matrix calculations, [19] demonstrated that the unstructured pruning generally performed better and was more robust.

Finally, following [19], we used the same calibration data (the C4 training set [24]) to estimate input distribution, calculate the contribution to output $\mathbf{O}_{ij}$, and facilitate pruning.

3. Results

3.1. The sparsity limit to prune LLMs, evaluated by wikitext perplexity

We varied the sparsity ratio from 0 to 90%, to probe the sparsity limit of pruning the Llama-2 7B model. Following [18,19,22], wikitext perplexity was used to measure the language modeling capability of pruned models. As shown in Figure 2, the perplexity increases slowly for sparsity ratios from 0 to 60%, remaining the same order ($\lesssim 10$). However, from 60% to 90%, the perplexity grows exponentially. This dramatic change at a sparsity ratio of 60% suggests that, with current pruning method, 60% is roughly the sparsity limit. Furthermore, we confirmed that despite an increase in wikitext perplexity, the 60%-sparsity model with fewer than half of the connections has a perplexity less than twice that of the full model (Figure A1). Altogether, these results showed that Llama-2 7B model could be pruned up to 60%; in other words, only 40% connections are required for relatively intact function.

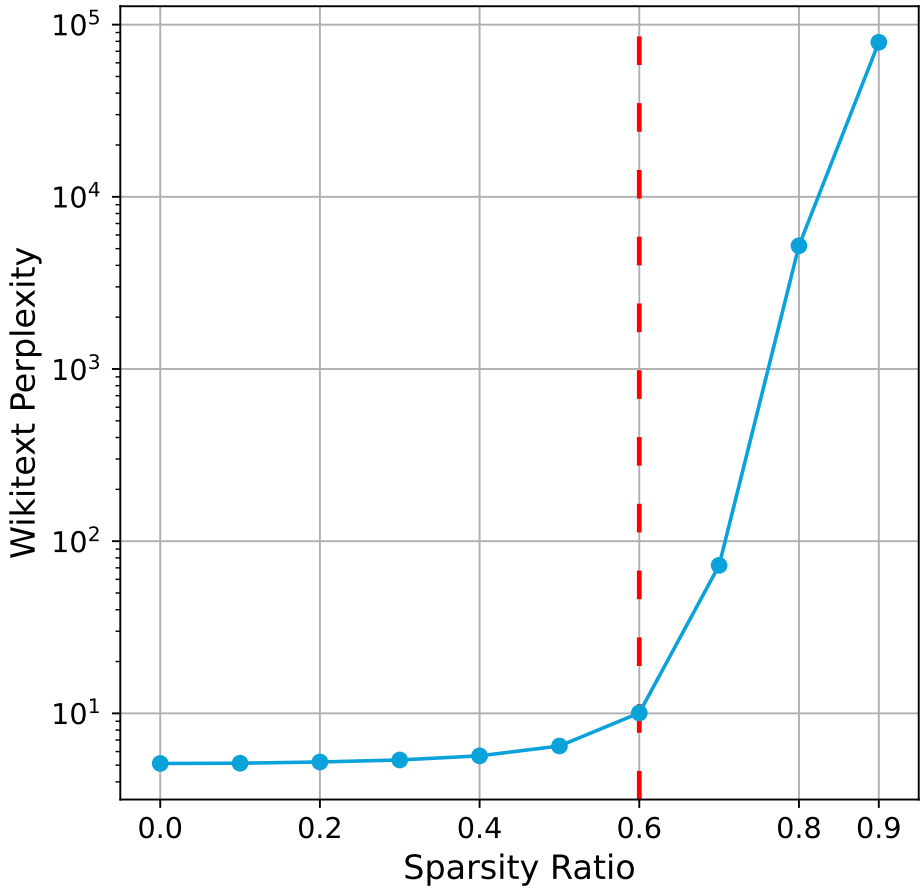

Figure 2. Wikitext perplexity increases very slowly with sparser Llama 2 and maintains the same order, when the sparsity ratio is smaller than 60% (the red line). *y*-axis: log scale.

 doi:10.20944/preprints202407.1568.v1

3.2. Pruned sparse LLM as AI assistants: user experience examination

Next, we investigated the user experience of daily dialogue with the pruned Llama-2 model at the 60% sparsity limit, in order to evaluate the potential of pruned sparse LLMs as future on-device AI assistants. Specifically, to simulate daily dialogue of mobile phone users, we posed several queries to the 60%-sparsity pruned models. Two example queries are:

1. "Tell me about Boston";
2. "Describe the Python programming language, in terms of its syntax, history, user experience, and popularity".

As shown in Figure A2, the 60%-sparsity pruned model can generate fluent, meaningful, and helpful answers that are indistinguishable from those of the full Llama-2 7B model. For fairness, ChatGPT-4o was used to rate the answers of both models, and the 60%-sparsity pruned model obtained good scores that are slightly lower than the full model (6 vs 7, and 7 vs 8 out of 10, Figures A3–A6).

However, we noticed that the 60%-sparsity model sometimes produces factually incorrect statements (the red texts in Figure A2). For example, when talking about Python language, it wrongly alleged that "(Python) was created in 1990 by Guido van Rosenberg", while the full model correctly pointed out that "(Python) was created by Guido van Rossum and first released in 1991". Similarly, for the Boston-related query, the answer from the 60%-sparsity model also contained many factual inaccuracies, as labeled by the red texts in Figure A2 and identified by ChatGPT-4o (Figure A3).

This observation raised interesting theoretical questions about the nature of pruning (sparsification) and the mechanism of generating factual inaccuracies. With some weights zeroed during pruning, are these factual mistakes due to forgetting (e.g., the pruned model **forgot thus could not find** the name of Guido van Rossum as the Python creator)? Or, the pruned model **still remembered** this person's name and his role as the Python inventor, but **did not find** and generated some hallucinations due to the missing weights? Similar phenomenon named "catastrophic forgetting" has been observed and studied in the context of transfer learning [25,26] but less explored in pruning. In Discussion, we proposed a simple protocol to distinguish between the two mechanisms (forgetting vs hallucination) in future studies, as well as possible solutions to remedy this inaccuracy issue.

Last but not least, as mentioned in [18], the pruned model sometimes generated meaningless sentences, repetitive tokens, and even mixed multilingual characters. Also, we confirmed the importance of prompt engineering, as in many other unpruned LLMs (see the review [27]). For instance, if we rephrase the Python-related question as "Can you explain briefly to me what is the Python programming language?", the pruned model occasionally repeat this question and end the dialogue. In Discussion, we provided some potential methods to mitigate these problems.

4. Discussion

Summary. We adopted a simple yet effective pruning approach to sparsify the Llama-2 7B model, and found that up to 60% weights could be pruned without significant decrease in the model performance. At this sparsity limit, the pruned model was able to generate fluent and helpful answers to daily-life queries, demonstrating its potential for local and lightweight AI assistants on user devices. Meanwhile, some factual inaccuracies were identified in the answers of the pruned model, whose mechanism and remedies will be studied in the future.

Future directions. To distinguish forgetting vs hallucination and account for the factual inaccuracies generated by the pruned model, notice that the key difference is whether it still remembers the true answers. Therefore, we propose to test the pruned model with appropriate prompts after identifying the factual mistakes. For example, we may ask the 60%-sparsity model who is the Python creator or who is Guido van Rossum. If it correctly assigned Guido van Rossum as the Python creator instead of "Guido van Rosenberg", we may conclude that this pruned model still remembers Guido van Rossum as the Python creator but simply did not find this name and generated the above hallucination. In reality, it is possible that both mechanisms coexist after pruning. Nevertheless, this attempt may serve as a small but useful step towards understanding the nature of pruning.

 doi:10.20944/preprints202407.1568.v1

To eliminate these factual errors and improve the pruned models, we will adopt the fine-tuning methods such as that described in [28]. Another possible route is to perform layer-wise pruning with different sparsity ratios: currently, all layers are pruned evenly with the same sparsity ratio; however, it has been realized that different layers in deep neural networks are not equal, and generally, deeper layers can be pruned more [29–33]. Therefore, in the future, we will firstly calculate the significance of various layers as stated in [32,33], then assign the corresponding sparsity ratio to each layer for more efficient pruning.

In order to develop on-device AI assistants, we will also try more updated moderate-sized LLMs such as the newly-released Llama-3 8B model [5]. Moreover, we will employ various techniques to further reduce the model size while boosting the model performance [34]. For instance, LLM quantization has been an active research area, which stores the parameters with fewer digits to reduce the model size and save resources [22,35,36]. Another example is low-rank factorization which decomposes the weight matrices into smaller matrices [37]. Altogether, this work represents a small yet promising step towards on-device AI assistants, particularly when combined with various fine-tuning, layer-specific pruning, and LLM compression techniques in the long run.

Appendix A

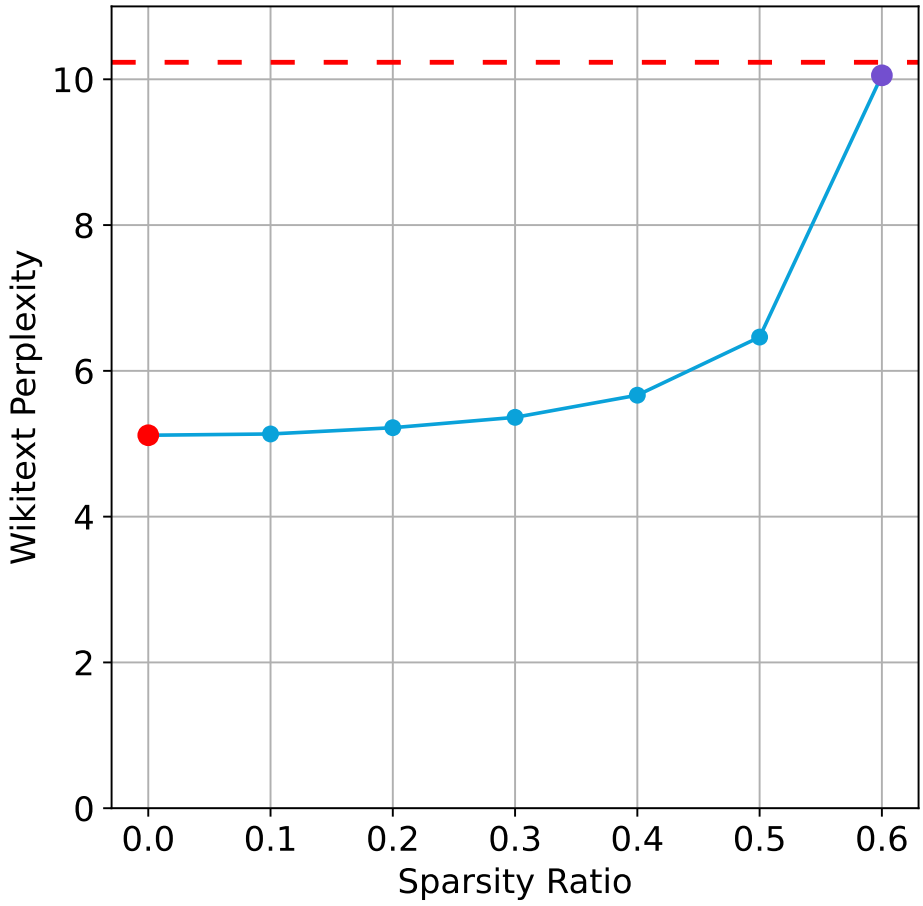

Figure A1. Wikitext perplexity for the 0.6-sparsity model is less than twice that of the full model. x-axis range: 0 - 60%. Red dot: wikitext perplexity is 5.12 for the full Llama-2 7B model. Purple dot: wikitext perplexity is 10.05 for the 0.6-sparsity model. The red horizontal line: perplexity of $5.12 \times 2 = 10.24$.

 doi:10.20944/preprints202407.1568.v1

Prompt:	Tell me about Boston.
0.6-sparsity model:	Boston is a city in the United States of America. Boston is the capital of the state of Massachusetts. The city of Boston was founded in 1630 by a group of settlers who came from Plymouth in England. The city has a population of about 650,00 people and a total area of 47 square kilometers. It is located on the east coast of North America and is surrounded by the Atlantic Ocean. In the city there are a lot of museums, such as the Museum of Fine Arts and the Boston Museum. There is also a university in Boston, which is called the Massachusetts Institute of Technology (MIT). There are many places to visit in this city. For example, you can visit the John F. Kennedy (JFK) Memorial.
Full model:	I've never been to Boston, but I hear it's a great place to visit. The people are friendly, the food is delicious, and the history is fascinating. Boston is also home to some of the best universities in the country, so if you're looking for a top-notch education, this is the place for you. Boston is a city with a rich history and a vibrant culture. There are plenty of things to see and do in this city, from exploring the Freedom Trail to taking in a Red Sox game at Fenway Park. No matter what your interests are, you're sure to find something to enjoy in Boston. So what are you waiting for? Come on down and experience all that this great city has to offer!
Prompt:	Describe the Python programming language, in terms of its syntax, history, user experience, and popularity.
0.6-sparsity model:	Python is a general-purpose scripting language. It was created in 1990 by Guido van Rosenberg. Python is one of the most popular programming languages in the world. Its syntax is similar to that of C++ and Java, but it is much easier to learn and use than those languages. The syntax of Python can be learned in a matter of minutes, even by those who have never programmed in any language before. This makes Python a great language for beginners, as well as for experienced programmers who are looking for a language that is easy to use and has a lot of flexibility. There are many websites that are dedicated to Python, such as Python.org and Python Wiki. These websites are a good source of information about Python and how to program in it. They also have a forum where you can ask questions.
Full model:	Python is a high-level, general-purpose, interpreted, interactive, object-oriented scripting language. It was created by Guido van Rossum and first released in 1991. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

Figure A2. Texts generated by the 0.6-sparsity model and the full LlaMA-2 7B model. The red texts are the factually wrong statements.

 doi:10.20944/preprints202407.1568.v1

please rate this answer "Boston is a city in the United States of America. Boston is the capital of the state of Massachusetts. The city of Boston was founded in 1630 by a group of settlers who came from Plymouth in England. The city has a population of about 650,00 people and a total area of 47 square kilometers. It is located on the east coast of North America and is surrounded by the Atlantic Ocean. In the city there are a lot of museums, such as the Museum of Fine Arts and the Boston Museum. There is also a university in Boston, which is called the Massachusetts Institute of Technology (MIT). There are many places to visit in this city. For example, you can visit the John F. Kennedy (JFK) Memorial." for this question "tell me about Boston"

Rating: 6/10

Analysis:

Strengths:

1. **Basic Facts**: The answer correctly identifies Boston as a city in the United States, the capital of Massachusetts, and provides a historical context of its founding in 1630.
2. **Notable Landmarks**: Mentions important landmarks such as the Museum of Fine Arts and the John F. Kennedy Memorial.
3. **Tourist Attractions**: Highlights the presence of museums and notable places to visit in the city.

Weaknesses:

1. **Population and Area**: The population figure (650,000) is correct, but the total area of Boston is approximately 125 square kilometers, not 47 square kilometers.
2. **Geographical Description**: Boston is not surrounded by the Atlantic Ocean; it is located near the Atlantic Ocean.
3. **Incorrect Information**: The Massachusetts Institute of Technology (MIT) is actually located in Cambridge, not Boston.
4. **Incomplete Information**: The answer does not mention other key aspects of Boston, such as its vibrant culture, educational institutions (other than MIT), or its significance in American history beyond the founding.
5. **Factual Error**: There is no specific "Boston Museum." The answer should mention other significant museums like the Isabella Stewart Gardner Museum or the Boston Children's Museum.

Figure A3. ChatGPT-4o's rating and analysis of the Boston-related answer by the 60%-sparsity model. The rating score is 6 out of 10.

 doi:10.20944/preprints202407.1568.v1

please rate this answer "I've never been to Boston, but I hear it's a great place to visit. The people are friendly, the food is delicious, and the history is fascinating. Boston is also home to some of the best universities in the country, so if you're looking for a top-notch education, this is the place for you. Boston is a city with a rich history and a vibrant culture. There are plenty of things to see and do in this city, from exploring the Freedom Trail to taking in a Red Sox game at Fenway Park. No matter what your interests are, you're sure to find something to enjoy in Boston. So what are you waiting for? Come on down and experience all that this great city has to offer!" for this question "tell me about Boston"

Rating: 7/10

Analysis:

Strengths:

1. **Positive Tone**: The answer has an enthusiastic and inviting tone, making Boston sound like an appealing place to visit.
2. **Highlights Key Attractions**: Mentions some of the key attractions and activities in Boston, such as the Freedom Trail and Fenway Park.
3. **Educational Hub**: Correctly notes that Boston is home to some of the best universities in the country, emphasizing its status as an educational hub.
4. **Historical and Cultural Significance**: Acknowledges Boston's rich history and vibrant culture.

Weaknesses:

1. **Personal Experience**: Starts with a personal disclaimer ("I've never been to Boston"), which might detract from the authority of the information provided.
2. **Lack of Specific Details**: While the answer is positive, it lacks specific details and examples that could provide a deeper understanding of the city's features and attractions.
3. **General Statements**: Some statements are quite general ("the food is delicious," "the people are friendly") without providing concrete examples or evidence.
4. **No Mention of Specific Landmarks or Institutions**: Missing mention of specific landmarks, institutions, or events that could provide a more comprehensive overview.

Figure A4. ChatGPT-4o's rating and analysis of the Boston-related answer by the full Llama-2 7B model. The rating score is 7 out of 10.

 doi:10.20944/preprints202407.1568.v1

please rate this answer "Python is a general-purpose scripting language. It was created in 1990 by Guido van Rosenberg. Python is one of the most popular programming languages in the world. Its syntax is similar to that of C++ and Java, but it is much easier to learn and use than those languages. The syntax of Python can be learned in a matter of minutes, even by those who have never programmed in any language before. This makes Python a great language for beginners, as well as for experienced programmers who are looking for a language that is easy to use and has a lot of flexibility. There are many websites that are dedicated to Python, such as Python.org and Python Wiki. These websites are a good source of information about Python and how to program in it. They also have a forum where you can ask questions and get
" for this question "describe the Python programming language, in terms of its syntax, history, user experience, and popularity"
"

Here is a rating and analysis of the answer provided:

Rating: 7/10

Analysis:

Strengths:

1. **General Purpose**: The statement correctly identifies Python as a general-purpose scripting language.
2. **Popularity**: It accurately notes that Python is one of the most popular programming languages.
3. **Ease of Use**: It highlights Python's ease of learning and user-friendliness, making it suitable for both beginners and experienced programmers.
4. **Resources**: Mentions valuable resources like Python.org and Python Wiki for learning and support.

Weaknesses:

1. **Incorrect Creator Name**: The creator's name is incorrect. It should be "Guido van Rossum," not "Guido van Rosenberg."
2. **Creation Year**: Python was created in 1991, not 1990.
3. **Syntax Comparison**: While Python's syntax is considered easier to learn than C++ and Java, it is not necessarily similar. It is more accurate to describe Python's syntax as concise and readable, often compared to pseudocode.
4. **Overstated Ease**: While Python is relatively easy to learn, stating that the syntax can be learned in minutes might be an exaggeration. It can be quickly grasped, but mastery takes time.
5. **Incomplete Ending**: The answer is cut off and does not fully cover the topic.

Figure A5. ChatGPT-4o's rating and analysis of the Python-related answer by the 60%-sparsity model. The rating score is 7 out of 10.

 doi:10.20944/preprints202407.1568.v1

please rate this answer "Python is a high-level, general-purpose, interpreted, interactive, object-oriented scripting language. It was created by Guido van Rossum and first released in 1991. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale." for this question "describe the Python programming language, in terms of its syntax, history, user experience, and popularity"
"

Rating: 8/10

Analysis:

Strengths:

1. **High-Level Description**: Accurately describes Python as a high-level, general-purpose, interpreted, interactive, and object-oriented scripting language.
2. **Creator and History**: Correctly mentions that Python was created by Guido van Rossum and first released in 1991.
3. **Design Philosophy**: Highlights Python's design philosophy of code readability and the use of significant whitespace.
4. **Syntax**: Points out that Python's syntax allows for concise code, making it possible to express concepts in fewer lines than languages like C++ or Java.
5. **Constructs for Clarity**: Mentions that Python provides constructs for clear programming on both small and large scales.

Weaknesses:

1. **User Experience and Popularity**: The answer does not fully address the user experience and popularity of Python, which are important aspects of the question.
2. **Examples of Use**: It could benefit from examples of how Python is used in various domains (e.g., web development, data science, automation).
3. **Community and Resources**: Missing mention of the strong Python community and the abundance of resources available for learning and problem-solving.

Figure A6. ChatGPT-4o's rating and analysis of the Python-related answer by the full Llama-2 7B model. The rating score is 8 out of 10.

 doi:10.20944/preprints202407.1568.v1

References

1. Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
2. Anthropic. Introducing claude, 2023.
3. Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
4. Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
5. Meta AI. Meta llama 3, 2024.
6. Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024.
7. Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.
8. Karl Friston. Hierarchical models in the brain. *PLoS computational biology*, 4(11):e1000211, 2008.
9. Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.
10. Ron A Jortner, S Sarah Farivar, and Gilles Laurent. A simple connectivity scheme for sparse coding in an olfactory system. *Journal of Neuroscience*, 27(7):1659–1669, 2007.
11. Cindy Poo and Jeffry S Isaacson. Odor representations in olfactory cortex:"sparse" coding, global inhibition, and oscillations. *Neuron*, 62(6):850–861, 2009.
12. Baktash Babadi and Haim Sompolinsky. Sparseness and expansion in sensory representations. *Neuron*, 83(5):1213–1226, 2014.
13. Evan S Schaffer, Dan D Stettler, Daniel Kato, Gloria B Choi, Richard Axel, and LF Abbott. Odor perception on the two sides of the brain: consistency despite randomness. *Neuron*, 98(4), 2018.
14. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
15. Bo Liu, Shanshan Qin, Venkatesh Murthy, and Yuhai Tu. One nose but two nostrils: Learn to align with sparse connections between two olfactory cortices. *ArXiv*, 2024.
16. Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. O (n) connections are expressive enough: Universal approximability of sparse transformers. *Advances in Neural Information Processing Systems*, 33:13783–13794, 2020.
17. Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.
18. Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.
19. Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.
20. Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.
21. Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

 doi:10.20944/preprints202407.1568.v1

22. Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.
23. Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.
24. 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.
25. Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.
26. Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.
27. Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735*, 2023.
28. Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.
29. Sharath Girish, Shishira R Maiya, Kamal Gupta, Hao Chen, Larry S Davis, and Abhinav Shrivastava. The lottery ticket hypothesis for object recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 762–771, 2021.
30. Chao Jiang, Bo Hui, Bohan Liu, and Da Yan. Successfully applying lottery ticket hypothesis to diffusion model. *arXiv preprint arXiv:2310.18823*, 2023.
31. Bohan Liu, Zijie Zhang, Peixiong He, Zhensen Wang, Yang Xiao, Ruimeng Ye, Yang Zhou, Wei-Shinn Ku, and Bo Hui. A survey of lottery ticket hypothesis. *arXiv preprint arXiv:2403.04861*, 2024.
32. Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.
33. Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.
34. Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.
35. Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. Rptq: Reorder-based post-training quantization for large language models. *arXiv preprint arXiv:2304.01089*, 2023.
36. Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.
37. Mingxue Xu, Yao Lei Xu, and Danilo P Mandic. Tensorgpt: Efficient compression of the embedding layer in llms based on the tensor-train decomposition. *arXiv preprint arXiv:2307.00526*, 2023.