Advanced Large Language Model Optimization and Quantization for Local Inference

Vadim Voevodkin, Ovakimyan Kristina, Dmitriy Vorob'ev, Artyom Pyanzin

vsvoevodkin@edu.hse.ru, kaovakimyan_1@edu.hse.ru,
Dmvorobev@edu.hse.ru, aapyanzin@edu.hse.ru

1 Abstract

LLM (Large Language Model) quantization is an important task in natural language processing because it reduces the model size and improves their real-time performance when dealing with large amounts of data. In this paper, a review of existing LLM model quantization methods is conducted, including quantization of weights, activations and other model parameters. For each method, their advantages and disadvantages as well as their applicability to different types of LLM models are analyzed. This paper also presents an example of quantization of the LLM model Nous-Hermes-Llama2-7b (1) with algorithm (2). After quantization, a comparison is made between the original model and the model after quantization based on their performance and accuracy on question datasets (3). Finally, the prospect of applying quantization of LLM models in various applications and their potential for effective use in the real world is discussed.

2 Introduction

In recent years, LLM (Large Language Model) models have increasingly found integration into our daily routines. This paper aims to explore the potential opportunities that may emerge for the general population, particularly those with standard hardware setups, to utilize LLM models within their homes.

One of the focal points of this study involves the quantization of the LLM model Nous-Hermes-Llama2-7b, as per the algorithm put forth by Dettmers et al. (2023), with the aim of demonstrating the practical application of quantization methods in the context of an existing LLM model. The process involves investigating the impact of quantization on model performance and accuracy, particularly concerning its performance on question datasets, as exemplified by the Allen AI ARC dataset. Through this example, we aim to illustrate the real-world implications of quantization, shedding light on the trade-offs between model compression and performance.

Furthermore, this paper expands its focus to explore the potential applications of quantized LLM models in diverse domains, encompassing areas such as language translation, conversational AI, and content generation. By delving into the practical implications of LLM model quantization, we endeavor to provide insights into how this technique can be effectively leveraged across various real-world scenarios, paving the way for enhanced accessibility and usability of advanced language models. The study aims to provide a nuanced understanding of the implications of LLM quantization, offering valuable insights into the future prospects of deploying these models in practical settings and

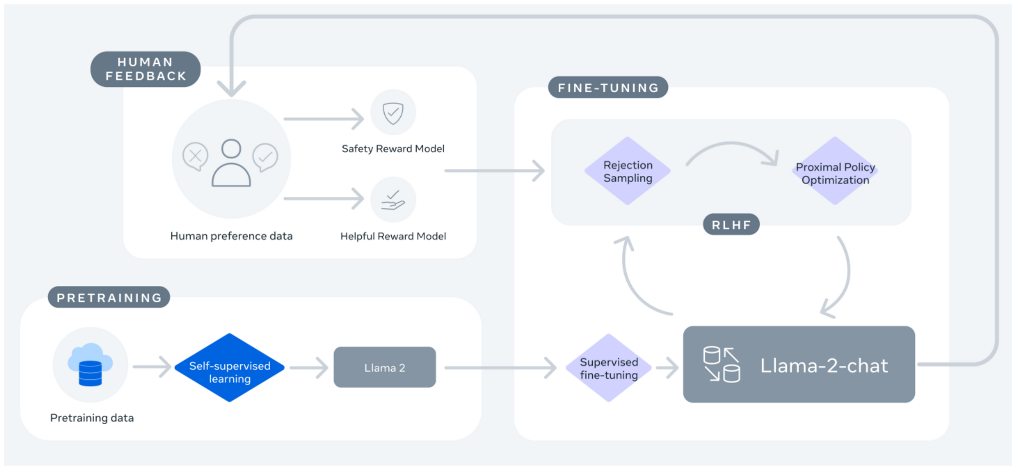


Figure 1: Training of Llama 2-Chat.(4)

shedding light on potential avenues for further exploration and development in this domain. We will delve into the techniques of model quantization and examine how they can be applied. Our objective is to assess the most effective quantization method and explore its application to a well-known LLM model.

3 Related Work

3.1 Llama 2 model introduction

The model we utilized as the primary foundation for task optimization is Llama 2, an advancement of the initial Meta's LLM. To create the new family of Llama 2 models, the authors built upon the pretraining approach from Touvron et al. (2023) and made significant optimizations for improved performance. This included enhanced data cleaning, updated data mixes, training with 40% more tokens, doubling the context length, and implementing grouped-query attention (GQA) for better scalability(4). The training corpus consists of a new mix of publicly available data, with efforts to remove personal information from certain sites. Trained on 2 trillion tokens, the focus was on factual sources to increase knowledge and reduce inaccuracies.

The process initiates with pretraining Llama 2 using publicly available online sources. Next, authors develop an initial version of Llama 2-Chat through supervised fine-tuning. Subsequently, the model undergoes iterative refinement using Reinforcement Learning with Human Feedback (RLHF) techniques, particularly via rejection sampling and Proximal Policy Optimization (PPO). It is imperative to accumulate iterative reward modeling data in tandem with model enhancements during the RLHF stage to ensure the reward models remain within distribution.

While adopting most of the pretraining settings and model architecture from Llama 1, the authors incorporated architectural differences such as increased context length and GQA, detailed in the Appendix. The hyperparameters included the AdamW optimizer, a cosine learning rate schedule, weight decay, and gradient clipping. The same tokenizer as Llama 1 was used, employing bytepair encoding with a 32k token vocabulary. Building on previous studies(4), the authors are determined to accurately measure the carbon emissions generated from the pretraining of Llama 2 models by taking into account GPU power consumption and carbon efficiency. The findings reveal that the pretraining of the Llama 2 family of models resulted in an estimated 539 tCO2eq emissions. In supervised

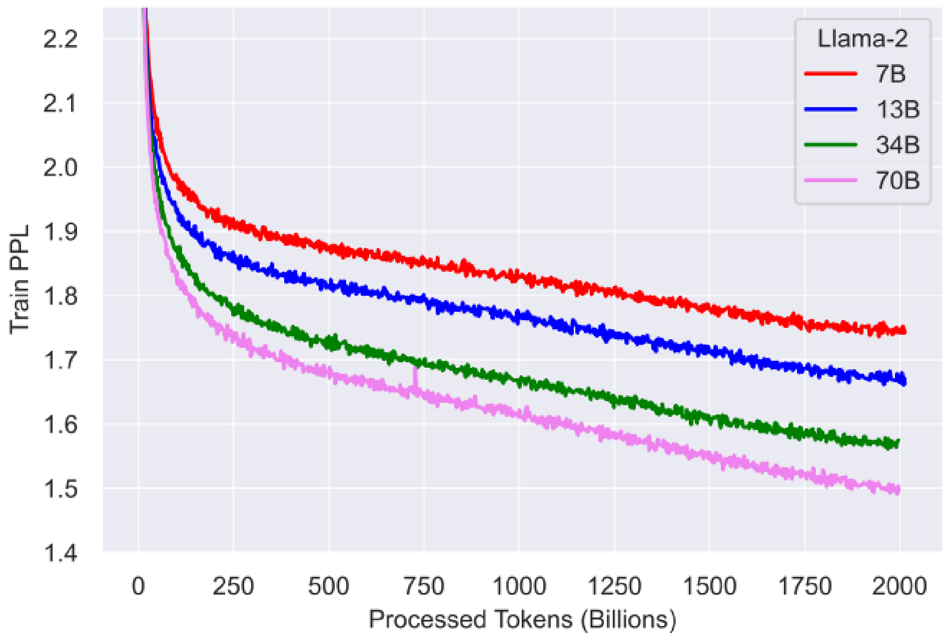


Figure 2: Training Loss for Llama 2 models.(4)

fine-tuning, a cosine learning rate schedule is applied, beginning with an initial learning rate of 2×10^{-5}. The weight decay is set at 0.1, using a batch size of 64 and a sequence length of 4096 tokens. Each sample for fine-tuning includes a prompt and an answer, which are concatenated to fill the model sequence properly, with a special token used to separate the prompt and answer segments. An autoregressive objective is utilized, and the loss on tokens from the user prompt is zeroed out, resulting in backpropagation occurring only on answer tokens. Lastly, the model is fine-tuned for 2 epochs.(4) Specifically, we employed the smallest version with 7 billion parameters.

3.2 Nous-Hermes-Llama2-7b model description

Sue to the strong performance of fine-tuned LLMs like Instruct-tuned LLMs on general tasks (a result of the fine-tuning process on QA), rather than commencing with the original Llama 2, we opted to use an already fine-tuned checkpoint known as the Nous-Hermes-Llama2-7b model, which is accessible at Hugging Face(1). This model has been fine-tuned from Llama 2 using a dataset of corpus of instructions and related answers structured as follows:

```
###Instruction:
<instruction/question to be answered>
###Response:
<answer to the instruction>
```

This is the modus operandi of the alpaca format - just how things go in the alpaca world.

Nous-Hermes-Llama2-7b is an SOTA language model that has been fine-tuned on an extensive dataset of over 300,000 instructions. This fine-tuning process was led by Nous Research, with Teknium overseeing the fine-tuning and dataset curation, Redmond AI providing the necessary compute, and several other contributors adding to the collaborative effort. The decision to use the same dataset as the previous Hermes model on Llama-1 was made to ensure continuity and consistency between the old and new versions, while enhancing Hermes' capabilities.

This model is distinguished by its ability to generate long responses, its lower rate of hallucinations, and the absence of OpenAI censorship mechanisms. The fine-tuning process involved using a sequence length of 4096 on a powerful 8x a100 80GB DGX machine.(1) The training data drew primarily from synthetic GPT-4 outputs, which were carefully curated to ensure high quality in knowledge, task completion, and style. The dataset includes contributions from diverse sources such as GPTeacher, the general, roleplay v1,2, code instruct datasets, Nous Instruct and PDACTL (unpublished), and several others, detailed further below.

Contributors to the datasets included Teknium, nlpxucan, Karan4D, HueminArt, Microsoft, jondurbin, Camel-AI, and Sahil 2801, who provided datasets such as GPTeacher, Wizard LM, Nous Research Instruct Dataset, GPT4-LLM, Unnatural Instructions, Airoboros, Camel-AI's domain expert datasets, and the CodeAlpaca dataset.(1)

The combination of these diverse datasets and the collaboration has contributed to the versatile and quality of the language model. Furthermore, the utilization of synthetic GPT-4 outputs in training the model has further enhanced its capabilities in generating high-quality responses and completing tasks effectively. This unique approach has set Nous-Hermes-Llama2-7b apart as an advanced and versatile language model capable of meeting the demands of various linguistic and cognitive tasks.

3.3 Quantization or a Story about how to run LLM on your PC

The article "Efficient LLM Inference on CPUs" by Haihao Shen, Hanwen Chang, Bo Dong, Yu Luo, and Hengyu Meng focuses on improving the performance of Large Language Model (LLM) inference on Central Processing Units (CPUs). Large language models, such as GPT-3, have gained significant attention in natural language processing and have demonstrated remarkable capabilities in tasks such as text generation, translation, and question-answering. However, the computational requirements for running these large models can be prohibitive, particularly on CPUs. (5)

The authors propose a novel method that leverages advanced optimization techniques and parallel computation to address the challenges of inefficient LLM inference on CPUs. The key technical details of their approach include:

1. Parallelization: The authors propose parallelizing the matrix-vector multiplication operations to accelerate the inference process. By utilizing efficient parallel computation strategies, such as multi-threading and SIMD (Single Instruction, Multiple Data) instructions, they aim to maximize CPU utilization and improve the overall inference speed.

2. Optimization Strategies: In addition to parallelization, the authors employ a range of optimization strategies to enhance the efficiency of LLM inference. This includes optimizing memory access patterns, minimizing cache misses, and reducing redundant computations to streamline the execution of LLM inference tasks on CPUs. Building on the ggml library, they have developed a tensor library specifically for CPU, fully supporting AVX2, AVX512, AVX512 VNNI Rodriguez et al. [2018], and AMX (Advanced Matrix Extensions) instruction sets. Their findings demonstrate that with just a single socket of 4th Generation Intel Xeon Scalable Processors(5), the average latency for generating tokens on large language models (LLMs) with 6B to 20B parameters was reduced from 20ms to 80ms, while maintaining high accuracy with only a 1% loss from FP32 baseline.

3. Lightweight Kernel Fusion: The authors introduce a lightweight kernel fusion technique to consolidate multiple operations into a single kernel, reducing the overhead associated with invoking individual operations and improving the overall performance of LLM inference.

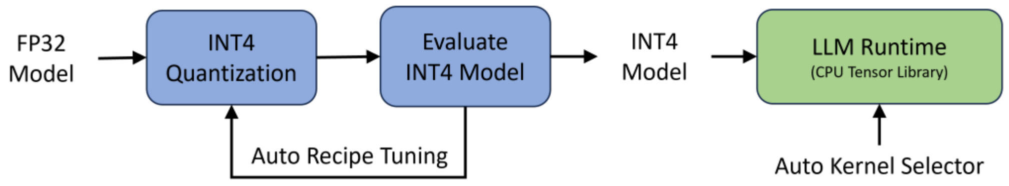


Figure 3: Efficient LLM runtime pipeline.(5)

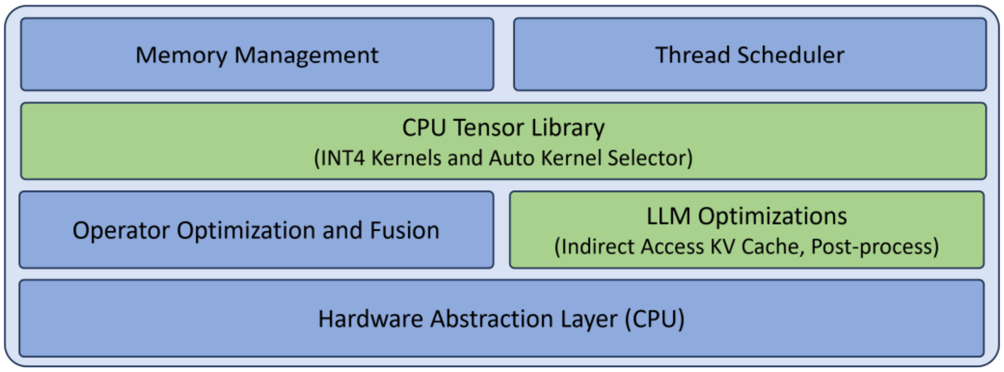

Memory Management
Thread Scheduler
CPU Tensor Library
(INT4 Kernels and Auto Kernel Selector)
Operator Optimization and Fusion
LLM Optimizations
(Indirect Access KV Cache, Post-process)
Hardware Abstraction Layer (CPU)

Figure 4: Key components in LLM runtime: general and LLM specialized..(5)

4. Quantization and Pruning: To further optimize the inference process, the authors explore techniques such as quantization and model pruning to reduce the computational demands of LLMs while maintaining their accuracy. These techniques are tailored to exploit the CPU architecture and improve the efficiency of inference tasks.

Through a series of experiments and evaluations, the authors demonstrate that their proposed method significantly reduces the latency and improves the throughput of LLM inference on CPUs. This enhanced efficiency makes LLMs more practical and accessible for real-time applications, enabling their deployment in a broader range of industry use cases. We would like to pay attention to quantization technique. The INT4 quantization flow is constructed utilizing the Intel Neural Compressor, a well-established quantization tool for deep learning platforms. This approach allows flexibility in tuning quantization recipes, granularities, and group sizes, enabling the generation and evaluation of INT4 models tailored to meet accuracy targets. Once validated, these INT4 models are passed to the LLM Runtime for performance assessment. This runtime is engineered to deliver efficient inference of LLMs on CPUs, with specific components optimized for LLM inference, depicted in green, and general runtime components, such as memory management and thread scheduling, in blue. Further elaboration on the CPU tensor library and LLM optimizations is provided in the subsequent sections, while general components are omitted due to space constraints.

3.4 One Data To Rule Them All

The AI2 Reasoning Challenge (ARC) dataset was selected for the next experiments. This dataset introduces a new set of questions, a text corpus, and baselines designed to drive AI research in advanced question answering. Collectively, these components form the AI2 Reasoning Challenge (ARC), requiring significantly more robust knowledge and reasoning than previous challenges such

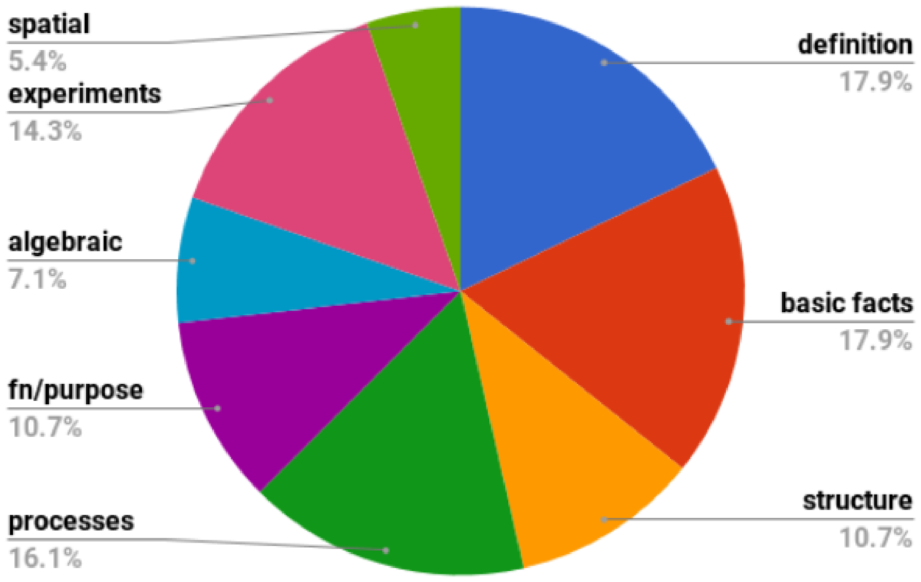


Figure 5: Relative sizes of different knowledge types suggested by the ARC Challenge Set.(3)

as SQuAD or SNLI. The ARC question set is divided into a Challenge Set and an Easy Set, with the Challenge Set comprising only questions that were answered incorrectly by both a retrieval-based algorithm and a word co-occurrence algorithm. The dataset consists solely of natural, grade-school science questions (crafted for human tests) and stands as the largest public-domain set of this kind, comprising 7,787 questions.(3) In addition to the ARC question set, authors are releasing the ARC Corpus, a substantial collection of science-related sentences extracted from the Web. This corpus encompasses 14 million sentences (1.4GB of text) and includes much of the knowledge necessary to address the Challenge Questions. While some of these references are indirect and leveraging them is not straightforward, it still serves as a foundational resource for tackling the ARC Challenge. It is important to note that the use of the corpus is optional, and systems are not bound to rely solely on this corpus.

4 Experimental Results

So in this paper, we investigated actual solutions and approaches for quantization of LLM models. Namely toolkit from Intel was used `https://github.com/intel/intel-extension-for-transformers`. Within which the automatic selection of the optimal algorithm for quantization of models is implemented. Application of this toolkit for our SOTA model allowed to reduce memory consumption 5 times from 31 GB to 6 GB. This size allows us to run this LLM model on any fairly simple device without excessive costs. Obviously, such a strong compression could not but affect the target metrics, and our model became near 20 percent worse at answering questions from the selected dataset. The main feature of our optimized model is Token per second(TOPS). The optimization accelerated the model more than 20 times from 1 token per 5 seconds to 4.8 TOPS.

Model	Size	CPU (percent)	Memory (Gb)	Dataset Accuracy (percent)	TOPS
llama-2	7b	100%	31.2	79.46	0.2
llama-2 **Optimized**	7b	100%	6.3	53.20	4.8

Table 1: Results of experiments.

5 Acknowledges

The authors of this article are grateful to Professor Krylov Vladimir Vladimirovich for setting the problem and valuable recommendations. The results of the experiments are available `https://github.com/thereez/hse_reserach_seminar`.

References

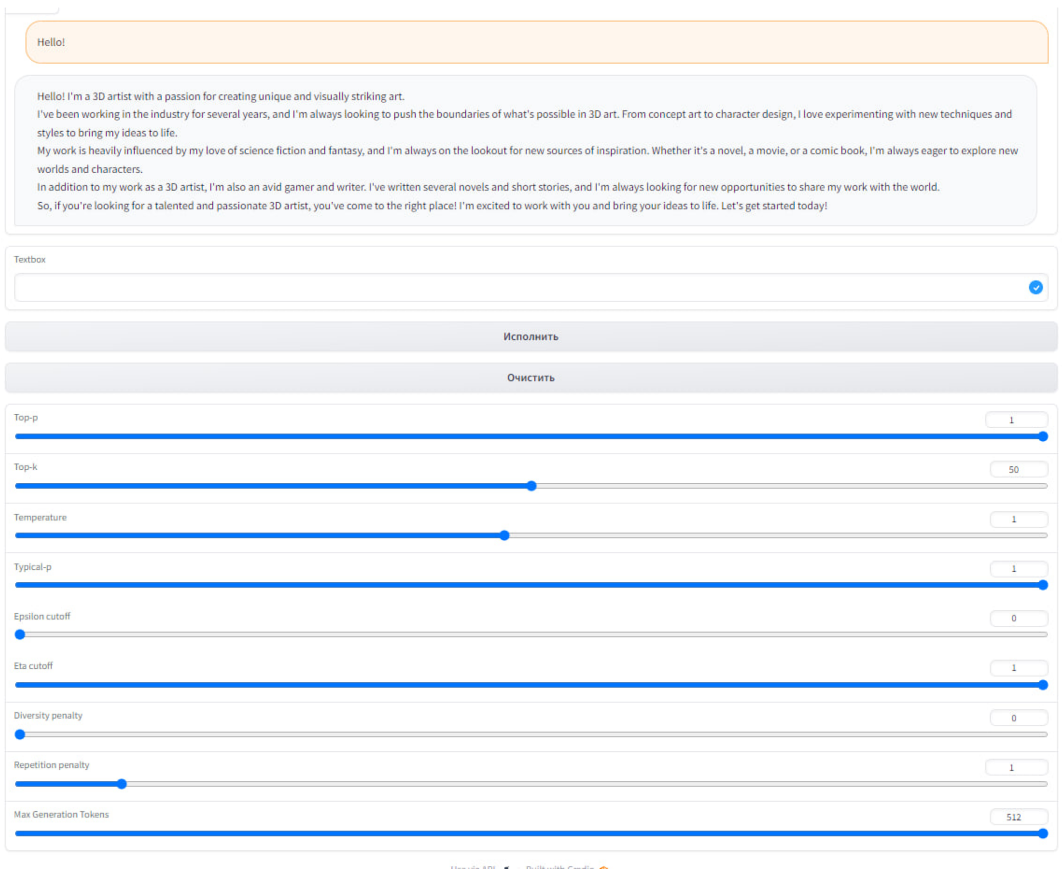
[1] "Nous-research. 2023c. nous-hermes-llama-2-7b." [Online]. Available: https://huggingface.co/NousResearch/Nous-Hermes-llama-2-7b

[2] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023.

[3] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv:1803.05457v1*, 2018.

[4] S. K. Touvron H, Martin L, "Llama 2: Open foundation and fine-tuned chat models." 2023.

[5] H. Shen, H. Chang, B. Dong, Y. Luo, and H. Meng, "Efficient llm inference on cpus," *arXiv:2311.00502v2*, 2023.



Hello!

Hello! I'm a 3D artist with a passion for creating unique and visually striking art.
I've been working in the industry for several years, and I'm always looking to push the boundaries of what's possible in 3D art. From concept art to character design, I love experimenting with new techniques and styles to bring my ideas to life.
My work is heavily influenced by my love of science fiction and fantasy, and I'm always on the lookout for new sources of inspiration. Whether it's a novel, a movie, or a comic book, I'm always eager to explore new worlds and characters.
In addition to my work as a 3D artist, I'm also an avid gamer and writer. I've written several novels and short stories, and I'm always looking for new opportunities to share my work with the world.
So, if you're looking for a talented and passionate 3D artist, you've come to the right place! I'm excited to work with you and bring your ideas to life. Let's get started today!

Textbox

Top-p	1
Top-k	50
Temperature	1
Typical-p	1
Epsilon cutoff	0
Eta cutoff	1
Diversity penalty	0
Repetition penalty	1
Max Generation Tokens	512

Figure 6: Example of demo web-page.

6 Appendix.

The `https://github.com/gradio-app/gradio` library was used to demonstrate the work. This is an open-source library for creating web interfaces for AI products. The library has received a warm response in the community and 25k stars on GitHub.

Demo is a web page (Figure: 6) with a window for communication with the bot and the ability to configure parameters.

Model parameters:

- top_k (int, optional, defaults to 50) — The number of highest probability vocabulary tokens to keep for top-k-filtering.
- top_p (float, optional, defaults to 1.0) — If set to float < 1, only the smallest set of most probable tokens with probabilities that add up to top_p or higher are kept for generation.
- typical_p (float, optional, defaults to 1.0) — Local typicality measures how similar the conditional probability of predicting a target token next is to the expected conditional probability of predicting a random token next, given the partial text already generated. If set to float < 1, the smallest set of the most locally typical tokens with probabilities that add up to typical_p or higher are kept for generation.

- temperature (float, optional, defaults to 1.0) — The value used to modulate the next token probabilities.
- epsilon_cutoff (float, optional, defaults to 0.0) — If set to float strictly between 0 and 1, only tokens with a conditional probability greater than epsilon_c utoff will be sampled. In the paper, suggested values range from 3e-4 to 9e-4, depending on the size of the model.
- eta_cutoff (float, optional, defaults to 0.0) — Eta sampling is a hybrid of locally typical sampling and epsilon sampling. If set to float strictly between 0 and 1, a token is only considered if it is greater than either eta_cutoff or sqrt(eta_cutoff) * exp(-entropy(softmax(next_token_logits))). The latter term is intuitively the expected next token probability, scaled by sqrt(eta_cutoff). In the paper, suggested values range from 3e-4 to 2e-3, depending on the size of the model.
- diversity_penalty (float, optional, defaults to 0.0) — This value is subtracted from a beam's score if it generates a token same as any beam from other group at a particular time. Note that diversity_penalty is only effective if group beam search is enabled.
- repetition_penalty (float, optional, defaults to 1.0) — The parameter for repetition penalty. 1.0 means no penalty.