The Journal of Open Source Software

# ReadmeReady: Free and Customizable Code Documentation with LLMs - A Fine-Tuning Approach

**Sayak Chakrabarty** [1] **and Souradip Pal** [2]

**1** Northwestern University **2** Purdue University

**DOI:** 10.21105/joss.07489

**Software**

- Review
- Repository
- Archive

**Editor:** Chris Vernon

**Reviewers:**

- @Manvi-Agrawal
- @camilochs

**Submitted:** 13 November 2024
**Published:** 12 April 2025

## Summary

Automated documentation of programming source code is a challenging task with significant practical and scientific implications for the developer community. ReadmeReady is a large language model (LLM)-based application that developers can use as a support tool to generate basic documentation for any publicly available or custom repository. Over the last decade, several research have been done on generating documentation for source code using neural network architectures. With the recent advancements in LLM technology, some open-source applications have been developed to address this problem. However, these applications typically rely on the OpenAI APIs, which incur substantial financial costs, particularly for large repositories. Moreover, none of these open-source applications offer a fine-tuned model or features to enable users to fine-tune custom LLMs. Additionally, finding suitable data for fine-tuning is often challenging. Our application addresses these issues.

## Statement of Need

The integration of natural and programming languages is a research area that addresses tasks such as automatic documentation of source code, code generation from natural language descriptions, and searching for code using natural language queries. These tasks are highly practical, as they can significantly enhance programmer efficiency, and they are scientifically intriguing due to their complexity and the proposed relationships between natural language, computation, and reasoning (Chomsky, 1956; Graves et al., 2014; Miller, 2003).

## State of the Field

Recently, large language models (LLMs) have become increasingly significant, demonstrating human-like abilities across various fields (Brown et al., 2020; Ouyang et al., 2022; Radford et al., 2019). LLMs typically employ transformer architecture variants and are trained on massive data volumes to detect patterns (Vaswani et al., 2017).

We present an LLM-based application that developers can use as a support tool to generate basic documentation for any code repository. Some open-source applications have been developed to address this issue, to name a few:

- **AutoDoc-ChatGPT** (Awekrx, 2023)
- **AutoDoc** (Labs, 2023)
- **Auto-GitHub-Docs-Generator** (Microsoft, 2023)

However, these applications suffer from two major issues. Firstly, all of them are built on top of the OpenAI APIs, requiring users to have an OpenAI API key and incurring a cost with each API request. Generating documentation for a large repository could result in costs reaching hundreds of dollars. Our application allows users to choose among OpenAI's GPT, Meta's

Llama2, and Google's Gemma models. Notably, apart from the first, the other models are open-source and incur no charges, allowing documentation to be generated for free.

Secondly, none of the existing open-source applications provide a fine-tuned model or an option for users to fine-tune. Our application offers a fine-tuning option using QLoRA (Dettmers et al., 2023), which can be trained on the user's own dataset. It is important to note that using this feature requires access to powerful GPU clusters. Some existing applications provide a command-line tool for interacting with the entire repository, allowing users to ask specific questions about the repository but not generating a README file.

## Methodology

The application prompts the user to enter the project's name, GitHub URL, and select the desired model from the following options:

- `gpt-3.5-turbo` (OpenAI, 2023a)
- `gpt-4` (OpenAI, 2023b)
- `gpt-4-32k` (OpenAI, 2023c)
- `TheBloke/Llama-2-7B-Chat-GPTQ` (quantized) (TheBloke, 2023b)
- `TheBloke/CodeLlama-7B-Instruct-GPTQ` (quantized) (TheBloke, 2023a)
- `meta-llama/Llama-2-7b-chat-hf` (Meta, 2023b)
- `meta-llama/CodeLlama-7b-Instruct-hf` (Meta, 2023a)
- `google/gemma-2b-it` (Google, 2023b)
- `google/codegemma-2b-it` (Google, 2023a)

For our experimentation and tests, we used 1 × NVIDIA Tesla V100 with 16GB of GPU memory which is ideal for running the application.

**Document Retrieval:** Our application indexes the codebase through a depth-first traversal of all repository contents and utilizes an LLM to generate documentation. All files are converted into text, tokenized, and then chunked, with each chunk containing 1000 tokens. The application employs the `sentence-transformers/all-mpnet-base-v2` (HuggingFace, 2023) sentence encoder to convert each chunk into a 768-dimensional embedding vector, which is stored in an in-memory vector store. When a query is provided, it is converted into a similar vector using the same sentence encoder. The neighbor nearest to the query embedding vector is searched using KNN (k=4) from the vector store, utilizing cosine similarity as the distance metric. For the KNN search, we use the HNSWLib library, which implements an approximate nearest-neighbor search based on hierarchical navigable small-world graphs (Malkov & Yashunin, 2018). This methodology provides the relevant sections of the source code, aiding in answering the prompted question. The entire methodology for Retrieval Augmented Generation (RAG) and fine-tuning is illustrated in Figure 1.

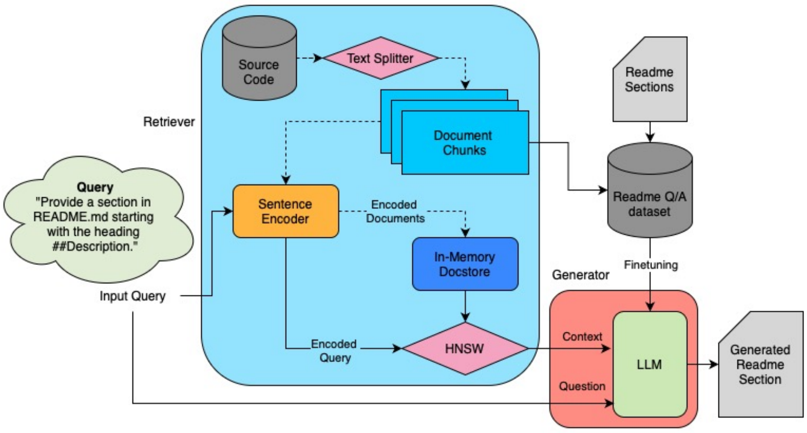

**Figure 1:** Input to Output Workflow showing the Retrieval and Generator modules. The retrieval module uses HNSW algorithm to create a context for the prompt to the Language model for text generation.

**Prompt Configuration:** Prompt engineering is accomplished using the Langchain API. For our purpose, a prompt template has been used. This template includes placeholders for questions, which users can edit and modify as needed. This flexibility allows the README to be generated according to the user's specific requirements. Our default README structure includes sections on description, requirements, installation, usage, contributing methods, and licensing, which align with standard documentation practices. The temperature for text generation is kept at the default value of 0.2. The current prompts are developer-focused and assume that the repository is code-centric.

## Fine Tuning

In our work, we fine-tune only one model, `TheBloke/Llama-2-7B-Chat-GPTQ` (TheBloke, 2023b), which is a 4-bit quantized model with 1.13 billion parameters. It supports a maximum sequence length of 4096 tokens and requires 3.9 GB of memory.

## Data Collection

We limit our scope to Python-based repositories; however, this approach is easily adaptable to multiple programming languages. A CSV file was created with three features: questions, context, and answers. Questions were derived from README file headings and subheadings, identified by markdown signatures # or ##. Answers correspond to the text under these headings.

The entire source code from the repositories is concatenated into a single string and separated into document chunks of 1000 tokens employing LangChain's text-splitter. Using the `sentence-transformers/all-mpnet-base-v2` (HuggingFace, 2023) sentence encoder, these chunks were converted into 768-dimensional vectors. Each question is then converted into a 768-dimensional vector and subjected to a KNN $(k = 4)$ search using HNSW (Malkov & Yashunin, 2018) to find the closest match from the entire set of document embeddings, stored as the context.

**Data Preprocessing:** Following the creation of the CSV file, we pre-process the data using regex patterns to clean the text. Since the context only captures source code, this eliminates the possibility of using offensive content. Regex is used to remove hashtags, email addresses, usernames, image URLs, and other personally identifiable information. Note that only repositories written entirely in English are used, with other languages filtered out. Prompt engineering in our source code ensures that the prompts are designed to avoid generating any personally identifiable data or offensive content.

# Experiments

We conducted the fine-tuning experiment on a small dataset consisting of randomly selected 190 README files, which may not address our default documentation questions. For each README, we examine its sections and subsections, frame relevant questions, and use the answers generated by our tool for training. For evaluation, we selected the rest of the 10 repositories and compared the original answers with the autogenerated documentation using BLEU and BERT scores to assess our model's performance.

## Before Fine-tuning

We conducted a series of experiments utilizing the `TheBloke/Llama-2-7B-Chat-GPTQ` model (TheBloke, 2023b) to demonstrate the functionality and efficacy of our proposed pipeline. The accompanying codebase is designed to be flexible, allowing the user to easily switch between different large language models (LLMs) by simply modifying the configuration file. Given the characteristics of LLMs, models with a greater number of parameters are generally expected to deliver enhanced performance.

## After Fine-tuning

We utilized the PEFT library from Hugging Face, which supports several Parameter Efficient Fine-Tuning (PEFT) methods. This approach is cost-effective for fine-tuning large language models (LLMs), particularly on lightweight hardware. The training configuration and hyperparameters are detailed in Table 1 and Table 2 respectively.

**Table 1: QLoRA Configuration**

| Parameter | Value |
|---|---|
| `r` | 2 |
| `lora_alpha` | 32 |
| `lora_dropout` | 0.05 |
| `bias` | None |
| `task_type` | CAUSAL_LM |

**Table 2: Training Hyper-parameters**

| Parameter | Value |
|---|---|
| `per_device_train_batch_size` | 1 |
| `gradient_accumulation_steps` | 1 |
| `num_train_epochs` | 3 |
| `learning_rate` | 1e-4 |
| `fp16` | True |
| `optim` | paged_adamw_8bit |
| `lr_scheduler_type` | cosine |
| `warmup_ratio` | 0.01 |

# References

Awekrx. (2023). *AutoDoc-ChatGPT*. https://github.com/awekrx/AutoDoc-ChatGPT.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., & others. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, *33*, 1877–1901. https://doi.org/10.48550/arXiv.2005.14165

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113–124. https://doi.org/10.1109/TIT.1956.1056813

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv Preprint arXiv:2305.14314*. https://doi.org/10.48550/arXiv.2305.14314

Google. (2023a). *Codegemma-2b-it*. https://github.com/google/codegemma-2b-it.

Google. (2023b). *Gemma-2b-it*. https://github.com/google/gemma-2b-it.

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural Turing Machines. *arXiv Preprint arXiv:1410.5401*. https://doi.org/10.48550/arXiv.1410.5401

HuggingFace. (2023). *Sentence transformers: All-mpnet-base-v2*. https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

Labs, C. (2023). *AutoDoc*. https://github.com/context-labs/autodoc.

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

Meta. (2023a). *CodeLlama-7b-instruct-hf*. https://github.com/meta-llama/CodeLlama-7b-Instruct-hf.

Meta. (2023b). *Llama-2-7b-chat-hf*. https://github.com/meta-llama/Llama-2-7b-chat-hf.

Microsoft. (2023). *Auto-GitHub-docs-generator*. https://github.com/microsoft/auto-github-docs-generator.

Miller, G. A. (2003). The cognitive revolution: a historical perspective. *Trends in Cognitive Sciences*, *7*(3), 141–144. https://doi.org/10.1016/S1364-6613(03)00029-9

OpenAI. (2023a). *Gpt-3.5-turbo*. https://github.com/openai/gpt-3.5-turbo.

OpenAI. (2023b). *Gpt-4*. https://github.com/openai/gpt-4.

OpenAI. (2023c). *Gpt-4-32k*. https://github.com/openai/gpt-4-32k.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., & others. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, *35*, 27730–27744. https://doi.org/10.48550/arXiv.2203.02155

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., & others. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, *1*(8), 9.

TheBloke. (2023a). *CodeLlama-7B-instruct-GPTQ*. https://github.com/TheBloke/CodeLlama-7B-Instruct-GPTQ.

TheBloke. (2023b). *Llama-2-7B-chat-GPTQ*. https://github.com/TheBloke/Llama-2-7B-Chat-GPTQ.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*, *30*. https://doi.org/10.48550/arXiv.1706.03762