# Отчет об оптимизации чат-бота LangChain/RAG

В существующей реализации чат-бота по документации платофрмы отмечены несколько проблем. Ниже приведены рекомендации по каждому пункту и дополнительные советы по улучшению.

## 1. Параметры разбиения документа (chunking)

* **Использовать семантическое разбиение по структуре Markdown.** Вместо простого разбиения по символам или словам рекомендуется применять специальные сплиттеры LangChain, учитывающие заголовки и структуру. Например, класс MarkdownHeaderTextSplitter группирует текст по секциям, основываясь на заголовках разного уровня[[1]](https://python.langchain.com/docs/how_to/markdown_header_metadata_splitter/#:~:text=As%20mentioned%2C%20chunking%20often%20aims,a%20specified%20set%20of%20headers).
* **Настройка размера и перекрытия.** Оптимальный размер чанка зависит от домена. Обычно используют диапазон ~500–1000 токенов с перекрытием 10–20%. Это обеспечивает сохранение связного контекста без избыточной информации[[2]](https://python.langchain.com/docs/concepts/text_splitters/#:~:text=Length)[[1]](https://python.langchain.com/docs/how_to/markdown_header_metadata_splitter/#:~:text=As%20mentioned%2C%20chunking%20often%20aims,a%20specified%20set%20of%20headers). Если тексты очень длинные, можно применять RecursiveCharacterTextSplitter, который пытается сохранить смысл целых абзацев и заголовков[[3]](https://python.langchain.com/docs/concepts/text_splitters/#:~:text=The%20most%20intuitive%20strategy%20is,based%20splitting).
* **Пример кода:**
* from langchain\_text\_splitters import MarkdownHeaderTextSplitter  
  splitter = MarkdownHeaderTextSplitter(headers\_to\_split\_on=[("#","H1"),("##","H2")], chunk\_size=1000, chunk\_overlap=200)  
  chunks = splitter.split\_text(markdown\_content)
* **Результат:** семантическое разбиение уменьшит «шум» в чанках (разрыв смысловых блоков), что улучшит качество поиска по векторам и понимание моделью контекста.

## 2. Модели эмбеддингов

* **Заменить all-MiniLM-L6-v2 на более качественные.** Этот небольшой SBERT (384-мерный) уступает более крупным моделям. Например, модель all-mpnet-base-v2 (768-мерная) дает более точные эмбеддинги (лучшая общая точность)[[4]](https://www.sbert.net/docs/sentence_transformer/pretrained_models.html#:~:text=The%20all,see%20all%20evaluated%20original%20models). Если нужна бесплатная локальная альтернатива, стоит обратить внимание на новые открытые эмбеддинг-модели высокого качества: Snowflake Arctic Embed (серия моделей от 22M до 334M параметров) или BGE (BaAI)[[5]](https://www.snowflake.com/en/blog/introducing-snowflake-arctic-embed-snowflakes-state-of-the-art-text-embedding-family-of-models/#:~:text=leading%20retrieval%20performance%2C%20give%20organizations,acquired%20last%20May%20via%20Neeva). Snowflake утверждает, что их Arctic Embed Large (334M) превосходит по качеству многие проприетарные эмбеддинги[[5]](https://www.snowflake.com/en/blog/introducing-snowflake-arctic-embed-snowflakes-state-of-the-art-text-embedding-family-of-models/#:~:text=leading%20retrieval%20performance%2C%20give%20organizations,acquired%20last%20May%20via%20Neeva).
* **Примеры из сравнений:** На рисунке ниже показано, что модель Snowflake Arctic Embed Large достигает лучших значений в MTEB-наборе задач (55–56 баллов) по сравнению с другими open-source и проприетарными эмбеддингами. Это говорит о её пригодности для RAG[[5]](https://www.snowflake.com/en/blog/introducing-snowflake-arctic-embed-snowflakes-state-of-the-art-text-embedding-family-of-models/#:~:text=leading%20retrieval%20performance%2C%20give%20organizations,acquired%20last%20May%20via%20Neeva).

*Рисунок: Сравнение качества эмбеддингов (метрика MTEB) различных моделей. Snowflake Arctic Embed (334M) показывает максимальные значения среди перечисленных (больше – лучше).*

*Рисунок: Сравнительная таблица параметров и результатов эмбеддингов. Arctic-Embed-L (334M) превосходит OpenAI и Cohere на WMT-16 пунктов (поле D) и превзошла предыдущий SoTA на 6.5% по MTEB*[*[5]*](https://www.snowflake.com/en/blog/introducing-snowflake-arctic-embed-snowflakes-state-of-the-art-text-embedding-family-of-models/#:~:text=leading%20retrieval%20performance%2C%20give%20organizations,acquired%20last%20May%20via%20Neeva)*.*

* **Реализация в коде:** например, вместо
* from langchain.embeddings import HuggingFaceEmbeddings  
  embeddings = HuggingFaceEmbeddings(model\_name="sentence-transformers/all-MiniLM-L6-v2")
* можно использовать
* embeddings = HuggingFaceEmbeddings(model\_name="sentence-transformers/all-mpnet-base-v2")
* или загрузить через Ollama: ollama start nomic-embed-text и указать этот эндпоинт.
* **Вывод:** более мощные эмбеддинги увеличат релевантность поиска: векторные представления документов будут точнее, что улучшит качество ответов RAG-системы.

## 3. Учет истории диалога (память)

* **Добавить хранение последних сообщений.** Текущая система игнорирует предыдущее общение. В LangChain для этого есть встроенные механизмы памяти. Например, класс ConversationBufferMemory сохраняет всю историю разговора и подставляет её в подсказку как {history}[[6]](https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/#:~:text=). При использовании return\_messages=True он возвращает историю в формате списка сообщений. Аналогично ConversationBufferWindowMemory держит только последние N сообщений. Использование этих классов позволит модели видеть предыдущие вопросы и ответы.
* **Пример кода:**
* from langchain.memory import ConversationBufferMemory  
  memory = ConversationBufferMemory(memory\_key="chat\_history", return\_messages=True)
* Затем при настройке цепочки можно передать эту память в ConversationChain или ConversationalRetrievalChain. Например:
* from langchain.chains import ConversationalRetrievalChain  
  qa\_chain = ConversationalRetrievalChain.from\_llm(  
   llm=llm, retriever=retriever, memory=memory)
* Эта цепочка автоматически объединит RAG с учётом истории.
* **Внешние решения:** в качестве альтернативы можно хранить историю в базе данных (Redis, SQLite) или отправлять каждый запрос в систему памяти (Memory API) для конденсации. Но в большинстве случаев встроенных классов LangChain достаточно.
* **Результат:** учёт истории поможет боту отвечать в контексте беседы (например, «в каком контексте был задан предыдущий вопрос?»), повысит когерентность ответов и облегчит решение уточняющих вопросов.

## 4. Ускорение генерации ответов

* **Модель и аппаратное ускорение.** Если текущее решение использует тяжёлую модель или API со значительной задержкой, есть несколько вариантов ускорения:
* **Локальные компактные модели:** на Mac M1 Ultra хорошо показывают себя лёгкие и оптимизированные модели. Например, Microsoft **Phi-2 (2.7B)** демонстрирует реальное “онлайн”-время отклика, генерируя ответы мгновенно[[7]](https://www.jeremymorgan.com/blog/generative-ai/how-to-llm-local-mac-m1/#:~:text=I%20found%20running%20the%20Microsoft,amounts%20of%20texts%20fairly%20quick). Модель **Mistral 7B** с квантованием (например, в формате llama.cpp Q6\_K) также показала высокую производительность и качество: в тестах на M1 Pro она превосходила Llama2-13B и генерировала ~18–47 токенов/с[[8]](https://medium.com/@mne/run-mistral-7b-model-on-macbook-m1-pro-with-16gb-ram-using-llama-cpp-44134694b773#:~:text=Share)[[9]](https://medium.com/@mne/run-mistral-7b-model-on-macbook-m1-pro-with-16gb-ram-using-llama-cpp-44134694b773#:~:text=1,69%20tokens%20per%20second).
* **Ограничение генерации:** при использовании OpenAI-интерфейсов можно выбрать более быстрые модели (например, GPT-3.5-turbo вместо GPT-4), или новые “Turbo”/Mini-версии. Так, GPT-4o-mini заметно быстрее и дешевле GPT-3.5-turbo, причем увеличенное окно контекста (128K) компенсирует меньшую точность[[10]](https://medium.com/@austin-starks/gpt-4-mini-vs-gpt-3-5-turbo-i-just-tried-out-the-new-model-and-am-beyond-impressed-e7a6abb54b1f#:~:text=Comparison%20between%20GPT,3.5%20Turbo). В целом, выбор модели должен отвечать компромиссу “скорость – качество”.
* **Квантование и оптимизация:** для локальных моделей используйте фреймворки типа llama.cpp или Hugging Face Accelerate с поддержкой Metal (MPS) для Apple. Квантованные веса (Q4/Q6) и пакетная генерация (n\_batch) могут значительно повысить пропускную способность.
* **Асинхронная генерация и стриминг:** в LangChain установить streaming=True у LLM для немедленной выдачи токенов, а также ограничить max\_tokens и добавить stop\_sequences, чтобы модель не генерировала больше необходимого.
* **Цитаты:** Например, в одном бенчмарке Mistral 7B на Mac M1 Pro с llama.cpp демонстрировал более 18 токенов/сек[[9]](https://medium.com/@mne/run-mistral-7b-model-on-macbook-m1-pro-with-16gb-ram-using-llama-cpp-44134694b773#:~:text=1,69%20tokens%20per%20second) (после настройки batch/pipeline скорость можно ещё повысить). Microsoft Phi-2 (3B) описывали как модель с «почти реальным временем диалога» на M1[[7]](https://www.jeremymorgan.com/blog/generative-ai/how-to-llm-local-mac-m1/#:~:text=I%20found%20running%20the%20Microsoft,amounts%20of%20texts%20fairly%20quick).
* **Результат:** Переход на оптимизированные модели и настройки позволит сократить время отклика бота. Это улучшит UX, особенно при диалогах в реальном времени.

## 5. Улучшение шаблона промпта и параметров LLM

* **Добавление системного сообщения (system prompt).** В чат-цепочке LangChain стоит задать системную инструкцию, определяющую стиль и цель бота. Например: «Вы — дружелюбный ассистент, отвечающий по вопросам взаимного кредитования. Отвечайте чётко и по существу». Это задаст тон и рамки ответа.
* **Четкость и лаконичность.** Нередко “много воды” возникает из-за непонятных инструкций. В промптах следует прямо указывать требуемый формат и объем ответа. Например, формулировать задачу как «В двух-трёх предложениях...» или «Используя приведённый контекст, ответьте кратко»[[11]](https://thetechjunction.net/chatgpt-prompts-how-to-you-use-them/#:~:text=,certain%20style%20or%20result%20in). Избегайте общих вопросов, добавляйте ограничения «не включайте личные мнения, только факты»[[12]](https://thetechjunction.net/chatgpt-prompts-how-to-you-use-them/#:~:text=,ChatGPT%20away%20from%20certain%20content). Таким образом модель будет ориентирована на краткость и релевантность.
* **Настройки генерации:**
* **Temperature:** Для получения точных и последовательных ответов рекомендуется понизить temperature (обычно 0–0.3). Низкая температура уменьшает случайность в выборе слов и делает текст более предсказуемым[[13]](https://www.ibm.com/think/topics/llm-temperature#:~:text=allows%20users%20to%20balance%20coherence,to%20their%20own%20desired%20outcome). Например, IBM отмечает, что «низкая температура предпочтительна для задач, требующих точности и достоверности, она делает ответы более осмысленными и избегает лишнего»[[13]](https://www.ibm.com/think/topics/llm-temperature#:~:text=allows%20users%20to%20balance%20coherence,to%20their%20own%20desired%20outcome).
* **Max tokens и stop-words:** Установите ограничение на максимальную длину ответа (max\_tokens), чтобы предотвратить излишнюю многословность, и, при необходимости, задайте stop\_sequence (например, «Перечень:», «Спасибо!») для завершения генерации по шаблону[[14]](https://www.ibm.com/think/topics/llm-temperature#:~:text=Maximum%20length%3A%20The%20maximum%20length,overly%20long%20or%20irrelevant%20responses).
* **Top-p/Top-k:** Строгие настройки (top\_p ≈ 0.8–1.0, top\_k ≈ 50–100) помогут избежать редких случайных отклонений, сохранив при этом гибкость речи.
* **Пример шаблона:**
* System: Вы — эксперт по кредитам. Отвечайте на вопросы клиента чётко и кратко.   
  User: {вопрос}  
  Assistant:
* Уточняющие описания можно вынести в PromptTemplate. Главное — всегда включать и **источник контекста** (документация из RAG), и **память истории**, и системные указания для формирования полных ответов.
* **Результат:** Правильно сформулированный промпт и параметры контролируют стиль ответов. Это обеспечит корректный тон общения, краткость, фокус на ключевых деталях и соответствие корпоративным стандартам общения.

## 6. Модели Ollama и варианты запуска (локально vs API)

* **Ollama – локальный сервер моделей:** Через Ollama можно запускать множество современных LLM. Например, доступны семейства **Llama 3** (8B, 70B, 405B)[[15]](https://ollama.com/library#:~:text=Updated%201%20month%20ago%20,3), **Mistral** (7B, а также **Mistral-nemo** 12B с длинным контекстом 128K)[[16]](https://ollama.com/library#:~:text=,scale)[[17]](https://huggingface.co/mistralai/Mistral-7B-v0.1/discussions/4#:~:text=edited%20Sep%2027%2C%202023), **Phi-2** (2.7B) и **Phi-4** (14B) от Microsoft, **Gemma** (2B, 9B, 27B) от Google[[18]](https://ollama.com/library#:~:text=understanding,Tags%20Updated%2011%20months%20ago), **WizardLM2** (7B и 8×22B)[[19]](https://ollama.com/library#:~:text=Tags%20Updated%201%20year%20ago,Tags%20Updated%201%20year%20ago), а также специализированные модели как **Dolphin** (8B, 70B) для кода. Среди эмбеддингов Ollama поддерживает **nomic-embed-text** (32.4M параметров)[[20]](https://ollama.com/library#:~:text=%2A%20%20nomic,3), **mxbai-embed-large** (335M)[[21]](https://ollama.com/library#:~:text=%2A%20%20mxbai,25), **Snowflake Arctic Embed** (22–335M)[[22]](https://ollama.com/library#:~:text=%2A%20%20snowflake,47), **BGE-M3** (567M)[[23]](https://ollama.com/library#:~:text=%2A%20%20bge,Tags%20Updated%2011%20months%20ago) и др.
* **Запуск на Mac M1 Ultra:** На мощных Mac лучше всего подойдут модели малого и среднего размера с поддержкой Metal (MPS). Например, **Phi-2 (2.7B)** и **Phi-3.5 (3.8B)** отлично работают на CPU/Metal, как показано в обзорах[[7]](https://www.jeremymorgan.com/blog/generative-ai/how-to-llm-local-mac-m1/#:~:text=I%20found%20running%20the%20Microsoft,amounts%20of%20texts%20fairly%20quick). **Mistral 7B** и **Llama 2 (7B, 13B)** можно запускать локально с приемлемой скоростью на M1. Большие модели (от 20B) потребуют много памяти, но при необходимости Ollama может сворачивать их в Web-инфраструктуру для MPS.
* **Через API:** Если подходит облачная интеграция, рассмотрите **OpenAI GPT-4o/GPT-4Turbo** (номинально дороже, но быстрые) или **Anthropic Claude**. Стоимость использования API следует сбалансировать со скоростью и качеством. Например, новая **GPT-4o mini** показывает лучшие результаты и в три раза больший контекст (128K) по сравнению с GPT-3.5-turbo[[10]](https://medium.com/@austin-starks/gpt-4-mini-vs-gpt-3-5-turbo-i-just-tried-out-the-new-model-and-am-beyond-impressed-e7a6abb54b1f#:~:text=Comparison%20between%20GPT,3.5%20Turbo).
* **Таблица сравнения моделей:** Ниже приведён пример сравнения некоторых популярных моделей для генерации (число параметров, максимальный контекст и особенности):

| Модель | Параметры | Окно контекста | Запуск | Особенности |
| --- | --- | --- | --- | --- |
| **GPT-4o-mini** | – | 128K[[10]](https://medium.com/@austin-starks/gpt-4-mini-vs-gpt-3-5-turbo-i-just-tried-out-the-new-model-and-am-beyond-impressed-e7a6abb54b1f#:~:text=Comparison%20between%20GPT,3.5%20Turbo) | API (OpenAI) | Очень широкий контекст, высокая стоимость после бесплатного лимита |
| **GPT-3.5-turbo** | – | 16K | API (OpenAI) | Дешёвая, но меньше контекста |
| **Mistral 7B** | 7B[[8]](https://medium.com/@mne/run-mistral-7b-model-on-macbook-m1-pro-with-16gb-ram-using-llama-cpp-44134694b773#:~:text=Share) | 8K[[17]](https://huggingface.co/mistralai/Mistral-7B-v0.1/discussions/4#:~:text=edited%20Sep%2027%2C%202023) | локально (M1) | Высокое качество, превосходит Llama2-13B[[8]](https://medium.com/@mne/run-mistral-7b-model-on-macbook-m1-pro-with-16gb-ram-using-llama-cpp-44134694b773#:~:text=Share) |
| **Phi-2 (Microsoft)** | 2.7B | 8K | локально (M1) | Очень быстрая (реальное время диалога)[[7]](https://www.jeremymorgan.com/blog/generative-ai/how-to-llm-local-mac-m1/#:~:text=I%20found%20running%20the%20Microsoft,amounts%20of%20texts%20fairly%20quick) |
| **Llama 2 7B/13B** | 7B/13B | 4K | локально (M1) | Открытая, хорошая поддержка, подходит для RAG |
| **WizardLM-7B** | 7B | 4K | локально | Инструкт-тюнED, от Microsoft |
| **Gemma-2B/9B** | 2B/9B | 4K | локально | От Google, эффективны при малой нагрузке |
| **StarCoder2** | 3B/7B/15B | 4K | локально | Оптимизированы для кода и тех. документов |

(Прим. «–» означает проприетарную модель без точной информации о параметрах.)

## 7. Прочие возможные улучшения

* **Векторное хранилище (VectorStore):** Убедитесь, что хранилище эмбеддингов выбрано оптимально. Для больших документов рекомендованы быстрые движки (Faiss, Qdrant, Chroma). При необходимости используйте гибридный поиск (dense + sparse) или кеширование в памяти для ускорения.
* **Обработка Markdown:** Если документация содержит код или специальные блоки, стоит очистить лишние маркеры. Можно убрать YAML-метаданные и кодовые блоки перед эмбеддингом, чтобы не разрежать информацию.
* **Отладка и метрики:** Добавьте логирование запросов и метрик (время, длина, токены) для анализа узких мест. Это поможет выявить узкие места (например, слишком длинные запросы к OpenAI).
* **Контроль качества:** Для уменьшения «галлюцинаций» можно внедрить пост-фильтр фактов (сверка ответов с базой знаний) или мультимодельную проверку.

## 8. Конкретные доработки к коду

Ниже перечислены ключевые шаги по доработке текущей реализации:

* **Переход на лучший сплиттер:** заменить текущий TextSplitter на MarkdownHeaderTextSplitter или RecursiveCharacterTextSplitter с параметрами chunk\_size≈500–1000, chunk\_overlap≈100–200.
* **Обновление эмбеддингов:** подключить новую модель, например:
* from langchain.embeddings import HuggingFaceEmbeddings  
  embeddings = HuggingFaceEmbeddings(model\_name="sentence-transformers/all-mpnet-base-v2")
* или использовать Ollama-сервер: embed = OllamaEmbeddings(model="snowflake-arctic-embed-l").
* **Включение истории:** добавить memory = ConversationBufferMemory(return\_messages=True) и передавать его в цепочки (ConversationChain или ConversationalRetrievalChain). Обеспечить хранение последних N сообщений и их включение в контекст.
* **Смена LLM:** в конструкторе цепочек LangChain заменить текущую LLM на более быструю/качественную (например, ChatOpenAI(model="gpt-3.5-turbo", temperature=0) или Ollama(model="mistral")). Настроить temperature=0–0.3, max\_tokens≈150–300 и, при возможности, streaming=True.
* **Системная инструкция:** при формировании ChatPromptTemplate добавить явный системный блок (напр. SystemMessagePromptTemplate) с ролью и стилем общения.
* **Оптимизация RAG:** если документы большие, использовать индексацию (FAISS/Pinecone) с обновленным векторным хранилищем. Перепроверить параметры поиска (Top-k) для баланса релевантности и скорости.
* **Дополнительная фильтрация:** ограничить пост-обработку ответа (например, убрать шаблонные приветствия, если не нужны) или добавить шаг проверки по ключевым словам.

Каждую из вышеперечисленных доработок следует реализовать и протестировать по отдельности, оценивая качество ответов и скорость. Также рекомендуется покрыть изменения юнит-тестами и провести A/B тестирование нового поведения бота.

**Вывод:** выполнение этих рекомендаций позволит добиться более быстрых, точных и «цельных» ответов от чат-бота. Качественные эмбеддинги и учёт истории улучшат релевантность, а оптимизированный промпт и выбор моделей – ясность и скорость общения.

**Источники:** использованы официальные руководства LangChain и обзоры LLM/эмбеддингов[[2]](https://python.langchain.com/docs/concepts/text_splitters/#:~:text=Length)[[1]](https://python.langchain.com/docs/how_to/markdown_header_metadata_splitter/#:~:text=As%20mentioned%2C%20chunking%20often%20aims,a%20specified%20set%20of%20headers)[[4]](https://www.sbert.net/docs/sentence_transformer/pretrained_models.html#:~:text=The%20all,see%20all%20evaluated%20original%20models)[[5]](https://www.snowflake.com/en/blog/introducing-snowflake-arctic-embed-snowflakes-state-of-the-art-text-embedding-family-of-models/#:~:text=leading%20retrieval%20performance%2C%20give%20organizations,acquired%20last%20May%20via%20Neeva)[[7]](https://www.jeremymorgan.com/blog/generative-ai/how-to-llm-local-mac-m1/#:~:text=I%20found%20running%20the%20Microsoft,amounts%20of%20texts%20fairly%20quick)[[8]](https://medium.com/@mne/run-mistral-7b-model-on-macbook-m1-pro-with-16gb-ram-using-llama-cpp-44134694b773#:~:text=Share)[[6]](https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/#:~:text=)[[11]](https://thetechjunction.net/chatgpt-prompts-how-to-you-use-them/#:~:text=,certain%20style%20or%20result%20in)[[13]](https://www.ibm.com/think/topics/llm-temperature#:~:text=allows%20users%20to%20balance%20coherence,to%20their%20own%20desired%20outcome)[[17]](https://huggingface.co/mistralai/Mistral-7B-v0.1/discussions/4#:~:text=edited%20Sep%2027%2C%202023)[[10]](https://medium.com/@austin-starks/gpt-4-mini-vs-gpt-3-5-turbo-i-just-tried-out-the-new-model-and-am-beyond-impressed-e7a6abb54b1f#:~:text=Comparison%20between%20GPT,3.5%20Turbo).

[[1]](https://python.langchain.com/docs/how_to/markdown_header_metadata_splitter/" \l ":~:text=As%20mentioned%2C%20chunking%20often%20aims,a%20specified%20set%20of%20headers) How to split Markdown by Headers | ️ LangChain

<https://python.langchain.com/docs/how_to/markdown_header_metadata_splitter/>

[[2]](https://python.langchain.com/docs/concepts/text_splitters/#:~:text=Length) [[3]](https://python.langchain.com/docs/concepts/text_splitters/#:~:text=The%20most%20intuitive%20strategy%20is,based%20splitting) Text splitters | ️ LangChain

<https://python.langchain.com/docs/concepts/text_splitters/>

[[4]](https://www.sbert.net/docs/sentence_transformer/pretrained_models.html#:~:text=The%20all,see%20all%20evaluated%20original%20models) Pretrained Models — Sentence Transformers documentation

<https://www.sbert.net/docs/sentence_transformer/pretrained_models.html>

[[5]](https://www.snowflake.com/en/blog/introducing-snowflake-arctic-embed-snowflakes-state-of-the-art-text-embedding-family-of-models/#:~:text=leading%20retrieval%20performance%2C%20give%20organizations,acquired%20last%20May%20via%20Neeva) Snowflake launches text-embedding model for retrieval use cases

<https://www.snowflake.com/en/blog/introducing-snowflake-arctic-embed-snowflakes-state-of-the-art-text-embedding-family-of-models/>

[[6]](https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/#:~:text=) Conversational Memory for LLMs with Langchain | Pinecone

<https://www.pinecone.io/learn/series/langchain/langchain-conversational-memory/>

[[7]](https://www.jeremymorgan.com/blog/generative-ai/how-to-llm-local-mac-m1/#:~:text=I%20found%20running%20the%20Microsoft,amounts%20of%20texts%20fairly%20quick) How to Run LLM Locally Mac M1

<https://www.jeremymorgan.com/blog/generative-ai/how-to-llm-local-mac-m1/>

[[8]](https://medium.com/@mne/run-mistral-7b-model-on-macbook-m1-pro-with-16gb-ram-using-llama-cpp-44134694b773#:~:text=Share) [[9]](https://medium.com/@mne/run-mistral-7b-model-on-macbook-m1-pro-with-16gb-ram-using-llama-cpp-44134694b773#:~:text=1,69%20tokens%20per%20second) Run Mistral 7B Model on MacBook M1 Pro with 16GB RAM using llama.cpp | by Gregory Zem | Medium

<https://medium.com/@mne/run-mistral-7b-model-on-macbook-m1-pro-with-16gb-ram-using-llama-cpp-44134694b773>

[[10]](https://medium.com/@austin-starks/gpt-4-mini-vs-gpt-3-5-turbo-i-just-tried-out-the-new-model-and-am-beyond-impressed-e7a6abb54b1f#:~:text=Comparison%20between%20GPT,3.5%20Turbo) GPT-4o mini vs GPT-3.5 Turbo. I just tried out the new model and am BEYOND Impressed | by Austin Starks | Medium

<https://medium.com/@austin-starks/gpt-4-mini-vs-gpt-3-5-turbo-i-just-tried-out-the-new-model-and-am-beyond-impressed-e7a6abb54b1f>

[[11]](https://thetechjunction.net/chatgpt-prompts-how-to-you-use-them/#:~:text=,certain%20style%20or%20result%20in) [[12]](https://thetechjunction.net/chatgpt-prompts-how-to-you-use-them/#:~:text=,ChatGPT%20away%20from%20certain%20content) ChatGPT Prompts: How Can You Use Them In 2025

<https://thetechjunction.net/chatgpt-prompts-how-to-you-use-them/>

[[13]](https://www.ibm.com/think/topics/llm-temperature#:~:text=allows%20users%20to%20balance%20coherence,to%20their%20own%20desired%20outcome) [[14]](https://www.ibm.com/think/topics/llm-temperature#:~:text=Maximum%20length%3A%20The%20maximum%20length,overly%20long%20or%20irrelevant%20responses) What is LLM Temperature? | IBM

<https://www.ibm.com/think/topics/llm-temperature>

[[15]](https://ollama.com/library#:~:text=Updated%201%20month%20ago%20,3) [[16]](https://ollama.com/library#:~:text=,scale) [[18]](https://ollama.com/library#:~:text=understanding,Tags%20Updated%2011%20months%20ago) [[19]](https://ollama.com/library#:~:text=Tags%20Updated%201%20year%20ago,Tags%20Updated%201%20year%20ago) [[20]](https://ollama.com/library#:~:text=%2A%20%20nomic,3) [[21]](https://ollama.com/library#:~:text=%2A%20%20mxbai,25) [[22]](https://ollama.com/library#:~:text=%2A%20%20snowflake,47) [[23]](https://ollama.com/library#:~:text=%2A%20%20bge,Tags%20Updated%2011%20months%20ago) library

<https://ollama.com/library>

[[17]](https://huggingface.co/mistralai/Mistral-7B-v0.1/discussions/4#:~:text=edited%20Sep%2027%2C%202023) mistralai/Mistral-7B-v0.1 · context window size

<https://huggingface.co/mistralai/Mistral-7B-v0.1/discussions/4>