llama_cpp_cached_prompts

Программа для организации кэширования прелюдий текстовых запросов к llama_cpp. Также помогает частично автоматизировать процесс разработки прелюдий.

Установка

Тестировалась на версии Python 3.10.12. Используемые Python пакеты:

- llama_cpp_python (тестировалось на версии 0.1.78)
- numpy

Для компиляции llama_cpp при установке llama_cpp_python с поддержкой OpenBLAS (немного ускоряет обработку входной части запроса) на Linux необходима цепочка операций (см. скрипт setup/install-llama-cpp-python.sh):

```
sudo apt-get install libopenblas-dev
CMAKE_ARGS="-DLLAMA_BLAS=ON -DLLAMA_BLAS_VENDOR=OpenBLAS" FORCE_CMAKE=1 pip install --force-reinstall l
```

Установка с OpenBLAS на Windows описана на сайте.

Работа программы на Windows не проверялась – возможно, могут возникнуть сложности с unicode буквами в путях при записи/загрузке кэша запросов.

Использование

Программа консольная, запускаемый файл: main.py. Всегда работает с файлам / директориями относительно запускаемого скрипта (в начале работы программы текущая папка меняется на корневую папку программы). В скрипте присутствует описание интерфейса через --full-help.

Основное предназначение программы – кэширование текстовых (.txt) прелюдий запросов и организация запуска новых запросов добавлением к кэшированным прелюдиям. Это позволяет в несколько раз увеличить скорость генерации результатов запросов в том случае, когда у большого числа запросов существует общая начальная часть.

Входные данные программы размещены по 3м папкам:

- prelude_drafts – не доконца готовые прелюдии, для которых предполагается проводить массовую генерацию ответов LLM (бота), чтобы из них выбрать самый для нас подходящий вариант. Начальную начать мы записали, но не знаем, какой ответ бота к нему записать в качестве примера для последующих запросов. Есть предположение, что в качестве примера вывода бота в прелюдии желательно брать текст, близкий к тому, что сам бот может сгенерировать. Поэтому предлагается для заготовок прелюдий пробовать генерировать большое количество ответов бота, из них выбрать самый подходящий к желаемому нами и немного доредактировать до желаемого формата / идеала. Наверное, если содержание примера ответа бота не сильно будет отходить от его “естественных” возможностей, то он сможет “увидеть” желаемую нами логику ответа на запрос, и “разумнее” копировать её в последующих наших запросах. Генерированные ответы ботов складываются в папку prelude_drafts_outputs.
- preludes – готовые прелюдии, которые мы предполагаем дальше использовать для реальных запросов.
- test_prompts – тестовые запросы, началами которых станут прелюдии из preludes. Предполагалось, что нужно для тестирования качества ответа ботов для имеющихся прелюдий. Формат названий файлов <название прелюдии без .txt части>---<обозначение запроса>.txt. При желании, можно использовать и как механизм получения ответов для реальных запросов (особенно, если хочется массово получить по несколько вариантов ответа). Генерированные ответы ботов складываются в папку test_prompts_outputs.

Кэш складывается в папку llm_cache. Для модели saiga2-13b-ggml-model-q4_1.bin размер кэша на прелюдию размера в 700 токенов примерно равняется 600mb.

Краткое описание основных команд / режимов:

- precache-preludes – прекэширование всех прелюдий из preludes для дальнейшего использования. Вручную запускать не обязательно, только если хочется сэкономить время в будущем.

- `outputs-for-drafts` – генерация ответов на все пред-прелюдии из `prelude_drafts`. По-умолчанию генерируется по 5 ответов.
- `outputs-test-prompts` – генерация ответов на все запросы из `test_prompts` (и соответствующие прелюдии из `preludes`). По-умолчанию генерируется по 5 ответов.
- `show-stats-on-tokens-number` – показать количество токенов во всех запросах и прелюдиях из `prelude_drafts`, `preludes`, `test_prompts`, `prelude_drafts_outputs`, `test_prompts_outputs`.
- `stdio-json-repl` – включить REPL-режим с обменом сообщений в виде JSON через stdin / stdout. Например, отправка в stdin команды вида

```
{command: "single_prompt", arguments: {prelude_id: "test_prelude", prelude: "user\n", prompt: "2+2=\nbo
```

выдаст в stdout ответ вида

```
{"status":"Ok","value":"4","message":""}
```

См. `lib/repl_mode.py` за списком и описанием текущих команд.

Все команды производят кэширование / генерацию файлов, только если выход старее входных файлов. Соответственно, все прелюдии можно спокойно менять и перезапускать программу – обновляться будут только новые. При желании есть ключ `--force-outputs` – генерировать ответы, даже если уже существуют достаточно новые ответы.

Модели по-умолючанию находятся в `models`, но это по желанию.

Пример использования

1. В `prelude_drafts` создаём файл `РПД-критерии_компетенций.txt` с содержанием

```
user
Название дисциплины:
"Байесовские и иерархические модели"

Содержание дисциплины:
"Основы байесовского статистического вывода. Базовые понятия статистичкского вывода: задача, модель, пр
связность графов и условная независимость в байесовских сетях. Нахождение наиболее вероятного исхода дл

Исходя из названия и содержания дисциплины, запиши список из 6 пунктов с фразами о том, что студент дол

bot
```

2. Запускаем `./main.py outputs-for-drafts`, в `prelude_drafts_outputs` появятся файлы `РПД-критерии_компетен 0.txt`, `РПД-критерии_компетенций.output-1.txt`, `РПД-критерии_компетенций.output-2.txt`, `РПД-критерии_компетенций.output-3.txt`, `РПД-критерии_компетенций.output-4.txt`. Выбираем, какой ответ бота нам понравился больше. Например:

```
1. Использование математических принципов в статистическом анализе.
2. Базовые приемы и методы байесовского статистического вывода.
3. Изучение основ программирования и информационных технологий для верификации и обработки данных в ста
4. Оценка вероятностей событий и моделей с использованием математических принципов.
5. Применение алгоритмов графа и байесовских сетей в статистическом анализе.
6. Создание и использование марковских сетей для решения задач статистического анализа.
```

3. Добавляем в `preludes` файл `РПД-критерии_компетенций.txt` с содержимым

```
user
Название дисциплины:
"Байесовские и иерархические модели"

Содержание дисциплины:
"Основы байесовского статистического вывода. Базовые понятия статистичкского вывода: задача, модель, пр
связность графов и условная независимость в байесовских сетях. Нахождение наиболее вероятного исхода дл

Исходя из названия и содержания дисциплины, запиши список из 6 пунктов с фразами о том, что студент дол

bot
1. Использование математических принципов в статистическом анализе.
```

```
2. Базовые приемы и методы байесовского статистического вывода.
3. Изучение основ программирования и информационных технологий для верификации и обработки данных в ста
4. Оценка вероятностей событий и моделей с использованием математических принципов.
5. Применение алгоритмов графа и байесовских сетей в статистическом анализе.
6. Создание и использование марковских сетей для решения задач статистического анализа.

user
```

Теперь все запросы с этой прелюдией будут начинаться с этих строк. Есть надежда, что это позволит боту опираться на приведённый нам пример, чтобы в ходе полуслучайного выбора токенов при генерации выбрать ту ветку, которая будет похожа на наш пример – а значит, если повезёт, даст желанный нами результат.

4. Добавляем в test_prompts несколько тестовых запросов, чтобы посмотреть, как бот справится. Например, добавим файл РПД-критерии_компетенций---Большие-данные--ОПК-3.txt:

```
Название дисциплины:
"Большие данные"

Содержание дисциплины:
"Введение. Линейная регрессия. Классификация данных. Нейронные сети. Введение. Исторический обзор теори
средних. Метрики. Деревья решений (Classification and Regression Trees). Наивный Байес. Метод опорных в
Learn. Инжиниринг данных. Упрощение функции. Регуляризация.  Проблема переобучения. Введение в нейронны

Исходя из названия и содержания дисциплины, запиши список из 6 пунктов с фразами о том, чем студент дол

bot
```

То, что в названии стоит РПД-критерии_компетенций до --- означает, что запрос будет приписывать к прелюдии из файла РПД-критерии_компетенций.txt.

5. Запускаем ./main.py outputs-test-prompts. В test_prompts_outputs появится, в частности, файл РПД-критерии_компетенций---Большие-данные--ОПК-3.output-3.txt:

```
1. Основы теории машинного обучения с примерами решения задач и описанием алгоритмов.
2. Обработка данных и использование научных пакетов Python для анализа больших объемов информации.
3. Анализ и классификация данных с помощью различных алгоритмов машинного обучения.
4. Разработка моделей, оптимизация параметров и выбор подходящей методики для решения задач.
5. Планирование и осуществление научных экспериментов с использованием больших данных.
6. Предоставление общего видения на темы теории машинного обучения, применение в научных исследованиях
```

Если результат (все результаты) нравятся – то прелюдия годится.

6. Запускаем ./main.py show-stats-on-tokens-number, выводится (в частности):

```
Context length: 4096

# Repository "prelude_drafts"

"prelude_drafts/РПД-критерии_компетенций.txt" tokens: 543
* with its output "prelude_drafts_outputs/РПД-критерии_компетенций.output-3.txt": 133 + 543 = 676
* with its output "prelude_drafts_outputs/РПД-критерии_компетенций.output-2.txt": 137 + 543 = 680
* with its output "prelude_drafts_outputs/РПД-критерии_компетенций.output-0.txt": 210 + 543 = 753
* with its output "prelude_drafts_outputs/РПД-критерии_компетенций.output-1.txt": 174 + 543 = 717
* with its output "prelude_drafts_outputs/РПД-критерии_компетенций.output-4.txt": 80 + 543 = 623

# Repository "preludes"

"preludes/РПД-критерии_компетенций.txt" tokens: 708

# Repository "test_prompts"

"test_prompts/РПД-критерии_компетенций---Большие-данные--ОПК-3.txt" tokens: 616
- with its prelude "preludes/РПД-критерии_компетенций.txt": 708 + 616 = 1324
  * with its output "test_prompts_outputs/РПД-критерии_компетенций---Большие-данные--ОПК-3.output-
3.txt": 185 + 708 + 616 = 1509
```

```
  * with its output "test_prompts_outputs/РПД-критерии_компетенций---Большие-данные--ОПК-3.output-
1.txt": 182 + 708 + 616 = 1506
  * with its output "test_prompts_outputs/РПД-критерии_компетенций---Большие-данные--ОПК-3.output-
4.txt": 180 + 708 + 616 = 1504
  * with its output "test_prompts_outputs/РПД-критерии_компетенций---Большие-данные--ОПК-3.output-
2.txt": 148 + 708 + 616 = 1472
  * with its output "test_prompts_outputs/РПД-критерии_компетенций---Большие-данные--ОПК-3.output-
0.txt": 117 + 708 + 616 = 1441
```

По этим цифрам можно оценить, насколько ещё можно увеличить прелюдии и/или запросы, чтобы не выходить (или выходить не слишком сильно) за пределы длины контекста модели (Context length).

Внешние прелюдии с режимом stdio-json-repl

Возможно задавать прелюдии с их идентификаторами сразу вместе с сообщением — тогда прелюдии будут автоматически кэшироваться и сохраняться в директорию с кэшом под заданной идентификацией. Например, команда

```
cat t1 | ../llama_cpp_cached_prompts/main.py stdio-json-repl
```

с содержимым файла `t1`

```
{command: "single_prompt", arguments: {prelude_id: "test_prelude", prelude: "user\n", prompt: "2+2=\nbo
{command: "single_prompt", arguments: {prelude_id: "test_prelude", prelude: "user\n", prompt: "2+1=\nbo
```

будет использовать общий кэш `test_prelude` для обоих запросов, и эта прелюдия не должна храниться где-то в структуре папок программы `llama_cpp_cached_prompts` или как-то дополнительно регистрироваться.

Замечание из документации llama_cpp_python

```
for Llama2 70b must set the n_gqa parameter (grouped-query attention factor) to 8 when loading
```