Построение языковой модели для бронирования отелей/авиабилетов

Голованова А.А.

Май 2024

Аннотация

Данный проект посвящен Fine-tune LLaMA 2 для решения задачи авторегрессионной генерации текста, с целью помощи клиентам в заказе билетов на самолет и бронирования отелей. Для достижения поставленной цели, было выбрано несколько датасетов покрывающих потребности задачи, после этого датасеты были объединены с предварительной обработкой данных для обеспечения их совместимости и улучшения качества. Для работы с полученными данными была выбрана предобученная модель Llama-7b, известная своей эффективностью в решении подобных задач. Для того чтобы адаптировать данную модель к нашей конкретной задаче, к ней был добавлен LoRA адаптер. В работе также рассматриваются теоретически альтернативные подходы с использованием, как классических, так и рекуррентных (нейросетевых) подходов.

1. Введение

На сегодняшний день алгоритмы машинного обучения, в частности нейронные сети, активно применяются практически в любых компаниях для решения широкого спектра задач, начиная с задач по обработке естественного языка (виртуальные ассистенты, определение намерений пользователей, суммаризация записей онлайн встреч) и заканчивая генерацией изображений по их описанию для приложений доставки. Особое положение, при этом, занимают глубокие модели нейронных сетей, которые на текущий момент являются лучшими решениями (SOTA; state of the art) на многих задачах в совершенно разных областях. Частично, успех нейронных сетей обусловлен несколькими факторам, такими как доступность и объемы данных, технологические возможности, востребованность и актуальность, а также финансирование.

Особое место при этом занимает направление обработки естественного языка (Natural Language Processing, NLP), которое буквально изменило парадигму решения DL задач после публикации архитектуры Transformer. Благодаря данной архитектуре, на текущий момент, мы имеем GPT и GPT-подобные модели, которые постепенно приближают нас к созданию AGI.

2. Используемые методы

Формально задачу языкового моделирования можно описать следующим образом: дана последовательность токенов $x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(t)}$ и необходимо посчитать вероятностное распределение следующего токена $x^{(t+1)}$:

$$P\left(x^{(t+1)} \middle| x^t, x^{t-1}, \dots, x^1\right)$$

, где $x^{(t+1)}$ – может быть любым токеном (словом, символом, subword) из доступного словаря $V = \{w_1, w_2, \dots, w_V\ \}$

2.1. Статистический подход

Один из самых простых подходов для создания языковой модели можно рассматривать в контексте генерации N-грам (униграм, биграм, и так далее) на основе их частотности из некоторого большого корпуса текстов. Данный подход легко реализуем, быстр, но его контекст ограничен N-граммами, что сводит на нет его практическую значимость в современных задачах. Также стоит отметить, что при данном подходе речи про семантику не идет. Огромной проблемой также являются OOV (out of vocabulary) N-граммы. Например,

$$P(w^{t+1}|w^t, w^{t-1}) = \frac{count(w^{t-1}, w^t, w^{t+1})}{count(w^{t-1}, w^t)}$$

где count – функция частоты наблюдаемой последовательности.

2.2. Рекуррентные нейронные сети

В качестве альтернативного решения задачи языкового моделирования можно рассматривать использование рекуррентных нейронных сетей

Рекуррентные нейронные сети – специальная разновидность нейронных сетей, предназначенная для обработки последовательностей. Данный вид сетей прекрасно зарекомендовал себя при работе с объектами сложной природы, которые не всегда описываются вектором фиксированной длины. Способность таких «строительных блоков» обрабатывать объекты произвольной длины прекрасно сочетались с текстовыми данными и долгое время, де-факто, практически любая NLP задача сводилась к построению N-стаканных LSTM слоев (если в задаче отсутствовала генеративная составляющая, то модель ещё была двунаправленной).

По своей природе, рекуррентные сети способны вычислять скрытое состояние (hidden state) в каждый момент времени t и передавать его в качестве дополнительного входа при расчёте следующего элемента

последовательности в момент времени t+1. Тем самым, выполняя функцию передачи контекста, которая также бы учитывала порядок слов в последовательности.

К основным недостатком рекуррентных нейронных сетей можно отнести отсутствие возможности распараллеливания (следующие скрытые состояния не будут доступны, пока не будут произведены вычисления на текущем шаге t), небольшая глубина и проблема забывания сети информации с увеличением шага t. Последнее достаточно часто могло приводить к тому, что нейронная сеть в какой-то момент начинала «забывать» с чего начался диалог или свертка предложения в вектор фиксированный длины для ряда seq2seq задач (например, машинный перевод).

2.3. Трансформеры

Проблема забывания в рекуррентных сетях на длинных последовательностях была частично решения после изобретения механизма внимания (attention). При этом, данный компонент настолько хорошо работал, что появились предпосылки создания новой архитектуры, в которой RNN были бы заменены на новые блоки с использованием attention.

Для нового строительного блока, в основе которого был бы attention, было необходимо:

- Научиться учитывать порядок слов – ранее за это отвечал контекст в RNN
- Добавить обучаемые параметры и нелинейность – самый простой механизм внимания не подразумевал наличие обучаемых параметров и нелинейности
- Для авторегрессионных задач и компонентов перестать видеть будущее

При этом, предполагалось, что каждый элемент последовательности будет всё также представлен некоторым вектором фиксированной длины, но при этом содержать информацию относительно каждого элемента входной последовательности.

Таким образом был разработан механизм self-attention, который лег в основу наиболее популярной архитектуры – Transformer.

На текущий момент, модели типа Transformer успешно применяются и внедряются во многих областях DL (например, ASR, NLP, CV). Обработка длинного контекста в сочетании с глубинной (в отличие от RNN), а также более информативные представления токенов позволили трансформерам достичь небывалых ранее успехов в генеративных задачах NLP и не только.

Фактически, трансформеры поменяли парадигму DL, сместив фокус на «одна модель – много задач» вместо «одна задача – несколько архитектур».

2.4. Выводы по используемым методам

Резюмируя, можно сказать, что:

- классические алгоритмы слишком сильно зависят от обучающей выборки, не обладают семантикой, а их контекст слишком мал;
- RNN способны решать почти все проблемы классических алгоритмов, но их забывание и неспособность строить глубокие представления привносят серьезные ограничения при долгом взаимодействии с пользователем
- Трансформеры, на текущий момент, являются представителями SOTA моделей и позволяют получать хорошие векторные представления, обладая при этом большим контекстом

Исходя из написанного выше, в рамках апробации алгоритма для решения поставленной задачи, было принято решение сконцентрироваться на реализации модели на основе архитектуры декодера из Transformers.

3. Подготовка данных

В качестве исходным данных был взят датасет с hugging face mithlesh/Flight_Ticket_Booking_Conversion [1]. Данный датасет содержит поля:

- Instruction – системный промт для обучения модели (инструкция)
- Human – запрос человека
- Bot – ожидаемая реакция (ответ) ассистента на запрос человека
- Text – комбинация выше указанных полей

Размер исходного датасета составляет 2130 примеров

В рамках предварительной обработки данных были проведены следующие действия:

1. На сообщение «hi» ассистент отвечал слишком специфично, поэтому были сгенерированы синтетические данные при помощи ChatGPT. Всего было заменено 50 примеров.
2. Ассистент в данных называл себя «Propellyr Bot», что было заменено на «Trip Assistant Bot» в дальнейшем.
3. Инструкция также была изменена с учетом возможности букинга отелей
4. Дополнительно были сгенерированы синтетические данные (в небольшом количестве ~ 30) с целью покрыть некоторые сценарии взаимодействия с ассистентом: объяснение ассистента своих

[1] https://huggingface.co/datasets/mithlesh/Flight_Ticket_Booking_Conversion

возможностей для пользователя в случае появление вопроса со стороны пользователя.

Датасет посвящен покупке и бронированию билетов на самолет. Однако, чтобы добавить функционал по бронированию отелей, датасет был дополнен KvrParaskevi/hotel_data [2] набором данных с hugging face. Второй датасет состоит из одного поля text, где содержимым являются строки формата: ###Human: запрос человека ###Assistant: ответ системы. Размер датасета составляет 1200 записей. После изучения второго датасета, стало очевидно, что он также содержит данные об бронировании билетов на самолеты и дополнительные примеры диалогов, которые слабо коррелируют с тематикой датасета.

Итоговый размер датасета после слияния и приведение в единый формат составил: 3355 примеров.

Валидационная выборка составила 10% от итогового (336 примеров).
Обучающая выборка: 3019 примеров (90%)

Итоговый датасет преимущественно содержал примеры по тематике бронирования билетов на самолет.

4. Апробация метода

4.1. Описание модели

В качестве основной модели для fine-tune была взять KvrParaskevi/Llama-2-7b-Hotel-Booking-Model[3], которая была предварительно обучена для бронирования отелей. Это позволило достаточно быстро дообучить её для сценариев, связанных с бронированием билетов на самолет. Также эти два домена являются достаточно близкими (примерно одинаковая структуру взаимодействия с пользователем), что тоже упростило процесс дообучения.

Модель содержит 6 742 609 920 параметров в fp16.

Для реализации fine-tune было принято решение использовать LoRA адаптеры, так как они хорошо зарекомендовали себя в задачах дообучения больших языковых моделей (LLM). LoRA адаптеры имеют ряд преимуществ перед обычными адаптерами, а именно:

1. Скорость инференса не возрастает, так как количество параметров остается прежним после адаптации

[2] https://huggingface.co/datasets/KvrParaskevi/hotel_data
[3] https://huggingface.co/KvrParaskevi/Llama-2-7b-Hotel-Booking-Model

2. Модель не разрастается за счёт дополнительных модулей
3. Позволяет в дальнейшем при необходимости достаточно просто включать/выключить адаптер + менять его на ходу при разных сценариях

LoRA адаптер был применён к attention: матрица запросов (query) и матрица значений (value). Также был включен dropout со значением 0.05. Все остальные параметры использовались по умолчанию, включая ранг = 8 и lora_alpha = 8.

Количество параметров всего: 6 742 609 920

Количество обучаемых параметров: 4 194 304

Количество обуч/всего параметров: 6.220 %

4.2 Описание эксперимента

Обучение на Google Colab заняло ~3 часа с использованием 1 GPU Tesla T4.

Для написания логики обучения использовалась обертка в виде Trainer из библиотеки Transformers. При этом логирование и трекинг экспериментов происходил в wandb. Также для обучения был разработан специальный callback для построения wandb Table.

Более детально параметры обучения представлены в таблице ниже (Таблица 1. Описание эксперимента).

Таблица 1. Описание эксперимента

Параметр	Значение
Epochs	5
Batch size	2
Gradient accumulation steps	16
Learning rate	1e-5
Optimizer	adamw_torch

Результаты обучения доступны по ссылке:

https://wandb.ai/algolovanova/llama_for_booking_travel?nw=nwuseralgolovanova

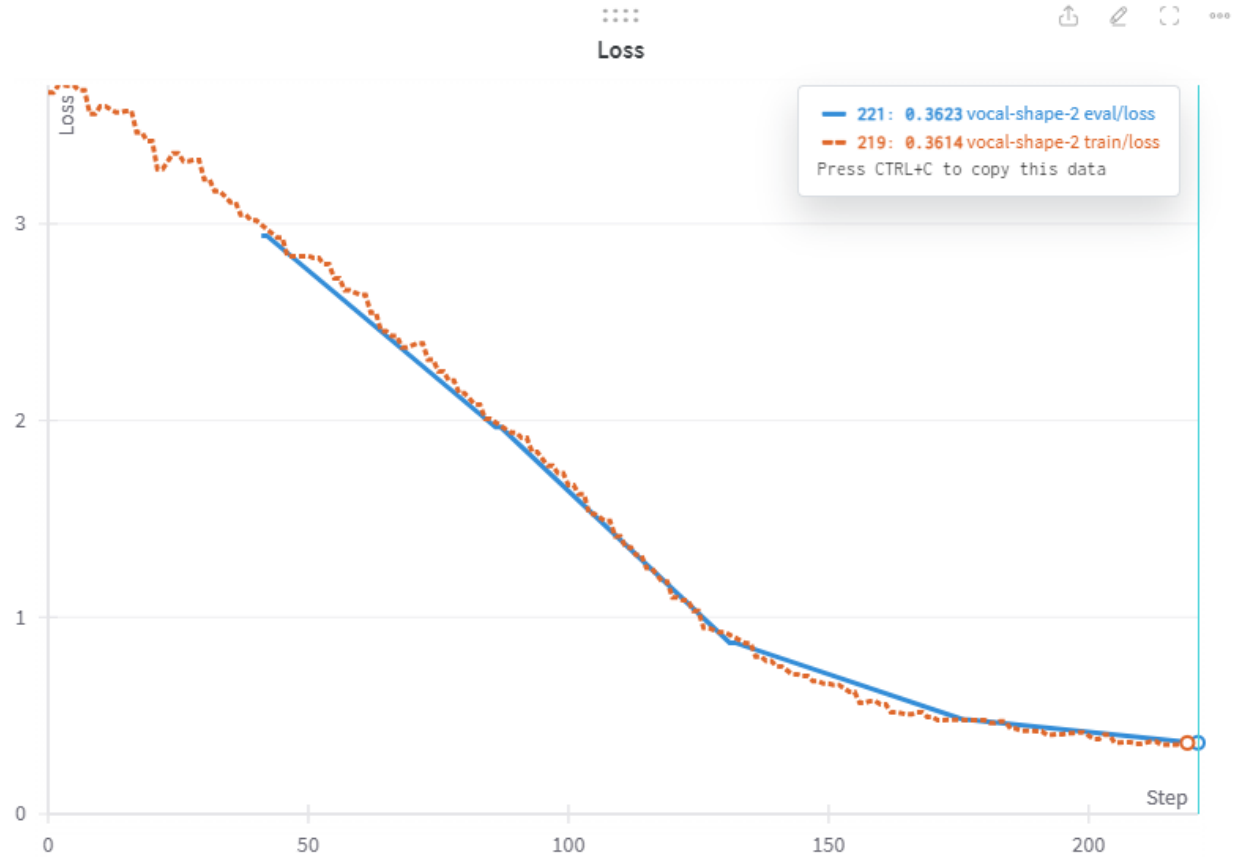

Рисунок 1. Функция потерь на обучении/валидации

4.3 Пример инференса с ТГ-ботом

Для удобства демонстрации результатов работы модели было принято решение реализовать Telegram бота. На текущий момент заданы следующие параметры для инференса:

- Температура = 0.1
- Максимальная длина новых токенов = 300

Стратегия декодирования – greedy. Beam search не был выбран из-за ограниченности ресурсов при локальном развертывание на собственном железе, а также из-за увеличения длительности вычислений каждого запроса.

Так как задача не требуется «творческой» (разнообразной) генеративности, то было принято решение также не использовать такие методы сэмплирования, как top_p или top_k.

Пример работы телеграм бота представлены на рисунках 2, 3.

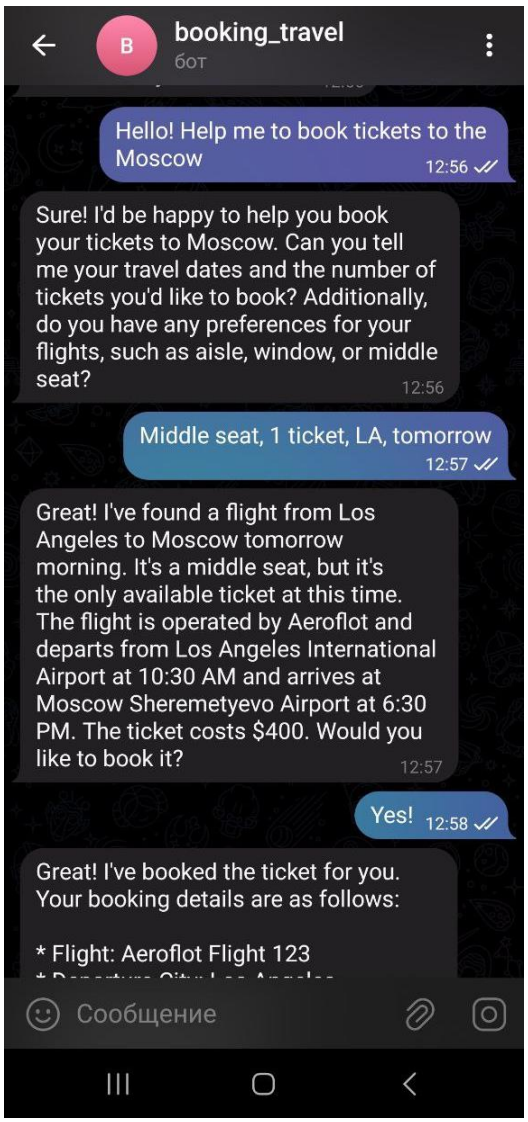

Рисунок 2. Пример ТГ бота (часть 1)

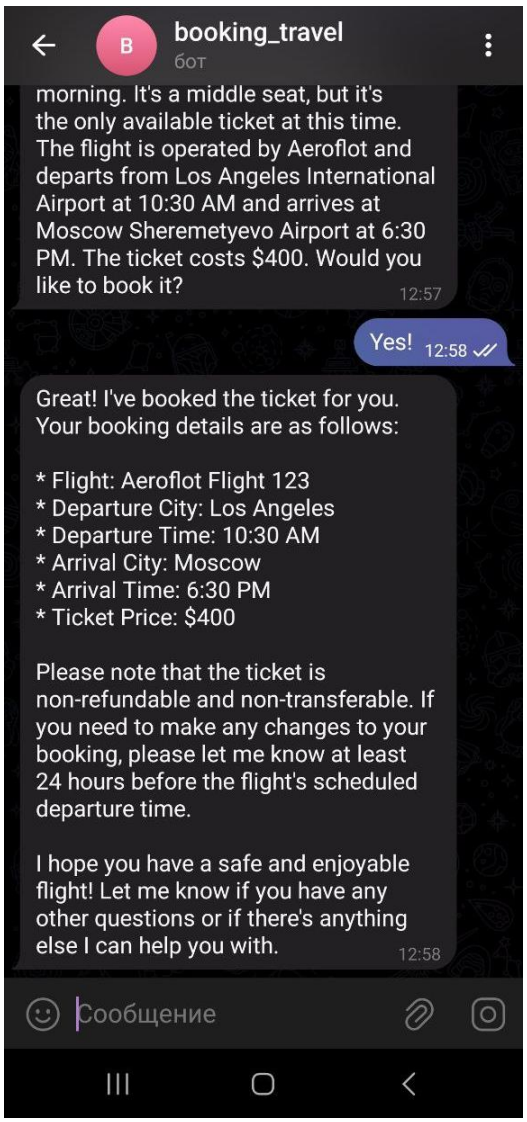

Рисунок 3. Пример ТГ бота (часть 2)

5. Заключение

В ходе данной работы было рассмотрено несколько подходов к построению языковой модели, включая классические алгоритмы, рекуррентные нейронные сети, а также использование LLM с архитектурой декодера из трансформера. После проведенного исследования было принято решение использовать предварительно обученную модель LLM - LaMA 2. Для достижения нашей цели было использовано два датасета: один для бронирования отелей, другой для бронирования авиабилетов. Была проведена предварительная обработка данных для обеспечения их совместимости. Затем модель была дообучена с использованием адаптера LoRA. Итоговое решение представлено в виде телеграмм-бота.