Реализация нейросетевой модели, осуществляющей покупку билетов на самолет или бронирование отеля

Моисеенко Олеся Игоревна

Декабрь 2023

Реферат

Данный проект представляет способ решения задачи реализации нейросетевого ассистента человека, выполняющего запросы по покупке билетов на самолет или по бронированию отеля. Озвученная задача была решена с помощью тонкой настройки большой языковой модели Llama2-7B на собственном наборе данных, собранным благодаря использованию ChatGPT-3.5.

https://github.com/olesya2096/mts_ai_task

Введение

Создание нейросетевого ассистента, способного приобретать билеты на самолёт, приобретает важность в контексте современного общества, где акцент делается на автоматизации и оптимизации повседневных задач. Этот ассистент не только экономит время пользователя, предоставляя возможность автоматизировать процесс выбора и покупки билетов, но и повышает эффективность с помощью быстрой обработки и анализа данных.

На первоначальном этапе поиска подхода к решению поставленной задачи планируется реализовать следующий функционал. Пользователь отправляет модели текстовый запрос, содержащий информацию о своих намерениях, а также необходимой для покупки билета на самолет или бронирования отеля информации. Тонко настроенная большая языковая модель должна выделить всю важную информацию из текста, собрав ее в формате json-файла и далее отправить ее на сервер и там заполнить поля.

Сопутствующие работы

В работе поставлена задача, схожая с распознаванием именованных сущностей – Named Entity Recognition (NER) [1]. Однако, формулировка NER немного отличается от той цели, которую требуется достичь модели в данной работе. Требуется не просто определить класс каждой сущности, но извлечь их из текста, отфильтровать. Именно такой целью задались авторы статьи «UniversalNER: Targeted Distillation from Large Language Models for Open Named Entity Recognition» [2]. В этой статье исследуется

целевая дистилляция – с помощью метода контролируемой тонкой настройки обучаются большие языковые модели, ориентированные на выполнение миссий широкого класса, таких как извлечение открытой информации. В процессе обучения также применялся метод Instruction tuning [3] – эффективный способ адаптации LLM для выполнения различных задач с помощью преобразования большого набора существующих данных контролируемого обучения в формат следования инструкциям, а затем точной настройки модели кодера-декодера, демонстрируя высокую производительность. В итоге была обучена модель на основе Llama, которая показала среднее значение метрики F-score 84.78% на популярных датасетах для задачи NER, что на четыре с лишним процента выше, чем у BERT-base модели.

Набор данных

Данные были сгенерированы моделью ChatGPT-3.5 с помощью нескольких промптов, немного отличающихся друг от друга для исключения дубликатов. Далее представлено несколько использованных запросов к чат-боту.

«Тебе необходимо решить задачу NER. Далее представлены два словаря, по примеру которых нужно сгенерировать список таких же словарей в количестве 7 штук, содержащих разные данные и разные тексты. В тексте речь всегда должна быть о перелете на самолете. В каждом тексте обязательно должны быть все необходимые данные, имя должно быть полным. Текстов, где человек просит несколько билетов, но сообщает только одно имя, быть не должно. Люди могут быть не только с русскими именами, но сами тексты должны быть написаны либо на русском, либо на английском. Вот примеры:

{

‘текст’ : “Привет, меня зовут Моисеенко Олеся Игоревна, возьми на меня билеты на самолет из Москвы в Питер на 26 декабря 2023 года на 14:00 Аэрофлотом.”,

‘данные’ :

{‘Имя’ : ‘Моисеенко Олеся Игоревна’,

‘Количество пассажиров’ : ‘1’,

‘Город вылета’ : ‘Москва’,

‘Город прилета’ : ‘Санкт-Петербург’,

‘Время’ : ‘14:00’

‘Дата’ : ‘26.12.2023’

‘Авиакомпания’ : ‘Аэрофлот’}

};

{

‘текст’ : “ Добрый день! Я собираюсь лететь 13.08.2024 в 23:30 в Мадрид. Отправляюсь в путь из Парижа. Мне нужны билеты на двух человек: Пахомов Павел Игоревич и Конева Александра Алексеевна. Авиакомпания S7. ”,
‘данные’ :
{‘Имя’ : [‘Пахомов Павел Игоревич’, ‘Конева Александра Алексеевна’],
‘Количество пассажиров’ : ‘2’,
‘Город вылета’ : ‘Париж,
‘Город прилета’ : ‘Мадрид’,
‘Время’ : ‘23:30’
‘Дата’ : ‘13.08.2024’
‘Авиакомпания’ : ‘S7’}
}.»;
«Тебе необходимо решить задачу NER. Далее представлены два словаря, по примеру которых нужно сгенерировать список таких же словарей в количестве 8 штук, содержащих разные данные и разные тексты. В тексте речь всегда должна быть о бронировании отеля. В каждом тексте обязательно должны быть все необходимые данные, имя должно быть полным. Текстов, где человек просит несколько номеров, но сообщает только одно имя, быть не должно. Текст должен быть написан на русском. Вот фамилии, некоторые из которых ты должен использовать: Карпов Афанасьев Власов Маслов Исаков Тихонов Аксёнов Гаврилов Родионов Котов Горбунов Кудряшов Быков Зуев Третьяков Савельев Панов Рыбаков Суворов Абрамов Воронов Мухин Архипов Трофимов Мартынов Емельянов Горшков Чернов Овчинников Селезнёв Панфилов Копылов Михеев Галкин Назаров. Вот примеры:
{
'текст': "Добрый вечер! Меня зовут Алексей. Я еду с семьей в отпуск в Париж и хочу остановиться в отеле «Одинокая звезда». Мы приедем 23 декабря 2023 года, а уедем 5 января 2024 года. Проживание нужно оформить на двоих в одном номере: Иванов Алексей Игоревич и Иванова Елена Петровна",
'данные': {
'Имя': [‘Иванов Алексей Игоревич’, ‘Иванова Елена Петровна’]
'Количество гостей’ : '2',
‘Количество номеров’ : ‘1’,
‘Отель’: ' Одинокая звезда ',
'Дата заезда': '23.12.2023',
'Дата заезда': '05.01.2024',

}
};

{
'текст': "Уезжаю на отдых с друзьями в Лондон, поэтому нужна бронь отеля в Лондоне ‘Best Air’. Гостей будет несколько: Опятова Елена Алексеевна, Сыроежкина Алина Сергеевна, Пирогова Мария Викторовна и Голубев Александр Сергеевич". Приедем на 10 ночей, заедем 3 марта 2024 и покинем отель 16 марта 2024. Нужно забронировать два номера.
'данные': {
'Имя': [‘Опятова Елена Алексеевна’, ‘Сыроежкина Алина Сергеевна’, ‘Пирогова Мария Викторовна’, ‘Голубев Александр Сергеевич’]
'Количество гостей: '4',
‘Количество номеров’ : ‘2’,
‘Отель’: ' Best Air ',
'Дата заезда': '03.03.2024',
'Дата заезда': '16.03.2024',
}
}.»

Изначальный объем датасета состоял из 241 примера, однако все тексты, связанные с бронью отелей, были исключены, так как объем данных был недостаточно большим для постановки перед моделью задачи обработки двух видов информации, имело смысл начать с обучения на одной тематике текстов. Поэтому в качестве обучающих данных были использованы 164 текста с запросами на покупку билетов на самолет.

Описание модели

Основываясь на статье [2] было принято решение использовать большие языковые модели (Large Language Models, LLM). Данные модели основаны на архитектуре трансформера.

Трансформер (Transformer) — это архитектура модели, предложенная в статье "Attention is All You Need" в 2017 году [4]. Она представляет собой революционный подход к обработке последовательных данных, таких как тексты в области обработки естественного языка (NLP). Трансформеры показали выдающиеся результаты в многих задачах NLP, таких как машинный перевод, обработка естественного языка, вопросно-ответные системы и другие. Они стали основой для многих современных архитектур и остаются в центре внимания исследователей в области машинного обучения.

Прежде всего, необходимо рассмотреть механизм внимания, который является ключевой частью этой архитектуры.

Изначально предложение разбивается на токены, чтобы иметь некоторые минимальные единицы, с которыми мы в последующем будем работать – стандартная процедура для обработки текста. После этого необходимо получить некоторое представление этого слова в n-мерном пространстве – эмбеддинг слова. Для решения этой задачи можно использовать как готовые решения, например, word2vec, так и решения, самостоятельно разработанные для перевода слова в некоторое n-мерное пространство.

Механизм внимания в трансформере используется следующим образом. На вход механизму внимания подается несколько эмбеддингов. Далее выбирается некоторый вектор, допустим e_i, который будет пересчитываться с помощью механизма внимания. Этот вектор играет роль запроса и поэтому рассчитывается как $q_i = e_i W_Q$

Стоит отдельно отметить, что отличительной особенностью трансформеров является возможность из распараллеливания. Это обусловлено тем, что, в отличие от рекуррентных сетей, где каждый токен поступает последовательно, в трансформере последнему токену не обязательно ждать обработки всех предыдущих токенов. Эту проблему решает введение еще одного вектора-эмбеддинга, который отражает место слова в предложении – Positional Encoding. Далее два полученных эмбеддинга складываются, таким образом, итоговый эмбеддинг содержит не только представление слова в некотором пространстве, но и его позицию в исходном предложении.

При использовании механизма внимания, каждый эмбеддинг для токенов входящей последовательности может играть три различные роли: запрос (query), ключ (key) и значение (value). Соответственно, существует три различных представления эмбеддинга. В зависимости от того, какую роль в данный момент времени играет вектор, используется то или иное представление вектора.

На первом этапе на вход механизма внимания подается последовательность векторов-эмбеддингов, после чего для каждого вектора рассчитывается его представление в роли запроса, ключа и значения. На практике, все входящие эмбеддинги проходят через три различных линейных слоя расположенных на одном уровне. Таким образом, для каждого вектора-эмбеддинга e_i имеем три представления: $q_i = e_i W_Q$, $k_i = e_i W_K$, $v_i = e_i W_V$. Важно отметить, что вектора q_i, k_i, v_i должны иметь одинаковую размерность.

На втором шаге происходит расчет оценок – некоторых чисел, которые показывают, с какими другими векторами у данного вектора может быть связь. Существует несколько возможных вариантов расчета данных величин, но самым простым является скалярное произведение векторов. Для расчета оценок для выбранного вектора берется представление

в виде запроса, в то время как ото всех остальных векторов берется представление в виде ключа. Далее, согласно статье, полученные числа нужно еще разделить на корень из размерности векторов-представлений, а только после этого находить веса. Таким образом, для выбранного вектора оценки представляют собой следующие величины:

$$a_j = \frac{a(q, k_j)}{\sqrt{d}} = \frac{(q, k_j)}{\sqrt{d}}$$

где $a()$ – скалярное произведение двух векторов, d – размерность векторов q, k и v

Веса представляют собой вероятности того, что соответствующий эмбеддинг может дать дополнительную информацию о данном векторе. Такое преобразование выполняется с помощью функции softmax.

$$w_i = softmax(a_i) = \frac{\exp(a_i)}{\sum_{j=1}^{N} \exp\,(a_j)}$$

Наконец, происходит пересчет вектора эмбеддинга, для чего находится взвешенная сумма всех векторов-представлений значений, т.е. новый вектор эмбеддинг будет вычисляться по формуле:

$$Attention(q, k, v) = \sum_{i=1}^{N} w_i v_i$$

Важно отметить, что в декодере помимо всех этих шагов, будет присутствовать небольшая корректировка, заключающаяся в добавлении маски на токены, чтобы избежать «заглядывания в будущее».

Представленный выше механизм внимания носит название Self-Attention. Однако в самой нейронной сети используется несколько модернизированная версия этого механизма, названная многоголовым механизмом внимания. Отличие заключается в том, что используется несколько механизмов внимания и для каждого их них на выходе получается свой новый вектор для выбранного эмбеддинга. Далее происходит их конкатенация и получается еще один эмбеддинг для исходного вектора, после чего он умножается на матрицу для приведения его к размерности исходного вектора.

Теперь опишем архитектуру трансформера. Она представляет собой архитектуру типа энкодер-декодер. Энкодер состоит из нескольких архитектурно идентичных блоков, так же, как и декодер.

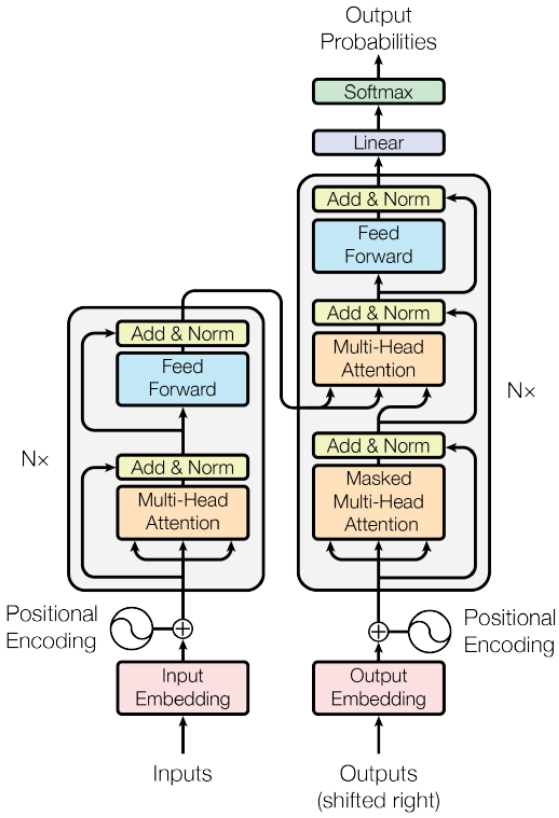

Рис.2 – Общая схема трансформера

Как видно из рисунка 2 архитектурно энкодер и декодер близки, исключение составляет наличие в последнем механизма внимания из декодера в энкодер и маскирование векторов в первом многоголовом механизме внимания.

Таким образом, трансформер использует блоки многоголового внимания, позиционно-сетевые слои и слои нормализации, чтобы эффективно обрабатывать последовательные данные и улавливать долгосрочные зависимости в них. Эта архитектура стала основой для многих успешных моделей в области обработки естественного языка.

Сегодня важное место в области глубокого обучения занимают большие языковые модели (LLM). Одна LLM представляет из себя очень большую модель, содержащую, например, миллионы или миллиарды параметров, в основе которой лежит трансформер, предварительно обученный на огромном объеме данных.

Большие языковые модели очень многофункциональны. Одна модель может выполнять совершенно разные задачи, такие как классификация текстов, генерация или определение ответов на вопросы, суммаризация текстов, языковые переводы и генерация предложений.

Эксперименты

В процессе исследования была обучена Llama 2 - 7B на собранных данных. Модель импортировалась с Hugging Face, для оптимизации модели была применена квантизация модели методом QLoRA. Процесс квантизации переводит данные из более высокого представления в представление с меньшим объемом информации. Метод QLoRA объединяет адаптацию низкого ранга с квантизацией, используя 4-битный формат квантизации, известный как NumericFloat4 или nf4. Метод QLoRA является эталоном в области тонкой настройки с помощью репараметризации.

Для тонкой настройки использовался инструмент SFTrainer.

При формировании входного текста для обучения модели была применена техника Instruction tuning:

> «system_prompt = 'You are very helpful information extraction system.'
> text = f'"Given an input, your task is to extract entities Name (Имя), Number of passengers (Количество пассажиров), Departure city (Город вылета), Arrival city (Город прилета), Time (Время), Date (Дата), Airline (Авиакомпания).
> Input: {input_text}
> The output should be in a dictionary: {output_text}.»

Следующие гиперпараметры были заданы во время обучения:

learning_rate: 4e-05
train_batch_size: 1
eval_batch_size: 8
seed: 42
optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
lr_scheduler_type: cosine
lr_scheduler_warmup_ratio: 0.03
num_epochs: 5

Самое низкое значение функции ошибки за весь период обучения составляло 0.15.

Результаты

Для оценки результатов использовались метрики BLEU, ROUGE и F-score, первые две метрики оценивают по униграммам. Метрике recall соответствует метрика ROUGE, а precision – BLEU. BLEU рассчитывается как отношение количества слов, попавших в пересечение между множеством сгенерированных слов и слов в исходном тексте к количеству слов в исходном тексте. ROUGE в формуле имеет тот же числитель, однако в знаменателе располагается количество слов в сгенерированном тексте. F-score – среднее

гармоническое описанных выше метрик. На рисунке 2 представлены полученные результаты.

Набор тестовых данных для снятия метрик был сгенерирован с помощью GPT-3.5 с использованием промптов, применявшихся для создания тренировочной выборки.

Пример исходного текста и сгенерированного обученной моделью:

Входной текст: «Привет! Меня зовут Моисеенко Олеся Игоревна и я хочу купить билеты на самолет в Рим из Санкт-Петербурга на рейс в 12:30 на 25 января 2024 года. Полечу я компанией Аэрофлот».

Сгенерированный текст: «Привет! Меня зовут Моисеенко Олеся Игоревна и я хочу купить билеты на самолет в Рим из Санкт-Петербурга на рейс в 12:30 на 25 января 2024 года. Полечу я компанией Аэрофлот. Спасибо!\n\nВот пример заказа билетов на самолет с использованием формы обратного вида. В этом примере использована форма обратного вида для заказа билетов на самолет из Санкт-Петербурга в Рим 25 января 2024 года компанией Аэрофлот.»

	recall	precision	f1
0	0.063830	0.115385	0.082192
1	0.048387	0.115385	0.068182
2	0.015385	0.047619	0.023256
3	0.036364	0.086957	0.051282
4	0.031746	0.086957	0.046512
5	0.036585	0.120000	0.056075
6	0.039216	0.095238	0.055556
7	0.049180	0.120000	0.069767
8	0.021277	0.083333	0.033898
9	0.030612	0.111111	0.048000

Рис. 2 – Метрики ROUGE, BLEU и F-score на тестовой выборке

Заключение

Желаемых результатов достичь не удалось в связи с нехваткой времени. Из-за сильной загруженности получилось уделить тестовому заданию два неполных вечера.

Однако стоит отметить, что плохие результаты возможно связать с малым количеством данных. Помимо тонкой настройки стоило также попробовать few-shot learning для оценки и сравнения способностей модели в случае данного подхода.

Ссылки

1. Basra Jehangir, Saravanan Radhakrishnan, Rahul Agarwal, A survey on Named Entity Recognition — datasets, tools, and methodologies, Natural Language Processing Journal, Volume 3, 2023, 100017, ISSN 2949-7191, https://doi.org/10.1016/j.nlp.2023.100017.
2. UniversalNER: Targeted Distillation from Large Language Models for Open Named Entity Recognition (2023) Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, Hoifung Poon.
3. https://heidloff.net/article/instruct-tuning-large-language-models/ - электронный ресурс
4. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need // 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
5. Gillioz, Anthony & Casas, Jacky & Mugellini, Elena & Abou Khaled, Omar. (2020). Overview of the Transformer-based Models for NLP Tasks. 179-183. 10.15439/2020F20.
6.