Сравнение полного и параметро-эффективного дообучения трансформеров для многоязычного анализа текстов

Подготовила студентка ВГУ, ФКН,
3-го курса 6-й группы Полина Шапиро

09.06.2025

01 Проблема

Языковые модели показывают высокую точность, но за это приходится платить огромным числом параметров. Практические задачи почти всегда требуют дообучения моделей, и из-за их размера растут и вычислительные затраты

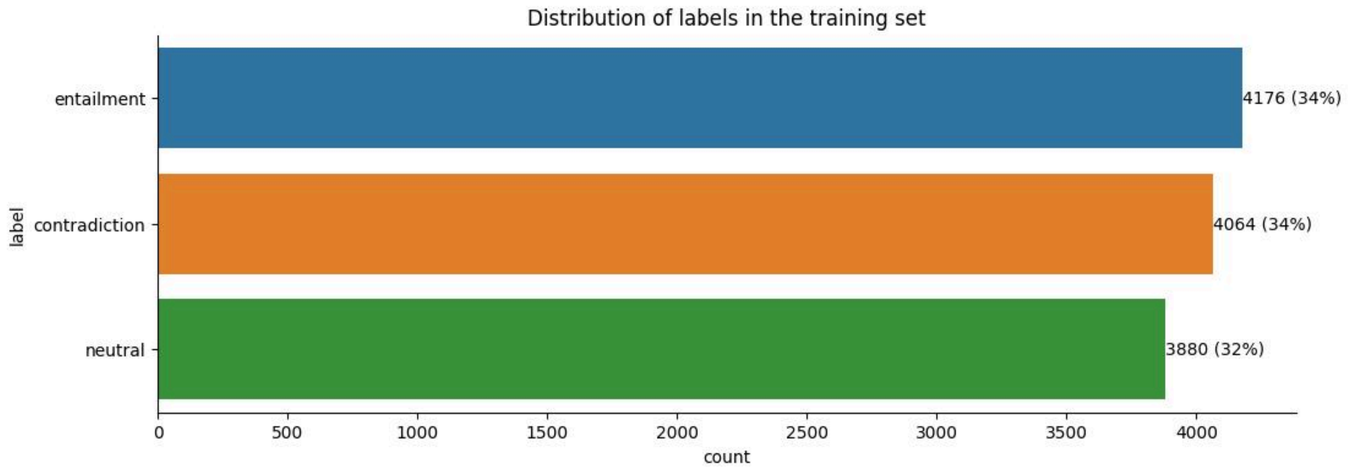
Исследование

В этой работе я исследую параметро-эффективное дообучение (PEFT) на примере метода LoRA и сравниваю его с полным fine-tuning. Сравнение проводится по времени обучения и потреблению ресурсов, а в завершение я демонстрирую веб-интерфейс, позволяющий «потрогать» полученную модель вживую

Тестовые данные

В качестве тренировочных данных был выбран публичный челлендж с Kaggle — “**Contradictory, My Dear Watson” .**

Для каждой пары предложений (premise, hypothesis) требуется определить, логически следует ли гипотеза из посылки (entailment), не связана ли она с ней (neutral) или противоречит ей (contradiction).

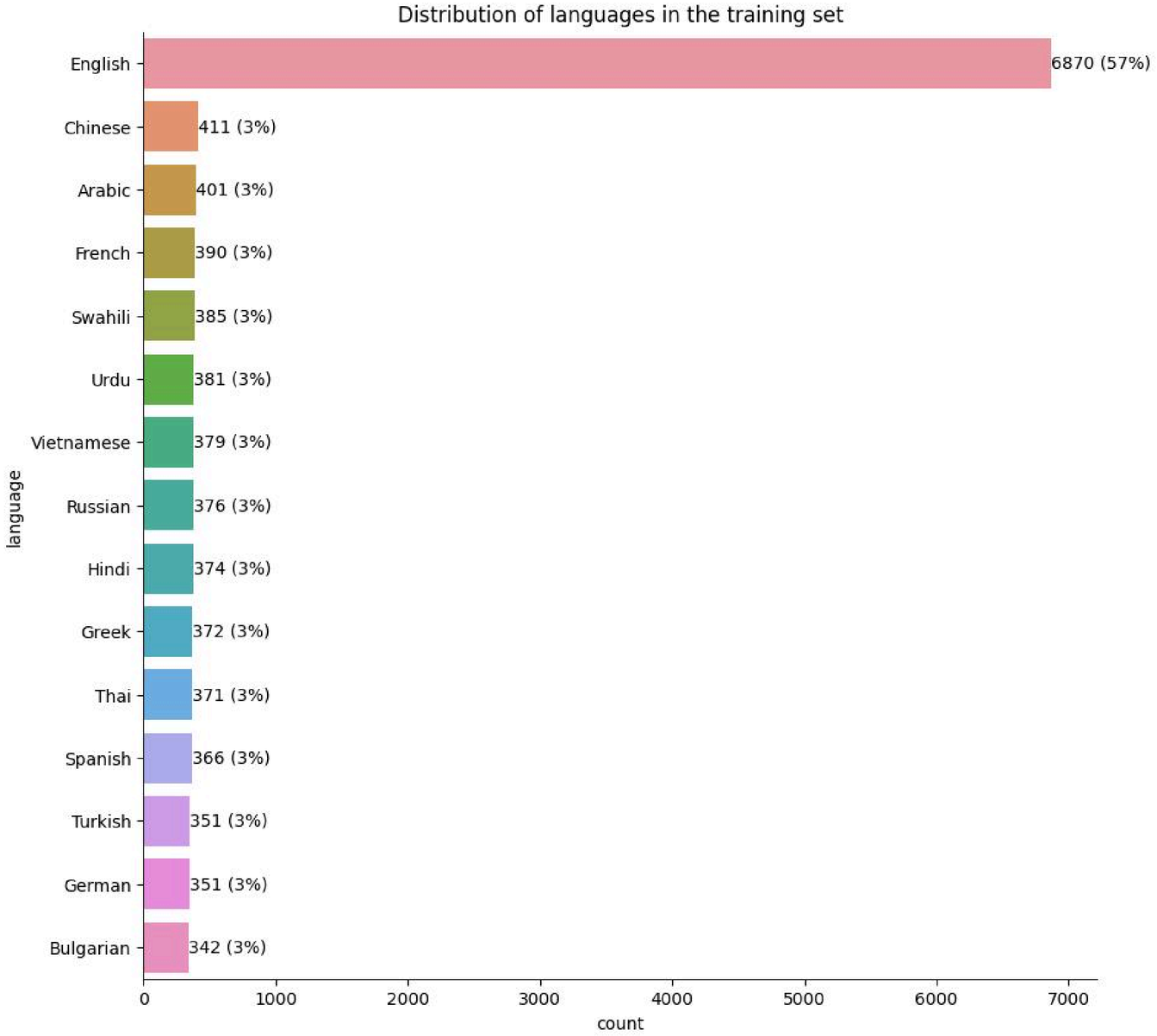


Статистика по сету

→ Корпус включает ~12 100 обучающих и ~5 200 тестовых пар на **15 языках**.

→ Английский формирует **~ 57 %** примеров

→ Остальные языки, в т. ч. низкоресурсные **— 3 %** (несколько сотен пар)



Что предлагается в работе?

→ Воспроизвести одно из представленных участниками решений полного дообучения модели mBERT (ресурсоемко, затратно по времени)

→ Дообучить модель mBERT с помощью адаптера LoRA (занимает считаные проценты параметров, обучается быстрее, возможна небольшая потеря точности)

→ Сравнить результаты

02 Трансформеры

Благодаря механизму само-внимания, они видят все слова в предложениях сразу и выбирают, что важнее. Такая одновременная оценка связей заменяет последовательное чтение в рекуррентных сетях и резко ускоряет обучение на видеокартах.

Самовнимание — как устроено формально

Самовнимание превращает каждый токен в три вектора — запрос, ключ и значение. Скалярные продукты «запрос–ключ» нормируются softmax-ом и дают веса, которыми токен взвешивает значения остальных слов. За один такой шаг слово получает контекст всей последовательности, поэтому трансформер без рекурсии захватывает и ближайшие, и далёкие зависимости.

Преимущество

→ Самовнимание считает связи между токенами за один шаг

→ Прямая связь между любыми двумя словами позволяет ловить длинный контекст, который сверточные модели часто упускают

→ Каждая голова фокусируется на своём паттерне (синтаксис, ко-референция, пунктуация) и модель сразу извлекает богатые представления

→ Предобучение на огромных корпусах, поэтому модель знает правила языка, и ее можно подучить под нужную задачу

Токены и пространство эмбеддингов

После токенизации каждое слово превращается в вектор; чем ближе два вектора, тем «похожее» их статистическое окружение в языке. Поэтому близость эмбеддингов отражает семантическую близость слов: cat и dog оказываются рядом, а cat и galaxy — далеко друг от друга

03 Что такое LoRA

LoRA не трогает основную матрицу весов: к ней добавляются две компактные матрицы A и B. Их произведение B*A образует низкоранговую матрицу ΔW, которая прибавляется к замороженной проекции W.

Именно A и B (доли процента от исходных весов) дообучаются, позволяя модели адаптироваться, не трогая основную матрицу.

Модель получает свободу подстроиться под новую задачу, не растеряв знаний, накопленных в предобучении.

04 mBERT

mBERT-base (multilingual BERT, cased)

Корпус обучения:	Википедия на 104 языках (≈ 100 GB текста)
Токенизатор:	WordPiece, общий словарь 110 000 подслов (Одинаковые корни и суффиксы разных языков делят одни и те же индексы)
Количество слоёв: 12	Глубина стека внимания
Ширина скрытого слоя: 768	Размер эмбеддинга токена и выходов каждого слоя
Голов в self-attention: 12	Параллельные подпространства внимания
Параметры:	≈ 110 млн

Мой пайплайн: mBERT + полный fine-tune vs mBERT + LoRA

Сначала я воспроизвела стандартный скрипт Kaggle, полностью дообучив mBERT-base на задаче логического вывода. Затем на тех же данных запустила тот же mBERT, но с LoRA-вставками ранга 16 только в матрицах query и value


Entailment
Neutral
Contradiction
Before LORA
After LORA

Гиперпараметры обеих моделей

Full fine-tune:

Эпохи = 3
LR = $2 \cdot 10^{-5}$,
batch = 16,
FP32

LoRA:

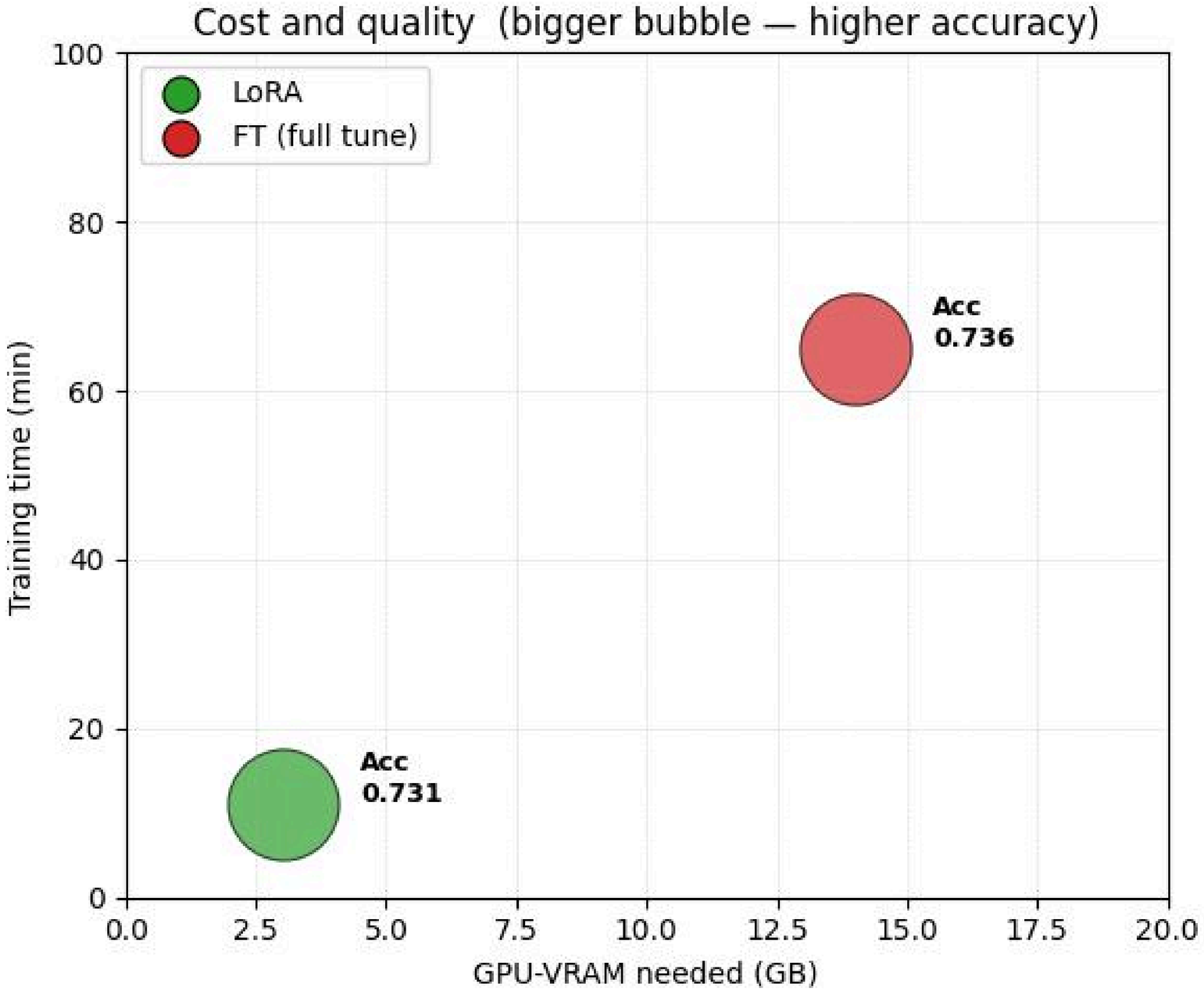
Эпохи = 3
LR = $5 \cdot 10^{-5}$,
физический batch = 8,
градиент
аккумулировался
два шага
FP16

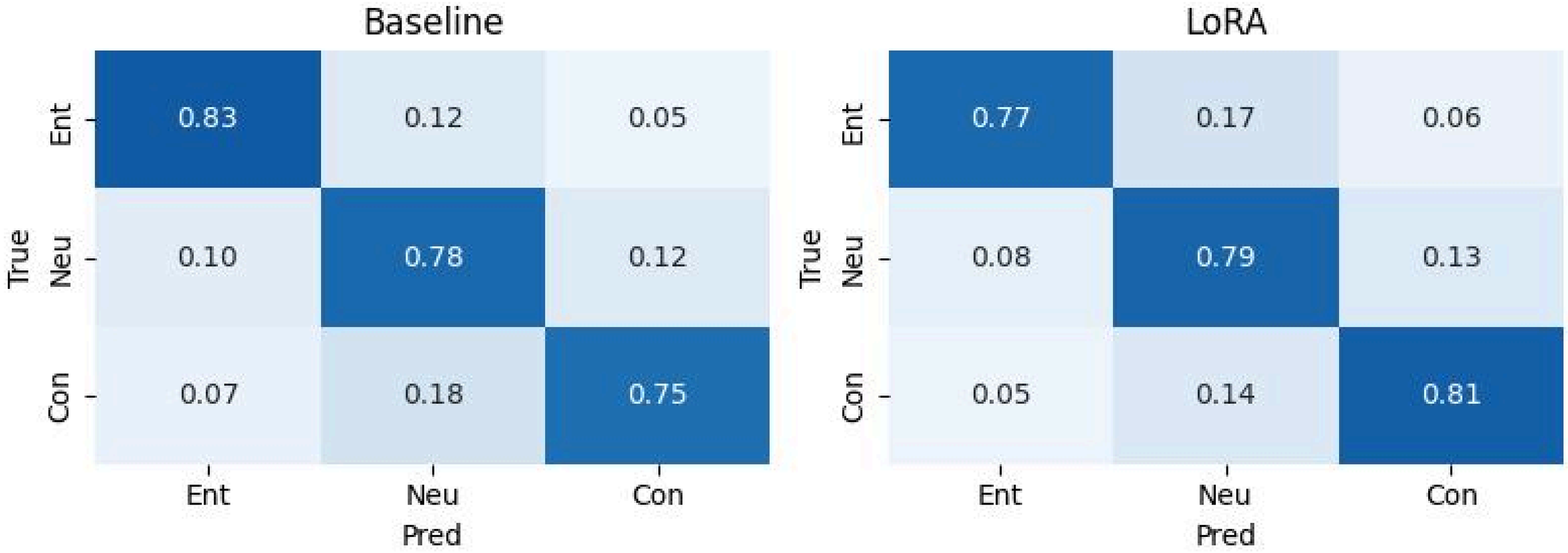
Метрики и ресурсы.

Полный fine-tune дал Accuracy 0,736, потребляя 12,8 GB VRAM и 24 мин на эпоху;
LoRA показал Accuracy 0,731 при 3,2 GB и 3.4 мин. Бар-диаграмма на соседнем слайде наглядно показывает четырёхкратную экономию памяти и троекратное ускорение.

Срез по языкам и эффект на низкоресурсных.

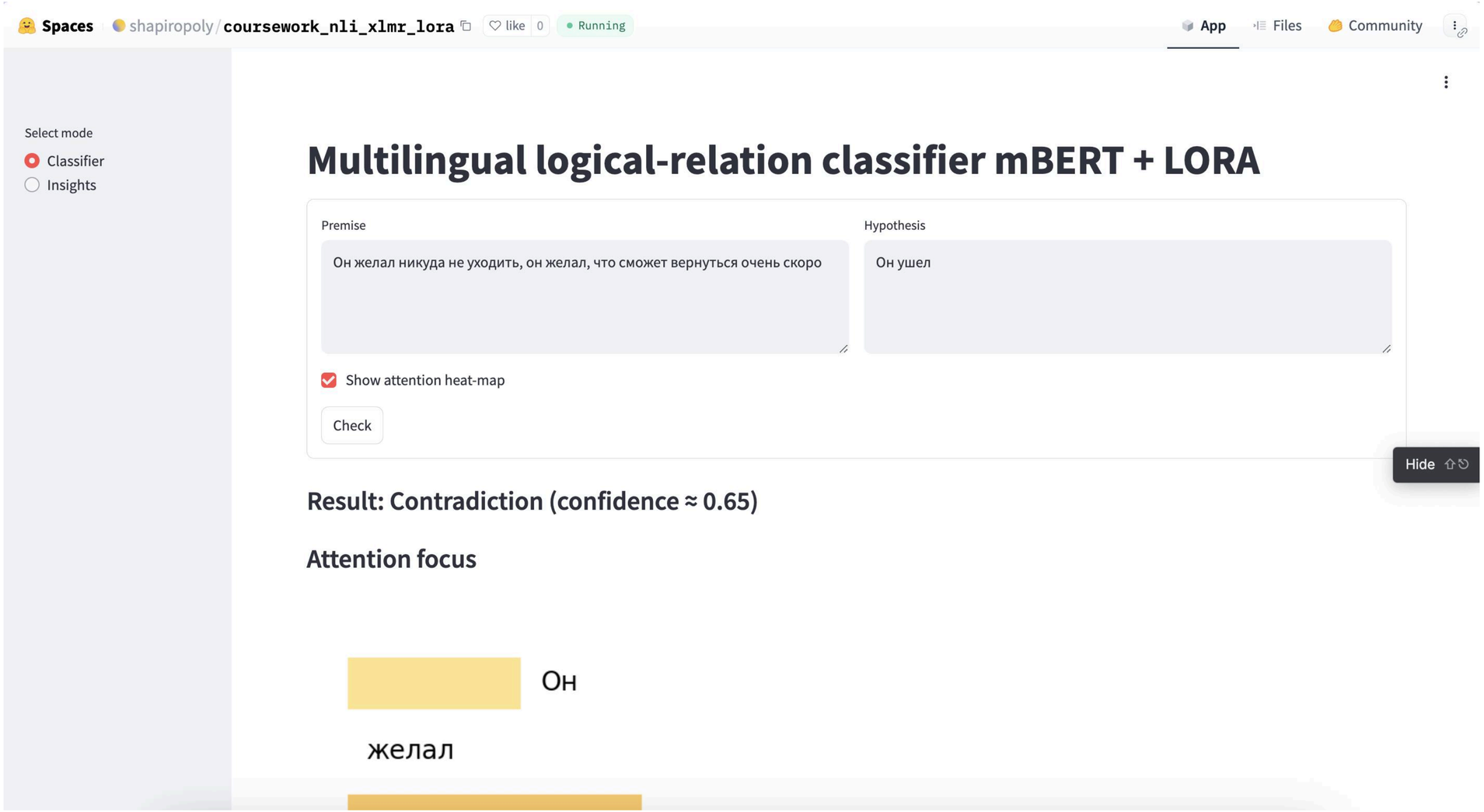
На английском результаты почти совпали, а на языках с сотнями примеров LoRA иногда даже выигрывает 2–3 pp: низкоранговая вставка действует как мягкий регуляризатор и меньше переобучается на редкие слова.


Cost and quality (bigger bubble — higher accuracy)
LoRA
FT (full tune)
Acc
0.736
Acc
0.731
Training time (min)
GPU-VRAM needed (GB)
0
20
40
60
80
100
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Baseline
True
Ent
Neu
Con
0.83
0.12
0.05
0.10
0.78
0.12
0.07
0.18
0.75
Ent
Neu
Con
Pred
LoRA
True
Ent
Neu
Con
0.77
0.17
0.06
0.08
0.79
0.13
0.05
0.14
0.81
Ent
Neu
Con
Pred

Веб-интерфейс Streamlit + Hugging Face.

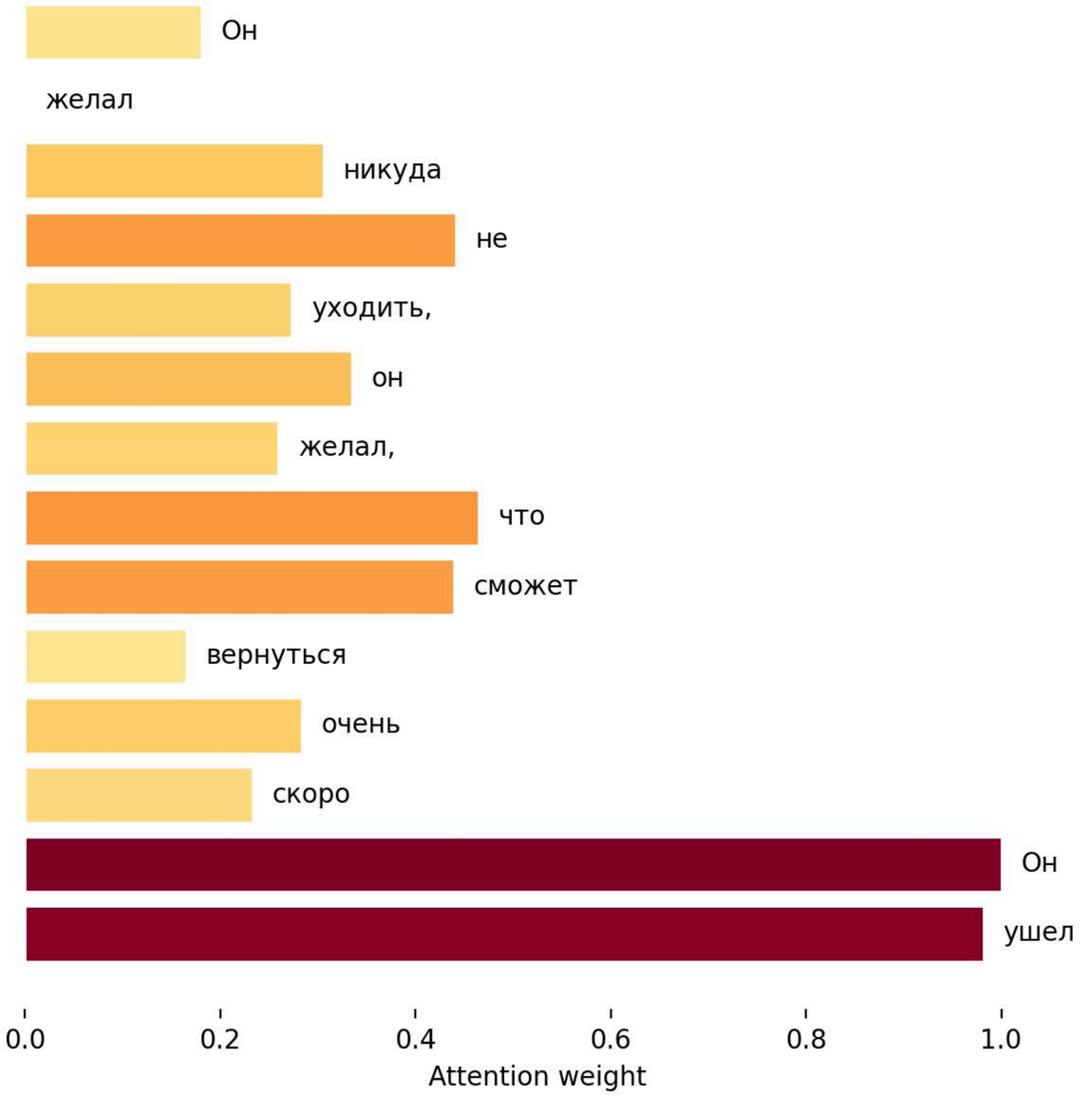
Слитый чек-пойнт загружен на Hub и обёрнут в одностраничное приложение: ввёл две фразы — получил метку и вероятность за 50 мс даже на CPU-сервере.


Spaces
shapiropoly / coursework_nli_xlmr_lora
like 0
Running
App
Files
Community
Select mode
Classifier
Insights
Multilingual logical-relation classifier mBERT + LORA
Premise
Он желал никуда не уходить, он желал, что сможет вернуться очень скоро
Hypothesis
Он ушел
Show attention heat-map
Check
Hide
Result: Contradiction (confidence ≈ 0.65)
Attention focus
Он
желал


Spaces
shapiropoly / coursework_nli_xlmr_lora
like 0
Running
Select mode
Classifier
Insights
Multilingual logical LORA

Тепловая карта токенов в интерфейсе

При нажатии «Show attention» страница вычисляет, какие слова получили наибольшее внимание от [CLS]; чем ярче полоса, тем сильнее слово влияет на решение, благодаря чему пользователь видит, почему модель считает пары противоречивыми или согласованными.


Он
желал
никуда
не
уходить,
он
желал,
что
сможет
вернуться
очень
скоро
Он
ушел
0.0
0.2
0.4
0.6
0.8
1.0
Attention weight

Выводы

LoRA сохраняет 93–99 % точности полного fine-tune, но требует в четыре раза меньше видеопамяти и учится втрое быстрее; а компактный чек-пойнт и веб-демо показывают, что многоязычный NLI-сервис можно поднять даже без GPU.

Стек

Библиотеки:

Tensorflow
Keras-NLP
Seaborn
Pandas

Среда обучения:

Google
Colab

Облако, контейнер, сервер:

Hugging
Face
Docker

Сет данных:

Kaggle

Язык:

Python

Веб-приложение:

Streamlit

Ссылка на веб-приложение

https://huggingface.co/spaces/shapiropoly/coursework_nli_xlmr_lora