Цель работы

Разработать модель машинного перевода с немецкого на английский язык. В качестве основной метрики качества использовалась BLEU.

Ход выполнения задания

1. Реализация на основе RNN

Первым шагом была попытка построить модель на основе рекуррентной нейронной сети (RNN). Это оказалось сложной задачей, и работа над этим вариантом заняла **10 дней**. Для реализации использовалась готовая архитектура из официального туториала PyTorch. (https://pytorch.org/tutorials/beginner/torchtext_translation_tutorial.html)

Основные проблемы:

- Изначально не был обучен кастомный токенизатор, что привело к низкому качеству перевода (BLEU **0.33**).
- После обучения токенизатора качество улучшилось, но все равно оставалось на уровне **6.7 BLEU**.

Обучение прошло на 8 эпохах и дальше качество не росло, уверен, что дело так в моем Dataloader, который у меня на тот момент был, сейчас он чуть модифицированный. Думаю качество было бы в районе 10-12.5 BLUE.

Вывод: RNN-архитектура показала себя неэффективной, особенно без предварительно обученного токенизатора. Было принято решение перейти к трансформерной архитектуре.

---

2. Реализация на основе Transformer

После RNN было решено использовать **Transformer**, однако процесс занял больше времени, чем ожидалось.

Основные проблемы и их решения:

- **Проблемы с масками и индексами.** Ошибки в маскировании последовательностей приводили к тому, что веса декодера превращались в `NaN`. После тщательной отладки удалось устранить этот баг.
- **Ограниченные вычислительные ресурсы.** Использование **медленного GPU на Kaggle** замедлило процесс обучения. Обучение заняло **10 часов**, что существенно замедляло эксперименты.
- **Проблемы с гиперпараметрами.** Изначально неправильно подобранные гиперпараметры и отсутствие LR-шедулера приводили к медленному улучшению качества модели.

Первоначальное качество после 16 эпох составило **22 BLEU**. На этом этапе стало ясно, что модель требует дальнейшего обучения. Реализацию подсмотрел в туториале pytorch.(https://web.archive.org/web/20240324185533/https://pytorch.org/tutorials/beginner/translation_transformer.html)

Какие гиперпараметры были выбраны:

EMB_SIZE = 512

NHEAD = 8

FFN_HID_DIM = 1024

NUM_ENCODER_LAYERS = 6

NUM_DECODER_LAYERS = 6

BATCH_SIZE = 128

NUM_EPOCHS = 25

В отличии от первоначальных гиперпараметров:

EMB_SIZE = 512

NHEAD = 8

FFN_HID_DIM = 512

NUM_ENCODER_LAYERS = 3

NUM_DECODER_LAYERS = 3

BATCH_SIZE = 64

NUM_EPOCHS = 12

Финальный результат:

- На **25-й эпохе** было достигнуто **25 BLEU**.
- Обучение было остановлено, чтобы избежать переобучения, так как прирост качества замедлился, а `loss` уже практически не уменьшался.

---

3. Оптимизация обучения

В ходе работы была обнаружена важная проблема с подачей данных:

- Изначально **все предложения дополнялись до 128 токенов** перед обучением, что привело к неэффективному расходу памяти.

- Решение: **динамический размер батча**, который уменьшает нагрузку на память и ускоряет обучение.
- Итог: прирост качества перевода.

---

4. Итоговые результаты

- Использование **Transformer-модели** позволило добиться **значительного прироста BLEU по сравнению с RNN**.
- Финальное качество перевода: **BLEU = 25**.
- Основные улучшения:
    - Использование **кастомного токенизатора** на основе SentencePiece.
    - Оптимизация **маскировки** и **паддинга**.
    - Подбор **гиперпараметров** и использование **LR-шедулера**.
    - Оптимизация размера **батча**.

---

5. Вывод

На основе проделанной работы можно сделать несколько ключевых выводов:

1. Использование **Transformer**-модели вместо RNN существенно улучшает качество перевода.
2. **Обучение кастомного токенизатора** является обязательным шагом для достижения приемлемого качества.
3. Важную роль играют **маскирование**, **правильный подбор гиперпараметров** и **динамичный батчинг**.
4. Для быстрого и эффективного обучения **необходим мощный GPU**.

Результат в **25 BLEU** говорит о том, что модель находится на достаточно высоком уровне, однако дальнейшие улучшения возможны за счет более мощного оборудования и тонкой настройки параметров.

6. Что можно было сделать еще?

Можно было внеси Аугментацию, перемешивая некоторые предложения, создавая шум. Также можно было воспользоваться не greedy переводом, а Beamом. Вроде как это дало бы прирост в 1.5 - 3 к BLUE. Также можно было посмотреть другие Шедулеры и сравнить качество, но на это времени не хватило. Уверен, будь у меня еще денек, все бы здесь попробовал.

В целом работа интересная, но я потратил очень много времени и сил. Оч тяжело было все успеть и держать фокус как на DL, так и на других предметах. Так как все приходилось делать параллельно, тем более есть еще военная кафедра, однозначно хорошая моя работа, несмотря на то, как сделали другие(у половины >28).

Честно для меня загадка, как все так затащили, даже видел, что у некоторых было высокое качество и без корректного трансформера… Но я все же доволен своим баллом.