Име: Кристиян Иванов Гърчев ФН: 0MI0800290
Спец.: КН Курс: 3. Група: 5

Доклад

Архитектура

Имплементирал съм модел от семейството на трансформърите.

Той съдържа само декодъри понеже:

- Имаме задача, за догенериране текста на английски и после преводът му на български. Това предвещава да третираме входните данни като 1 редица, а не 2 отделни редици, както е описано в “Attention is all you need”.
- Щом ще имаме 1 входна редица, то можем да използваме GPT архитектура.

Моделът приема бачове от tokenized редици, и ги падва за са изгради нормален тензор (ако такъв не е подаден).

Tokenization

Използвам BPE (Byte Pair Encoding) за избирането на кои subwords да използвам за токени.

Пред избрал съм размера на речника да е **vocab_size=8_000**. Това е базирано на статиите, посочени в условието, и на мощността на компютърът ми да изкара една тренировка с този размер. С този размер пускаме алгоритъма и накрая получаваме token-ите и съответно merge таблицата, която после се използва в последствие за tokenization на изреченията.

Имплементацията ми се различава от главния алгоритъм с това, че не позволявам нито на скритите токъни (за начало, край, ...), нито на white-space символите, нито на пунктуационните символи да се обединяват с други символи в по-дълги токъни. Този избор е базиран, на факта примерно, че ако някой токен представлява често срещана представка, ако пред нея слеем с празно място, то ще ограничим модела думите винаги да започвам с тази представка, докато може и да има по-сложни думи, които да започва с още една представка пред първата. По-този начин ограничаваме алгоритъма една идея по-семантично да разделя теста на морфеми. От друга страна, имаме отделен токен за празно място, който се среща много често; това увеличава дължината на редиците в токъни спрямо базовия алгоритъм.

Embedding Layer

Всеки token от редиците се замества с вектор отговарящ на token-а, взет от Embedding матрицата. Това е прост look-up по индекси.

Positional Encoding

След това към всеки вектор се добавя отместване базирано на позицията на съответстващия token. Използваме схемата, представена в “Attention is all you need”.

Decoder blocks

Имаме 5 такива блока. Данните минават линейно пред тях. Всеки декодър блок съдържа:

- MultiheadAttention layer

    - 8 Attention глави.
    - Подават му се тензора от предходния блок или от Embedding layer-а за Query, Key, Value.
    - Подават се key_padding_mask и causal attention mask, за пропускане на изчисленията на корелацията със Padding tokens и future tokens.
    - Не се взимат изчислените тегла за ключовете, защото не се използват в останалата част на архитектурата.
- Residual connection – с входа на блока
- Layer Normalization
- Feed Forward Block:
    - 2 Linear Blocks с Rely activation между тях, и с Dropout слой накрая и на двата.
    - Разширяват размерността на векторите от 512 до 1536. Тук реално модела "заучава" информация по време на тренировката.
- Процента за Dropout-а е съставен от аргумента на декодър блока и сумарно от двата Dropout-а е аргумента.
    - Layer Normalization

Linear layer

Накрая данните се подават на линеен слой който преоразмерява векторите до vocab_size. Резултатът третираме като logits за Softmax layer-а

Inference mode

Logits за последния токен в редицата се подават на Softmax, който ни връща вероятностното разпределение за следващия токен. Преди да се подадат, те се разделят на temperature=0.2 на модела, който променя вероятността на моделада избере най-вероятния следващ token. Избора е през multinomial дистрибуция.

Training mode

Всяка редица с дължина L реално представлява L – 1 екземпляра за трениране. Затова, ние като input приемаме редицатите без последния им елемент и като target приемаме редиците без първия им алгумент. Logits за всички токени в input-а се подават заедно с target, предвалително flattened, на Cost функцията.

Трениране

Tokenizer

Отне около половин-един час

Model

Моделът е трениран за 14 епохи върху training dataset-а, разделен на бачове по 16 екземпляра, като dataset-а бива shuffle-нат преди всяка епоха.

На всеки 10 итерации се логва loss-а на модела върху текущия бач (от training set-а).

На всеки 2000 итерации се логва перплексията на модела върху dev-сета.

Име: Кристиян Иванов Гърчев ФН: 0MI0800290
Спец.: КН Курс: 3. Група: 5

Loss/Cost function

Използвам CrossEntropyLoss както е описано в “Attention is all you need”.

Отделните оценки на екземплярите се осредняват за numerical stability.

Овен това прилагаме label_smoothing=0.1 към loss-а, както е описано в “Attention is all you need”.

Optimizer

Използвам Adam optimizer с β_1=0.9 и β_2=0.98 и ε=10^{-9}, както е описано в “Attention is all you need”.

Основния Learning rate е 0.001, като в началото се клипва отдолу с 0.0005, понеже той се варира по начинът описан в “Attention is all you need” за **warmup_steps=4000**. В началото, рейта расте линейно (преди клипването), а след това намаля експоненциално към 0.

Всички параметри

```
vocab_size = 8_000
max_seq_len = 1532
num_blocks = 5
d_model = 512
num_heads = 8
dropout = 0.1
d_ff = 1536
label_smoothing = 0.1


learning_rate = 0.001
batch_size = 16
clip_grad = 5.0
max_epochs = 10
warmup_steps = 4000

log_every = 10
test_every = 2000

temperature = 0.2
```

Експерименти

Пробвал съм около 4-5 тренировки. Тренирал съм както в Google Colab, Kaggle Colab, така и локално.

В тях пробвах:

- num_blocks да е 5 или 6
- d_ff да е 2048 или 1536

- batch_size да е 32 или 16. Това указваше влияние при трениране в облака заради ограниченията по GPU памет в безплатния вариант на услугите.

Направих $O(n^2)$ вариант на generation-а, но на практика е по-бавно от наивния $O(n^3)$ вариант, затова на края реших да ползвам наивния.

Крайни Резултати

И двете метрики са изчислени върху тест сета.

Перплексия: 8.406167109942377

BLEU: 37.17

Как да ръннем проекта

Почти всичко е както в условието, само в началото вместо prepare, имаме:

- python run.py train_tokenizer
  - Изисква да има поддиректория model/tokenizer
  - Тренира токънайреза.
- python run.py tokenize_corpus
  - Изисква да има поддиректория data
  - Токенизира корпусите, които са ни дадени, и запазва резултатите в pickle файлове.

Източници

https://huggingface.co/learn/nlp-course/en/chapter6/5

https://medium.com/nearist-ai/word2vec-tutorial-the-skip-gram-model-c7926e1fdc09

https://medium.com/@hunter-j-phillips/positional-encoding-7a93db4109e6

https://marinafuster.medium.com/cross-entropy-loss-for-next-token-prediction-83c684fa26d5

https://pytorch.org/get-started/previous-versions/#linux-and-windows