Доклад

към курсов проект на тема

Невронен машинен превод чрез генеративен езиков модел

Търсене и извличане на информация

зимен семестър, 2024/2025 г.

Калоян Цветков
4MI0800017

14.02.2025г.

1 Резюме

Проектът цели реализация на модел за едновремено генериране на последователност на английски по зададен префикс, както и преводът ѝ от английски на български. Моделът е обучаван единствено върху предоставените тренировъчни данни. Предадените скриптове включват настройка на модела, довършване по дадена начална последователност и превод на цял корпус на български.

Сред предадените файлове присъства и т.н. *jupyter notebook* на име **TII_project.ipynb**, където има посочени конкретни команди за работа с проекта.

2 Архитектура

За предварителната обработка на текстовите данни се използва вграден токенизатор от библиотеката `tokenizers`. Вграденият токенизатор осигурява бърза и оптимизирана обработка на текста, като прилага предварително обучени алгоритми за сегментиране на думите. Началните опити с посимволно влагане и влагане по думи имаха проблеми съответно трудно научаване на зависимости между цели думи и прекалено голям речник. Експериментите със собствената имплементация на генериране на токени, кодиране и декодиране на текст показаха, че липсата на умения за паралелизация на токенизирането на множество данни увеличава значително времето за изпълнение и я правят неизползваема. Предоставен е скрипт `generateTokens.py`, генериращ конкретен брой токени, както и примерен изход (`tokens.json`) върху тренировъчното множество.

Моделът представлява невронна мрежа за езиково моделиране, базирана на трансформър архитектурата, предложена в **[2]**. Той използва трансформър енкодери, които включват механизъм за самообучение (*self-attention*), което позволява паралелната обработка на входни последователности.

Основната структура на модела включва слой `torch.nn.Embedding`, който преобразува думите в числови вграждания. След това тези вграждания се обогатяват с позиционен енкодинг, който се използва, за да се добави информация за реда на думите в последователността. Използваният позиционен енкодинг е имплементираният на упражнения чрез клас `PositionalEncoding`.

В същината на модела се намира енкодер, използващ архитектура *трансформър*, реализиран чрез `TransformerEncoder` от `PyTorch`, представляващ многослойно наслагване на `TransformerEncoderLayer`, реализиращ описаната в **[2]** архитектура. Важно е използването на маска, представена чрез горно триъгълна матрица, скриваща информация за следващи последователности. Според описаното в **[2]**, най-добро представяне се постига, когато размерността на скритият слой на *feed-forward* мрежата е 4 пъти по-голяма от размера на влагането, поради което така е постъпено и тук. Чрез линейна проекция, изхода от трансформъра се преобразува в разпределение на думите от речника. Използвана е емпирична крос ентропия като функция за загуба при обучението.

Моделът също така включва механизъм за генерация на текст, като използва семплиране с температурен коефициент за регулиране на разпределението на вероятностите при генериране на следваща дума. При стартиране на генерацията моделът използва даден префикс и генерира нови думи (чрез семплиране) последователно, докато не достигне зададения лимит или не генерира специален (*end*) токен.

3 Източници

- Официална документация към `TransformerEncoderLayer` от `PyTorch`: https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoderLayer.html
- Официална документация към `TransformerEncoder` от `PyTorch`: https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html

Литература

[1] Dzmitry Bahdanau, Kyunghyun Cho и Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". В: *3rd International Conference on Learning Representations (ICLR)*. 2015. URL: `https://arxiv.org/abs/1409.0473`.

[2] Ashish Vaswani и др. "Attention Is All You Need". В: *Advances in Neural Information Processing Systems*. Т. 30. 2017. URL: `https://arxiv.org/abs/1706.03762`.

4 Процес на обучение

Подготовката на корпус, а след това и обучение се извършват стандартно чрез аргументите съответно `prepare` и `train`/`extratrain` на програмата `run.py` при наличие на *json* файл с токени
(`tokensFileName`).
Ако такъв няма е необходимо предварително изпълнение на прорамата (`generateTokens.py`).

Обучението на предадения модел е извършено на графична карта *T4*, предоставена от платформата *Google Colab*. Тази платформа предлага безплатен достъп до *GPU* ресурси, което позволява по-бързо и ефективно обучение на дълбоки невронни мрежи.

Следните хиперпараметри бяха използвани за обучението на модела:

- **Параметри на модела:**
  - Размер на влагането (`d_model`) е зададен на 128
  - Размерът на скрития слой на *feed-forward* частта е зададен на 512
  - 8 глави на вниманието (`n_heads`)
  - 4 слоя на вниманието (`num_layers`)
- **Скорост на обучение:** Скоростта на обучение е зададена на 0.001 (`learning_rate`).

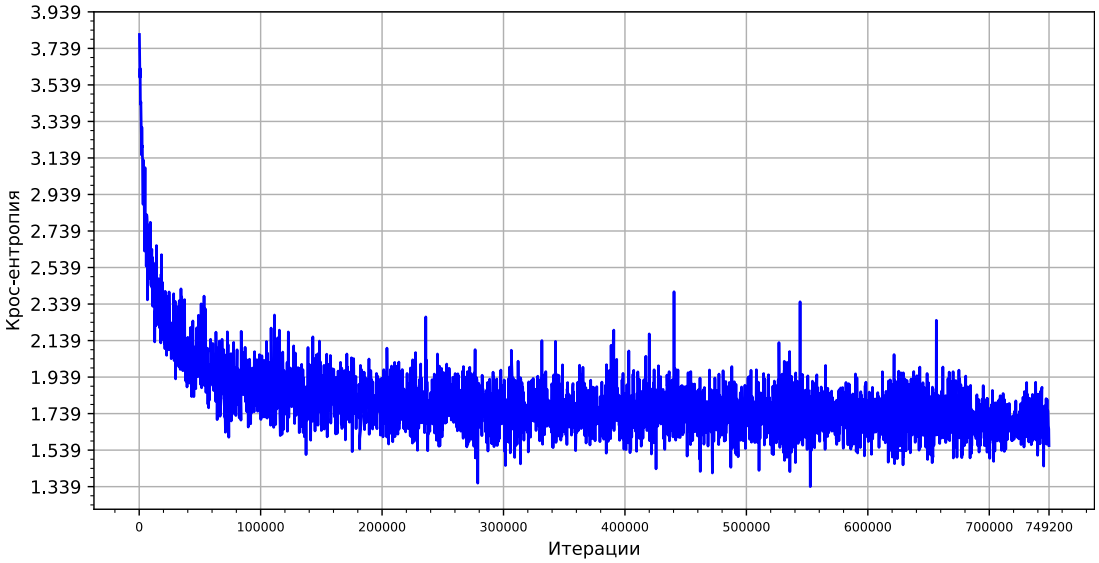
- **Размер на партидата:** 32 (`batchSize`).
- **Нормата на градиентите** се ограничава до максимална стойност от 10.0 (`clip_grad`).
- **Логване:** На всеки 10 стъпки от обучението (`log_every`) в конзолата се извежда текущото състояние на обучението, като то се записва на всеки 200 стъпки (`log_file_every`) в посочен файл (`log_filename`). Тестовете за перплексия се изпълняват също на всеки 200 стъпки (`test_every`).

5 Резултати

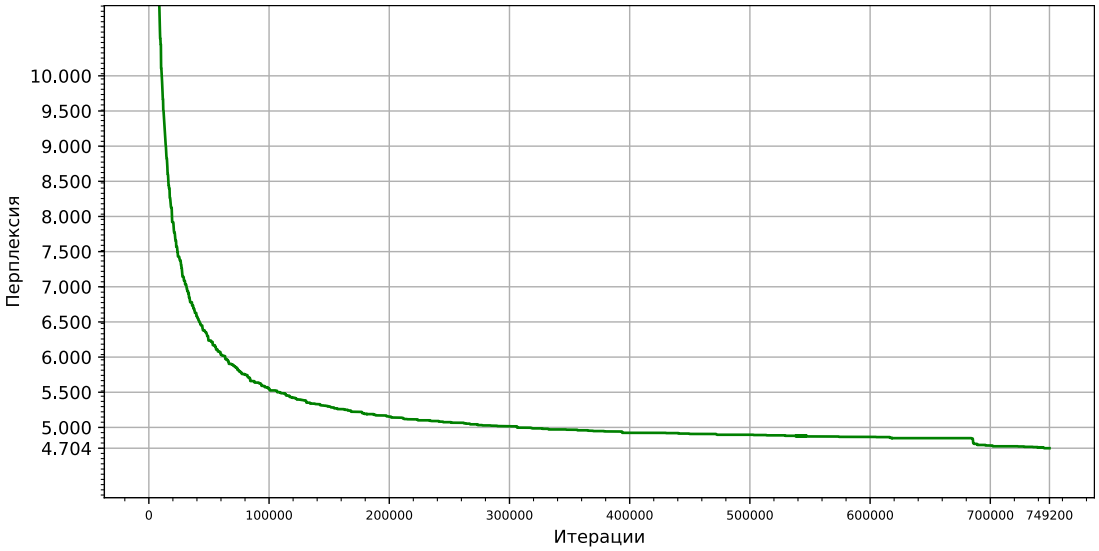
След приблизително 7 часа обучение моделът достига перплексия **4.704** и *BLEU* резултат от **41.865** върху тестовото множество данни. Резултатите по време на обучението според итерациите с посочения размер на партида са изложени в Таблица 1 на всеки 50000 итерации, както и на всяки 200 във Фигура 1 и Фигура 2.

Итерация	**Загуба**	**Думи/сек**	**Най-добра перплексия**
50 000	1.988	36890.72	6.235
100 000	1.977	38979.05	5.548
150 000	1.645	37719.01	5.296
200 000	1.773	34864.27	5.153
250 000	1.918	38086.78	5.074
300 000	1.704	35146.69	5.016
350 000	1.685	34397.45	4.967
400 000	1.780	31387.54	4.923
450 000	1.692	39530.54	4.907
500 000	1.725	39798.69	4.895
550 000	1.542	33088.33	4.871
600 000	1.676	37576.14	4.864
650 000	1.907	40292.59	4.847
650 000	1.731	30246.79	4.847
700 000	1.623	34977.17	4.743
749 200	1.563	36504.24	4.704

Таблица 1: Резултати от обучението



Фигура 1: Отношение на крос-ентропия спрямо броя итерации



Фигура 2: Отношение на перплексията спрямо броя итерации

6 Заключение

Моделът **NMTmodel**, постигащ BLEU резултат от приблизително 41.8, се спрява задоволително както със задачата за довършване на последователност на английски, така и със задачата за превод на последователност на английски на български, при това едновременно. Тематиката на трнировъчните данни ограничават модела в някаква степен до добър превод предимно на публицистични текстове.

Бъдещи подобрения включват паралелизация на токенизирането на корпус с цел употреба на токени, създадени от скрипт на автора, както и паралелизиран превод на корпус.