Търсене и извличане на информация. Приложение на дълбоко машинно обучение

Зимен семестър 2022/2023

Курсов проект на тема

Невронен машинен превод от български към английски език

Студент: Климент Стоянов Бербатов ФН: 81946

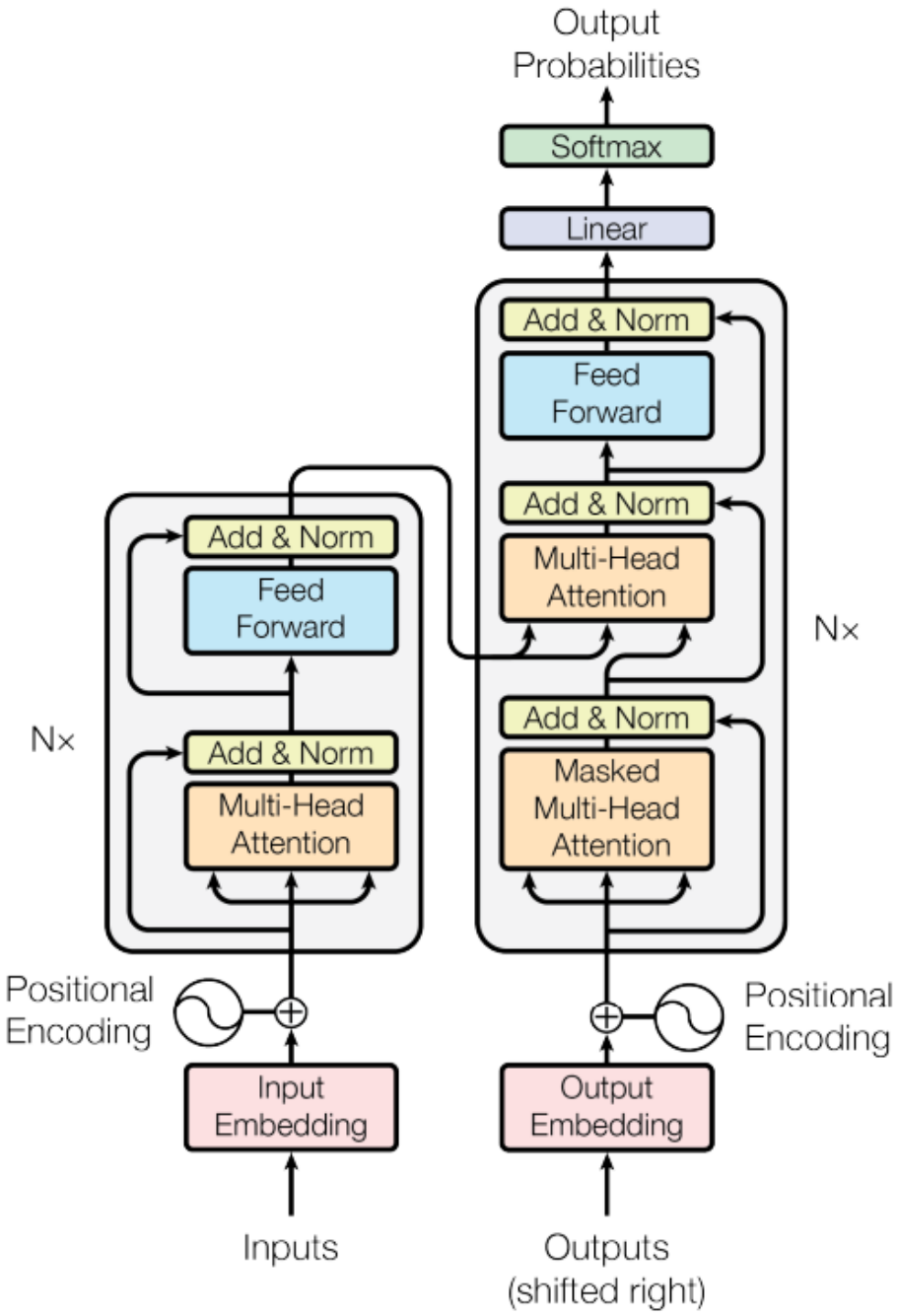
1. Увод

Проектът е реализация на трансформер архитектура на невронна мрежа за превод на текстове (изречения) от български към английски език. Голяма част от кодът в проекта е писан паралелно спрямо предоставената реализация [1], базирана на научната статия „Attention is all you need”[2].

2. Архитектура

Както вече споменах, избраната архитектура за НМ е трансформер. Тя сер разделя на 2 основни слоя – encoder и decoder. За реализацията на encoder-а се използва механизъм за внимание (или многоглаво внимание), различни линейни слоеве (positional encoding, feedforward) и нормализация на входа. Decoder-ът е реализиран почти по същия начин, с разликата че след първия слой за многоглаво внимание и преди feed forward има втори слой а многоглаво внимание, който приема изходът от encoder-а като параметър за ключовете и стойностите. Изходът на decoder-а преминава през един линеен слой и softmax за да

се получи вероятностно разпределение за изходът от НМ.



Фигура 1. Трансформер архитектура [2]

2.1 Multiheaded attention

Механизъмът за внимание представлява матрично произведение на три матрици – една за заявки(Q), една за ключове(K) и една за стойности(V). По-подробно тя е произведението на Q с K, след което се прилагат линейни слоеве за скалиране(Scale), маскиране(Mask) и вероятностно разпределение(softmax), и накрая

резултатът е умножен по V. Многоглавото внимание е просто конкатенацията на няколко „глави“ за внимание. Допълнително входовете на многоглавото внимание представляват тензори, на които първото измерение е размера на партидата. Малка разлика между кодът в този проект, и кодът от [1] е, че тук е пропуснат един dropout преди последното матрично умножение, защото не става ясно защо го има.

2.2 Encoder, Decoder

Имплементацията на encoder, съответно decoder, слоя е директно съответстваща на описанието му, като единствено е използван допълнителен клас EncoderLayer, съответно DecoderLayer, който представлява единичен слой encoder, съответно decoder, за да може по лесно да се реализира НМ с няколко слоя Encoder/Decoder.

2.3 NMTmodel

Този клас обединява реализациите на encoder и decoder модулите в цялостната архитектура на НМ. Освен метода за подравняване на партида има допълнителни методи за създаване на маски за двата входа на модела – този за входния език, и този за изходния, и метод за превеждане на изречение.

Последният метод наподобява на циклично прилагане на forward метода, като за вход първоначално приема само едно изречение на входния език и изкуствено подадено изречение от само начален токен и всеки път резултатът, или индексът с най-висока вероятност (greedy search), се добавя до второто изречение, докато не се генерира токен за край на изречение, или не се достигне лимита за дължина(1000 по подразбиране).

3. Параметри

Хиперпараметрите на моделът и оптимизационния алгоритъм са избрани подобни на тези по подразбиране дадени в [1] с малки изключения и след тестване на различни комбинации от различни стойности:

- Nx = 4, брой слоеве encoder/decoder (в [1] този параметър е 6, но поради странно държане, а именно астрономически голяма

перплексия по време на обучение от сорта на няколко хиляди, на програмата е намален до 4).

- n_heads = 4, брой глави на многоглавото внимание.
- d_model = 128, размер на скритите вектори във всеки слой на encoder/decoder.
- pf_size = d_model * 2, размер на скритите вектори в feedforward модулите.
- dropout = 0.3/0.2/0.1, стойности за dropout параметъра използван навсякъде из кода.
- lr = 0.001/0.0005/0.0003, стойностите за lr параметъра за adam алгоритъма.
- Epochs = 10, броя епохи едновременно обучение (общо 40 епохи, или 4 обучения).

  Останалите параметри са оставени както са, защото нямат влияние върху резултатите на модела.

4. Обучение

Обучението на модела стана посредством средата Google Colab и предоставените от Google графични ядра. Обучението продължи общо 40 епохи. Първите 10 епохи обучение са използвани dropout = 0.3 и lr=0.001, като на всеки следващи 10 епохи съответните параметри са намалявани. При последните 15 епохи, т.е. от 25 до 30 и от 31 до 40, алгоритъмът за обучение няколко пъти използва механизма за търпение и се върна на предишен модел, поради което моделът не е бил обучаван повече епохи. В архива на проекта е добавена и тетрадката от Google Colab, с която е обучаван моделът.

5. Резултати

Резултатите на модела спрямо двата корпуса – dev и train, са следните:

Корпус	**Перплексия**	**BLEU score**
dev	3.695	43.342
test	3.982	41.182

Допълнително резултатите могат да се видят във файловете bleu_score.txt и perplexity.txt.

6. Източници

[1]. Ben Trevett, “6 – Attention is all you need.ipynb” 2021, https://github.com/bentrevett/pytorch-seq2seq/blob/master/6%20-%20Attention%20is%20All%20You%20Need.ipynb

[2]. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[3]. Pytorch Documentation, https://pytorch.org/docs/stable/index.html