Week 3 Assignment

Generative AI

Saarland University – Winter Semester 2024/25

Martínez
7057573
cama00005@stud.uni-saarland.de

1 Exercise Assignment: E1

The algorithm to compute the attention vector $\mathbf{a}$ as output from the Multi-Head Attention Layer is the following:

Algorithm 1: Multi-Head Attention Layer

Input : $N-1$ embedding vectors of dimension d, $\mathbf{e}_{t-N+1}, \ldots, \mathbf{e}_{t-2}, \mathbf{e}_{t-1}$
Output : Attention vector $\mathbf{a}$
MHA $(\mathbf{e}_{t-N+1}, \ldots, \mathbf{e}_{t-2}, \mathbf{e}_{t-1})$

foreach *attention head indexed* $i < \eta_{\mathbf{a}}$ **do**

foreach *pair of* $\mathbf{e}$ *vectors indexed* m **do**

$$\text{score}(\mathbf{e}_{t-1}, \mathbf{e}_m) \leftarrow \frac{W_i^q \mathbf{e}_{t-1} \cdot W_i^k \mathbf{e}_m}{\sqrt{d_k}}$$

where $W_i^q \in \mathbb{R}^{d_k \times d}$ and $W_i^k \in \mathbb{R}^{d_k \times d}$, and are the query and key matrices respectively, of the i-th attention head.

Compute attention weights α_m using *softmax* over all scores:

$$\alpha_m \leftarrow \text{softmax}_m \left(\{\text{score}(\mathbf{e}_{t-1}, \mathbf{e}_j) \mid j = t-N+1, \ldots, t-1\} \right)$$

Compute the attention vector $\mathbf{a}_i$ as a weighted sum given by:

$$\mathbf{a}_i \leftarrow \sum_{m=t-N+1}^{t-1} \alpha_m \, W_i^v \mathbf{e}_m, \quad \text{s.t. } \mathbf{a}_i \in \mathbb{R}^{d_v}$$

where $W_i^v \in \mathbb{R}^{d_v \times d}$ is the value matrix of the i-th attention head.

Concatenate the outputs $\mathbf{a}_i$ of all η_a attention heads:

$$\mathbf{a} \leftarrow \left[\mathbf{a}_1, \ldots, \mathbf{a}_{\eta_a}\right], \quad \text{s.t. } \mathbf{a} \in \mathbb{R}^{\eta_a d_v}$$

return $\mathbf{a}$;

---

2 Exercise Assignment: E2

As we saw in Week 2's Assignment: E3, the number of parameters introduced by the single attention head in the *Simple Feed-forward Neural Language Model* are the number of parameters in the query, key and value matrices, i.e., W^q, W^k and W^v respectively, which came up to be equal to $2d_k d + d_v d$. Since there are η_a attention heads, the total number of parameters introduced by the attention layer is naturally $\eta_a(2d_k d + d_v d)$.

3 Exercise Assignment: E3

Consider the transformer block in Week 3's lecture slides #22-29.

- **[Input vectors]** There are no parameters introduced by the input vectors.
- **[Attention layer]** As we saw in **E2**, the number of parameters introduced by the attention layer is $\eta_a(2d_k d + d_v d)$.
- **[Attention projection, sum, norm]** Firstly, we introduce the number of parameters in the projection matrix $W^{\text{AProj}} \in \mathbb{R}^{d \times \eta_a d_v}$. Secondly, we introduce the two learnable parameters, namely γ^{ANorm} and β^{ANorm}, which appear in the *LayerNorm* operation. Thus, the total number of parameters introduced by the attention projection, sum and normalization steps is $d\eta_a d_v + 2$.
- **[Feedforward layer]** For each neural unit i, we have a weight vector $\mathbf{w}_i \in \mathbb{R}^d$ and a bias term $b_i \in \mathbb{R}$. Since we have d_h neural units, the total number of parameters introduced by the feedforward layer is $d_h(d+1)$.
- **[Feedforward projection, sum, norm]** The number of parameters are given by the number of elements in $W^{\text{FfProj}} \in \mathbb{R}^{d \times d_h}$; $\mathbf{b}^{\text{FfProj}} \in \mathbb{R}^d$, and the two learnable parameters γ^{FfNorm} and β^{FfNorm} in the *LayerNorm* operation. Thus, the total number of parameters introduced by this step is $d(d_h+1)+2$.
- **[Output vectors]** There are no parameters introduced by the output vectors.

Finally, the final number of parameters is given by the sum of all previously introduced parameters, as follows:

$$\begin{aligned}
\#\text{ parameters} &= \underbrace{\eta_a(2d_k d + d_v d)}_{\text{Attention Layer}} + \underbrace{d\eta_a d_v + 2}_{\text{Attention projection, sum, norm}} + \underbrace{d_h(d+1)}_{\text{Feedforward layer}} + \underbrace{d(d_h+1)+2}_{\text{Feedforward projection, sum, norm}} \\
&= \eta_a(2d_k d + d_v d) + d\eta_a d_v + 2d_h d + d_h + d + 4
\end{aligned}$$

4 Exercise Assignment: E4

Consider the transformer architecture in Week 3's lecture slides #31–33.

1. The input to this transformer architecture is given by $N-1$ words denoted by $\mathbf{x}_{t-N+1}, \ldots, \mathbf{x}_{t-2}, \mathbf{x}_{t-1}$ and represented as one-hot encoded vectors of size $|V|$. These one-hot encoded vectors are first converted to embedding vectors of size d using the embedding matrix $E \in \mathbb{R}^{d \times |V|}$ before being processed by the transformer blocks.

   $\Rightarrow$ Thus, the number of parameters introduced so far are $d \times |V|$.

2. After that, they will get processed by the L transformer blocks.

   $\Rightarrow$ If the number of parameters in one transformer block is B (as calculated in **E3**), then the total number of parameters introduced here are $L \times B$.

3. The output from these transformer blocks is then converted back to a vector of size $|V|$ using the unembedding matrix $U \in \mathbb{R}^{d \times |V|}$, which has shared weights with E.

   $\Rightarrow$ Since the unembedding matrix U has shared weights with the embedding matrix E, these parameters are already accounted for in **1.**.

Thus, the total number of parameters in this transformer architecture in terms of L, B, $|V|$, and d is given by[1]:

$$d \times |V| + L \times B$$

[1]Note that the context window size N does not influence the number of parameters in the transformer architecture.

5 Exercise Assignment: E5

OpenAI's ***GPT-4o-mini model*** performed the following on the translation task of the Old Romanian sentence "Ie' de-nbucă oarişce!":

- **Zero-shot:** $1/3$ times correct. See Figure 1 for the results.
- **Few-shot:** $3/3$ times correct. See Figure 2 for the results.

The two different prompting strategies, zero-shot and few-shot, show that the model performs better when given a few examples of the task it is supposed to perform. This is because the model can learn the patterns and structure of the task from the examples, and then apply this knowledge to new examples. In the zero-shot case, the model has to rely solely on its pre-trained knowledge, which may not be sufficient to perform the task accurately, whereas the few-shot case provides the model with additional information that helps it make a better translation of the given sentence.

6 Exercise Assignment: E6

Table 1 shows the results for the Basic vs. Chain-of-Thought (CoT) Prompting Strategies on Task-A, Task-B, and Task-C. As we can see, OpenAI's ***GPT-4o-mini model*** fails on all three tasks when using the Basic Prompting Strategy, but has 100% accuracy when using CoT. This shows that a "step-by-step" approach is more effective, at least in these particular type of tasks, than a "one-shot" approach. It forces the model to develop a structured answer by breaking down the task into smaller, more manageable steps, that it can handle[2]. See Figure 3 for an example of the CoT Prompting Strategy in action.

Table 1: Results for Basic vs. Chain-of-Thought (CoT) Prompting Strategies on the given tasks.

Task	**Basic Prompting**			**CoT Prompting**		
	Trial 1	*Trial 2*	*Trial 3*	*Trial 1*	*Trial 2*	*Trial 3*
Task-A	900	900	900	990	990	990
Task-B	72	72	72	1080	1080	1080
Task-C	660	720	740	520	520	520

7 Exercise Assignment: I3

First of all, OpenAI's GPT-2 XL is the 1.5B parameter version of GPT-2, pretrained on English language using a causal language modeling (CLM) objective. One of its intended use case, as OpenAI mentions in its model card, is "(...) creative writing and art: exploring the generation of creative, fictional texts; aiding creation of poetry and other literary art (...)" [1]. This means that the model is not specifically designed for completing Shakespearean texts, but it can still *creatively* generate text that is coherent and stylistically similar to Shakespeare's works, as seen in Figure 4. It is worth noting that all of the completions made by GPT-2 XL were related to the original prefix. Although in some cases, it started repeating lines over and over again, until the end of the completion.

On the other hand, Microsoft's Phi-1.5 is a custom model with 1.3B parameters, whose "training involved a variety of data sources, including subsets of Python codes from The Stack v1.2, Q&A content from StackOverflow, competition code from code contests, and synthetic Python textbooks and exercises generated by gpt-3.5-turbo-0301" [2]. Also, as its intended use mentions in the Hugging Face model card, "(...) Phi-1.5 is best suited for prompts using the Q&A format, the chat format, and the code format. Note that Phi-1.5, being a base model, often produces irrelevant text following the

[2] I am specifically avoiding the words "reason" and "thinking", because it is highly debatable whether this is thinking at all, at least in the *human* sense. Based on my knowledge and personal interpretation of it, it is nothing but a guided answer, without any *human*-like reasoning behind it, finding the patterns and the most-probable pieces of text from the *billions* it was once trained on to minimize a loss function, that end up in the right answer. If it actually did reason, the one-shot answer would not be any different from the CoT answer.

main answer (...)". Thus, unlike GPT-2 XL, Phi-1.5 is more suited for Q&A and code generation tasks given its training data, which explains why the completion (a) deviated greatly from the original prefix, as seen in Figure 5. These kinds of hallucinations were very common in the completions made by Phi-1.5, but some of them did seem to be related to some degree, as seen in completion (b) of Figure 5.

Acknowledgements

This week's slides.

References

[1] OpenAI. Gpt-2 model card. https://github.com/openai/gpt-2/blob/master/model_card.md. Accessed: 2024-11-06.

[2] Hugging Face. Microsoft phi-1 model card. https://huggingface.co/microsoft/phi-1. Accessed: 2024-11-06.

Annex

```
The Old Romanian sentence "Ie’ de-nbucă oarişce!" translates to
English as:  "Take a bite of something!" or "Eat something!"

The phrase "Ie’ de-nbucă" is an older or dialectal expression that
suggests taking a bite or eating, and "oarişce" means "something" or
"anything."
```

(a) First result.

```
The Old Romanian sentence "Ie’ de-nbucă oarişce!" can be translated
to modern English as:

"Go and enjoy anything!" or "Go and have fun with whatever!"

Note that Old Romanian had a more archaic structure and vocabulary,
so the exact meaning can vary slightly depending on context.  The
phrase seems to be an imperative, urging someone to enjoy themselves
or engage with something in a carefree manner.
```

(b) Second result.

```
The Old Romanian sentence "Ie’ de-nbucă oarişce!" translates to:

"Let anything happen!" or "Let whatever happen!"

In this context, "Ie’" is a form of "a ieși" (to happen), "de-nbucă"
can be interpreted as a colloquial expression meaning "whatever" or
"anything," and "oarişce" is an archaic form meaning "something" or
"anything." The overall sense is a kind of expression of accepting
whatever comes.
```

(c) Third result.

Figure 1: Zero-shot translation by OpenAI’s ***GPT-4o-mini model***.

```
English:  Take and eat anything!
```

(a) First result.

```
English:  Take and eat anything!
```

(b) Second result.

```
English:  Eat whatever you want!
```

(c) Third result.

Figure 2: Few-shot translation by OpenAI’s ***GPT-4o-mini model***.

```
Let's break this down step by step:

1. Calculate the Down Payment: The total cost of the laptop is
$1000, and the shop requires a 20% down payment. To find the down
payment amount:
                Down Payment = 1000 × 0.20 = 200
So, the down payment is $200.

2. Tara's Additional Down Payment: Tara wants to pay an additional
$20 on top of the required down payment of $200. Therefore, her total
down payment will be:
              Total Down Payment = 200 + 20 = 220

3. Calculate the Remaining Balance: The total cost of the laptop is
$1000, and Tara has made a down payment of $220. So, the remaining
balance to be paid is:
              Remaining Balance = 1000 − 220 = 780

4. Monthly Installment Payment: The computer shop accepts monthly
installments of $65. Tara will pay $65 each month. After 4 months,
she will have made:
              Total Paid in 4 Months = 65 × 4 = 260

5. Calculate the Remaining Balance After 4 Months: After 4 months
of payments, Tara has paid $260 toward the remaining balance of $780.
So, the new balance is:

                New Balance = 780 − 260 = 520
Final Answer:
After paying for 4 months, Tara's remaining balance will be $520.
```

Figure 3: Example of OpenAI's ***GPT-4o-mini model*** response to Task-C using the Chain-of-Thought (CoT) Prompting Strategy. See E6.

```
===PREFIX===
look in thy glass and tell the face thou viewest

===COMPLETION===
look in thy glass and tell the face thou viewest,

That thou art not the same man that thou art seen to be.

And if thou wilt not, I will tell thee the truth,

That thou art not the same man that thou art seen to be.

And if thou wilt not, I will tell thee the truth,

That thou art not the same man that thou art seen to be.

And if thou wilt not
```

Figure 4: One of the prefixes completion by OpenAI's GPT-2 XL.

```
===PREFIX===
O, what a noble mind is here o'erthrown!

===COMPLETION===
O, what a noble mind is here o'erthrown!

(Scene:  The Alien, intrigued by the conversation, approaches the
group.)

Alien:  Greetings, Earthlings!  I am Zog, an extraterrestrial being
from a distant galaxy.  I have been observing your planet and its
inhabitants.  Your concept of "daily life" is fascinating.  Can you
explain more about the role of a janitor in your society?

Earthling 1:  Welcome, Zog
```

(a) First example, which shows how the model can easily diverge from the original text.

```
===PREFIX===
Thou canst not see one wrinkle in my brow,

===COMPLETION===
Thou canst not see one wrinkle in my brow, but I can see many.

(Chorus)
Oh, the world is a stage, and we're all just players, In this grand
performance, we're all just trying to find our place.  But sometimes,
it feels like we're all just trying to survive, In this chaotic world,
where we're all just trying to survive.

(Verse 3)
I wake up every morning
```

(b) Second example, which shows a better completion with related content.

Figure 5: Examples of prefixes completion by Microsoft's Phi-1.5.