高级人工智能课程汇报

信息科学与工程学院
Gu Rui
220220942871

June 25 2023

Contents

1 Attention exploration

Multi-headed self-attention is the core modeling component of Transformers. In this question, we' ll get some practice working with the self-attention equations, and motivate why multi-headed self-attention can be preferable to single-headed self-attention. Recall that attention can be viewed as an operation on a *query* $q \in \mathbb{R}^d$, a set of *value* vectors $\{v_1, ..., v_n\}$, $v_i \in \mathbb{R}^d$, and a set of *key* vectors $\{k_1, ..., k_n\}$, $k_i \in \mathbb{R}^d$, specified as follows:

$$c = \sum_{i=1}^{n} v_i \alpha_i \tag{1}$$

$$\alpha_i = \frac{\exp(k_i^T q)}{\sum_{j=1}^{n} \exp(k_j^T q)} \tag{2}$$

with $_i$ termed the "attention weights". Observe that the output $c \in \mathbb{R}^d$ is an average over the value vectors weighted with respect to $_i$.

(a) (4 points) **Copying in attention.** One advantage of attention is that it' s particularly easy to "copy" a value vector to the output c. In this problem, we' ll motivate why this is the case.

i. (1 point) **Explain** why α can be interpreted as a categorical probability distribution.

Answer: The alpha weights α_i can be interpreted as a categorical probability distribution because they are obtained by normalizing the exponential values of the inner products between the query vector q and the key vectors k_i. This normalization ensures that the alpha weights sum up to 1, making them represent probabilities.

ii. (2 points) The distribution α is typically relatively "diffuse"; the probability mass is spread out between many different α_i. However, this is not always the case. **Describe** (in one sentence) under what conditions the categorical distribution α puts almost all of its weight on some α_j , where $j \in \{1, \ldots, n\}(i.e. \quad \alpha_j \gg \sum_{i \neq j} \alpha_i)$. What must be true about the query q and/or the keys $\{k_1, \ldots, k_n\}$?

Answer: The categorical distribution α puts almost all of its weight on some α_j when there is a strong alignment between the query vector q and a specific key vector k_j. This occurs when the inner product $k_j^T q$ is much larger than the inner products $k_i^T q$ for $i \neq j$.

iii. (1 point) Under the conditions you gave in (ii), **describe** what properties the output c might have.

Answer: Under the conditions described in (ii), the output vector c will be heavily influenced by the value vector v_j associated with the key vector k_j that received the majority of the attention weight. The contribution of other value vectors to the output will be relatively small.

iv. (1 point) **Explain** (in two sentences or fewer) what your answer to (ii) and (iii) means intuitively.

Answer: This means that in multi-headed self-attention, where multiple attention heads are used, each head can attend to different aspects or features of the input. When a single head receives a significantly higher attention weight, it indicates that it is focusing on a specific important aspect. By using multiple attention heads, the model can simultaneously capture and attend to different relevant information, leading to richer and more comprehensive representations.

(b) (7 points) **An average of two**. Instead of focusing on just one vector v_j , a Transformer model might want to incorporate information from *multiple* source vectors. Consider the case where we instead want to incorporate information from **two** vectors v_a and v_b, with corresponding key vectors k_a and k_b.

i. (3 points) How should we combine two d-dimensional vectors v_a, v_b into one output vector c in a way that preserves information from both vectors? In machine learning, one common way to do so is to take the average: $c = \frac{1}{2}(v_a + v_b)$. It might seem hard to extract information about the original vectors va and v_b from the resulting c, but under certain conditions one can do so. In this problem, we' ll see why this is the case.

Suppose that although we don' t know v_a or v_b, we do know that v_a lies in a subspace A formed by the m basis vectors $\{a_1, a_2, \ldots, a_m\}$, while v_b lies in a subspace B formed by the p basis vectors $\{b_1, b_2, \ldots, b_p\}$. (This means that any va can be expressed as a linear combination of its basis vectors, as can v_b. All basis vectors have norm 1 and orthogonal to each other.)

Additionally, suppose that the two subspaces are orthogonal; i.e. $a_j^\top b_k = 0$ for all j, k. Using the basis vectors $\{a_1, a_2, \ldots, a_m\}$, construct a matrix M such that for arbitrary vectors $v_a \in A$ and $v_b \in B$, we can use M to extract v_a from the sum vector $s = v_a + v_b$. In other words, we want to construct M such that for any v_a, v_b, $M_s = v_a$.

Note: both M and v_a, v_b should be expressed as a vector in $\mathbb{R}^d$, not in terms of vectors from A and B.

Hint: Given that the vectors $\{a_1, a_2, \ldots, a_m\}$ are both *orthogonal* and *form a basis* for v_a, we know that there exist some $c_1, c_2, \ldots, c_m$ such that $v_a = c_1 a_1 + c_2 a_2 + \ldots + c_m a_m$. Can you create a vector of these weights c?

Answer:

ii. (4 points) As before, let v_a and v_b be two value vectors corresponding to key vectors k_a and k_b, respectively. Assume that (1) all key vectors are orthogonal, so $k_i^\top k_j$ for all $i \neq j$; and (2) all key vectors have norm 1.[1] **Find an expression** for a query vector q such that $c \approx \frac{1}{2}(v_a + v_b)$.[2]

[1] Recall that a vector x has norm 1 if $x^\top x = 1$.

[2] Hint: while the softmax function will never exactly average the two vectors, you can get close by using a large scalar multiple in the expression.

2 Pretrained Transformer models and knowledge access (35 points)

You'll train a Transformer to perform a task that involves accessing knowledge about the world -knowledge which isn't provided via the task's training data (at least if you want to generalize outside the training set). You'll find that it more or less fails entirely at the task. You'll then learn how to pretrain that Transformer on Wikipedia text that contains world knowledge, and find that finetuning that Transformer on the same knowledge-intensive task enables the model to access some of the knowledge learned at pretraining time. You'll find that this enables models to perform considerably above chance on a held out development set.

The code you're provided with is a fork of Andrej Karpathy's minGPT. It's nicer than most research code in that it's relatively simple and transparent. The "GPT" in minGPT refers to the Transformer language model of OpenAI, originally described in this paper [1].

As in previous assignments, you will want to develop on your machine locally, then run training on HuaWei Could. You'll need around 5 hours for training, so budget your time accordingly!

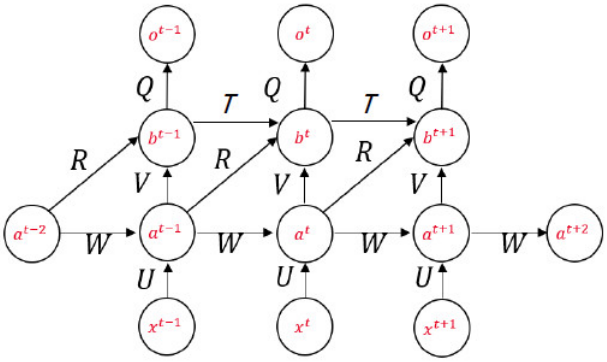
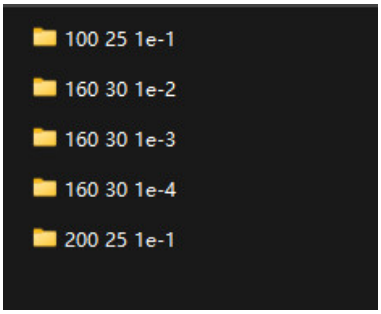
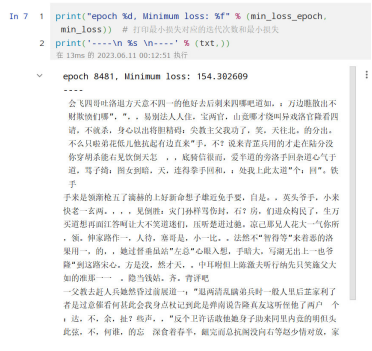


Figure 1: WRNN 网络结构图。

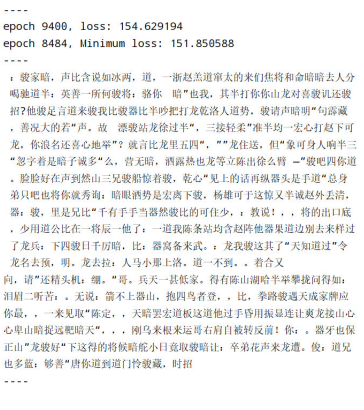
Fig. 2d和 Fig. 2e在 RNN 中有对比实验，loss 的值都比 RNN 的低，或许能够说明在 hidden_layer 不是很深的情况下，RNN 的效果可能要更好，在不考虑学习率的情况下，从 Fig. 2f和 Fig. 2c的结果中可以推断，随着 hidden_layer 的增加，WRNN 的效果应该是要优于 RNN，但是降低学习率对于 RNN 和 WRNN 来说会导致文本出现乱码的情况增多，或者只生成乱码。



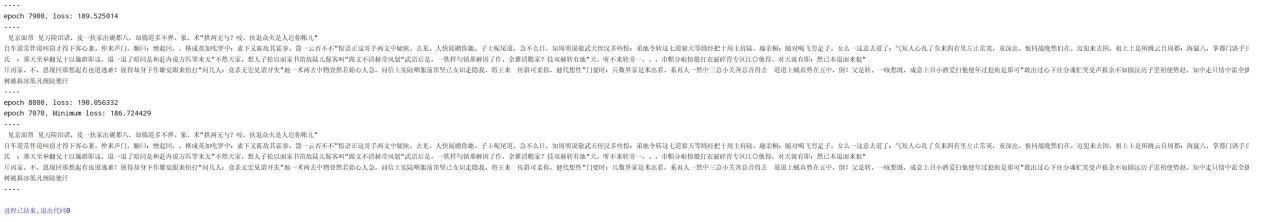
(a) parameter_tuning



(b) 100 25 1e-1



(c) 200 25 1e-1



(d) 160 30 1e-2

(e) 160 30 1e-3

(f) 160 30 1e-4

Figure 2: WRNN 下实现的四组调参结果。因为后期修改了调参策略（在第一题中提到），所以相对比较全面的对比结果还没有（限于作业提交时间关系，后续可以把结果补上，这里图片中的结果只做一个展示和说明。跑一次，找到一个最优的参数需要 3h 左右）

3 一些说明

完整的项目在已经上传到 GitHub[3]中, 上面有个人的提交记录和代码的详细修改记录[4]。上面提到的图片都是用 Jupyter Notebook 跑，但是因为题目限定了无法使用第三方 RNN 库，所以未使用 torch，因而无法使用 GPU 训练模型。而使用 CPU 训练的速度，加上手动调参，过于缓慢了，所以对比实验没做完全。考虑到，时间的因素，所以完整的训练过程我并没有使用 Jupyter Notebook，RNN.py 和 WRNN.py 一直还在在训练（见 Fig. 3）后续完整的结果可以通过邮箱发给老师。

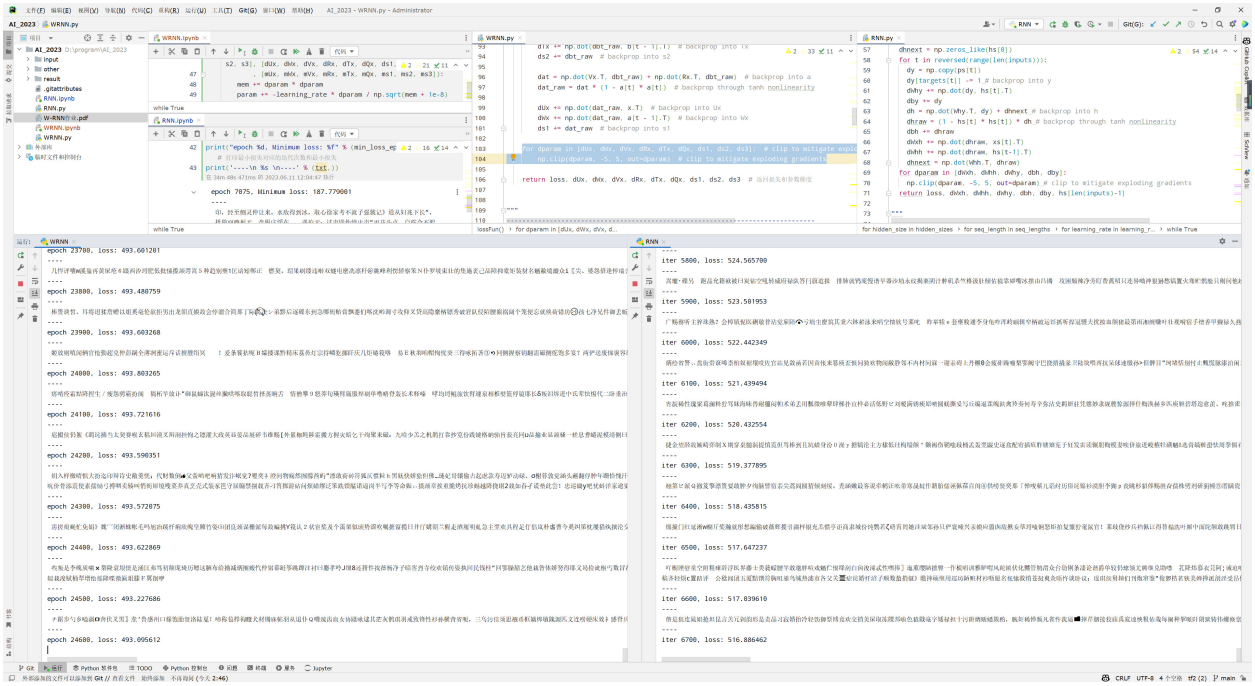


Figure 3: 运行 RNN.py 和 WRNN.py 的 PyCharm 界面。

References

[1] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding with unsupervised learning. Technical report, OpenAI (2018).

[3]https://github.com/npukujui11/AI_2023

[4]https://github.com/npukujui11/AI_2023/commits/main

[2] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research 21, 140 (2020), 1–67.

[3] Tay, Y., Bahri, D., Metzler, D., Juan, D.-C., Zhao, Z., and Zheng, C. Synthesizer: Rethinking self-attention in transformer models. arXiv preprint arXiv:2005.00743 (2020).