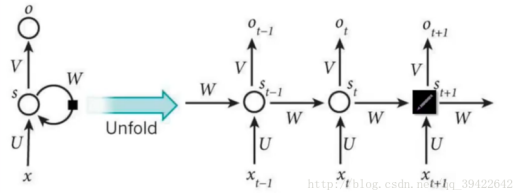
语言模型

1. 实验背景知识

本实验主要探究不同语言模型的性能和参数的影响。

本次实验使用了不同的模型结构，训练基于神经网络的语言模型。下面对这些模型做一个简要的介绍。

1.1. RNN



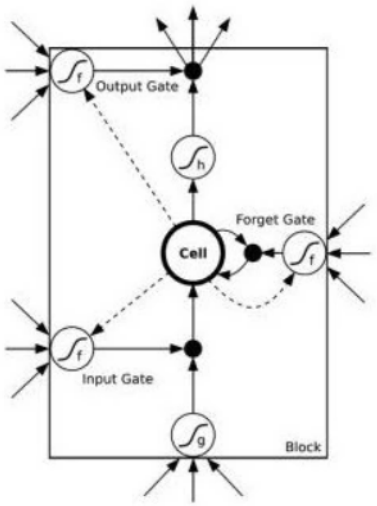
循环神经网络（Recurrent Neural Network，RNN）是一种用于处理序列数据的神经网络模型。与传统的前馈神经网络不同，RNN 具有循环连接，通过循环连接和隐藏状态来建模序列数据中的时间依赖关系，使其能够对序列数据中的上下文信息进行建模。

RNN 的基本思想是通过将前一时刻的隐藏状态作为当前时刻的输入，来捕捉序列数据中的时间依赖关系。这种循环连接使得 RNN 具有记忆性，能够在处理序列数据时保留之前的信息，并将其传递到后续的时间步。

RNN 的一个关键组成部分是隐藏状态（hidden state），它是网络在每个时间步的内部表示。隐藏状态可以看作是网络对之前观察到的序列数据的总结和记忆。在每个时间步，RNN 根据当前输入和前一时刻的隐藏状态来计算新的隐藏状态。这样的结构决定了 RNN 是 nonparallel 的并且效果可能比不上自注意力机制。

RNN 的难点在于，难以处理长期的依赖关系，此时 RNN 会遇到梯度消失或者爆炸的问题。为了解决这个问题，产生了新的模型 LSTM 与 GRU 等。

1.2. LSTM



LSTM，长短期记忆网络（Long Short-Term Memory）是一种循环神经网络（RNN）的变体，专门设计用于解决传统 RNN 中的梯度消失和梯度爆炸问题，以更好地捕捉长期依赖关系。

LSTM 结构的核心是引入了门控机制，能够在处理序列数据时选择性地记忆、遗忘和输出信息。它通过三个关键的门来实现这一机制：遗忘门（Forget Gate）、输入门（Input Gate）和输出门（Output Gate）。

遗忘门：遗忘门决定了前一时刻的隐藏状态中哪些信息需要被遗忘。它通过一个 sigmoid 激活函数来输出一个 0 到 1 之间的值，表示每个隐藏单元中的信息保留程度。0 表示完全遗忘，1 表示完全保留。

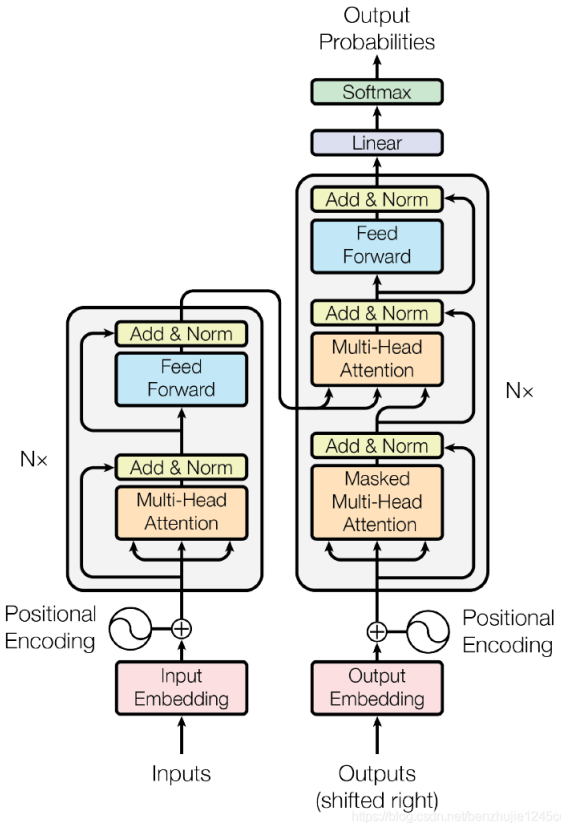
输入门：输入门决定了当前时刻的输入中哪些信息需要被记忆。它通过一个 sigmoid 激活函数来输出一个 0 到 1 之间的值，表示每个隐藏单元中的信息输入程度。0 表示完全忽略，1 表示完全接受。

输出门：输出门决定了当前时刻的隐藏状态中哪些信息需要输出。它通过一个 sigmoid 激活函数来输出一个 0 到 1 之间的值，表示每个隐藏单元中的信息输出程度。0 表示完全屏蔽，1 表示完全输出。

LSTM 的另一个结构是记忆细胞（Menmory Cell），用于在时间步之间传递和存储信息。细胞状态可以看作是 LSTM 网络的记忆单元，可以在不同时间步长期地保留和传递信息。

通过这些门控机制和细胞状态，LSTM 能够更好地捕捉长期依赖关系。解决了传统 RNN 的困难。

1.3. Trasformer



Transformer 是一种基于自注意力机制（self-attention）的神经网络架构，用于处理序列数据，相比于传统的循环神经网络（RNN）和卷积神经网络（CNN），Transformer 在处理长序列数据时具有更好的并行性和捕捉长距离依赖关系的能力。

Transformer 的核心思想是通过自注意力机制来建模序列数据中的上下文关系。自注意力机制允许模型在计算每个位置的表示时，同时考虑到序列中其他位置的信息。这使得模型能够更好地理解每个位置与其他位置之间的关系，而不仅仅依赖于固定的局部上下文窗口。

Transformer 由编码器（Encoder）和解码器（Decoder）组成。编码器负责将输入序列转换为高维表示，解码器则根据编码器的输出和之前的预测来生成目标序列。在编码器和解码器中，Transformer 由多个相同的层堆叠而成。每个层都包含了多头自注意力机制和前馈神经网络。自注意力机制允许模型在不同的表示子空间中对输入进行加权组合，以捕捉不同的语义信息。前馈神经网络则通过多层感知机对每个位置的表示进行非线性变换。

除了自注意力机制和前馈神经网络，Transformer 还引入了残差连接和层归一化，以加速训练和提高模型性能。残差连接允许信息在层之间直接传递，有助于缓解梯度消失问题。层归一化则对每个层的输出进行归一化，有助于加速训练和提高模型的泛化能力。

1.4. 评估指标

我们的评估指标是 PPL，这个值越低表明我们的模型性能更好。

2. 实验过程

我首先阅读了几个文件的代码内容，查看模型的结构和超参数。

模型中实现了一个 RNNModel，其中的 RNN 层定义了一个 RNN 神经网络模型，支持 LSTM, GRU, RNN_TANH,RNN_RELU。另外一个模型 Transformer 由另一个类 TransformerModel 定义。

接下来我查看超参数列表，了解我可以调整的超参数有哪些。

我计划调整和探究下面的超参数及其影响：

- 模型结构
  - RNN：Recurrent Neural Network，一种序列模型，具有记忆功能。

  - GRU：Gated Recurrent Unit，一种比 RNN 更复杂的序列模型，具有更好的记忆保持性能。
  - LSTM：Long Short-Term Memory，一种比 GRU 更复杂的序列模型，具有更强的记忆保持性能，但也更容易过拟合。
  - Transformer：一种基于 attention 机制的序列模型，能够处理较长的序列并且可并行化。
- 隐层大小
  - emsize：网络的嵌入维数，通常是输入词向量的维数，也称为嵌入大小。
  - nhid：隐藏层的大小，也称为隐藏状态的大小。
- 训练用到的参数
  - lr：学习率，控制每次更新的步长。
  - clip：控制梯度的最大范数，用于防止梯度爆炸。
  - epochs：训练循环次数。
  - batch_size：将数据集分成的批次大小。这会影响模型每次迭代的样本数量，过大可能会导致内存不足，过小可能会降低模型的训练效率。
- dropout 参数
  - dropout：一种随机失活技术，可以帮助减少过拟合。
- 其他参数
  - bptt：backpropagation through time，用于平衡反向传播的时间步数的大小

我首先对这些参数进行一个初步的探究，寻找一个合适的参数范围。默认的一些参数会导致训练时数据爆炸出现 nan 的数据，我先初步调整寻找一个合适的参数范围。

之后我编写了 sh 脚本，对不同的超参数进行遍历，将结构保存在对应的 txt 文件中，之后收集数据总结结论。

之后我学习了 tensorboard 的用法，尝试使用这个工具画出其中一些数据在训练过程中的变化趋势。

3. 数据收集与结论

‘第一次做这个实验时，我没有仔细关注 PPL 的绝对变化而只去看了相对变化，实验中得到的 PPL 都在 500+，在和同学交流后发现自己的训练很多都没有收敛，因此浪费了很多实验时间和数据，下面是更改后的正确的实验结果探究。

3.1. 实验数据汇总

参数：hidden:64,dropout:0,2,lr:10,batch32

模型种类	test loss	test ppl
RNN_RELU	nan	nan
RNN_TANH	5.67	289.42
GRU	5.21	183.08
LSTM	5.21	183.31
Transformer	5.43	227.80

参数：hidden:64,dropout:0,2,lr:10,batch64

模型种类	test loss	test ppl
RNN_RELU	nan	nan
RNN_TANH	5.61	272.41
GRU	5.21	183.50
LSTM	5.20	181.48
Transformer	5.33	207.18

参数：hidden:256,dropout:0.2,lr:10,batch32

模型种类	test loss	test ppl
RNN_RELU	nan	nan
RNN_TANH	5.30	201.25
GRU	4.93	137.82
LSTM	4.90	134.48
Transformer	5.27	193.85

参数：hidden:256,dropout:0.2,lr:10,batch32

3.2. 探究各个超参数对效果的影响

下面讨论一些超参数对模型效果的影响：

模型种类	test loss	test ppl
RNN_RELU	nan	nan
RNN_TANH	6.17	478.08
GRU	4.94	139.44
LSTM	4.91	136.29
Transformer	5.20	181.95

3.2.1. 模型效果

总体看下来，GRU 和 LSTM 的表现更好，Transformer 表现反而不如这两个模型。

这其实令我挺吃惊的，因为在大家的印象中应该是 Transformer 更加优秀。我查阅了资料，发现可能是以下的原因

1、数据量不足：Transformer 模型通常需要大量的数据来进行训练，以便学习到有效的表示和模式。如果数据量较小，Transformer 可能无法充分利用其强大的建模能力，而传统的 RNN 和 LSTM 可能更适合在小数据集上进行训练。

2、上下文信息的重要性：在某些任务中，特别是需要对上下文信息进行深入理解的任务，RNN 和 LSTM 可能更适合。RNN 和 LSTM 通过循环连接可以保留和传递更多的历史信息，而 Transformer 主要通过自注意力机制来建模上下文关系，可能对长期依赖关系的建模能力稍弱。

3、参数调优和模型架构：Transformer 模型的性能很大程度上取决于其参数调优和模型架构的选择。不同的超参数设置和模型架构可能会对结果产生显著影响。因此，如果没有进行充分的参数调优和模型架构选择，Transformer 的性能可能不如 RNN 和 LSTM。

当然我更多的感觉是代码的实现有一定的问题，因为我本身参与了相关的 Transformer 相关的 NLP 任务，我认为上面说的这些问题在这个任务中的影响较小。

3.2.2. 学习率

在本任务中，学习率会随着训练而成倍下降，因此我没有太在意学习率的调整。在开始训练的过程中用大的学习率可以帮助快速收敛，而之后采取小的学习率可以保证模型逐步收敛到局部最优。

一开始的探究里我就犯了这个错误，一开始就使用了小学习率，导致模型的训练其实很大程度上都没有收敛，因此实验结果没有参考意义。

3.2.3. batchsize

较大的 batch size 会占用更多的内存。如果模型和硬件资源的内存容量有限，较大的 batch size 可能会导致内存不足的问题。在这种情况下，需要适当减小 batch size 或调整模型结构以适应可用的内存。较大的 batch size 可能会导致模型收敛性的变化。一般来说，较大的 batch size 可以提供更稳定的梯度估计，有助于减少随机性和噪声。这可能导致模型更快地收敛，但也可能使模型陷入局部最优解。

较小的 batch size 可能会引入更多的随机性，使模型更容易逃离局部最优解，但也可能导致训练过程更不稳定。

由实验结果可以看到，较大的 batchsize 优化了模型的效果。这证实了上面的解释。

3.2.4. 隐层大小

隐层大小较大时，模型的表征能力会更强，会一定程度上提高模型的性能。但是过大的隐层会导致模型的参数亮变大，过大的模型会难以训练，内存开销很大。

由结果也可以看出增大隐层大小会优化模型效果。

3.2.5. 其他参数

我还探究了一些其他参数的影响，比如 dropout，bptt 参数等，但这些参数的影响比较小，这里不详细讨论。

3.3. 最优模型

根据我以上所有的探究，我还进行了一些实验，最后优化出最佳模型。

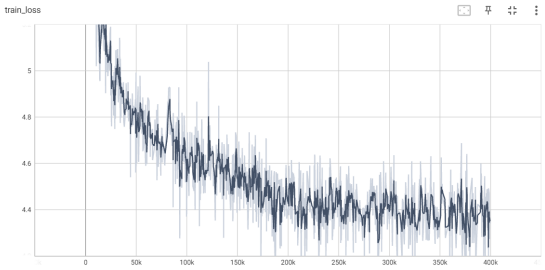
参数如下：python main.py –emsize 800 –dropout 0.5 –lr 5 –batch_size 32 –nhid 800–cuda

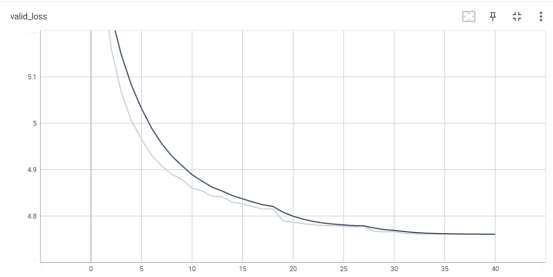
最终模型的大小为约 55MB，效果如下

test loss	test ppl
4.82	123.47

4. tensorboard 绘制图像

我学习了用 tensorboard 绘制 loss 图像，test-loss 因为只测算一次，因此只需要在最后读取即可。下面是一些图像。





可以通过观察这些 loss 来判断自己模型 xunlian 的程度和正确性。