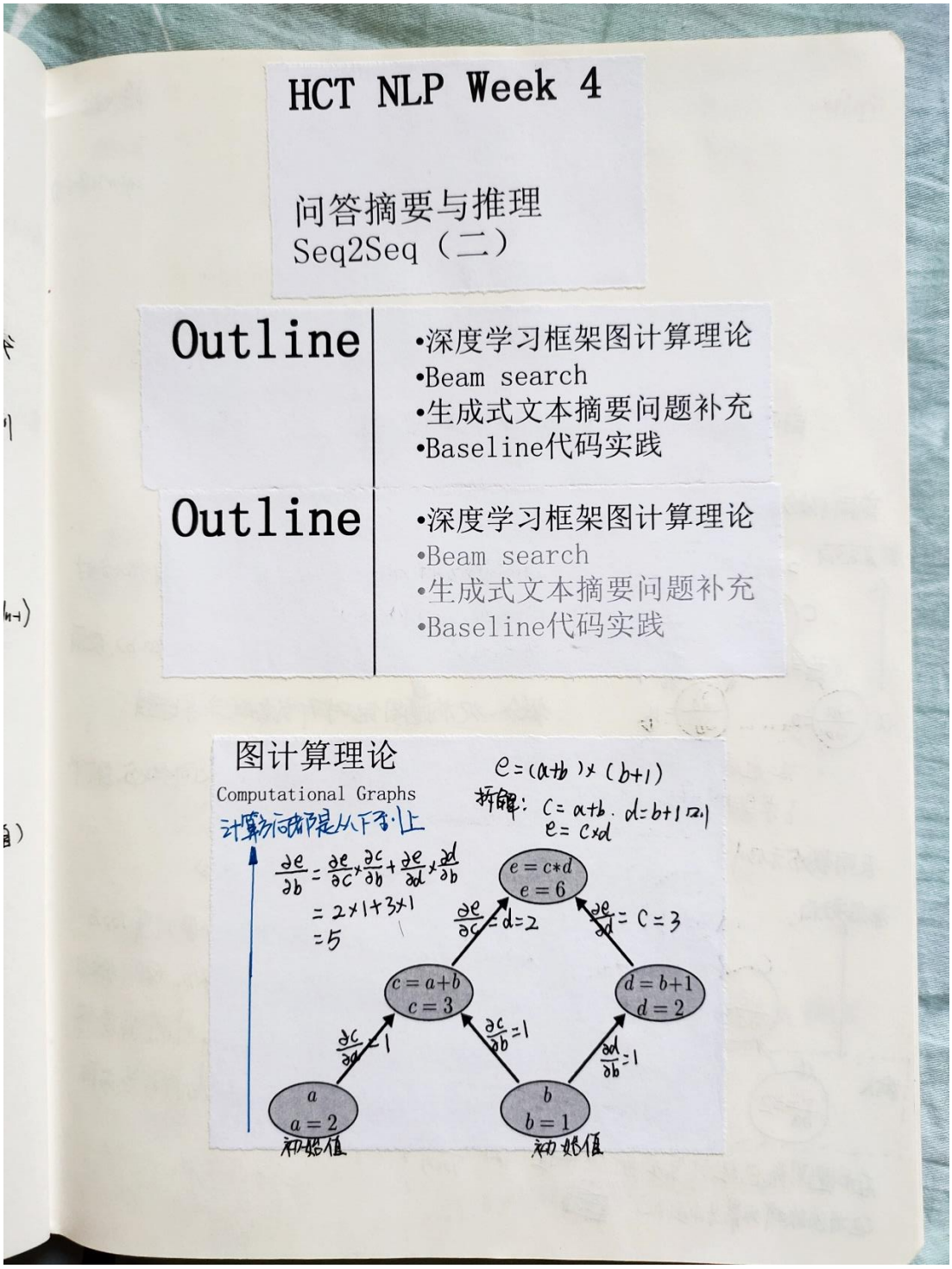
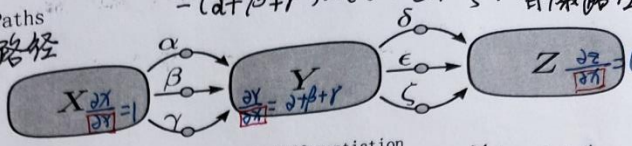
第三章 问答摘要与推理-Seq2Seq（二）



图计算理论

Factoring Paths
因式分解路径

$\frac{\partial z}{\partial x} = \alpha\delta + \alpha\varepsilon + \alpha\zeta + \beta\delta + \beta\varepsilon + \beta\zeta + \gamma\delta + \gamma\varepsilon + \gamma\zeta$
$= (\alpha+\beta+\gamma)\times(\delta+\varepsilon+\zeta)$ 有9条路径



forward-mode differentiation 前向微分

$\frac{\partial z}{\partial x}=(\alpha+\beta+\gamma)(\delta+\varepsilon+\zeta)$ 反向微分 $\frac{\partial z}{\partial y}=\delta+\varepsilon+\zeta$ $\frac{\partial z}{\partial z}=1$

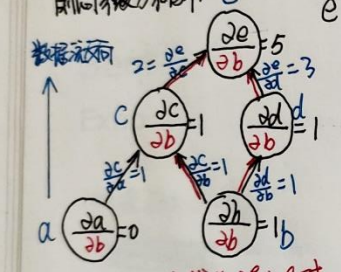
reverse-mode differentiation

微分求导的一种方式

前向微分和反向微分是两种方式构建图，

前向微分示例：

$e = c \times d = (a+b)\times(b+1)$

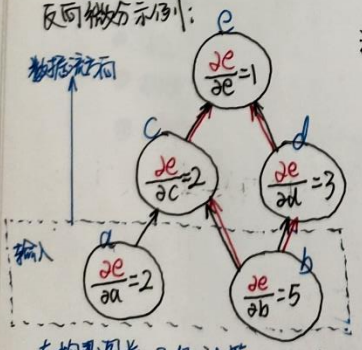


注：红线为 $(\frac{\partial e}{\partial b})$ e对b求导的路径

1. 前向微分时，分母不变（如都为 ∂b），反向微分时分子不变（如都为 ∂e）
2. 前向微分，一次构建图能对一个参数求导（如 b），反向微分一次构建图能对所有参数求导
3. 前向微分，一个输入影响每个节点，反向微分，每个节点影响最终输出。

注：tensorflow，Pytorch都是采用反向微分

反向微分示例：



在构建图前，已经计算出每个节点的值（把运算拆为最基本运算）

若有100万个参数，正向（前向）微分要计算100万次才能算出对所有参数的微分，反向微分只需要计算1次就能算出对所有参数的微分。（所以反向微分时的时间为前向的 $\frac{1}{100万}$）

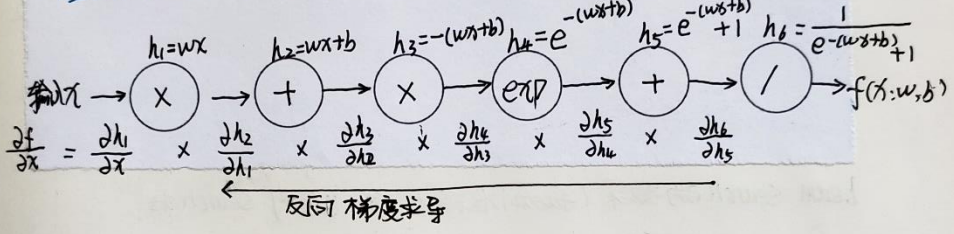
设有：$z=\sigma(wx+b)$，$y=wz+b$，神经网络基本流程为：输入$(x,y)\to\hat{y}\to$ loss 得$L(y,\hat{y})$

反向梯度求导：$\frac{\partial L}{\partial w}=\frac{\partial z}{\partial \sigma}\times\frac{\partial \sigma}{\partial w}$　　$\frac{\partial L}{\partial b}=\frac{\partial z}{\partial \sigma}\times\frac{\partial \sigma}{\partial b}$

图计算理论

$$f(x;w,b)=\frac{1}{\exp(-(wx+b))+1}$$

一个基本操作（如加、减）为一个算子

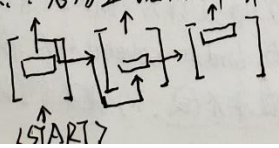


Outline

- 深度学习框架图计算理论
- Beam search
- 生成式文本摘要问题补充
- Baseline代码实践

greedy search algorithm 贪婪搜索算法，即在每步运算中只输出概率最大的那个值，并代入下一步进行运算

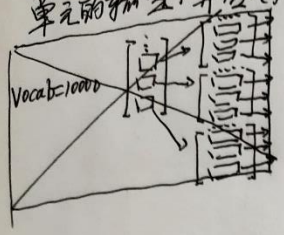
缺点：容易出现局部最优（整体不是最优）



每步预测都只输出概率最大的一个

beam_size=3

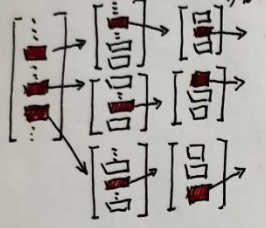
Beam search algorithm：即保概率最大的几个（如3个）做为本单元的输出，并传入下一个单元，预测下一个单元的输出。



① beam search 搜索中，若 beam_size=1，则 beam search 变为 greedy search

② beam search 是在广度优先下进行搜索空间的优化（类似剪枝）

③ beam search 比 greedy search 慢（因为传入下一步有多个参数）



设 beam_size=3，则每层取3个概率最大的做为输出，假设词表大小为 10000，则每层要做 30000 次搜索。

设 B= beam_size，词典大小为 V，序列长度为 L，则 beam search 的算法复杂度为：$O(B \cdot V \cdot L)$

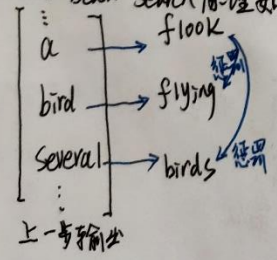
beam search 的效果（预测准确度）比 greedy search 好

beam search 中 B 越大，则输出选择越多（输出为稳性越多），但计算的时间也越长，同时消耗的内存也越大

beam search 的缺点：多样性不明显（输出的差异小）

为了解决 beam search 的缺点，可使用 Diverse Beam Search

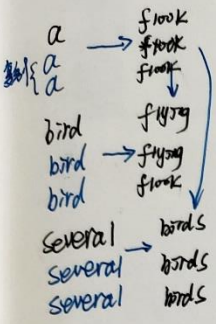
Diverse Beam Search 原理如下：设 beam_size=3



① 假设上一步输出为 a，bird，several.

② 下一步计算出，假设 a 单词作为输入时 flook 的概率最大，则 bird 和 several 作为输入时，在计算 flook 时进行一定的惩罚（如减1分），使 a 作为输入时 flook 作为输出的概率与 bird 和 several 作为输入 flook 作为输出的概率不同，以此来增加输出的多样性

借鉴 diverse beam search 增加多样性的思想，通过复制多份输入（上一步输出）
在代码实现中
函数词根可以增加输出多样性



复制后再传入下一步进行计算

注：diverse beam search 中惩罚的实现位置。
①实现 score 函数，初始化 score 如 [10, 10, 10 ……] 初始都为10
②在此步计算中，若可能输出的词在前面的 beam 组全出现过，则对这个词的分数减1，若没出现过，则不减(不惩罚)

SUMMARY GENERATION

Diverse Beam Decoding

the top-B hypotheses may differ by just a couple tokens at the end of sequences, which not only affects the quality of generated sequences but also wastes computational resources

Outline

- 深度学习框架图计算理论
- Beam search
- 生成式文本摘要问题补充
- Baseline代码实践

houchang

Scheduled Sampling 解决模型泛化能力不够的问题

A method for avoiding the problem of exposure bias.

是一种解决训练和生成时输入数据分布不一致的方法。在训练早期该方法主要使用目标序列中的真实元素作为解码器输入，可以将模型从随机初始化的状态快速引导至一个合理的状态。随着训练的进行，该方法会逐渐更多地使用生成的元素作为解码器输入，以解决数据分布不一致的问题。该方法应用在模型的训练阶段，生成阶段不使用。

① 定义系数：$\varepsilon \in [0,1]$ 在第 t 步训练样本真实值为 x_t，预测为 $\hat{x}_t$

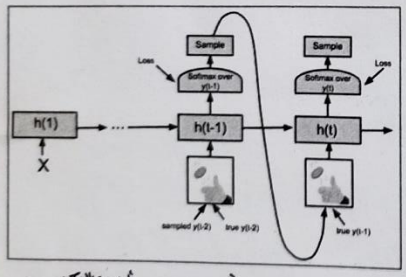
将 $\varepsilon \cdot x_t + (1-\varepsilon)\hat{x}_t$ 做为下一步 $(t+1)$ 的输入，

② 开始的时候 ε 的取值较大，随着训练步数的增多 ε 的值取得较小。

Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks

Scheduled Sampling

houchangtech.com



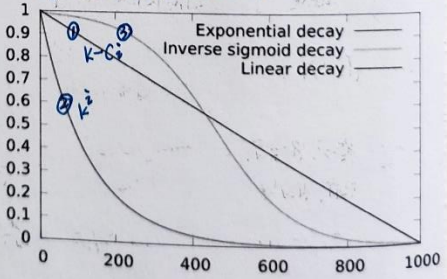
① $\varepsilon_i = \max(\varepsilon, k - c_i)$ 初始值 衰减率 线性衰减

最小的 ε

② 指数线：$\varepsilon_i = k^i$

③ 逆 Sigmod $\varepsilon_i = \frac{k}{k + e^{\frac{i}{k}}}$ $(k \geq 1)$

参数 ε 的选取方法有3种



17

代码中 input 与 target 说明

input <START> 后厂 理工 学院 <END> target

Datasets

数据集的选择很关键

CNN dailymail数据集：文本摘要最具代表性的数据集

First highlight: Argentina coach Sabella believes Messi's habit of being sick during games is down to nerves.

First 2 sentences: Argentina coach Alejandro Sabella believes Lionel Messi's habit of throwing up during games is because of nerves. The Barcelona star has vomited on the pitch during several games over the last few seasons and appeared to once again during Argentina's last warm-up match against Slovenia on Saturday.

数据组成形式

单文本摘要数据集：

- Gigaword
- LCSTS
- Newsroom
- Xsum

新浪微博摘要数据集（679898 条数据）

{"干杯 大哥！"外卖小哥点头那一刻 泪目} 近日，一男生和外卖小哥之间的互动，在网上刷屏。视频中，外卖小哥坐在电瓶车上啃干粮，男生请他帮拧瓶盖。当外卖小哥拧开第二瓶水时，男生说"干杯，大哥，天气很热，加油！"。外卖小哥这才反应过来，点头致谢❤（央视）

train_text.txt	短文本的内容，约100-200字
train_label.txt	短文本的摘要，约10-20字

ROUGE 评价模型的好坏，基于召回模型的评价指标

recall-oriented understand for gisting evalution

语言模型中的n-gram

- ROUGE-N RN
- ROUGE-L RL：L(longest common subsequence) 最长公共序列
- ROUGE-S
- ROUGE-W
- ROUGE-SU

注：分类问题可用准确率衡量

生成式问题（摘要生成）不适合用准确度来衡量

BLEU为基于准确度的评价指标

参考摘要即参考答案（标准答案）

自动摘要即生成的摘要

注：Pyrouge包（pip安装）

生成摘要问题通常用rouge评价

在参考摘要和自动摘要中同时出现

$$ROUGE\text{-}N = \frac{\text{参考摘要和自动摘要中共有的n-gram个数}}{\text{参考摘要中n-gram的个数}}$$

通常取 rouge-1，rouge-2

示例：参考摘要：the cat was under the bed (x)

生成摘要：the cat was found under the bed (y)

Rouge-1(x,y) = $\frac{6}{6}$ = 1.0 （黄色标注）

Rouge-2(x,y) = $\frac{4}{5}$ （红色标注）

ROUGE-L： Lcs(X,Y)：X,Y两个序列数据集最长子序列相同的个数

$R_{lcs}=\frac{Lcs(X,Y)}{m}$ ：回计算召回（最长子序列的召回）

m为参考答案的长度

$P_{lcs}=\frac{Lcs(X,Y)}{n}$ ：准确率

n为候选集摘要的长度

$F_{lcs}=\frac{(1+\beta^2)RP}{R+\beta^2P}$

S_1：Police killed the gunman 参考

S_2：police ended the gunman 基线1

S_3：the gunman murded police 基线2

句子的方向 →

当$\beta=1$时，Rouge-L为召回R_{LCS}，以1.0时：

$Rouge\text{-}L_{S_2}=\frac{3}{4}$　　$Rouge\text{-}L_{S_3}=\frac{2}{4}$

从性能的序列S_2优于S_3

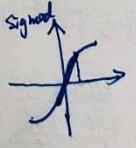
Initializing neural networks 网络初始化，初始化-给权重、参数。

期望 方差 $N(0,\delta^2)$

1、基于固定方差的初始化方法
- ① 符合高斯分布的初始化（使期望为0）
- ② 均匀分布初始化

① 若方差过小，会使神经元的输出过小，经过多层后信号消失

② 若方差过小，采用Sigmod作为激活函数时，表现为非线性能力，容易出现梯度消失



2、基于方差缩放的参数初始化方法
- ① Xavier初始化方法（Glorot）：根据每层神经元数量自动计算合适的参数初始化方法
  - 适用于Sigmod、tanh作为激活函数，但不适用于Relu，
- ② He初始化，适合Relu作为激活函数。

取值很小时，使输出神经元的输出为0

Batch size

1. Batch size是用于在每次迭代中训练模型的数据数量。一般的设置是32，64，128，256，512。

2. 选择正确的Batch size对于确保cost function和参数值的收敛，以及模型的泛化能力。

3. Batch size决定更新的频率。Batch size越小，更新就越快。

4. Batch size越大，梯度越精确。也就是说，在迭代计算的时候更容易跳过局部区域。

5. 比较大Batch size，往往GPU memory是不够用的，就需要通过并行计算的方式解决。

Choice of optimizer

优化器

(Stochastic) Gradient Descent

Momentum

RMSprop

Adam 最常用，会自动调节学习率，但Adam容易造成不收敛，此时需要更换其他优化器

Outline

- 深度学习框架图计算理论
- Beam search
- 生成式文本摘要问题补充
- Baseline代码实践 见jupyter

附：

（1）后厂理工 ai github：https://github.com/HouchangX-AI

（2）微分代码 github：

https://github.com/ZhaoYi1031/automatic_differentiation/blob/master/autodiff_test.py

（3）后厂理工代码参考：

https://github.com/HouchangX-AI/Question-and-answer-summary-and-reasoning

（4）优化器：https://deeplearning.ai/ai-notes/optimization/

（5）网络参数初始化：https://www.deeplearning.ai/ai-notes/initialization/

（6）华为云：https://auth.huaweicloud.com/authui/login.html#/login

（7）python argparse 讲解：https://zhuanlan.zhihu.com/p/28871131