NLP

# 花式自然语言处理

苏剑林 · 2018 年 1 月 23 日

中山大学

# Overview

1. CNN 与 NLP
2. Attention 新谈
3. 一些训练技巧
4. 交叉验证与融合
5. 无监督 NLP
6. 小杂烩

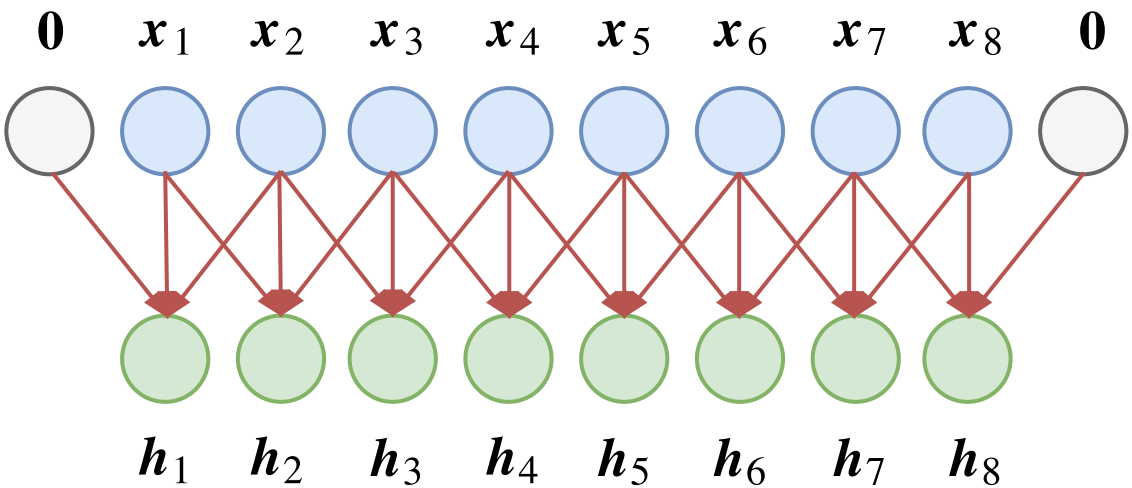
# 1. CNN 与 NLP

是时候抛开固有的 RNN 思维，尝试迎接 CNN 的拥抱了。

# 1.1. 一维卷积

CNN 实际上是 NLP 的标配方法，其思路比 RNN 更加自然。去年 facebook 的大作《Convolutional Sequence to Sequence Learning》充分体现了 CNN 在 NLP 中还有很大的挖掘空间。

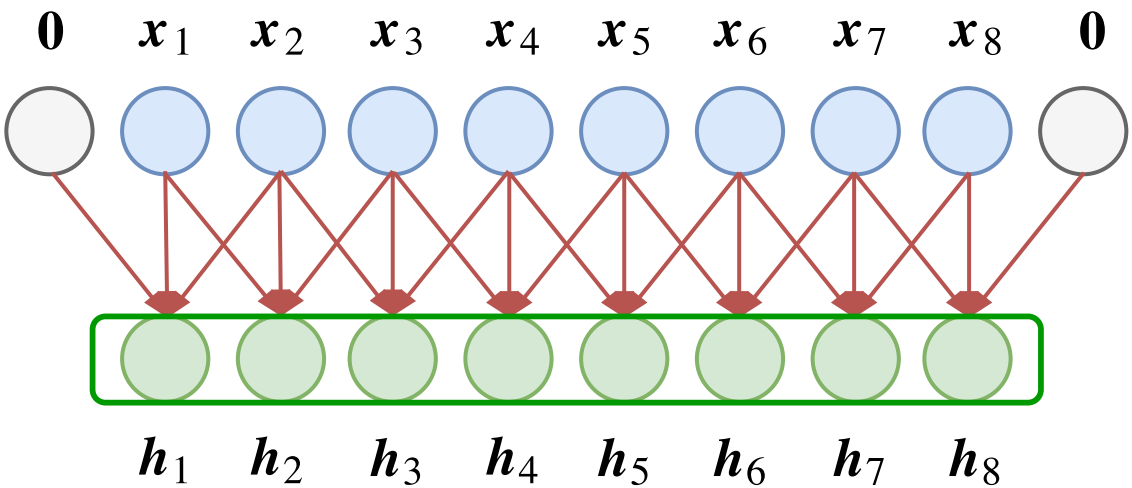
$$\boldsymbol{h}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}) = \boldsymbol{W}[\boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}] + \boldsymbol{b}$$



# 1.1. 一维卷积

CNN 实际上是 NLP 的标配方法，其思路比 RNN 更加自然。去年 facebook 的大作《Convolutional Sequence to Sequence Learning》充分体现了 CNN 在 NLP 中还有很大的挖掘空间。

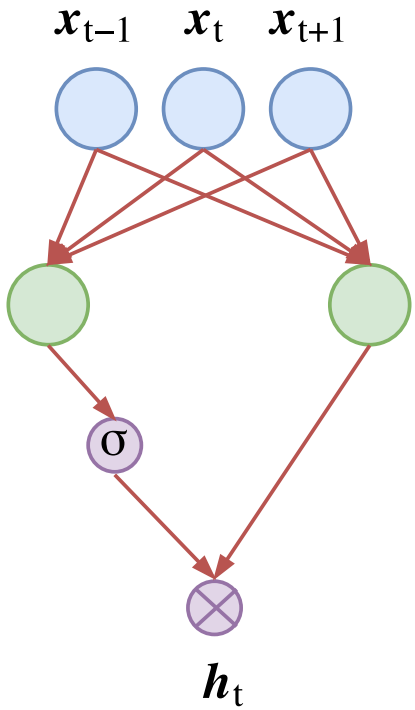
$$\boldsymbol{h}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}) = \boldsymbol{W}[\boldsymbol{x}_{t-1}, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}] + \boldsymbol{b}$$



# 1.2. 门激活机制

经过多番测试，可以比较肯定的是：在自然语言处理中，使用GLU（Gated Linear Unit，门线性单元）作为激活函数，效果最佳。

$$GLU(\boldsymbol{x}) = \boldsymbol{f}_{W_1}(\boldsymbol{x}) \otimes \sigma\Big(\boldsymbol{f}_{W_2}(\boldsymbol{x})\Big)$$



# 1.3. 残差机制

残差机制本来是为了解决深层神经网络的而提出的，但事实上残差有助于加速信息流动，使得简单的问题可以用简单的路径。

残差计算公式：$\boldsymbol{o} = \boldsymbol{x} \pm \boldsymbol{f}(\boldsymbol{x})$

如果与一维卷积、GLU 激活函数配合使用，则它在数学上等效于

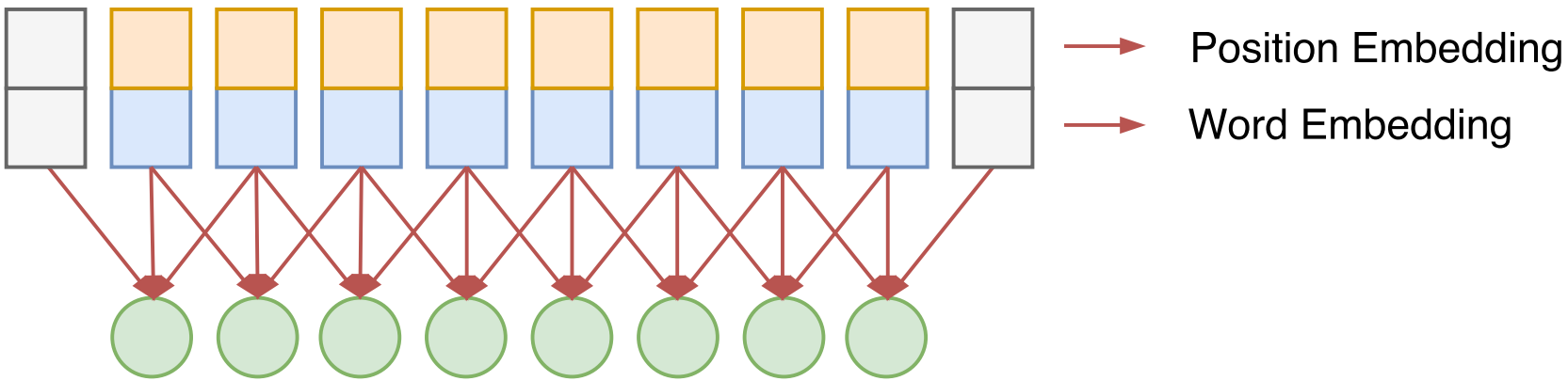
$$\boldsymbol{o} = \boldsymbol{x} \otimes \left[1 - \sigma\left(\boldsymbol{f}_{\boldsymbol{W}_2}(\boldsymbol{x})\right)\right] + \boldsymbol{f}_{\boldsymbol{W}_1}(\boldsymbol{x}) \otimes \sigma\left(\boldsymbol{f}_{\boldsymbol{W}_2}(\boldsymbol{x})\right)$$

这体现了信息在双通道中的选择性流动。

# 1.4. 位置向量

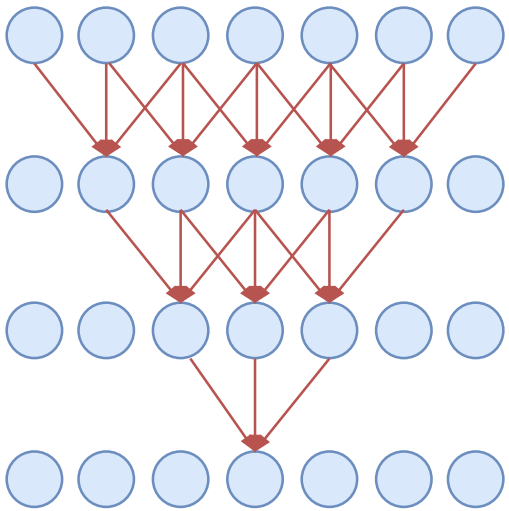
CNN 容易捕捉序列的结构信息，但不能很好地捕捉序列的位置信息（位置信息是 RNN 擅长的）。一个有趣的方案是给 CNN 的输入加入“位置向量”，增强 CNN 的“位置感”。

$$\begin{cases} PE_{2i}(p) = \sin\left(p/10000^{2i/d_{pos}}\right) \\ PE_{2i+1}(p) = \cos\left(p/10000^{2i/d_{pos}}\right) \end{cases}$$



# 1.5. 膨胀 CNN

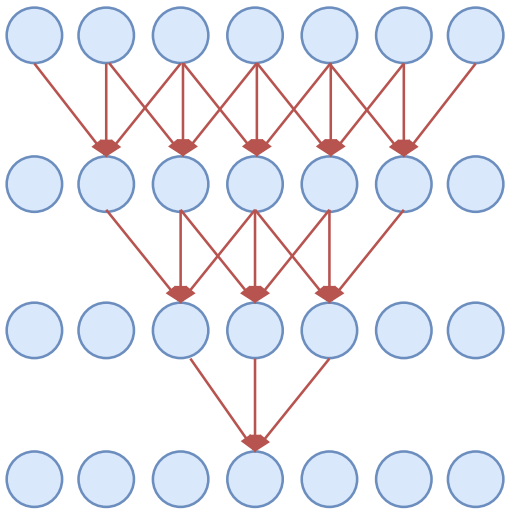
膨胀 CNN，又叫空洞 CNN，它通过在卷积核中增加“空洞”，从而在不增加参数量的情况下让捕捉更远的距离，目前已经广泛用于图像、语音、NLP 中。



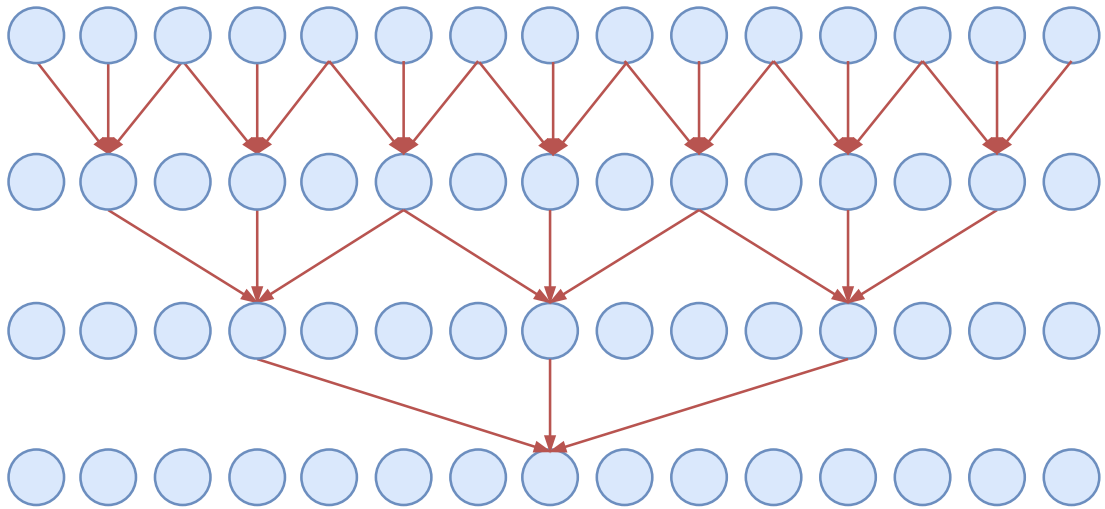
普通 CNN

# 1.5. 膨胀 CNN

膨胀 CNN，又叫空洞 CNN，它通过在卷积核中增加“空洞”，从而在不增加参数量的情况下让捕捉更远的距离，目前已经广泛用于图像、语音、NLP 中。



普通 CNN



膨胀 CNN

# 2. Attention 新谈

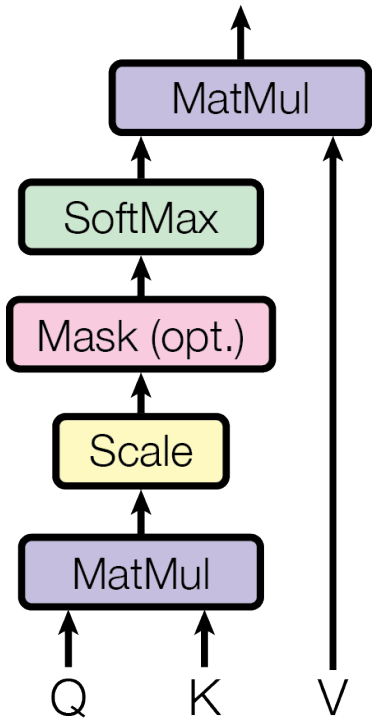
有所注意有所不注意。注意力集中了，接下来就好办了。

# 2.1. Attention

Attention，即注意力机制，是模拟人在某一时刻会有选择地关注某些事情的算法。Attention 本已广泛用于 NLP 中，而 Google 的大作《Attention is All You Need》将 Attention 推到了高潮。

$$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}$$

$$\boldsymbol{Q}\in\mathbb{R}^{n\times d_k}, \boldsymbol{K}\in\mathbb{R}^{m\times d_k}, \boldsymbol{V}\in\mathbb{R}^{m\times d_v}$$



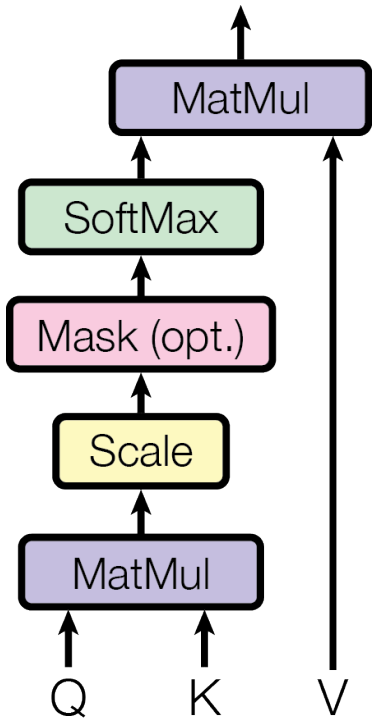
# 2.1. Attention

Attention，即注意力机制，是模拟人在某一时刻会有选择地关注某些事情的算法。Attention 本已广泛用于 NLP 中，而 Google 的大作《Attention is All You Need》将 Attention 推到了高潮。

$$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}$$

$$\boldsymbol{Q}\in\mathbb{R}^{n\times d_k}, \boldsymbol{K}\in\mathbb{R}^{m\times d_k}, \boldsymbol{V}\in\mathbb{R}^{m\times d_v}$$

$$\mathbb{R}^{n\times d_k} \quad \to \quad \mathbb{R}^{n\times d_v}$$



# 2.1. Attention

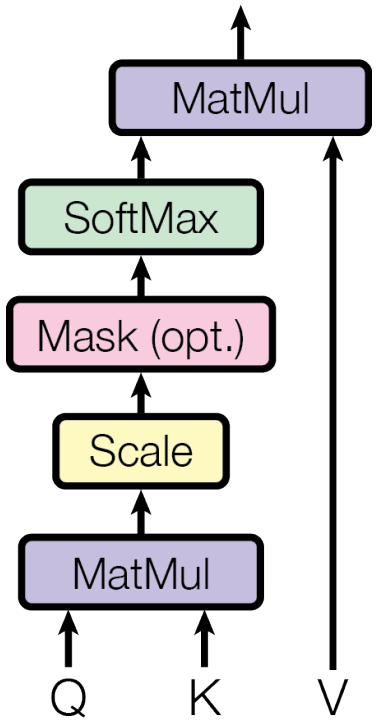
Attention，即注意力机制，是模拟人在某一时刻会有选择地关注某些事情的算法。Attention 本已广泛用于 NLP 中，而 Google 的大作《Attention is All You Need》将 Attention 推到了高潮。

$$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}$$

$$\boldsymbol{Q}\in\mathbb{R}^{n\times d_k}, \boldsymbol{K}\in\mathbb{R}^{m\times d_k}, \boldsymbol{V}\in\mathbb{R}^{m\times d_v}$$

$$\mathbb{R}^{n\times d_k} \quad \to \quad \mathbb{R}^{n\times d_v}$$

$$Attention(\boldsymbol{q}_t,\boldsymbol{K},\boldsymbol{V}) = \sum_{s=1}^{m}\frac{1}{Z}\exp\left(\frac{\langle \boldsymbol{q}_t,\boldsymbol{k}_s\rangle}{\sqrt{d_k}}\right)\boldsymbol{v}_s$$



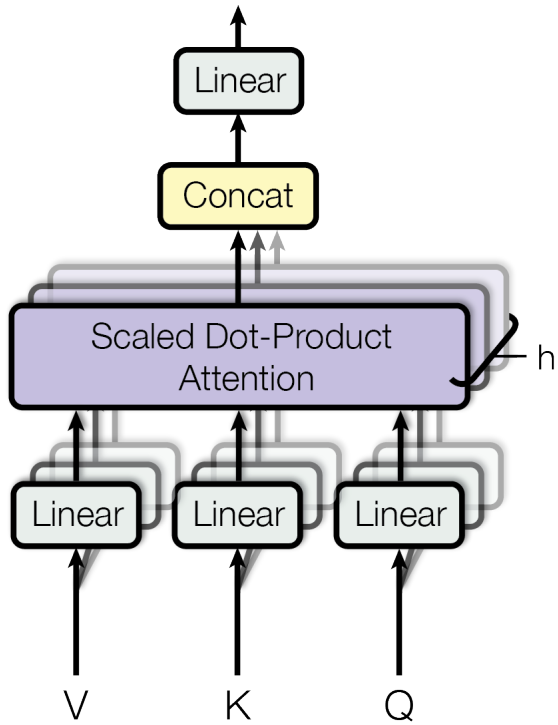
# 2.2. Multi-Head Attention

Multi-Head Attention 是 Google 在《Attention is All You Need》提出的新概念，类比多个卷积核的方式，将 Attention 重复多次并把结果拼接起来，从而实现多角度集中注意力。

$$head_i = Attention(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V)$$

$$\boldsymbol{W}_i^Q \in \mathbb{R}^{d_k \times \tilde{d}_k}, \boldsymbol{W}_i^K \in \mathbb{R}^{d_k \times \tilde{d}_k}, \boldsymbol{W}_i^V \in \mathbb{R}^{d_v \times \tilde{d}_v}$$

$$MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(head_1, ..., head_h)$$



# 2.3. Self Attention

Attention 中的 $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ 选取存在很大的自由度。如果做阅读理解的话，$\boldsymbol{Q}$ 可以是篇章的词向量序列，取 $\boldsymbol{K} = \boldsymbol{V}$ 为问题的词向量序列，那么输出就是所谓的 Aligned Question Embedding。

对于一般的序列任务，如文本分类等，我们更常用的是 Self Attention（自注意力），也就是

$$Attention(\boldsymbol{X}, \boldsymbol{X}, \boldsymbol{X}) \quad 和 \quad MultiHead(\boldsymbol{X}, \boldsymbol{X}, \boldsymbol{X})$$

# 3. 一些训练技巧

介绍从训练角度提高模型准确率的技巧，又名曰“深度炼丹”。

# 3.1. focal loss

focal loss 是 Kaiming 在《Focal Loss for Dense Object Detection》提出的新 loss，用于解决类别不平衡、分类难度差异的问题，效果良好。具体形式为

$$-\alpha_t(1-\hat{p}_t)^\gamma \log \hat{p}_t$$

# 3.1. focal loss

focal loss 是 Kaiming 在《Focal Loss for Dense Object Detection》提出的新 loss，用于解决类别不平衡、分类难度差异的问题，效果良好。具体形式为

$$-\alpha_t(1-\hat{p}_t)^\gamma \log \hat{p}_t$$

用在二分类问题中，形式为

$$-\alpha(1-\hat{p})^\gamma \log \hat{p} - (1-\alpha)\hat{p}^\gamma \log(1-\hat{p})$$

经过 Kaiming 的调参，发现如果正样本的数目远小于负样本的数目，那么取 $\alpha = 0.25, \gamma = 2$ 比较好。

# 3.2. 学习率下降

深度学习的调参主要指超参数，如节点数、批大小、输出阈值等。训练方面，一般而言精调的 sgd 会更好，而用 adam 算法进行优化，那么可以参考的经验如下：

1. 以默认学习率 ($10^{-3}$) 将模型迭代足够多次数，保留验证正确率最高的模型；
2. 加载上一步最优模型，学习率降到 $10^{-4}$，继续训练模型，保留验证正确率最高的模型；
3. 加载上一步最优模型，去掉正则化策略（dropout 等），学习率调为 $10^{-5}$，训练至最优。

# 3.3. 数据扩增

解决过拟合的最好方法是找更多的数据！数据扩增（data augmentation）是指在原标注数据的基础上造更多的数据，来增强模型的泛化能力。常用的数据扩增手段有

1. 想办法找更多的标签数据（很多时候很难做到）；
2. 一个句子重复拼接、随机去掉若干个词、打乱词序，类别不变；
3. 随机挑两个同类句子，拼接成一个新句子，类别不变；
4. none of above：收集更多的其它语料，将分类器多加 1 类，新语料都归为新类（Universum Prescription- Regularization Using Unlabeled Data）。

# 3.4. 其它细节

可以优化的地方还有很多，比如分词、词向量、正则化方法等，需要多尝试、多看论文，总结出自己的调参方案。

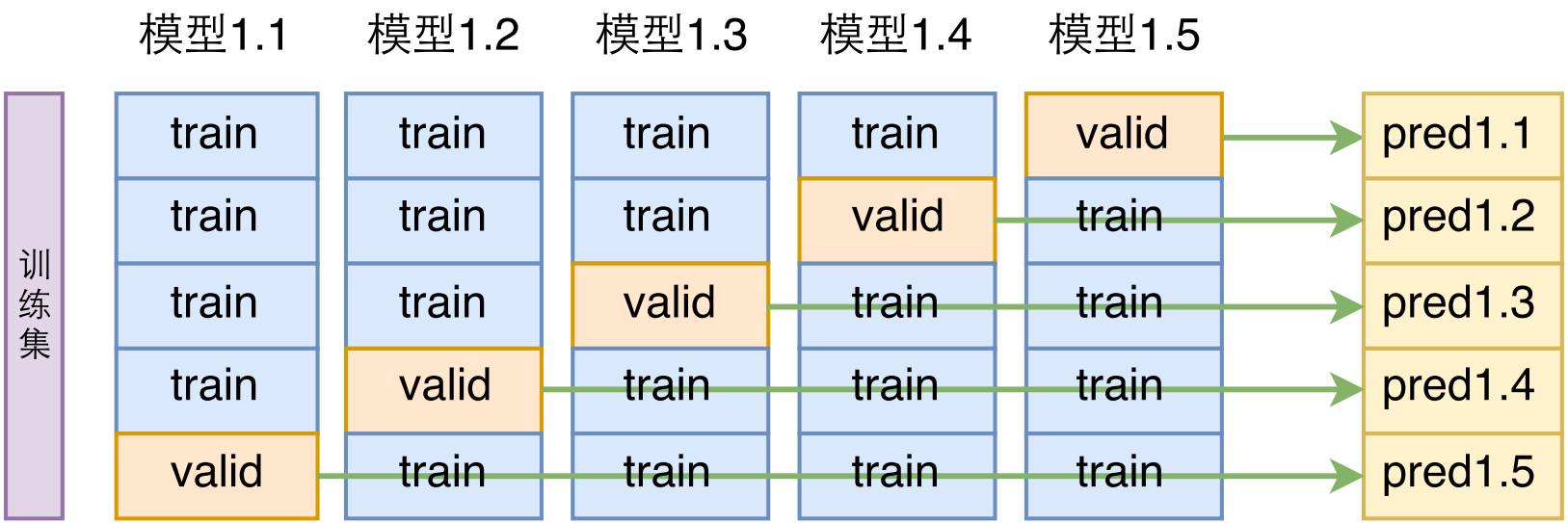
1. 分词：想办法准备一个领域相关的词表，优化分词效果，如果用 jieba 分词，关闭新词发现功能；
2. 词向量：准备领域相关的语料，训练 Skip Gram 模型（耗更多时间，但能保证效果）；
3. 正则化：适当使用 Dropout（要设置 noise_shape），如果使用 CNN 的话，还可以考虑使用 DropPath（出自《FractalNet: Ultra-Deep Neural Networks without Residuals》）。

# 4. 交叉验证与融合

模型融合是提高模型准确率的“杀手锏”，尤其是在打比赛时。

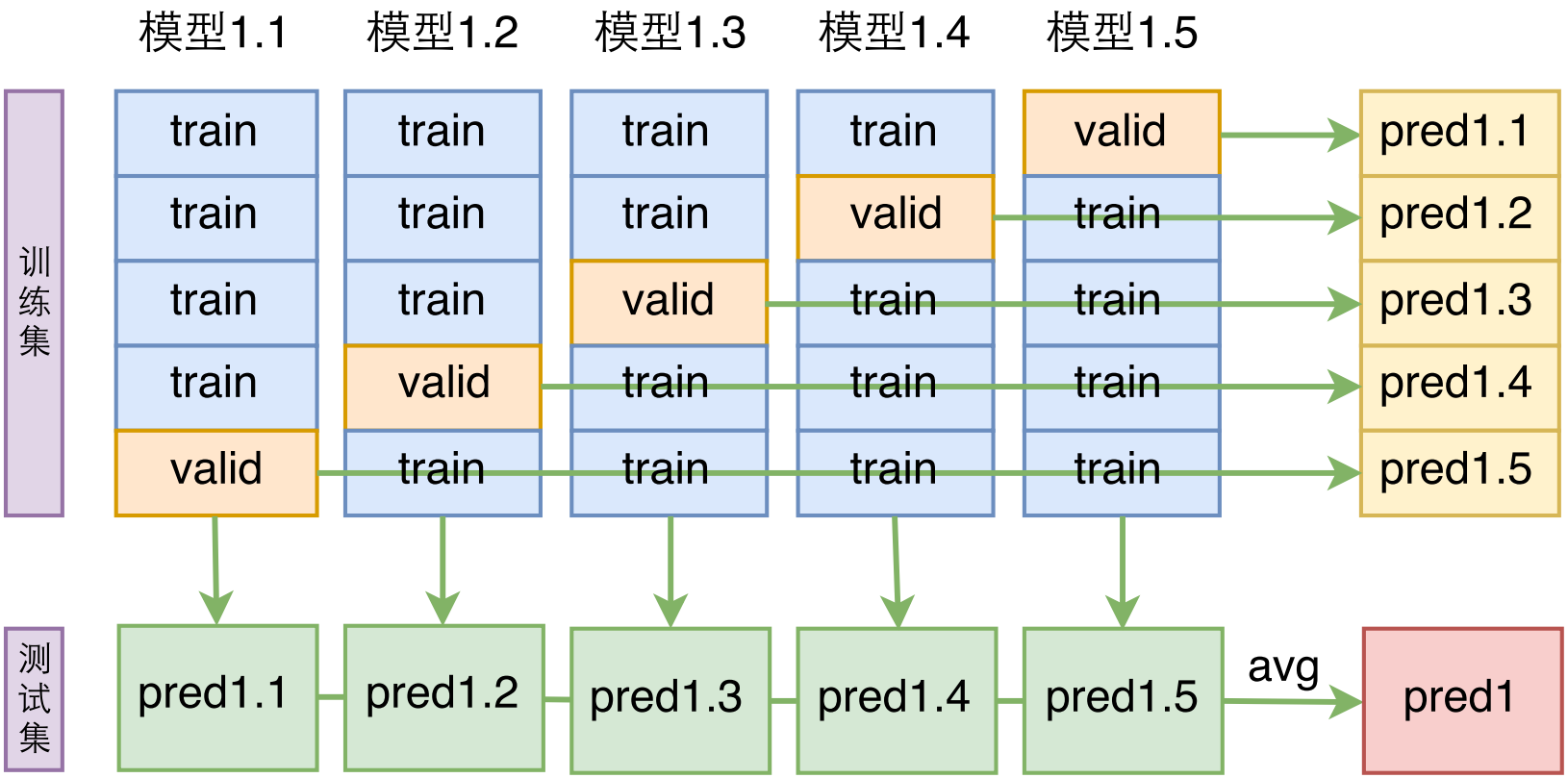
# 4.1. 交叉验证

事实上，交叉验证才是评估模型性能的最可靠方法。交叉验证，就是把数据打乱，然后分为 $n$ 分，挑一份做验证集，剩下 $n-1$ 做训练集，如此验证完整个数据集（训练 $n$ 个模型）。



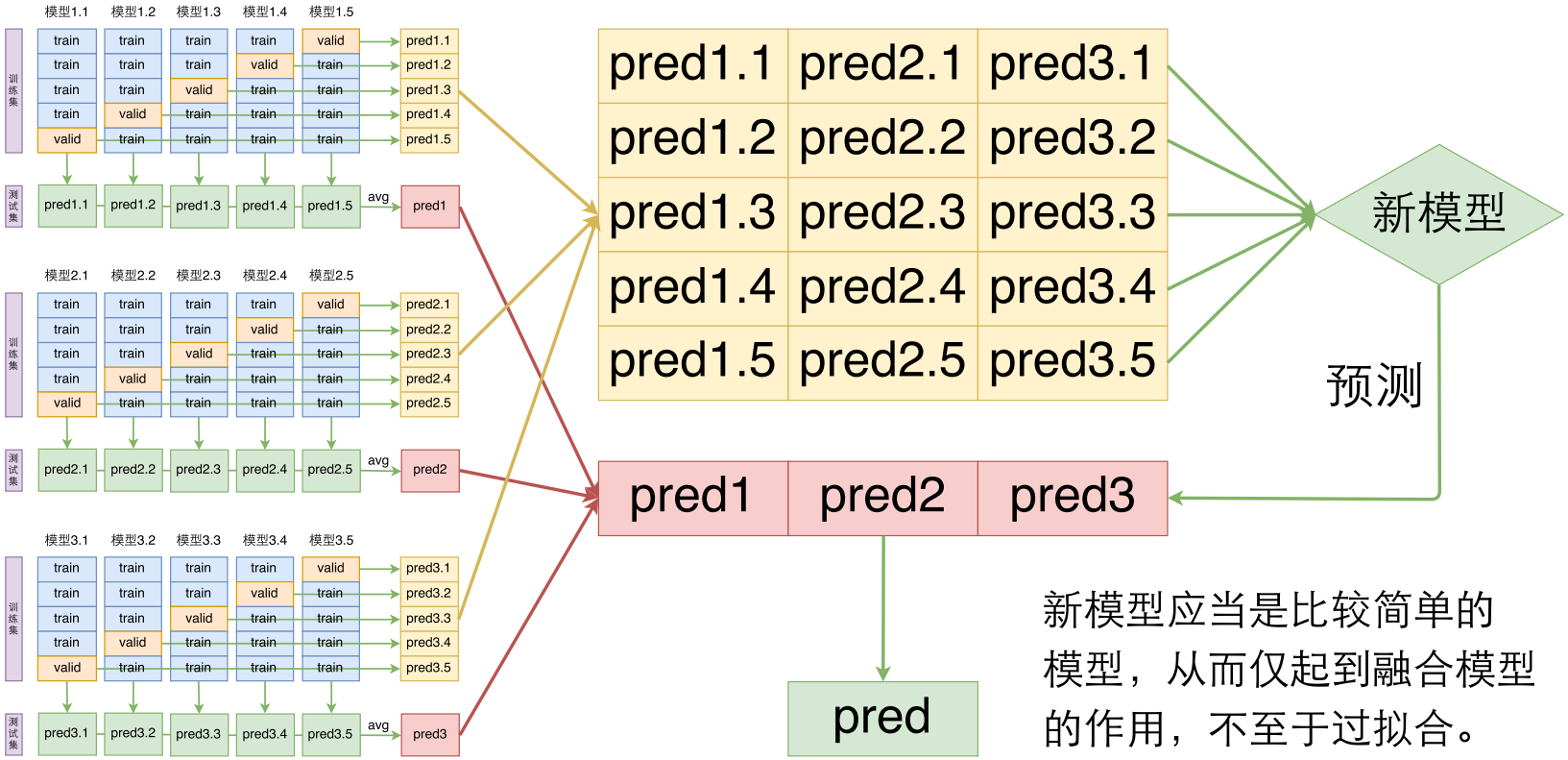
# 4.2. 单模型融合

模型融合能大幅提升准确率，是打比赛的“杀器”。把交叉验证训练的 $n$ 个模型直接求一下平均，就是最简单的模型融合方案。



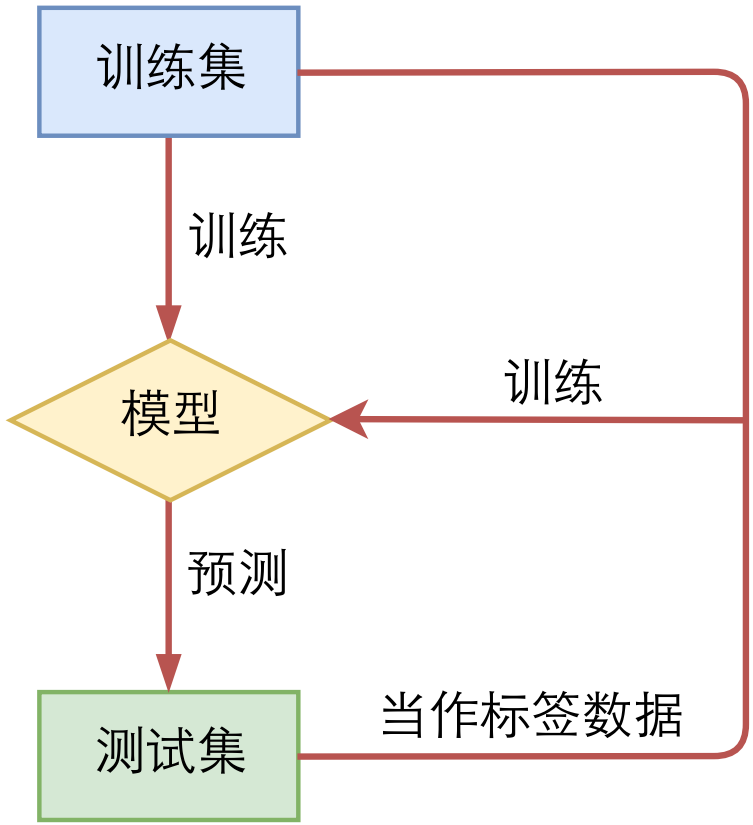
# 4.3. 多模型融合

假如有 $m$ 种不同的模型，每种模型做 $n$ 划分交叉验证，可以得到 $mn$ 个不同的模型，通过一个新模型来融合这 $mn$ 个模型。



# 4.4. 迁移学习

迁移学习是一种半监督学习过程，是一种充分利用无标签数据的技巧。首先我们利用标签数据训练一个模型，然后用这个模型来预测测试集，得到测试集的“伪标签”，把测试集连同其“伪标签”当作额外的训练集，加入到训练集中继续训练。这样有助于提高测试集的准确率（因为提前过了一遍测试集），也能提高模型的泛化能力。



# 5. 无监督 NLP

在满大街深度学习的今天，你还愿意思考 NLP 背后的原理么？

# 5.1 最小熵原理

熵衡量了一个事物的信息量，而我们接收信息的速度是基本固定的，那么熵就是衡量了我们的学习成本！不管作为教师还是学生，我们都想办法降低学习成本，这就是最小熵原理——通过降低熵来完成自然语言的无监督学习。

$$\text{平均单字平均熵：}\mathcal{L}=\frac{\mathcal{H}}{\bar{s}}=\frac{-\sum_w p_w \log p_w}{\sum_w p_w s_w}$$

如果不分词，按字算，平均单字信息熵约为 9.65；分词后，算得约为 7.2。也就是说，分词有助于降低学习成本！这也就是为什么很多 NLP 模型都需要先分词。

# 5.2 构建词库

既然分词有助于降低学习成本。那么反过来，能找到一种降低学习成本的切分方案，那么它就是好的分词方案，这就是无监督构建词库的思路。（《Redundancy re- duction as a strategy for unsupervised learning》，1993 年的文章）

$$\text{组合指标：}\mathcal{F} \approx p_{\boldsymbol{a}}\left(\log \frac{p_{\boldsymbol{a}}}{\prod_{i\in \boldsymbol{a}} p_i} - 1\right)$$

该指标的意思是：连续的 $k$ 个字 $\boldsymbol{a} = (a_1, a_2, \ldots, a_k)$，如果它是一个“词”，那么它应该使得上式越大越好。可以沿着这条路走下去，直到无监督句法分析！（参考《一种基于生语料的无监督的语法规则学习方法》）

# 6. 小杂烩

还有这些内容可能也是大家会感兴趣的～

# 6.1 有意思的玩意

深度学习最初的成功领域是图像，后来迁移到 NLP 中。目前图像也有很多新的成果，这些成果其实也有可能迁移到 NLP 中。

1. FractalNet：“分形”网络，亮点有两个：1、通过函数的复合来增强非线性；2、DropPath 来正则化；
2. CapsuleNet：Hinton 最新提出的神经网络的新概念，目前只在图像上做了实验，还有很大挖掘潜力；
3. Attention：在 Google 的《Attention is All You Need》中还提出了一个 restrict 版的 Attention，有待研究；
4. ...

# 6.2 有价值的资源

即使我们平时的研究难以做到面面俱到，然而我们还是有必要去关注各个领域的最新内容，寻求想法，“他山之石，可以攻玉”。

1. 公众号：PaperWeekly
2. 公众号：机器之心
3. 公众号：机器学习算法与自然语言处理
4. 知乎专栏：西土城的日常搬砖
5. 博客：科学空间
6. QQ 群 & 微信群：相互推荐加入

# Thanks

有想法或疑问欢迎再聊～