待解决问题：

* Ppt内容
* 整理自己的idea
* 确定2、3三个方向和老师讨论
* 整理前段时间做的工作
* 基础是不是比较重要，是否需要多学学图像分割之类知识
* 汇报学习计划
* 用代码实现自己的想法会花费比较多时间？应该怎么去判断自己的想法值不值得用代码实现？不用代码实现能否证明其有效性？
* 分析自己的优缺点，希望老师能点拨一下
* 关于GPU
* Pytorch安装方法？匹配和冲突问题？
* 什么时候用GPU，什么时候用CPU?
* 模型是为了捕捉数据的某种潜在特征，或者说规律，仔细去体会好的模型中数据可能存在的某种规律，或者采用结构化attention能提升效果背后的原因
* 注意总结连接点结合方法，分析内在原因
* Seq2seq翻译句子长度怎么确定？什么时候结束？花书P237
* 动量？花书P251
* Attention相对于LSTM是不是有点权重不共享的感觉？理论上各步相互之间作用的权重不应该相同才对？
* 卷积神经网络的优化方法？
* LSTM一般有多少参数，都共享参数的吗？它为啥能记住数据？神奇？输出一般是one-hot还是词向量？词向量不是不好判断是哪个词吗？
* 跑代码时可能 会面临的问题
* 如何调参数
* 如何设计合适的宽度和深度
* 如何选择合适的模型
* 如何将模型和直观想法联系起来？如《tensorflow》P145
* 正负样本的作用？
* 注意词、句、文章表示的具体形式？及长度不一怎么解决？
* 分成不同模块的训练方法？是联合训练还是分开训练?怎么选择？
* 有时间自己敲一个多层简单神经网络程序，并自己设计激活函数和反向传播（可能pytorch会比较好用）
* 不同核函数的更深层的作用理解
* 学习两种比较典型的VQA方法并复现代码：CNN+LSTM和attention，查看其历史，搞清来龙去脉。
* 模型深度、宽度、激活函数的设计方法？
* seq2seq输出是词向量还是one-hot？
* 贝尔曼公式
* 遗传算法
* 记忆网络
* DNN：交叉熵损失函数，普通RNN会出现梯度消失和长时间输入的梯度爆炸,常不能使用?
* Different variants: Neural Turing Machine [Graves et al., arXiv ’14],

Memory Networks [Weston et al., ICLR ’15] and End-to-end Memory

Networks [Sukhbaatar et al., NIPS ’15], Dynamic Memory Networks [Kumar et

al., ICML ’16], Neural Differentiable Computer [Graves et al., Nature ’16],

Recurrent Entity Network [Henaff et al., ICLR ’17]

* 低资源？
* 知识图谱：按箭头方向可以练成一句话，稀疏性是指实体间的关系连线较少？在实体命名识别和关系抽取过程中，有两个比较棘手的问题：实体统一和指代消解

<https://www.jiqizhixin.com/articles/2018-06-20-4>

* Stage表示一个功能层？包括多层layer？
* 词向量空间太稠密（即词与词离得太近，不容易区分）会不会造成预测出错？
* 图像处理背景的影响？作人脸识别是要先经过预处理把人脸抠出来吗？使用的是传统学习方法吗？
* Bootstrap
* Jurafsky 和 Martin 的 Speech and Language Processing 是领域内的经典教材
* Few-shot learning？meta learning？
* 做对话有必要了解一下语音识别
* 强化学习与机器人：<https://blog.csdn.net/m0_37290785/article/details/82734572>
* 各种激活函数选择原因以及适用场合？
* CV预训练网络：VGG16、Faster R-CNN、ResNet。CNN和R-CNN区别
* bottom-up attention：<https://blog.csdn.net/u012991166/article/details/80659909>
* VQA比赛，后面有两个源码：
* <https://blog.csdn.net/yH0VLDe8VG8ep9VGe/article/details/81277584>
* 训练好模型后，尝试自己随机乱弄些数据，看看结果怎么样，看看模型是否具有通用性
* 注意内积和外积有两种不同定义，有张量积和几何积

<https://www.jianshu.com/p/f58a8b30f460>

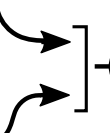
* 双线性插值<https://blog.csdn.net/qq_37577735/article/details/80041586>
* 30 ensembled models和Single-model是将模型简单叠加吗？
* spatial output layer of a CNN？
* 注意体会激活函数不同输出范围的影响
* binary cross-entropy？
* backforward 和forward GRU？正向和反向编码
* 元学习：<https://blog.csdn.net/langb2014/article/details/84953307>

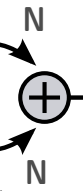
1. 自己总结：

* stage表示一个组成部分，可能有多层
* 用共享权重、神经网络的方法相当于把不同输入映射到了同一个连续分布的空间，有利于捕获差异和相同点，也有利于计算机处理
* 深度学习是输入-输出模型，特征提取由隐层来完成，非常简单方便，但是计算量比较大，这就需要算力的提高和算法的优化
* 斯坦福大学建立的 GQA 数据集可用于VQA
* 看模型首先看是什么任务？输入输出是什么？再看目标函数是什么？怎么优化？
* 图像的每个像素点可以看成不同的维度，这样就可以把图像识别看成分类任务
* 机器人之所以叫observe r，因为在现实生活中机器人应该自己通过观察需要寻找目标物体，然后确定回报大小，损失太大（可以通过房间大小等设置上限损失，如撞墙、时间过长）或者进入黑洞要退回原点重新搜索，机器人应该会自己优化路径回到原点，如果是在仿真环境里，直接回到原点即可
* batch梯度下降是随机梯度下降和批梯度下降的折中
* 多线程会出现log，另外，由于s+=i的出现，使得s的增长速度变快而非每次加1，因此与sqrt(n)为一个数量级，至于为什么正好跟sqrt(n)一个数量级，会出现根号
* 可以通过验证集来判断是否收敛
* L1范数允许有少数较大权重出现，L2不允许较大权重出现，较大平方会比较大，正则化一般只管权重，不管偏置
* 数据集增强之对象识别：沿训练图像每个方向平移几个像素的操作通常可以大大改善泛化。许多其他操作如旋转图像或缩放图像已被证明非常有效
* 当看论文时遇见正则化等问题时再去细看《深度学习》书
* GPU通常采用2的幂数运行速度更快
* CNN能否尝试三通道分开算
* 用导数定义求导会比较低效，但可以解决不可微的情况，以及用于验证求导是否正确
* 初始点选择对优化和泛化的影响
* 花书P213页给了根据数据集设计模型的启示
* 这样我们心里会有数，神经网络模型至少不应该比n-gram模型差的。神经网络模型因为其超参数、梯度爆炸等问题，有时候我们不太容易决定是真的模型不行、参数没调好还是代码有 bug。那么通过 N-gram LM 的给出的下限，我们就可以直观地知道神经网络是有 bug 还是没调好参数。
* GPU选择：<https://zhuanlan.zhihu.com/p/61411536>

<https://www.cnblogs.com/lantingg/p/7810732.html>

<https://product.pconline.com.cn/itbk/software/dnyw/1709/9997014.html>

* 待拟合的函数是客观的，不以超参数的改变而改变，我们的目的就是通过调节超参数，使模型更好的拟合这个函数，超参数过大或过小都不好
* 想让模型与某种事物发生联系，直接将这个事物其嵌入到端到端的输入端，让模型自动捕获隐性关系
* 需注意观察模型各神经元激活状态，激活及饱和太多或太少都可能有问题
* 在生成句子中作贪婪搜索或beam搜索，会提高语言的流畅度
* 注意激活函数的求导范围，如sigmoid为0-0.25，梯度消失可能由这个造成
* 文本挖掘相关：<http://www.sohu.com/a/239822761_387904>
* Ablation study: 消融研究，指通过移除某个模型或者算法的某些特征，来观察这些特征对模型效果的影响
* 一般要调用某些生数据，先经过一层神经网络编码会比较好，
* 种类的细分等细粒度问题值得研究，如狗的种类
* 默认一般是列向量，加T变成行向量。⊗叉积：注意和几何积的区别；◦Hadamard (element-wise) product，两张量各元素分别相乘，如下最左图几何积叫dot product；，串联各个向量，如下次左图；累加符号表示将所有对应元素相加，如下次右图，表示加权累加；圆圈里+表示element-wise product，两张量各元素分别相加，如下右图



* 要完成一个深度学习项目，首先要做的是收集数据，分析数据，处理数据，设计模型，训练模型。
* 可能有用软件：zotero、坚果云（两者可配合使用）
* 词袋：看图像比较好理解

<https://blog.csdn.net/Jaster_wisdom/article/details/81539623>

1. 调参和优化方法

* 模型太复杂（参数多）、数据集小会造成过拟合，可以会使用dropout、正则化来减小过拟合；模型太简单会造成欠拟合
* 相同的神经元数，深度较大的泛化能力强
* AdaDelta优化方法不需要固定的学习率、对初始化参数不敏感，注意区分参数（parameter）和超参数（hyperparameter）
* 为了防止过拟合，可采用如下方法
* 提前停止训练
* dropout
* 正则化
* 加大数据集
* 减小模型复杂度
* 集成（ensemble）
* 判断何时停止训练
* 用验证集或训练集误差或评估标准设定阈值判断，或根据是否波动判断，或在多少iteration或epoach不再提升判断
* 方差（几次训练的差异）大，可能原因是过拟合和不稳定的训练
* 1

1. 分析数据集方法

* 查找系统分析数据集书籍或课程

1. 印象较深文章：

* 自然语言处理：<https://mp.weixin.qq.com/s/-W6SiE1G70JQryWbkY3i5g>
* 可否通过训练有效控制激活函数阈值，实现有效切换？VQA中的不同方法可否根据不同问题或图片设定不同的阈值激活函数，或者通过契合的契合度设定知识库和图像读取答案的切换

1. 常用数据集和模型

* 数据集
* Dialog：<https://github.com/AtmaHou/Task-Oriented-Dialogue-Dataset-Survey>，文件已下载至H:\learning\postgraduate\download\_self\dialog，任务型（Amazon、微博）非任务型（豆瓣，得会自己爬数据）
* NLP：wordnet（英语语义库）、
* Visual：mnist、cifar、ImageNet 、COCO
* VQA: clevr（专门针对视觉推理任务，几何体）、GQA（也具有很强推理性，实物）、COCO-QA dataset、Visual7W、Visual Genome、VQA datasets、the VQA Real Multiple Choice challenge、COCO dataset
* Environment simulator：Matterport3D
* Referring expression：RefCOCO、RefCOCO+、RefCOCOg
* 语言模型学习数据集：PTB（penn treebank dataset）、
* KB：ConceptNet （https://conceptnet.io）(Speer and Havasi 2012)、
* Visual Grounding： Flickr9k, Flickr30k and MS COCO.
* Slam数据集：KITTI, Cityscapes dataset、RobotCar dataset（牛津大学一年）、EuRoc MAV dataset（flying robot）、TUM dataset [40] and NYU dataset（室内，手持RGB-D）、
* 3D：ShapeNet [1]、Blobby objects and Sculptures
* Semantic segmention：Cityscapes/ADE20k/Pascal-VOC
* 模型：、FC？
* Visual：CNN、ReLU
* VQA预训练特征提取模型：AlexNet [28], GoogLeNet [1], ResNet-34、50、101（不同深度）、VGG
* Sequence（language）：RNN、LSTM、BI-LSTM、attention
* EQA：
* 预训练的词向量：GloVE、Google News
* 评价（evaluation）
* BLEU, METEOR, and ROUGE
* 强化学习相关：AUC、ROC curve
* CV: IoU(Intersection over Union)

<https://blog.csdn.net/iamoldpan/article/details/78799857>

* SLAM：Absolute Trajectory Error (ATE),
* 1

1. 专业名词缩写：

* 级联：concatenated
* Neural Machine Translation：NMT
* intersection-over-union (IoU)
* region of interest (RoI)
* 深度前馈网络=前馈神经网络（feedforward neural network）=多层感知机（multilayer perceptron，MLP）

1. 常用模型：循环神经网络（Recurrent Neural Network, RNN）：

<https://blog.csdn.net/qq_32241189/article/details/80461635>

卷积神经网络（Convolutional Neural Networks, CNN）：

<https://www.cnblogs.com/skyfsm/p/6790245.html>

递归神经网络（recursive neural network，RNN，ANN?）：

<https://blog.csdn.net/u010164190/article/details/78172289>

强化学习（reinforcement learning）

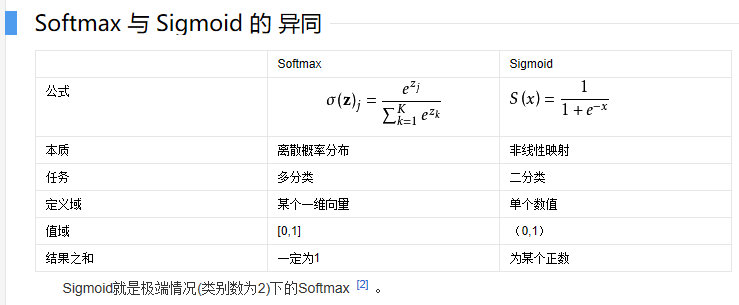
1. 常见NLP任务：机器翻译、问答、对话系统、词性标注、句法分析、命名实体识别、情感分析、上下文嵌入、信息检索、信息抽取、知识工程、自然语言生成、推荐系统、对话交互系统、语义分析、摘要生成、文本挖掘、语篇语用学、意见挖掘、社会计算等自然语言处理领域众多研究方向。
2. 人工智能方向：自然语言处理（NLP）、计算机视觉（CV）、知识图谱、专家系统
3. 自然语言处理（NLP）是指对人类语言进行自动分析和表示的计算技术
4. 词嵌入是自然语言处理（NLP）中语言模型与表征学习技术的统称。概念上而言，它是指把一个维数为所有词的数量的高维空间嵌入到一个维数低得多的连续向量空间中，每个单词或词组被映射为实数域上的向量。目前词嵌入已经是 NLP 任务的标配，大多数 NLP 任务的顶尖结果都需要借助它的能力。词嵌入真正走向 NLP 主流还是 Mikolov 等人在 2013 年做出的研究《Distributed Representations of Words and Phrases and their Compositionality》。Mikolov 等研究者在这篇论文中提出了连续词袋模型（CBOW）和 Skip-Gram 模型，这两种方法都能学习高质量的分布式词表征。Pennington et al. (2014) 提出了另一个非常出名的词嵌入方法 GloVe
5. visual grounding：给定图片和描述性语言，找出图中相关物体或区域
6. 人工智能左膀右臂：深度学习和知识图谱
7. 人工智能>机器学习>深度学习
8. 无人驾驶：面对堵车、大规模的人员流动等情况，只要出现十倍以上的差别，算法一定会失效，中国的路况永远会出现见所未见的情况。做小范围、室内无人车、高速公路会比较靠谱
9. 如果说每出来一个新的 CVPR 论文，你都能找到论文、找到原代码、找到模型，真正能够去理解它后面的东西，在这个过程当中，你其实就在积累你的语言能力，而不仅仅是运用这门技术的能力。以前有一句话：你要选择一条少有人走的路。随便拿一个模型过来，训练一下马上就能用，虽然很爽，但是没有什么用，你要走那条少有人走的路，这样才能在技术的迭代当中，不断增强你的竞争力。
10. 深度学习对自然语言的影响主要体现在以下 6 个方面：

<https://mp.weixin.qq.com/s/-W6SiE1G70JQryWbkY3i5g>

* 端到端训练（输入-输出）：不需要人为设定和优选Feature
* 语义表示（Embedding）和预训练模型（Pretrained Model）：词和词之间距离，句和句之间距离
* Attention（注意力模型）
* 句子的编码方法（RNN/LSTM/GRU/Transformer）：更多是用 Transformer 来对句子编码。对句子编码之后，就可以做翻译、问答、检索等各种应用。
* 编码 - 解码模型（Encoder-Decoder）
* 强化学习：系统根据用户的反馈或者环境的反馈信号，会迭代地修正参数，整个系统得以不断改进。比如对话系统很多用到了强化学习。不过在很多其他NLP任务中，如何体现强化学习是一个还在不停探索的问题。

1. 算力和算法相辅相成，数据和算力是训练模型的重要因素
2. 未来可能有两条路，一是基于数据驱动的。两件事，一把数据掌握好，二是把算力掌握好，就能把模型很好地训练出来。还有一条路就是基于知识，以及推理的这条线，它背后也要靠一些算力，但我们现在还没有到以算力取胜的阶段。
3. 只要你会Python编程，网上找到训练语料，基本上就能把自然语言的第一个模型走出来了。请大家不要被一大堆公式所迷惑，还是要找一个具体的任务试一试。我建议大家拿机器翻译为例，把端到端的训练过程玩起来，沉浸其中，很快就能理解整个自然语言的精髓了。第一件事做好了，比什么都重要。比如你把第一个机器翻译模型训练好，确实Work了，水平也还可以，至少和别人发表的水平差不多，你这时就会信心大涨。只要有一个任务通了，其他自然语言的任务也可以通的。之后可以很容易拓展到深度学习其他领域。
4. 机器学习：监督学习、无监督学习、半监督学习、强化学习
5. 自然语言处理概述：中文信息处理前沿技术进展-刘挺-2018-1105-1628-FINAL（H:\learning\postgraduate\given by teacher）
6. 名：语言符号；实：客观事实、主观事实；知：知识；人：语言使用者。

形式：名；语义：名+实；推理：名+实+知；语用：名+实+知+人。

1. 线性整流函数（rectified linear unit，RElu，修正线性单元）：max（0，z）或sigmoid函数
2. 传统机器学习需要手工提取特征，但深度神经网络不需要
3. 深度学习的重点难点是模型的理解和优化方法
4. softmax函数（归一化指数函数）和sigmoid函数，其中z可能为负
5. CBOW

1 输入层：上下文单词的onehot. {假设单词向量空间dim为V，上下文单词个数为C}

2 所有onehot分别乘以共享的输入权重矩阵W. {VN矩阵，N为自己设定的数，初始化权重矩阵W}

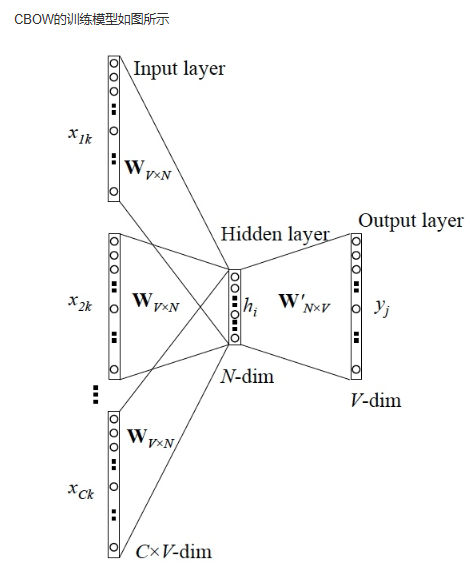
3 所得的向量 {因为是onehot所以为向量} 相加求平均作为隐层向量, size为1N.

4 乘以输出权重矩阵W' {NV}

5 得到向量 {1V} 激活函数处理得到V-dim概率分布 {PS: 因为是onehot嘛，其中的每一维斗代表着一个单词}

6 概率最大的index所指示的单词为预测出的中间词（target word）与true label的onehot做比较，误差越小越好（根据误差更新权重矩阵）

所以，需要定义loss function（一般为交叉熵代价函数），采用梯度下降算法更新W和W'。训练完毕后，输入层的每个单词与矩阵W相乘得到的向量的就是我们想要的词向量（word embedding），这个矩阵(W，那W'呢？)（所有单词的word embedding）也叫做look up table（其实聪明的你已经看出来了，其实这个look up table就是矩阵W自身），也就是说，任何一个单词的onehot乘以这个矩阵都将得到自己的词向量。有了look up table就可以免去训练过程直接查表得到单词的词向量了。

1. 嵌入是高维到低维，如下图CBOW，W可看成嵌入矩阵，将显式表示变为隐式，W’还原，将隐式再变成显式
2. NLP挑战：未见过的生词；词歧义；句歧义；隐喻；指代；不能在翻译员译后学习
3. beam-search：<https://www.zhihu.com/question/54356960>

<https://blog.csdn.net/qq_16234613/article/details/83012046>

1. 熵的概念：注意对数是2

<https://baijiahao.baidu.com/s?id=1615832462981347284&wfr=spider&for=pc>

1. nivdia嵌入式芯片，带cuda：

<https://blog.csdn.net/chengyq116/article/details/82981925>

1. 1
2. 1