# 自然语言处理第二次实验

### 实验目的

* 了解神经网络中的基础模型

神经网络是指一系列受生物学和神经科学启发的数学模型。主要是通过对人脑的神经元网络进行抽象，构建人工神经元，并按照一定拓扑结构来建立神经元之间的连接。前馈神经网络（FNN）是最早发明的简单人工神经网络；卷积神经网络（CNN）是一种具有局部连接、权重共享等特性的深层前馈神经网络；循环神经网络（RNN）是一类具有短期记忆能力的神经网络。

* 了解深度学习框架Pytorch的使用

PyTorch使用python作为开发语言，近年来和TensorFlow, keras, caffe等热门框架一起，成为深度学习开发的主流平台之一。PyTorch的基本元素包含张量(Tensor)、变量(Variable)、神经网络模块(nn.Module)等。

* 了解使用深度学习解决文本分类任务基本流程

以PyTorch为例，一个常规的文本分类任务代码开发流程是：安装并导入相关的深度学习库、数据获取和预处理、定义神经网络、定义损失函数(loss function)和优化器(optimizer)、训练网络和测试网络。

### 实验环境

python 3 + jieba + PyTorch + NumPy + Sklearn + TensorboardX + tqdm

* python3

除了高性能外，拥有NumPy、SciPy等优秀的数值计算、统计分析库。TensorFlow、Caffe等著名的深度学习框架都提供了Python接口。

* jieba

jieba是一款优秀的Python第三方中文分词库，支持三种分词模式：精确模式、全模式和搜索引擎模式。

* PyTorch

PyTorch是一个针对深度学习，并且使用GPU和CPU来优化的tensor library，它是一个以Python优先的深度学习框架，不仅能够实现强大的GPU加速，同时还支持动态神经网络。

* NumPy

NumPy是Python语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

* Sklearn

Sklearn (全称 Scikit-Learn) 是基于 Python 语言的机器学习工具。它建立在 NumPy, SciPy, Pandas 和 Matplotlib 之上，Sklearn 里面有六大任务模块：分别是分类、回归、聚类、降维、模型选择和预处理。

* TensorboardX

Tensorboard 是 TensorFlow 的一个附加工具，可以记录训练过程的数字、图像等内容，以方便研究人员观察神经网络训练过程。可是对于 PyTorch 等其他神经网络训练框架并没有功能像 Tensorboard 一样全面的类似工具，一些已有的工具功能有限或使用起来比较困难。TensorboardX 这个工具使得 TensorFlow 外的其他神经网络框架也可以使用到 Tensorboard 的便捷功能。

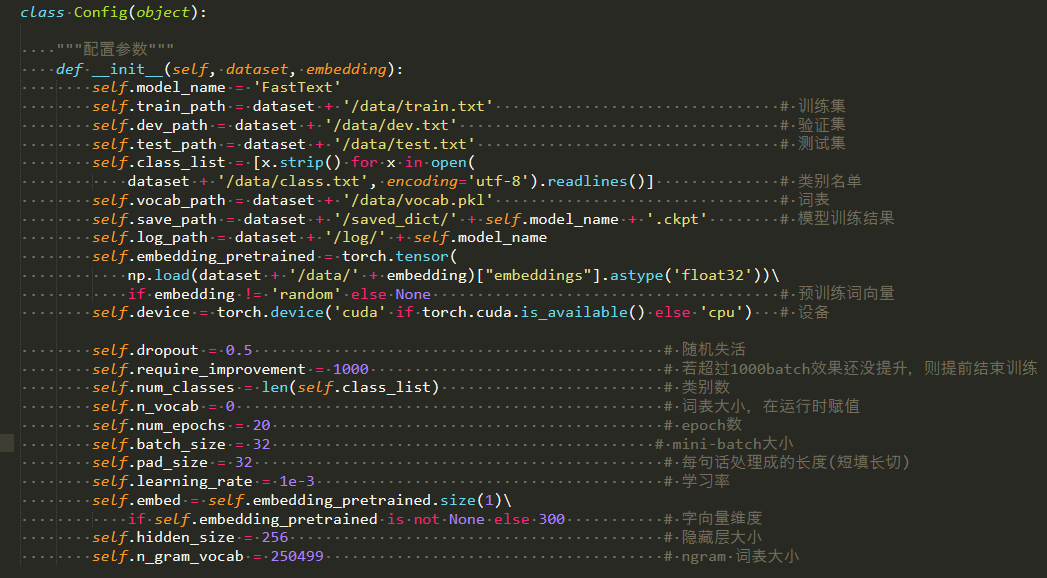
* tqdm

tqdm 是一个快速，可扩展的Python进度条，可以在 Python 长循环中添加一个进度提示信息，用户只需要封装任意的迭代器 tqdm(iterator)。

### 实验步骤

1. 阅读代码，运行并得到结果。

2. 根据已有的TextCNN模型格式，写出BiLSTM模型，代码实现可参考[Chinese-Text-Classification-Pytorch](https://github.com/649453932/Chinese-Text-Classification-Pytorch)。TexcCNN模型相关超参数配置在\Chinese-Text-Classification-Pytorch \models\TextCNN.py



3. 调整batch size参数，取值分别为[8, 16, 32, 64] ，画出TextCNN和BiLSTM训练集和验证集的loss折线图。

4. 调整embedding size参数，取值分别为[64, 128, 256] ，画出TextCNN和BiLSTM训练集和验证集的loss折线图。

5. 调整hidden size参数，取值分别为[64, 128, 256] ，画出TextCNN和BiLSTM训练集和验证集的loss折线图。

6. 调整learning rate参数，取值分别为[1e-2, 5e-3, 1e-3, 5e-4, 1e-4] ，画出TextCNN和

BiLSTM训练集和验证集的loss折线图。

7. 调整dropout参数，取值分别为[0.1, 0.2, 0.3, 0.4, 0.5] ，画出TextCNN和BiLSTM训练集和验证集的loss折线图。

8. 选出最好的参数组合（不局限于以上参数组合），列出参数并将实验结果写入表格进行对比分析，例如：

|  |  |
| --- | --- |
| 模型 | 准确率 |
| TextCNN | 86.80% |
| BiLSTM | 87.68% |

loss图示例：



### 提交时间

**11月3号截止**

由吴雨欣同学负责收集，文件命名方式：姓名-学号-第2次实验

### 实验要求

* 完成所有实验内容
* 良好的代码风格
* 完整的实验报告

### 参考资料

1. [pytorch1.0.0官方文档](https://pytorch.org/docs/1.0.0/)

2. [《神经网络与深度学习》](https://nndl.github.io/)

3. [Convolutional Neural Networks for Sentence Classification](https://arxiv.org/abs/1408.5882)

4. [Chinese-Text-Classification-Pytorch](https://github.com/649453932/Chinese-Text-Classification-Pytorch)

**武汉大学国家网络安全学院**

**实验报告**

**课程名称 自然语言处理**

**专业年级 网络空间安全**

**姓 名 戴挺**

**学 号 2018302060188**

**协 作 者 无**

**实验学期 2021-2022　 学年 第一 学期**

**课堂时数 课外时数**

**填写时间 2021 年 10 月 28 日**

|  |
| --- |
| **实验介绍** |
| **【实验名称】：神经网络基础模型** |
| **【实验目的】：**   * 了解神经网络中的基础模型   神经网络是指一系列受生物学和神经科学启发的数学模型。主要是通过对人脑的神经元网络进行抽象，构建人工神经元，并按照一定拓扑结构来建立神经元之间的连接。前馈神经网络（FNN）是最早发明的简单人工神经网络；卷积神经网络（CNN）是一种具有局部连接、权重共享等特性的深层前馈神经网络；循环神经网络（RNN）是一类具有短期记忆能力的神经网络。   * 了解深度学习框架Pytorch的使用   PyTorch使用python作为开发语言，近年来和TensorFlow, keras, caffe等热门框架一起，成为深度学习开发的主流平台之一。PyTorch的基本元素包含张量(Tensor)、变量(Variable)、神经网络模块(nn.Module)等。   * 了解使用深度学习解决文本分类任务基本流程   以PyTorch为例，一个常规的文本分类任务代码开发流程是：安装并导入相关的深度学习库、数据获取和预处理、定义神经网络、定义损失函数(loss function)和优化器(optimizer)、训练网络和测试网络。    **【实验环境】：**  python 3 + jieba + PyTorch + NumPy + Sklearn + TensorboardX + tqdm   * python3   除了高性能外，拥有NumPy、SciPy等优秀的数值计算、统计分析库。TensorFlow、Caffe等著名的深度学习框架都提供了Python接口。   * jieba   jieba是一款优秀的Python第三方中文分词库，支持三种分词模式：精确模式、全模式和搜索引擎模式。   * PyTorch   PyTorch是一个针对深度学习，并且使用GPU和CPU来优化的tensor library，它是一个以Python优先的深度学习框架，不仅能够实现强大的GPU加速，同时还支持动态神经网络。   * NumPy   NumPy是Python语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。   * Sklearn   Sklearn (全称 Scikit-Learn) 是基于 Python 语言的机器学习工具。它建立在 NumPy, SciPy, Pandas 和 Matplotlib 之上，Sklearn 里面有六大任务模块：分别是分类、回归、聚类、降维、模型选择和预处理。   * TensorboardX   Tensorboard 是 TensorFlow 的一个附加工具，可以记录训练过程的数字、图像等内容，以方便研究人员观察神经网络训练过程。可是对于 PyTorch 等其他神经网络训练框架并没有功能像 Tensorboard 一样全面的类似工具，一些已有的工具功能有限或使用起来比较困难。TensorboardX 这个工具使得 TensorFlow 外的其他神经网络框架也可以使用到 Tensorboard 的便捷功能。   * tqdm   tqdm 是一个快速，可扩展的Python进度条，可以在 Python 长循环中添加一个进度提示信息，用户只需要封装任意的迭代器 tqdm(iterator)。  **【参考文献】：**  1. [pytorch1.0.0官方文档](https://pytorch.org/docs/1.0.0/)  2. [《神经网络与深度学习》](https://nndl.github.io/)  3. [Convolutional Neural Networks for Sentence Classification](https://arxiv.org/abs/1408.5882)  4. [Chinese-Text-Classification-Pytorch](https://github.com/649453932/Chinese-Text-Classification-Pytorch) |
| **实验内容** |
| **【实验方案设计】：**   1. 阅读代码，运行并得到结果。 2. 根据已有的TextCNN模型格式，写出BiLSTM模型，代码实现可参考[Chinese-Text-Classification-Pytorch](https://github.com/649453932/Chinese-Text-Classification-Pytorch)。代码如下：     3. 调整batch size参数，取值分别为[8, 16, 32, 64] ，画出TextCNN和BiLSTM训练集和验证集的loss折线图。  4. 调整embedding size参数，取值分别为[64, 128, 256] ，画出TextCNN和BiLSTM训练集和验证集的loss折线图。  5. 调整hidden size参数，取值分别为[64, 128, 256] ，画出TextCNN和BiLSTM训练集和验证集的loss折线图。  6. 调整learning rate参数，取值分别为[1e-2, 5e-3, 1e-3, 5e-4, 1e-4] ，画出TextCNN和BiLSTM训练集和验证集的loss折线图。  7. 调整dropout参数，取值分别为[0.1, 0.2, 0.3, 0.4, 0.5] ，画出TextCNN和BiLSTM训练集和验证集的loss折线图。  上述3-7为5个调参过程为：控制其他超参数不变，调整某个超参数来观测train\_loss以及dev\_loss随训练过程的变化。这里为了不每次手动调参，在此我们基于此对原始代码run.py进行修改得到run\_v2.py。接下来对部分代表性代码块做解释。详细代码见代码文件。**关于超参数的一些概念和知识，会在实验总结中归纳。**  首先提前在run\_v2.py预设5个超参数列表  batch\_size\_list = [8, 16,32, 64,128]  embedding\_size\_list = [64,128,256]  hidden\_size\_list = [64,128,256]  learning\_rate\_list = [1e-2,5e-3,1e-3,5e-4,1e-4]  drop\_out\_list = [0.1,0.2,0.3,0.4,0.5]    每次调完参数后，我们保存对应的loss数据到对应的文件。  为方便操作，我们给不同的超参数打上label\_num。  Batch:1、Embed:2、Hidden:3、learning\_rate:4、drop\_out:5  将原本run.py的中的main函数修改为train\_what(),入口参数为label\_num以及需要调整的超参数的值（红框内config参数调整），函数返回训练过程中的train\_loss,dev\_loss,并写入文件。    同时需要对train\_eval.py中train函数作一定如下修改（原本的train函数不会返回loss）      **由上述代码也可以看出这里输出的loss的计算方法：每个epoch中，取所有训练，验证的loss取均值作为本次epoch对应的loss。**  运行该代码run\_v2.py，执行python3 run\_v2.py --model TextCNN --embedding random。之后再运行plot\_data.py即可绘制图像。（plot\_data.py文件无重要的过程，故不在此处贴代码，详见文件代码）  8. 选出最好的参数组合（不局限于以上参数组合），列出参数并将实验结果写入表格进行对比分析。  在CNN中超参数的调整我们**选取遗传算法**来进行。以下为算法大致流程：  种群初始化采用随机初始化的方式，在每一轮迭代中，会经过选择，交叉和变异等过程，然后用得到的每组超参数组合训练TextCNN并计算每种超参数组合对应的模型的适应值，这里使用在模型测试集上的准确率作为适应值。  在选择的过程中，首先对各个个体根据适应值进行排序，随后基于轮盘赌模型，选择得到继续遗传的个体。下一步是交叉和变异过程，在t条被保留的染色体中每次随机选择两条染色体作为父母，染色体进行随机交配，产生后代染色体，直到本轮种群数量大于等于pop\_size。由于超参数的取值范围各不相同，因此，在进行交叉的时候，只允许父母染色体同种基因之间相互交叉。对于每条后代染色体，则随机选择一个基因进行变异。  接下来为具体分析部分代表性过程及相关代码实现（完整详见GA.py）  首先介绍染色体编码，本次实验把每个超参数视为一个基因，超参数类型在本次实验中均为数值类型，因此基因的取值可能是整数，小数。所有超参数的组合构成一条染色体。  本次实验中，我们观察实验给出的初始超参数  batch\_size\_list = [8,16,32,64,128]  embedding\_size\_list = [64,128,256]  hidden\_size\_list = [64,128,256]  learning\_rate\_list = [1e-2,5e-3,1e-3,5e-4,1e-4]  drop\_out\_list = [0.1,0.2,0.3,0.4,0.5]  基于上述参考的超参数的取值范围，我们定义batch\_size、embedding\_size、hidden\_size为8bit长度的二进制码，0x00-0xff映射到1-256。learning\_rate的最大值与最小值的比率达到100，因此设置为7bit长度的二进制码，0000000-1111111映射到（1-128） / 10^5。drop\_out由于自身属性只能设置在0-1之间，故考虑用3bit长度的二进制码表示，000-111映射到（1-8）/ 10。  由于定义的5个参数对应的染色体长度不同，因此为了更方便的操作，我们在存储染色体时，**考虑每个超参数即染色体片段不以二进制形式存储，而是直接存储对应的值。**这时原本的个体就是一个1\* 5的向量，而不是一个1 \* （8 + 8 + 8 + 7 + 3）的向量。  在染色体交叉互换，变异的时候再考虑将对应的染色体片段切换到二进制形式进行对应的操作，再转换为对应的数值存储到染色体。  为方便操作，我们给不同的超参数打上label\_num。  Batch:1、Embed:2、Hidden:3、learning\_rate:4、drop\_out:5  给出超参数数值，二进制切换函数，代码如下：      在染色体的定义确定之后，我们可以开始初始化种群。通过传入起始给定的5组超参数，希望的种群的大小，随机的生成预期数量的个体，最终初代种群以pop\_size \* parametert\_size的二维数组返回，代码如下。（p1-p5为上述给出的5组超参数。）    接下来为自然选择过程，由于我们希望求得最优的超参数组合使得模型最终的测试精度最高，因此训练和选择的参数指标因为test\_accuracy。如下为训练完成后并计算test\_acc代码，每个个体作为入口参数，依据不同的5个染色体片段的值对模型的超参数进行重置。(红框内对超参数进行调整)    基于计算出每代种群的fit\_value即test\_acc，便可基于此进行选择，这里利用轮盘模型进行个体筛选。轮盘模型的优点在于对测试准确率较低的个体不会直接失去遗传的机会，每个个体均有遗传的机会，只是fit\_value更大的个体遗传的几乎更大，反之亦然。代码如下：    经过选择后，新的种群大小依然维持不变，接下来则是遗传（交叉互换）和变异的过程。  对于种群中每个个体两两交配，发生染色体互换，产生新的个体。而每个个体均有5段染色体（对应5个不同超参数），每段染色体均有可能可能发生互换，提前预设染色体交叉互换的概率pc。通过生成0-1随机数rand，若rand < pc,该段染色体发生交叉互换；若rand >= pc,该段染色体不发生交叉互换。再根据发生互换的染色体种类随机生成互换点位cpoint，染色体保留自身cpoint之前的片段，互换cpoint之后的片段。代码如下：    交叉互换生成的新的子代个体再进行变异操作，同样是每个个体的5段的染色体片段同样都可能发生变异。通过预设变异概率pm，生成0-1随机数rand，若rand < pm,该段染色体发生变异；若rand >= pm,该段染色体不发生变异。再根据染色体种类即每段染色体的长度随机生成变异位点mpoint，mpoint位由0变异为1或者1变异为0。代码如下：    至此一轮进化完成，此后就是依据循环次数，进行进化迭代，以下为main函数：    在main函数中设置初始的5组参数列表，以及迭代次数，以及变异&交叉互换的概率。每次迭代输出该代中test\_acc最高的参数组合。而后进过选择、交叉互换、变异，生成新的子代，循环迭代。遗传的终止条件可以设置固定次数，或者设置一个阈值，当best\_test\_acc的增幅小于这个阈值时，则停止迭代。  修改train\_eval.py文件中的train（）    同时修改train\_eval.py的 test()函数中添加test\_acc作为返回值。  ps:执行遗传算法代码，执行命令python3 GA.py —model TextCNN —embedding random  **【实验结果分析】：**  显示3-7输出的图片：（每个epoch中，取所有训练，验证的loss取均值作为本次epoch对应的loss）  1、以下为控制其他4个参数不变，调整batch\_size训练过程中每个epoch对应的train\_loss以及dev\_loss.。      观察上述实验数据可知，batch\_size越大，模型收敛越快，loss会更快降低到比较小的数值。  2、以下为控制其他4个变量不变，调整embedding\_size的大小每个epoch对应的train\_loss&dev\_loss      由上述数据可以看出，embedding size越大，验证集起始的loss越大，表现出过拟合的现象。  3、以下为控制其他4个变量不变，调整hidden\_size的大小每个epoch对应的train\_loss&dev\_loss      由上述数据可以看出，hidden\_size越大，验证集起始的loss越大，表现出过拟合的现象，且模型收敛速度也更慢。  4、以下为控制其他4个变量不变，调整learning\_rate的大小每个epoch对应的train\_loss&dev\_loss      可以看出learning\_rate在给出的几个参数范围内，数值越大，训练集&验证集起始的loss越大，表现出过拟合的现象模型收敛越慢。  5、以下为控制其他4个变量不变，调整dropout的大小每个epoch对应的train\_loss&dev\_loss      在这里模型参数调整好像没有多大差别（可能原因是由于loss定义的是一个epoch的均值，细化后也许可以看出区别）。  遗传算法求解出的结果如下：    但是由于设置的参数遗传算法参数不是很好（种群个体数量每代只有10个个体，且只迭代了10代），其实训练的最佳正确率并不是最佳的，中途有一次识别正确率达到90.5%左右，但是相信在调整参数之后再运行，应该可以调出最优参数组合。但由于时间原因暂时没能调出更优的参数组合。 |
| **【实验总结】：**  1、TextCNN循环神经网络  TextCNN模型主要使用了一维卷积层和时序最大池化层。假设输入的文本序列由nn个词组成，每个词用dd维的词向量表示。那么输入样本的宽为nn，高为1，输入通道数为dd。textCNN的计算主要分为以下几步。  1、定义多个一维卷积核，并使用这些卷积核对输入分别做卷积计算。宽度不同的卷积核可能会捕捉到不同个数的相邻词的相关性。  2、对输出的所有通道分别做时序最大池化，再将这些通道的池化输出值连结为向量。  3、通过全连接层将连结后的向量变换为有关各类别的输出。这一步可以使用丢弃层应对过拟合。  这里给出参考资料中的一个很好理解的图显示TextCNN模型框架    **2、超参数相关介绍**  BATCH\_SIZE:即一次训练所抓取的数据样本数量；BATCH\_SIZE的大小影响训练速度和模型优化。同时iter\_per\_epoch = num\_train // epoch\_size，依据此可知其大小同样影响每一epoch训练模型次数。  调整batch\_size最大的好处在于使得CPU或GPU满载运行，提高了训练的速度。若BATCH\_SIZE设置合适，梯度会变准确。此时再增加BATCH\_SIZE大小，梯度也不会变得更准确。  Embedding\_size：在TextCNN模型中的embedding层的神经元的个数，输入的文本经过分词后，进过embedding层处理后生成对应embedding\_size维的向量，维度从几十维到上千维，一般会在中间存在一个效果最好的维度，维度过低模型的表现能力不够，维度过高容易过拟合现象。  Hidden\_size:隐藏层的层数，显然，该参数越大，模型越复杂，也容易导致过拟合。  Learning\_rate:参数学习的步长，依据神经网络前馈信息调整参数的幅度。该参数过小，降低模型收敛速度；过大同样容易造成模型收敛速度，最坏的情况是模型永远无法收敛，不停地在收敛点的两侧震荡。  Dropout：可以作为训练深度神经网络的一种trick供选择。在每个训练批次中，通过忽略drop\_out比率的特征检测器，可以明显地减少过拟合现象，提高模型的泛化能力。这种方式可以减少特征检测器（隐层节点）间的相互作用，检测器相互作用是指某些检测器依赖其他检测器才能发挥作用。 |
| **评语及评分（指导教师）** |
| **【评语】：**      **评分：**  **日期：** |

**附件2：**

**实验报告说明**

**1．实验名称：**要用最简练的语言反映实验的内容。

**2．实验目的**：目的要明确，要抓住重点。

**3．实验环境**：实验用的软硬件环境（配置）。

**4．实验方案设计（思路、**步骤和方法等**）**：这是实验报告极其重要的内容。包括概要设计、详细设计和核心算法说明及分析，系统开发工具等。应同时提交程序或设计电子版。

对于**设计型和综合型实验**，在上述内容基础上还应该画出流程图、设计思路和设计方法，再配以相应的文字说明。

对于**创新型实验**，还应注明其创新点、特色。

**5．实验结果分析：**即根据实验过程中所见到的现象和测得的数据，进行对比分析并做出结论（可以将部分测试结果进行截屏）。

**6．实验总结：**对本次实验的心得体会，所遇到的问题及解决方法，其他思考和建议。

**7．评语及评分：**指导教师依据学生的实际报告内容，用简练语言给出本次实验报告的评价和价值。