|  |
| --- |
|  |
| 软件设计文档 |
| 基于PyTorch的前沿深度学习算法集成应用程序接口 |
|  |
| Team A |
| 2020/5/12 |

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 版本说明 | 审核人 |
| V1.0 | 2020/5/12 | 张崇智，秦浩桐，黄涵，王茵迪，赵永驰，吴振赫 |  | 高明骏 |
| V2.0 | 2020/5/17 | 张崇智，赵永驰 | 对软件所需的环境依赖以及软件具体的使用规范做了修改 | 高明骏 |
| V3.0 | 2020/5/18 | 王茵迪 | 对阅读理解模块文档进行了修改 | 张崇智 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 引言 4](#_Toc40644171)

[1.1 目的 4](#_Toc40644172)

[1.2 系统概述 4](#_Toc40644173)

[1.3 文档概述 5](#_Toc40644174)

[2 基本框架搭建 5](#_Toc40644175)

[2.1 安装Python 5](#_Toc40644176)

[2.2 安装PyTorch 6](#_Toc40644177)

[2.3 安装distutils并进行配置 6](#_Toc40644178)

[3 系统设计说明 7](#_Toc40644179)

[3.1 对抗样本生成模块设计说明 7](#_Toc40644180)

[3.1.1 功能描述 7](#_Toc40644181)

[3.1.2 输入项目 7](#_Toc40644182)

[3.1.3 输出项目 8](#_Toc40644183)

[3.1.4 接口介绍 8](#_Toc40644184)

[3.1.5 程序逻辑 8](#_Toc40644185)

[3.1.6 测试要点 10](#_Toc40644186)

[3.2 模型量化模块设计说明 10](#_Toc40644187)

[3.2.1 功能描述 10](#_Toc40644188)

[3.2.2 执行项目 10](#_Toc40644189)

[3.2.3 输出项目 11](#_Toc40644190)

[3.2.4 模块介绍 11](#_Toc40644191)

[3.2.5 程序逻辑 11](#_Toc40644192)

[3.2.6 测试要点 13](#_Toc40644193)

[3.3 目标检测模块设计说明 13](#_Toc40644194)

[3.3.1 功能描述 13](#_Toc40644195)

[3.3.2 输入项目 13](#_Toc40644196)

[3.3.3 输出项目 14](#_Toc40644197)

[3.3.4 接口介绍 14](#_Toc40644198)

[3.3.5 程序逻辑 14](#_Toc40644199)

[3.3.6 测试要点 15](#_Toc40644200)

[3.4 阅读理解模块设计说明 16](#_Toc40644201)

[3.4.1 功能描述 16](#_Toc40644202)

[3.4.2 输入项目 16](#_Toc40644203)

[3.4.3 输出项目 16](#_Toc40644204)

[3.4.4 接口介绍 16](#_Toc40644205)

[3.4.5 程序逻辑 17](#_Toc40644206)

[3.4.6 测试要点 18](#_Toc40644207)

[3.5 主动学习模块设计说明 18](#_Toc40644208)

[3.5.1 功能描述 18](#_Toc40644209)

[3.5.2 输入项目 19](#_Toc40644210)

[3.5.3 输出项目 19](#_Toc40644211)

[3.5.4 接口介绍 19](#_Toc40644212)

[3.5.5 程序逻辑 20](#_Toc40644213)

[3.5.6 测试要点 22](#_Toc40644214)

# 引言

## 目的

在pytorch框架的基础上，对项目计划的方案、实现的目标进行介绍，以指导开发和测试工作的进行。

## 系统概述

API是指一些预先定义的函数，或指软件系统不同组成部分衔接的约定。目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。这使得使用者可以充分利用到前人所做出的成果，不必再将已有的工作重复做一遍。

PyTorch是Torch的Python版本，是由Facebook开源的神经网络框架，专门针对 GPU 加速的深度神经网络（DNNs）编程。本平台在基于PyTorch框架的情况下将一些深度学习领域的经典或优秀算法封装成了API，以便使用者可以简单的调用这些已有的算法或工作。它对于学生、科研人员、教师、企业员工等工作群体具有较高的使用价值，可以提高深度学习实验的效率，有助于推动研究和工程项目的开展进行。

本平台主要包含了多个子模块，分别对应五个常用的深度学习领域。它们分别是对抗样本生成模块、模型量化模块、阅读理解模块、主动学习模块。我们为每一个子模块都提供了多种实用算法的API，通过调用相应的函数以及输入参数，可以快速的实现对应的算法内容。对抗样本生成模块中提供了多种深度学习对抗攻防领域中的前沿算法，包括FGSM、PGD、C&W等。模型量化模块中提供了多种在线量化压缩算法，包括IR-Net、LQ-Nets、BWN等。阅读理解模块中提供了多种自然语言文本问答算法，包括VNET、SNET、BERT等。主动学习模块提供了多种主动学习方向的经典算法，包括Uncertainty、Coreset、Bayesian等。系统的架构图如图1.1所示。

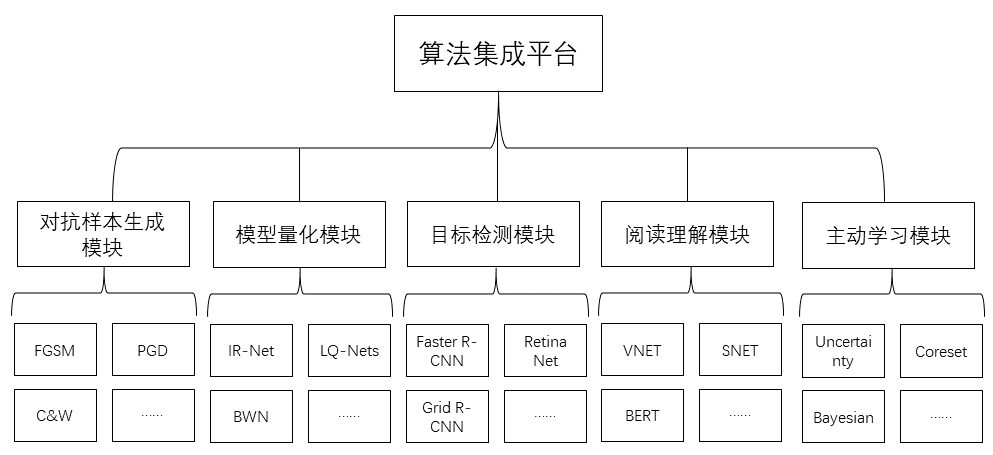


图 1.1 系统架构图

## 文档概述

本文档是在PyTorch的基础上，介绍了基于PyTorch框架的算法集成平台的架构与功能实现。

* 以用例图和流程图的形式给出平台中各模块的设计和实现方案，并对用例图和流程图中的用例和逻辑做详细的描述。
* 描述了算法集成平台中各子模块内部的数据定义、数据结构和功能描述。
* 给出初步的测试要点。

# 基本框架搭建

## 安装Python

Unix & Linux 平台安装 Python:

* 打开 WEB 浏览器访问<https://www.python.org/downloads/source/>
* 选择适用 于Unix/Linux 的源码压缩包。
* 下载及解压压缩包。
* 如果你需要自定义一些选项修改Modules/Setup
* 执行 ./configure 脚本
* make
* make install

执行以上操作后，Python 会安装在 /usr/local/bin 目录中，Python 库安装在 /usr/local/lib/pythonXX，XX 为你使用的 Python 的版本号。

（2）Window 平台安装 Python:

* 打开 WEB 浏览器访问<https://www.python.org/downloads/windows/>
* 在下载列表中选择Window平台安装包，包格式为：python-XYZ.msi 文件 ， XYZ 为你要安装的版本号。
* 要使用安装程序 python-XYZ.msi, Windows 系统必须支持 Microsoft Installer 2.0 搭配使用。只要保存安装文件到本地计算机，然后运行它，看看你的机器支持 MSI。Windows XP 和更高版本已经有 MSI，很多老机器也可以安装 MSI。
* 下载后，双击下载包，进入 Python 安装向导，安装非常简单，你只需要使用默认的设置一直点击"下一步"直到安装完成即可。

## 安装PyTorch

访问Pytorch官网：(<https://pytorch.org/>) ，选择与自己的环境版本，复制官网上给出的安装指令，进入命令行输入指令进行安装。

## 安装distutils并进行配置

distutils可以用来在Python环境中构建和安装额外的模块。本项目构建的算法集成平台中实现各个模块和算法最终依赖于distutils模块的功能来进行打包发布。作为开发者，除了编写源码之外，还需要：

* 编写setup脚本（一般是setup.py）；
* 编写一个setup配置文件（可选）；
* 创建一个源码发布；
* 创建一个或多个构建（二进制）发布（可选）;

之后方可将项目进行发布，以便于用户安装使用。

# 系统设计说明

## 对抗样本生成模块设计说明

### 功能描述

对抗样本生成模块是为深度学习对抗攻防领域的研究人员提供的当今经典以及前沿对抗攻击算法库。用户可以调用该模块完成多种对抗样本的生成，减少科研人员复现方法的时间，提高科研效率。具体地，用户应将使用的模型和数据集以及其他辅助变量（下文有详细规定），作为函数的输入。对应的对抗攻击算法将针对模型信息，在数据集的每一个样本上生成对抗样本，并将新生成的样本作为一个数据集输出给用户。

### 输入项目

* model：待攻击的模型，为torch.nn.module类型的对象，即其包含forward()函数以及支持PyTorch的自动求导机制；
* data\_loader：存放了待攻击样本集合的torch.utils.data.DataLoader类型的对象，其是一个迭代器，可以依次取出各个批次的数据，同时该数据集对应图像数据，所经历的transformation过程应仅包含toTensor()操作，如有normalization操作，在下面的normalizer中进行处理；
* criterion：损失函数，即torch.nn中的loss函数，例如torch.nn. CrossEntropyLoss()，用于作为对抗攻击中的辅助评判函数；
* epsilon：float类型，表示攻击噪声的最大幅度，取值为[0,1]；
* iteration：int类型，表示攻击迭代次数（迭代攻击所需参数）；
* normalizer：PyTorch中的运算函数，用于进行图像的normalization过程，若攻击模型对于输入样本有normalization要求，则需要构造，一个例子如下：

CUDA\_AVAILABLE = torch.cuda.is\_available()

mu = torch.Tensor((0.485, 0.456, 0.406)).unsqueeze(-1).unsqueeze(-1).cuda() if(CUDA\_AVAILABLE) else torch.Tensor((0.485, 0.456, 0.406)).unsqueeze(-1).unsqueeze(-1)

std = torch.Tensor((0.229, 0.224, 0.225)).unsqueeze(-1).unsqueeze(-1).cuda() if(CUDA\_AVAILABLE) else torch.Tensor((0.229, 0.224, 0.225)).unsqueeze(-1).unsqueeze(-1)

normalizer = lambda x: (x-mu)/std

其中，mu和std中的(0.485, 0.456, 0.406)和(0.229, 0.224, 0.225)分别代表三个通道的均值μ和标准差σ，因此，当不需要normalization过程时，mu和std中分别填写(0, 0, 0)和(1, 1, 1)。

**注：本模块函数可在GPU和CPU两种环境下进行运行，会自动检测当前是否支持CUDA加速运算，而选择最优的环境执行（如可能则自动加载到GPU），因此在所有的输入都应在CPU环境中，不应事先加载GPU。**

### 输出项目

* 运行正常：模型在生成对应集合的对抗样本x\_adv和对应的标签label，均为torch.Tensor类型，仅在Fourier\_based\_Corruption方法中只返回对抗样本x\_adv；
* 运行失败：抛出异常。

### 接口介绍

该模块包含6种对抗攻击算法，它们的输入和输出定义如上文所示：

* fgsm(model, data\_loader, criterion, epsilon, normalizer)
* pgd(model, data\_loader, criterion, epsilon, normalizer, iteration)
* step\_ll(model, data\_loader, criterion, epsilon, normalizer)
* momentum\_ifgsm(model, data\_loader, criterion, epsilon, normalizer, iteration, attack\_momentum)
* CarliniWagnerL2(model, data\_loader, steps, search\_steps, normalizer, debug=False)
* Fourier\_based\_Corruption(dataset, imgsize, position)

### 程序逻辑

下面是该部分的流程图，以及对流程图的介绍。



图3.1 对抗样本生成流程图

如图3.1所示,首先判断参数是否合法，如果不合法则抛出异常，该模块结束。接下来逐批次生成对抗样本。如果生成过程出现设备问题，抛出异常，该模块结束。表3.1为整个模块的RUCM说明。

表3.1 生成对抗样本用例描述表

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 生成对抗样本 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的对抗攻击方法对提供的模型在指定数据集上生成对抗样本 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | CV 研究人员 | | |
| Secondary Actor | None | | |
| Dependency | INCLUDE USE CASE 加载,INCLUDE USE CASE 保存 | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户载入模型 | |
| 3 | 用户载入数据 | |
| 4 | 用户处理数据成满足生成对抗样本模块输入要求的格式 | |
| 5 | 用户调用工具包中的指定对抗攻击算法 | |
| 6 | 用户将模型和数据集传入对抗样本生成函数中 | |
| 7 | 系统生成对抗样本结果 | |
| Postcondition | | 系统不报告错误信息，正常运行 |
| Bounded  Alternative  Flow | **RFS** 2，3，4，5 | | |
| 1 | 函数调用失败 | |
| 2 | 系统报告命令不合法信息 | |
| 3 | ABORT | |
| Postcondition | | 程序中断，用户依据报告信息修改代码 |

### 测试要点

* 导入工具包是否会抛出异常
* 输入不符合规范时是否会抛出异常
* 生成过程中设备异常是否会抛出异常
* 符合规范且设备正常时是否能正常运行

## 模型量化模块设计说明

### 功能描述

模型量化模块是为神经网络模型量化领域的研究人员提供的当今经典以及前沿的在线量化压缩算法库。用户可以调用该模块完成多种尺寸、位宽的模型量化，减少科研人员复现方法的时间，提高科研效率。具体地，用户应将使用的模型和数据集处理成符合要求的格式，作为函数的输入。对应的模型量化算法将针对模型的具体信息，针对每个模型以及指定的算法、位宽生成替换模型，并将新的模型保存以便用户使用。

### 执行项目

* QNNs.bnn\_res20\_1w1a().\_\_main\_\_().main():生成使用bnn算法量化的量化模型，精度为1bit weight/1bit activation
* QNNs.bwn\_res20\_1w32a().\_\_main\_\_().main():生成使用bwn算法量化的量化模型，精度为1bit weight/32bit activation
* QNNs.xnor\_res20\_1w1a().\_\_main\_\_().main():生成使用xnor算法量化的量化模型，精度为1bit weight/1bit activation
* QNNs.horq\_res20\_1w2a().\_\_main\_\_().main():生成使用horq算法量化的量化模型，精度为1bit weight/2bit activation
* QNNs.irnet\_res20\_1w1a().\_\_main\_\_().main():生成使用irnet算法量化的量化模型，精度为1bit weight/1bit activation

### 输出项目

正常运行：模型在生成对应集合的对抗样本/抛出异常

### 模块介绍

QNNs模块包含5种模型量化算法：

* QNNs.bnn\_res20\_1w1a
* QNNs.bwn\_res20\_1w32a
* QNNs.xnor\_res20\_1w1a
* QNNs.horq\_res20\_1w2a
* QNNs.irnet\_res20\_1w1a

### 程序逻辑

下面是该部分的流程图，以及对流程图的介绍。

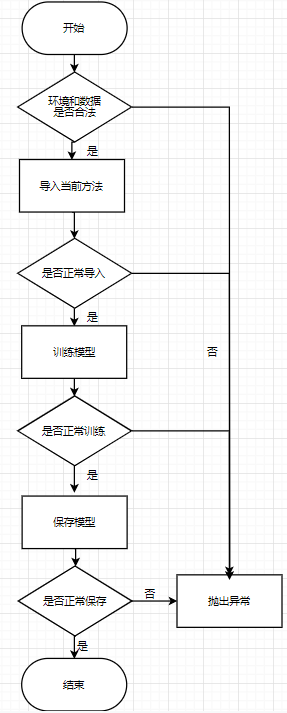


图3.2 生成量化模型流程图

如图3.2所示,首先判断环境和数据是否合法并导入模型，如果不合法则抛出异常，该模块结束。接下来训练量化模型、保存模型。如果训练、保存过程出现设备问题，抛出异常，该模块结束。表3.2为整个模块的RUCM说明。

表3.2 生成量化模型用例描述表

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 生成量化模型 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的神经网络量化模型方法训练生成模型。 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | CV 研究人员 | | |
| Secondary Actor | None | | |
| Dependency | INCLUDE USE CASE 加载,INCLUDE USE CASE 保存 | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户载入模型 | |
| 3 | 用户载入数据 | |
| 4 | 用户处理数据成满足模型量化模块输入要求的格式 | |
| 5 | 用户调用工具包中的指定模型量化算法 | |
| 6 | 系统生成量化模型结果 | |
| Postcondition | | 系统不报告错误信息，正常运行 |
| Bounded  Alternative  Flow | **RFS** 1，2，3，4，5，6 | | |
| 1 | 函数调用失败 | |
| 2 | 系统报告命令不合法信息 | |
| 3 | ABORT | |
| Postcondition | | 程序中断，用户依据报告信息修改代码 |

### 测试要点

* 导入工具包是否会抛出异常
* 输入不符合规范时是否会抛出异常
* 生成过程中设备异常是否会抛出异常
* 符合规范且设备正常时是否能正常运行

## 目标检测模块设计说明

### 功能描述

目标检测模块是为深度学习目标检测领域的研究人员提供当今经典及前沿的目标检测算法库。用户可以调用该模块以多种算法检测图像中的前景物体，减少科研人员复现方法的时间，提高科研效率。具体地，用户应将使用的数据集处理成符合要求的格式并选择预训练模型，以参数的形式输入。对应的目标检测算法将根据指定的参数信息，载入预训练模型并对数据集的图片进行物体的候选框检测的训练或测试。

### 输入项目

* main\_function：根据参数搭建网络结构的函数
* args：用户输入的参数及默认参数

### 输出项目

训练记录及保存的模型/抛出异常

### 接口介绍

该模块包装了Fast RCNN、Faster RCNN和RetinaNet三种，它们的都通过统一的接口调用，通过参数config-file传入不同的yaml文件来选择不同算法：

* launch\_interface (main, args)

### 程序逻辑

下面是该部分的流程图，以及对流程图的介绍。

图片包含 游戏机

描述已自动生成

图3.3 目标检测流程图

如图3.3所示,首先判断参数是否合法，如果不合法则抛出异常，该模块结束。接下来逐批次进行目标检测。如果生成过程出现设备问题，抛出异常，该模块结束。表3.3为整个模块的RUCM说明。

表3.3目标检测用例描述表

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 调用目标检测算法生成物体候选框 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的目标检测方法使用指定的预训练模型在指定数据集上生成物体候选框，并保存到本地 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | CV 研究人员 | | |
| Secondary Actor | None | | |
| Dependency | INCLUDE USE CASE 加载模型，INCLUDE USE CASE 处理数据 | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户加载模型 | |
| 3 | 用户载入已处理成满足目标检测函数输入规格要求的数据 | |
| 4 | 用户调用工具包中的指定目标检测算法，将模型和数据集传入函数中 | |
| 5 | 函数返回生成的物体候选框 | |
| Postcondition | | 系统不报告错误信息，正常运行 |
| Bounded  Alternative  Flow | **RFS** 2，3，4 | | |
| 1 | 函数调用失败 | |
| 2 | 系统报告命令不合法信息 | |
| 3 | ABORT | |
| Postcondition | | 程序中断，用户依据报告信息修改代码 |

### 测试要点

* 导入工具包是否会抛出异常
* 输入不符合规范时是否会抛出异常
* 检测过程中设备异常是否会抛出异常
* 符合规范且设备正常时是否能正常运行

## 阅读理解模块设计说明

### 功能描述

阅读理解模块是为想要构建问答型阅读理解系统的研究人员提供的工具，融合了NLP中的技术，提供便捷的文本预处理和深度神经网络构建、训练、测试框架， 用户可以处理自己的文本数据，然后使用训练好的模型进行问答预测，或重新训练自己的模型。

### 输入项目

* config：声明了模型参数、训练参数，结果文件位置的文件
* device：指定运行位于cup或gpu设备

### 输出项目

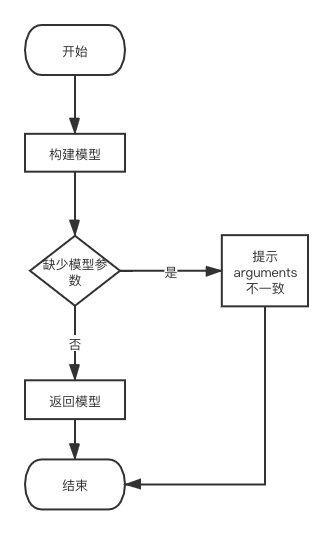
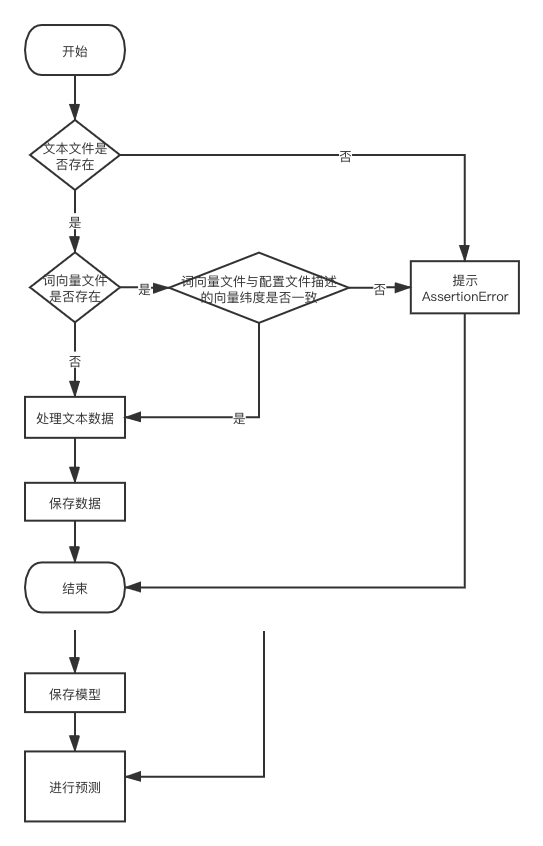
* [train, dev, test]\_record\_file：特征文件
* [train, dev]\_eval\_file：用于评价的文件
* [word, char]2idx\_file：记录词序号的文件
* [word, char]\_emb\_file：json格式词向量文件
* [word, char]\_mat：保存词向量矩阵的文件
* model：神经网络模型
* 训练时屏幕输出每轮损失情况
* 预测后文件输出预测结果

### 接口介绍

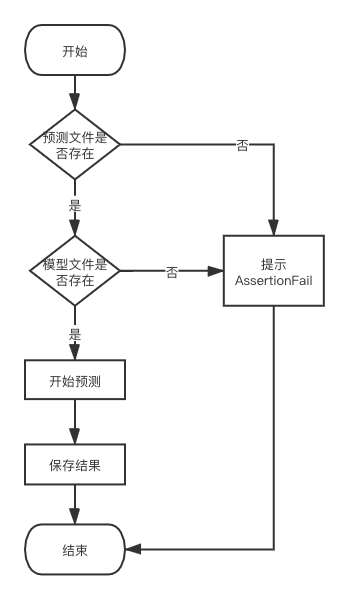
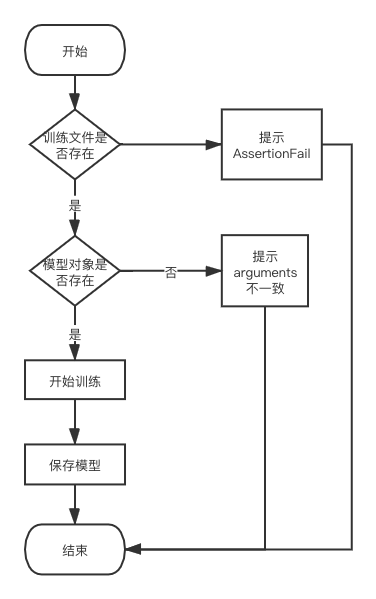
包括处理文本数据，构建模型，训练，预测4个接口，其输入和输出定义相同：

* process\_train\_file(config)
* process\_dev\_file(config)
* process\_predict\_file(config)
* QAModel(config): return model
* QATrain(config, model, device)
* QAPredict(config, model, device)

### 程序逻辑



|  |  |
| --- | --- |
| 图3.4.1 数据处理流程图 | 图3.4.2 模型构建流程图 |



|  |  |
| --- | --- |
| 图3.4.3 模型训练流程图 | 图3.4.4 模型预测流程图 |

表3.4 调用阅读理解模块用例表

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 调用阅读理解模型 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的阅读理解模型在指定数据集上训练网络，使用训练好的网络回答指定数据集的问题。 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | 工具包用户（自然语言处理领域研究人员） | | |
| Secondary Actor | None | | |
| Dependency | None | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户使用数据处理工具处理待训练文本 | |
| 3 | 用户使用模型构建工具 | |
| 4 | 用户使用训练工具训练模型 | |
| 5 | 用户使用数据处理工具处理待预测文本 | |
| 6 | 用户使用预测工具 | |
| Postcondition | | 整体过程不报告错误信息，运行结束退出 |
| Specific  Alternative Flow | **RFS** 1, 2, 3, 4,5,6 | | |
| 1 | 系统提示运行时出错位置 | |
| Postcondition | | 中断退出 |

### 测试要点

* 导入工具包是否会抛出异常
* 输入不符合规范时是否会抛出异常
* 处理过程中设备异常是否会抛出异常
* 符合规范且设备正常时是否能正常运行
* 不同设备、操作系统上运行是否一致

## 主动学习模块设计说明

### 功能描述

主动学习模块是为主动学习研究人员提供多种基础的采样策略，方便研究人员进行对比实验，并快速针对某个问题试验多种主动学习策略。用户可以将所有的数据提供给采样器，在模型训练过程中，采样器根据对应的采样策略不断选择样本进行标注，并将标注样本返回给用户，用户利用标注数据对模型进行训练。

### 输入项目

* model：torch.nn.module类型的对象，用户利用当前已标注数据训练的模型
* X：list类型的对象，包含了目前所有已标注和未标注的样本数据
* Y：list类型的对象，包含目前已标注样本的所有标签信息
* seed：int类型的对象，用来随机采样的种子
* reward\_function：python中的函数对象，作为某些采样策略的奖励函数
* gamma：float类型的对象，表示组合权重信息
* mixture：dict类型的对象，混合采样器的配置信息
* labeled：list类型，已标注的样本数据对应的索引集合
* budget：int类型，表示标注预算
* N：int类型，表示每回合标注的样本数目
* metric：str类型，表示特征度量类型
* eval\_acc：float类型的对象，表示模型的准确度

### 输出项目

正常运行：主动学习过程中标注的样本和训练的模型/抛出异常

### 接口介绍

该模块包含9种主动学习采样策略，各个采样类的参数分别定义为如下：

* BanditDiscreteSampler (X, Y, seed)
* GraphDensitySampler (X, Y, seed)
* HierarchicalClusterAL (X, Y, seed)
* InformativeClusterDiverseSampler (X, Y, seed)
* kCenterGreedy (X, Y, seed)
* MarginAL (X, Y, seed)
* MixtureOfSamplers (X, Y, seed)
* RepresentativeClusterMeanSampling (X, Y, seed)
* UniformSampling (X, Y, seed)

每个主动学习策略类都定义了1个采样方法select\_batch\_，用户调用此采样方法对未标注数据进行采样。

对于GraphDensitySampler、kCenterGreedy、UniformSampling和HierarchicalClusterAL类，采样方法定义为如下：

* select\_batch\_ (N, labeled)

对于BanditDiscreteSampler类，采样方法定义为如下：

* select\_batch\_(N, labeled, eval\_acc, model)

对于其他采样类，采样方法定义为：

* select\_batch\_(N, labeled, model)

### 程序逻辑

下面是该部分的流程图，以及对流程图的介绍。

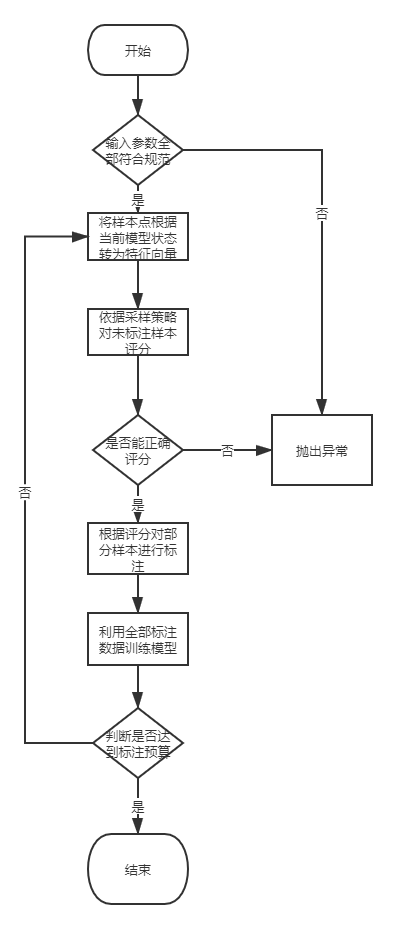


图3.5 主动学习流程图

如图3.5所示,首先判断参数是否合法，如果不合法则抛出异常，该模块结束。接下来将样本点根据当前模型状态转化为特征向量，并依据采样策略对未标注样本点评分，然后依据评分对部分样本进行标注。如果评分过程出现问题，则抛出异常。之后利用全部标注数据训练模型，如果达到标注预算，则结束主动学习过程，否则再次进行评分标注过程。表3.5为整个模块的RUCM说明。

表3.5 主动学习用例描述表

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 主动学习 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的主动学习采样策略采样未标注样本并进行标注，然后利用标注样本对模型进行训练 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | NLP研究人员 | | |
| Secondary Actor | None | | |
| Dependency | None | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户载入模型 | |
| 3 | 用户载入数据 | |
| 4 | 用户处理数据为满足主动学习模块输入要求的格式 | |
| 5 | 用户选择并加载主动学习采样器 | |
| 6 | 用户调用主动学习采样器的采样算法 | |
| 7 | 用户利用标注数据对模型进行训练 | |
| 8 | 系统返回全部标注数据和训练的模型 | |
| Postcondition | | 系统不报告错误信息，正常运行 |
| Bounded  Alternative  Flow | **RFS** 2，3，4，5 | | |
| 1 | 函数调用失败 | |
| 2 | 系统报告命令不合法信息 | |
| 3 | ABORT | |
| Postcondition | | 程序中断，用户依据报告信息修改代码 |

### 测试要点

* 导入工具包是否会抛出异常
* 输入不符合规范时是否会抛出异常
* 主动学习过程中设备异常是否会抛出异常
* 符合规范且设备正常时是否能正常运行