

# AI算法说明

需要使用AI算法实现的步骤包括：

- 利用关键词找到最相关的论文
  - 参考之前发过的学长论文，对 SciBERT 模型进行微调
- 由论文等信息对专家进行排序
  - 自己设计规则，难度不大

## 微调 SciBERT 模型

- SciBERT
  - 一个基于科学文本数据集的预训练表示模型
- 微调思路
  - 将论文的关键词和标题成对输入，使其向量表示尽可能相似，采用对比学习方法
- 参考代码
  - 以下为 `model.py` 文件参考，训练和使用的代码请大家自行补充

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from transformers import AutoTokenizer, AutoModel

class ContrastiveSciBERT(nn.Module):

    def __init__(self, out_dim, tau, device='cpu'):
        """用于对比学习的SciBERT模型

        :param out_dim: int 输出特征维数
        :param tau: float 温度参数τ
        :param device: torch.device, optional 默认为CPU
        """
        super().__init__()
        self.tau = tau
        self.device = device
        self.tokenizer = AutoTokenizer.from_pretrained('allenai/scibert_scivocab_uncased')
        self.model = AutoModel.from_pretrained('allenai/scibert_scivocab_uncased').to(device)
        self.linear = nn.Linear(self.model.config.hidden_size, out_dim)

    def get_embeds(self, texts, max_length=64):
        """将文本编码为向量

        :param texts: List[str] 输入文本列表，长度为N
```

```

:param max_length: int, optional padding最大长度，默认为64
:return: tensor(N, d_out)
"""
    encoded = self.tokenizer(
        texts, padding='max_length', truncation=True, max_length=max_length,
return_tensors='pt'
    ).to(self.device)
    return self.linear(self.model(**encoded).pooler_output)

def calc_sim(self, texts_a, texts_b):
    """计算两组文本的相似度


    :param texts_a: List[str] 输入文本A列表，长度为N
    :param texts_b: List[str] 输入文本B列表，长度为N
    :return: tensor(N, N) 相似度矩阵,  $S[i, j] = \cos(a[i], b[j]) / \tau$ 
    """
    embeds_a = self.get_embeds(texts_a) # (N, d_out)
    embeds_b = self.get_embeds(texts_b) # (N, d_out)
    embeds_a = embeds_a / embeds_a.norm(dim=1, keepdim=True)
    embeds_b = embeds_b / embeds_b.norm(dim=1, keepdim=True)
    return embeds_a @ embeds_b.t() / self.tau

def forward(self, texts_a, texts_b):
    """计算两组文本的对比损失（直接返回损失）

    :param texts_a: List[str] 输入文本A列表，长度为N
    :param texts_b: List[str] 输入文本B列表，长度为N
    :return: tensor(N, N), float A对B的相似度矩阵，对比损失
    """
    # logits_ab等价于预测概率，对比损失等价于交叉熵损失
    logits_ab = self.calc_sim(texts_a, texts_b)
    logits_ba = logits_ab.t()
    labels = torch.arange(len(texts_a), device=self.device)
    loss_ab = F.cross_entropy(logits_ab, labels)
    loss_ba = F.cross_entropy(logits_ba, labels)
    return logits_ab, (loss_ab + loss_ba) / 2

```

## 数据

- 文件: `computer_paper.txt`
  - 从知兔计算机领域下抽出的约 110 万数据，数据量比较大，可能无法直接打开
- 数据格式
  - 一行为一个字典
  - 举例 

```
{  
  "id": '3438751784',  
  "title": "Automatic extraction of street trees' nonphotosynthetic components  
from MLS data",  
  "keywords": ['Nonphotosynthetic components', 'Stem', 'Individual tree', 'MLS',  
'Urban environment', 'Clustering']  
}
```