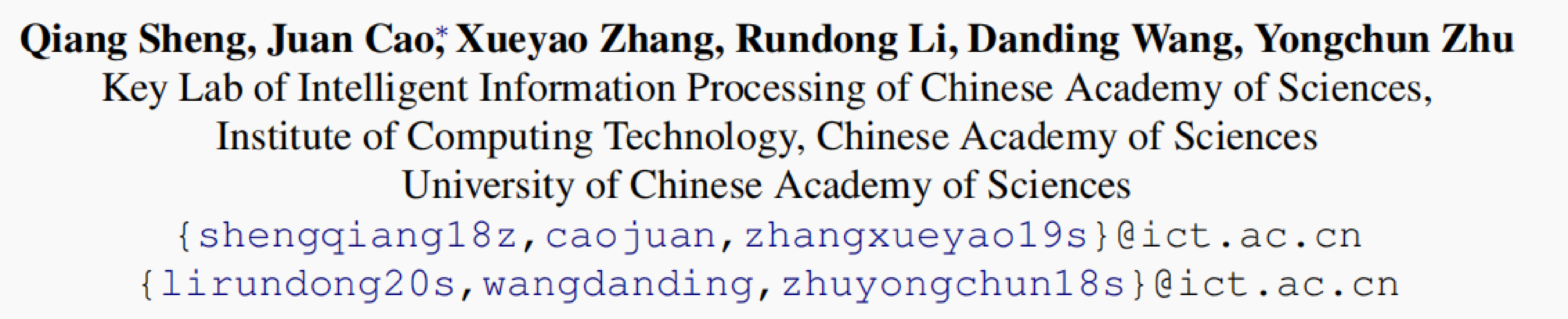
Zoom Out and Observe:
News Environment Perception for Fake News Detection



Qiang Sheng, Juan Cao*, Xueyao Zhang, Rundong Li, Danding Wang, Yongchun Zhu
Key Lab of Intelligent Information Processing of Chinese Academy of Sciences,
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
{shengqiang18z,caojuan,zhangxueyao19s}@ict.ac.cn
{lirundong20s,wangdanding,zhuyongchun18s}@ict.ac.cn

code:https://github.com/ICTMCG/News-Environment-Perception

CIKM 2022

Reported by Minqin Li

Method

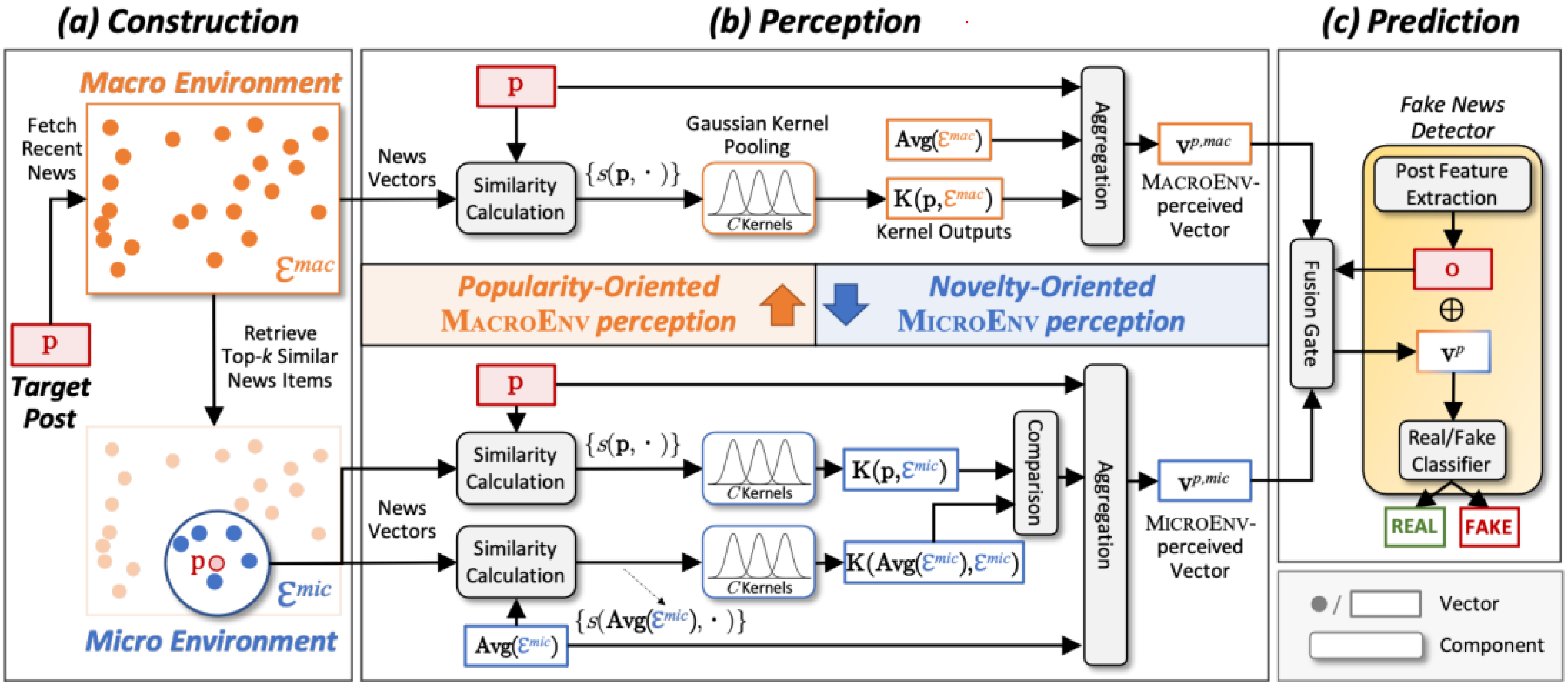


Figure 3: Architecture of the News Environment Perception Framework (NEP).

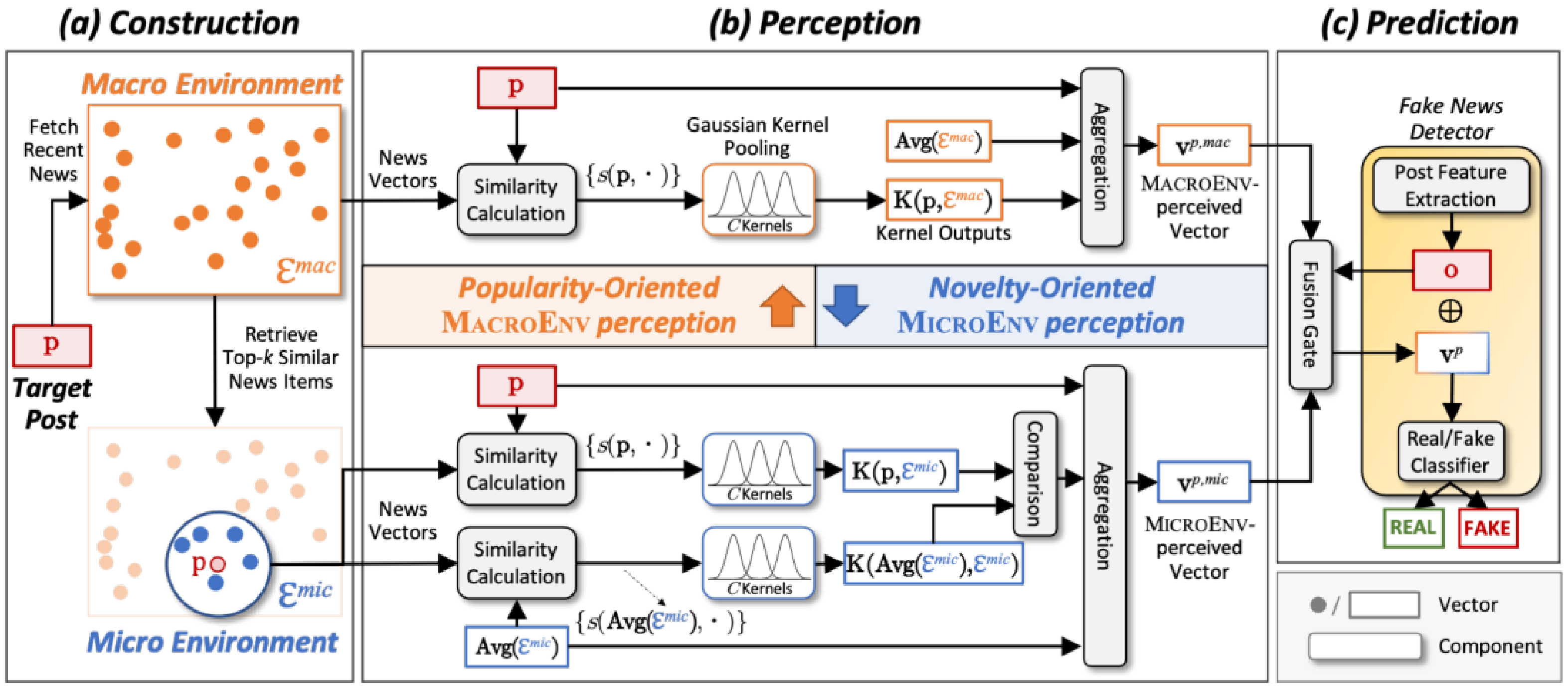
Method

'''Gaussian Kernel Pooling
input:The similarity list.
output:c-dimensional vector.(c is the number of Gaussian kernels)
'''

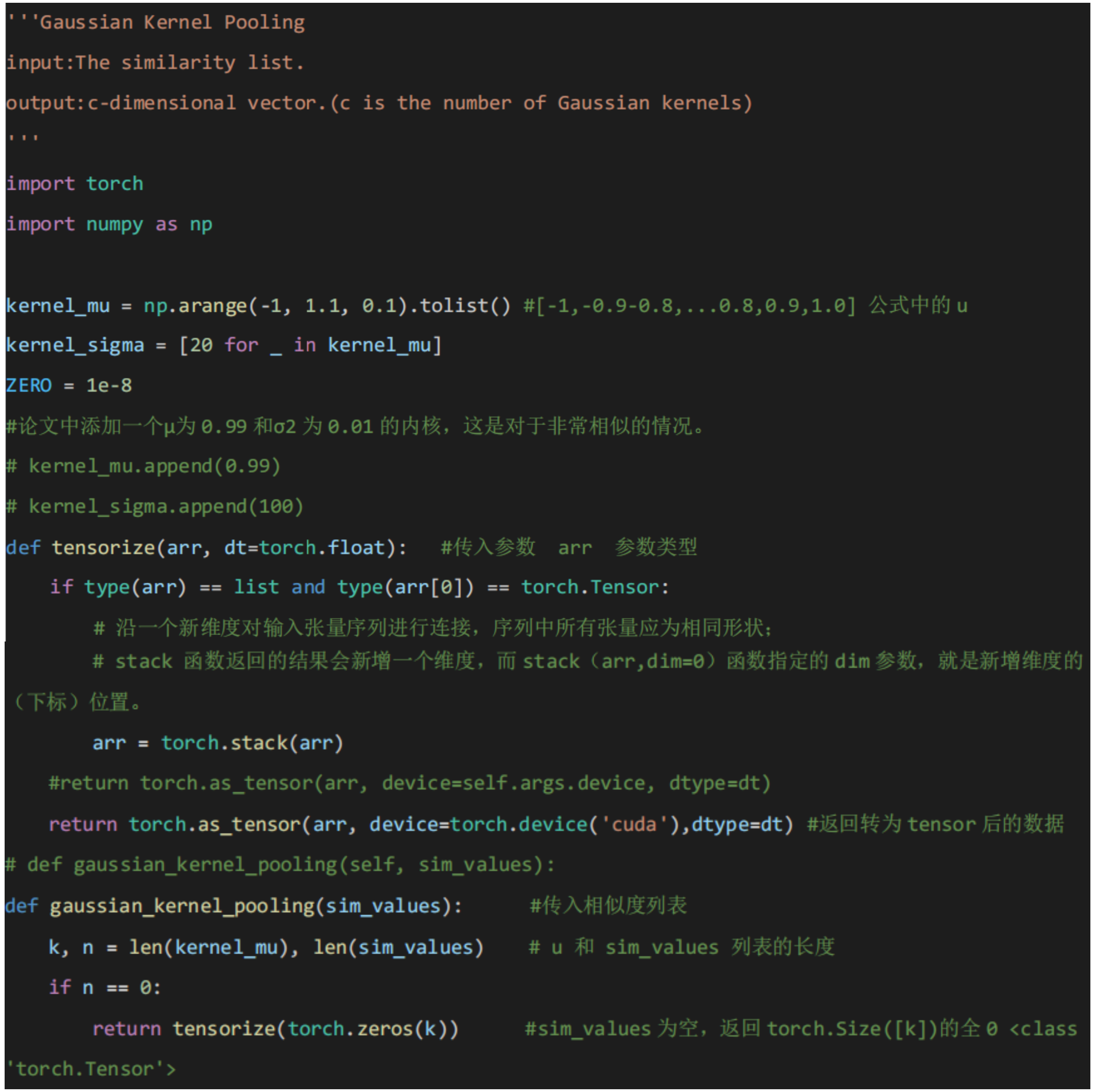
$$\mathbf{K}_k^i = \exp\left(-\frac{(s(\mathbf{p},\mathbf{e}_i)-\mu_k)^2}{2\sigma_k^2}\right)$$

$$\mathbf{K}_k(\mathbf{p},\mathcal{E}^{mac}) = \sum_{i=1}^{|\mathcal{E}^{mac}|}\mathbf{K}_k^i$$

$$\mathbf{K}(\mathbf{p},\mathcal{E}^{mac}) = \text{Norm}\left(\bigoplus_{k=1}^{C}\mathbf{K}_k(\mathbf{p},\mathcal{E}^{mac})\right)$$

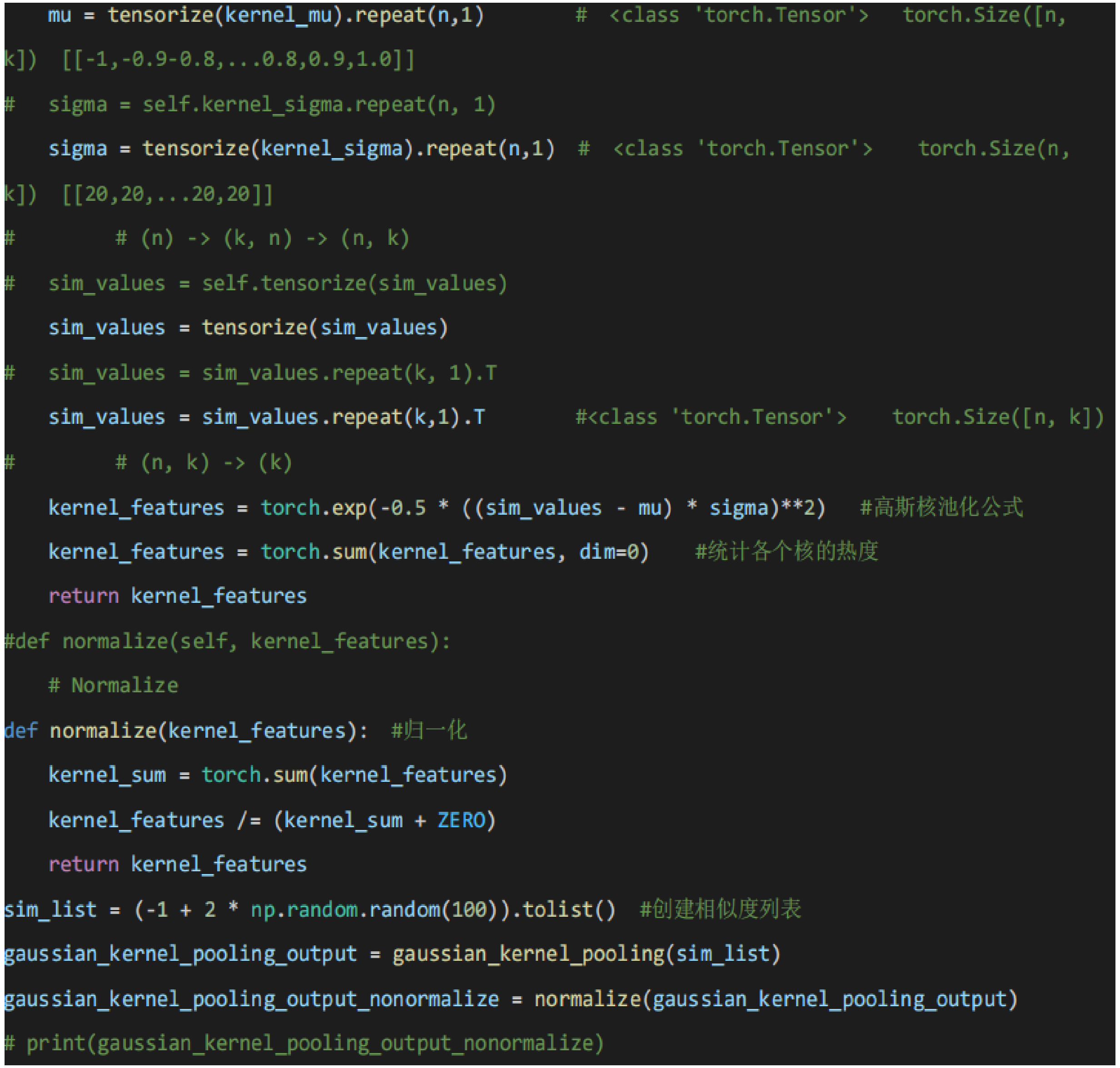


Method



```
'''Gaussian Kernel Pooling
input:The similarity list.
output:c-dimensional vector.(c is the number of Gaussian kernels)
'''
import torch
import numpy as np

kernel_mu = np.arange(-1, 1.1, 0.1).tolist() #[-1,-0.9-0.8,...0.8,0.9,1.0] 公式中的 u
kernel_sigma = [20 for _ in kernel_mu]
ZERO = 1e-8
#论文中添加一个μ为 0.99 和σ2 为 0.01 的内核，这是对于非常相似的情况。
# kernel_mu.append(0.99)
# kernel_sigma.append(100)
def tensorize(arr, dt=torch.float):   #传入参数  arr  参数类型
    if type(arr) == list and type(arr[0]) == torch.Tensor:
        # 沿一个新维度对输入张量序列进行连接，序列中所有张量应为相同形状；
        # stack 函数返回的结果会新增一个维度，而 stack（arr,dim=0）函数指定的 dim 参数，就是新增维度的
（下标）位置。
        arr = torch.stack(arr)
    #return torch.as_tensor(arr, device=self.args.device, dtype=dt)
    return torch.as_tensor(arr, device=torch.device('cuda'),dtype=dt) #返回转为 tensor 后的数据
# def gaussian_kernel_pooling(self, sim_values):
def gaussian_kernel_pooling(sim_values):      #传入相似度列表
    k, n = len(kernel_mu), len(sim_values)    # u 和 sim_values 列表的长度
    if n == 0:
        return tensorize(torch.zeros(k))      #sim_values 为空，返回 torch.Size([k])的全 0 <class
'torch.Tensor'>
```



```
    mu = tensorize(kernel_mu).repeat(n,1)        #  <class 'torch.Tensor'>   torch.Size([n,
k])  [[-1,-0.9-0.8,...0.8,0.9,1.0]]
#   sigma = self.kernel_sigma.repeat(n, 1)
    sigma = tensorize(kernel_sigma).repeat(n,1)  #  <class 'torch.Tensor'>    torch.Size(n,
k])  [[20,20,...20,20]]
#        # (n) -> (k, n) -> (n, k)
#   sim_values = self.tensorize(sim_values)
    sim_values = tensorize(sim_values)
#   sim_values = sim_values.repeat(k, 1).T
    sim_values = sim_values.repeat(k,1).T        #<class 'torch.Tensor'>    torch.Size([n, k])
#        # (n, k) -> (k)
    kernel_features = torch.exp(-0.5 * ((sim_values - mu) * sigma)**2)   #高斯核池化公式
    kernel_features = torch.sum(kernel_features, dim=0)    #统计各个核的热度
    return kernel_features
#def normalize(self, kernel_features):
    # Normalize
def normalize(kernel_features):  #归一化
    kernel_sum = torch.sum(kernel_features)
    kernel_features /= (kernel_sum + ZERO)
    return kernel_features
sim_list = (-1 + 2 * np.random.random(100)).tolist()  #创建相似度列表
gaussian_kernel_pooling_output = gaussian_kernel_pooling(sim_list)
gaussian_kernel_pooling_output_nonormalize = normalize(gaussian_kernel_pooling_output)
# print(gaussian_kernel_pooling_output_nonormalize)
```