第六周周报

本周工作

完成路径规划模块,核心代码如下

核心算法为 Dijkstra 最短路径堆优化算法

```python
import time
import heapq
from src.DataType import *

'''
@param start_degree 起点度
@param destination_degree 终点度
@param average_degree 平均度
@param 目标道路
@return 拥挤度
'''
def get_crowd(road:Road,start_degree,destination_degree,average_degree):
    current_time = time.localtime(time.time())
    if current_time.tm_hour >= 0 and current_time.tm_hour < 7:time_coefficient = 0.5
    elif current_time.tm_hour >= 7 and current_time.tm_hour < 10:time_coefficient = 2
    elif current_time.tm_hour >= 10 and current_time.tm_hour < 17:time_coefficient = 1.5
    elif current_time.tm_hour >= 17 and current_time.tm_hour < 20:time_coefficient = 2
    else:time_coefficient = 0.5
    busy_coefficient:float = (start_degree + destination_degree) / 2.0 / average_degree
    road.crowd = time_coefficient * busy_coefficient
    return road.crowd

'''
@param area 区域
@param road 道路
@param v 速度
@return 时间
'''
def get_time(area:Area,road:Road,v:float):
    check = False
    for value in area.road_group.values():
        for r in value:
            if r == road:check = True
    if not check:raise AttributeError('Road not in area')
    road.crowd = get_crowd(road,len(area.road_group[road.start]),
                           len(area.road_group[road.destination]),area.get_average_degree())
    return road.length/ v / (1 + road.crowd)

def get_shortest_road(area:Area,start:int,destination:int,mode:int = 0):
    heap = []
    prev = dict()
    dis = dict()
    dis[start] = 0
```

```
    heapq.heappush(heap,(0,start))
    for building_id in area.building_group.keys():
        if building_id != start:dis[building_id] = float('inf')
    while(len(heap)>0):
        current = heapq.heappop(heap)
        if current[1] == destination: return current[0],prev
        for e in area.road_group[current[1]]:
            if dis[e.start]+e.length < dis[e.destination]:
                dis[e.destination] = dis[e.start]+e.length
                heapq.heappush(heap,(dis[e.destination],e.destination))
                prev[e.destination] = current[1]
    return -1,prev
```

完成查找模块,核心代码如下

核心算法为 KMP 算法

```
from DataType import *
from Recommend import *
from Route_select import *

# KMP算法用于模式搜索
def KMPSearch(pat, txt):
    M = len(pat)
    N = len(txt)
    # 创建lps[]将会为模式持有最长的前缀后缀值
    lps = [0]*M
    j = 0 # pat[]的索引
    # 预处理模式（计算lps[]数组）
    computeLPSArray(pat, M, lps)
    i = 0 # txt[]的索引
    results = []
    while i < N:
        if pat[j] == txt[i]:
            i += 1
            j += 1
        if j == M:
            results.append(i-j)
            j = lps[j-1]
        # j个匹配后的不匹配
        elif i < N and pat[j] != txt[i]:
            # 不匹配lps[0..lps[j-1]]字符，
            # 它们将 anyway 匹配
            if j != 0:
                j = lps[j-1]
            else:
                i += 1
    return results

def computeLPSArray(pat, M, lps):
    len = 0 # 前一个最长前缀后缀的长度

    lps[0] = 0 # lps[0]始终为0
    i = 1
    while i < M:
        if pat[i] == pat[len]:
            len += 1
            lps[i] = len
            i += 1
        else:
```

```
            if len != 0:
                len = lps[len-1]
            else:
                lps[i] = 0
                i += 1

# 函数在类实例的文本属性中搜索关键字
def search_keyword_in_classes(keyword, instances):
    results = {}
    for instance in instances:
        total_occurrences = 0
        for attribute, value in vars(instance).items():
            if isinstance(value, str):
                found_positions = KMPSearch(keyword, value)
                if found_positions:
                    occurrences = len(found_positions)
                    total_occurrences += occurrences
                    if instance not in results:
                        results[instance] = []  # 确保此处已初始化
                    results[instance].append((attribute, occurrences))
            elif isinstance(value, list) or isinstance(value, set):
                for item in value:
                    if isinstance(item, str):
                        found_positions = KMPSearch(keyword, item)
                        if found_positions:
                            occurrences = len(found_positions)
                            total_occurrences += occurrences
                            if instance not in results:
                                results[instance] = []  # 确保此处已初始化
                            results[instance].append((attribute, occurrences))
        results[instance] = total_occurrences  # 使用总出现次数更新实例
    return results
comment2 = Comment("Bob", ["Not as expected"], 2, time.localtime(time.time()))
journal2 = journal("Weekend Getaway", 3.0, ["Short trip", "Okayish experience"], {comment2}, tir
keyword_results = search_keyword_in_classes("Not", [comment2, journal2])
print(keyword_results)
```

下周安排

完善推荐模块和日记模块