part2

October 8, 2024

[20]:
```python
class Agent:
    # Mapping of social group descriptions to numeric values
    social_group_mapping = {'small': 3, 'medium': 6, 'large': 9}

    def __init__(self, name, function, employment_time, friendliness, openness,␣
 ↪neuroticism,
                 conscientiousness, agreeableness, extraversion, social_group,␣
 ↪hobby,
                 experiences, reputation_score):
        self.name = name
        self.function = function
        self.employment_time = employment_time
        self.friendliness = friendliness
        self.openness = openness
        self.neuroticism = neuroticism
        self.conscientiousness = conscientiousness
        self.agreeableness = agreeableness
        self.extraversion = extraversion
        # Convert social group from string to numeric value using the mapping
        self.social_group = self.social_group_mapping[social_group.lower()]
        self.hobby = hobby
        self.experiences = experiences
        self.reputation_score = reputation_score
```

[31]:
```python
def calculate_trust(agent, context):
    if context == 'social':
        weights = {
            'friendliness': 0.20,
            'openness': 0.15,
            'neuroticism': -0.10,
            'agreeableness': 0.20,
            'extraversion': 0.15,
            'reputation_score': 0.10,
            'experiences': {
                'direct': {'helpful': 0.10, 'creativity': 0.10},
                'communicated': {'helpful': 0.05, 'creativity': 0.05}
```

```
            }
        }
    else:  # 'work' context
        weights = {
            'employment_time': 0.20,
            'conscientiousness': 0.20,
            'reputation_score': 0.15,
            'experiences': {
                'direct': {'efficiency': 0.20, 'accuracy': 0.20, 'flexibility':␣
↪0.10},
                'communicated': {'efficiency': 0.10, 'accuracy': 0.10,␣
↪'flexibility': 0.05}
            }
        }

    score = 0
    # Handle non-experience attributes
    for attr, weight in weights.items():
        if attr in ['friendliness', 'openness', 'agreeableness',␣
↪'extraversion', 'conscientiousness', 'reputation_score']:
            score += getattr(agent, attr) * weight
        elif attr == 'experiences':
            # Handle experiences separately
            for exp_type, exp_dict in agent.experiences.items():
                if exp_type in weights['experiences']:
                    for exp_attr, exp_weight in␣
↪weights['experiences'][exp_type].items():
                        if exp_attr in exp_dict:
                            score += exp_dict[exp_attr] * exp_weight

    return score

def select_agent(agents, context, function=None):
    filtered_agents = agents
    if function:
        # Filter agents by the specified function (role)
        filtered_agents = [agent for agent in agents if agent.function ==␣
↪function]

    best_agent = None
    highest_score = -1
    for agent in filtered_agents:
        trust_score = calculate_trust(agent, context)
        if trust_score > highest_score:
            highest_score = trust_score
            best_agent = agent
```

```
    return best_agent
```

[32]:
```
agents = [
    Agent(
        name="Agent 1",
        function="manager",
        employment_time=10,
        friendliness=8,
        openness=8,
        neuroticism=7,
        conscientiousness=6,
        agreeableness=9,
        extraversion=9,
        social_group="medium",
        hobby="marathons",
        experiences={
            'direct': {'efficiency': 6, 'accuracy': 8, 'flexibility': 2,␣
 ↪'creativity': 5, 'helpful': 8},
            'communicated': {'efficiency': 4, 'flexibility': 9, 'accuracy': 1,␣
 ↪'creativity': 3, 'helpful': 8}
        },
        reputation_score=5
    ),

    Agent(
        name="Agent 2",
        function="loader",
        employment_time=1,
        friendliness=4,
        openness=4,
        neuroticism=2,
        conscientiousness=6,
        agreeableness=9,
        extraversion=9,
        social_group="small",
        hobby="dance",
        experiences={
            'direct': {'efficiency': 5, 'accuracy': 8, 'flexibility': 3,␣
 ↪'creativity': 5, 'helpful': 4},
            'communicated': {'efficiency': 6, 'flexibility': 1, 'accuracy': 9,␣
 ↪'creativity': 3, 'helpful': 6}
        },
        reputation_score=7
    ),
    Agent(
        name="Agent 3",
        function="associate",
```

```
        employment_time=4,
        friendliness=8,
        openness=4,
        neuroticism=7,
        conscientiousness=9,
        agreeableness=6,
        extraversion=8,
        social_group="large",
        hobby="novelist",
        experiences={
            'direct': {'efficiency': 8, 'accuracy': 8, 'flexibility': 8,␣
↪'creativity': 9, 'helpful': 8},
            'communicated': {'efficiency': 9, 'flexibility': 8, 'accuracy': 6,␣
↪'creativity': 8, 'helpful': 9}
        },
        reputation_score=8
    ),
    Agent(
        name="Agent 4",
        function="forklift driver",
        employment_time=15,
        friendliness=6,
        openness=4,
        neuroticism=3,
        conscientiousness=9,
        agreeableness=3,
        extraversion=8,
        social_group="medium",
        hobby="photographer",
        experiences={
            'direct': {'efficiency': 8, 'accuracy': 5, 'flexibility': 6,␣
↪'creativity': 6, 'helpful': 5},
            'communicated': {'efficiency': 6, 'flexibility': 7, 'accuracy': 6,␣
↪'creativity': 4, 'helpful': 6}
        },
        reputation_score=7
    ),
    Agent(
        name="Agent 5",
        function="machine operator",
        employment_time=4,
        friendliness=8,
        openness=4,
        neuroticism=7,
        conscientiousness=9,
        agreeableness=6,
        extraversion=8,
```

```
        social_group="large",
        hobby="gym",
        experiences={
            'direct': {'efficiency': 5, 'accuracy': 8, 'flexibility': 5,␣
↪'creativity': 5, 'helpful': 3},
            'communicated': {'efficiency': 6, 'flexibility': 6, 'accuracy': 9,␣
↪'creativity': 6, 'helpful': 5}
        },
        reputation_score=9
    ),
    Agent(
        name="Agent 6",
        function="loader",
        employment_time=4,
        friendliness=2,
        openness=6,
        neuroticism=5,
        conscientiousness=5,
        agreeableness=6,
        extraversion=3,
        social_group="small",
        hobby="fishing",
        experiences={
            'direct': {'efficiency': 5, 'accuracy': 5, 'flexibility': 3,␣
↪'creativity': 4, 'helpful': 6},
            'communicated': {'efficiency': 3, 'flexibility': 1, 'accuracy': 3,␣
↪'creativity': 1, 'helpful': 3}
        },
        reputation_score=2
    ),
    Agent(
        name="Agent 7",
        function="loader",
        employment_time=10,
        friendliness=7,
        openness=7,
        neuroticism=7,
        conscientiousness=3,
        agreeableness=3,
        extraversion=8,
        social_group="large",
        hobby="writing",
        experiences={
            'direct': {'efficiency': 7, 'accuracy': 6, 'flexibility': 3,␣
↪'creativity': 7, 'helpful': 8},
            'communicated': {'efficiency': 6, 'flexibility': 6, 'accuracy': 9,␣
↪'creativity': 8, 'helpful': 7}
```

```
        },
        reputation_score=6
    )
]
```

[33]:
```
selected_agent_social = select_agent(agents, 'social')
selected_agent_work = select_agent(agents, 'work',function='loader')
print("Selected agent for social event:", selected_agent_social.name)
print("Selected agent for work task:", selected_agent_work.name)
```

```
Selected agent for social event: Agent 1
Selected agent for work task: Agent 2
```

1 Step 1

For the first context, since it's a present-buying task, the person needs to be observant and understand/know what people like, and also honest and reasonable with spending money. Based on this, we think hobby and function of the person doesn't does not matter in the context, but rather their relationship with other colleagues. Features like employment time might tell if they know the company or colleagues long enough. Friendliness and extraversion might increase chances of gathering information of which present is suitable, which could also be contributed by the size of his/her social group. Openness and ensures clarity and honesty when it comes to decision-making (of buying gift/spending company's budget). Conscientiousness is important to evaluate if one could finish the task properly. You want a person with less neuroticism. For the second context, you need someone who's reliable and experienced with loading, preferably a loader. In the social value, employment time and conscientiousness are the most valuable criteria. The task also needed to be done fastly, so efficiency, accuracy, flexibility and is highly valued.

2 Step 2

Can be checked in the code block

3 Step 3

We use the direct and communicated features to identify the reliability of an employee, we tend to take the direct (observed) skill more valuable than the communicated (he/she said so).

4 Step 4

Yes! Just as what as I expected. Maybe but as there's no real answer, I'm still ok with the infomation chosen. If more types are involved in the caculation or the weights are changed, it might change the result of the question.

5 Step 5

We can either reflect on the features again, and change the weight, or give the person a minus.value when he/she performed task poorly.