Materials Project API - 高级端点详解

11. 磁性性质 (Magnetism)

11.1 端点信息

URL: `/materials/magnetism/`
用途: 获取磁序、磁化强度等磁性性质
数据覆盖: ~50,000 材料
重要性: ★★★★ 对磁性材料、自旋电子学重要

11.2 可用字段

基础磁性信息:

```
"material_id"               # 材料ID
"ordering"                  # 磁序类型
"total_magnetization"       # 总磁化强度 (μB)
"total_magnetization_normalized_vol"     # 体积归一化磁化强度 (μB/Å³)
"total_magnetization_normalized_formula_units"  # 每分子式磁化强度 (μB/f.u.)
```

磁性位点信息:

```
"num_magnetic_sites"        # 磁性位点总数
"num_unique_magnetic_sites"  # 唯一磁性位点数
"types_of_magnetic_species"  # 磁性元素列表
```

磁序类型:

- `FM` - 铁磁性 (Ferromagnetic)
- `AFM` - 反铁磁性 (Antiferromagnetic)
- `FiM` - 亚铁磁性 (Ferrimagnetic)
- `NM` - 非磁性 (Non-magnetic)
- `Unknown` - 未确定

11.3 查询示例

示例1: 铁磁材料搜索

```
url = f"{BASE_URL}/materials/magnetism/"
params = {
    "ordering": "FM",
    "_fields": "material_id,ordering,total_magnetization",
    "_sort_fields": "total_magnetization",
```

```
    "_ascending": False,
    "_limit": 20
}

response = requests.get(url, headers=headers, params=params)
data = response.json()["data"]

print("高磁化强度的铁磁材料:")
print("-" * 60)
for mat in data:
    print(f"{mat['material_id']:<12} "
          f"磁化强度: {mat['total_magnetization']:.2f} μB")
```

示例2: 特定元素的磁性分析

```
# 搜索含Fe的磁性材料
url_summary = f"{BASE_URL}/materials/summary/"
params_summary = {
    "elements": "Fe",
    "is_stable": True,
    "_fields": "material_id,formula_pretty",
    "_limit": 50
}

materials = requests.get(url_summary, headers=headers,
                         params=params_summary).json()["data"]

# 获取磁性信息
url_mag = f"{BASE_URL}/materials/magnetism/"
for mat in materials[:10]:
    params_mag = {
        "material_ids": mat["material_id"],
        "_fields": "material_id,ordering,total_magnetization"
    }

    mag_response = requests.get(url_mag, headers=headers, params=params_mag)
    if mag_response.status_code == 200:
        mag_data = mag_response.json()["data"]
        if mag_data:
            mag_info = mag_data[0]
            print(f"{mat['formula_pretty']:<15} "
                  f"{mag_info.get('ordering', 'N/A'):<5} "
                  f"{mag_info.get('total_magnetization', 0):.2f} μB")

    time.sleep(0.3)
```

11.4 应用：永磁材料筛选

```python
def screen_permanent_magnet_materials():
    """
    筛选潜在的永磁材料
    标准:
    - 铁磁性 (FM)
    - 高磁化强度
    - 稳定相
    - 含有稀土或过渡金属
    """
    # 第一步: 搜索铁磁材料
    url_mag = f"{BASE_URL}/materials/magnetism/"
    params = {
        "ordering": "FM",
        "_fields": "material_id,total_magnetization",
        "_sort_fields": "total_magnetization",
        "_ascending": False,
        "_limit": 100
    }

    mag_response = requests.get(url_mag, headers=headers, params=params)
    fm_materials = mag_response.json()["data"]

    # 第二步: 获取详细信息并筛选
    results = []
    for mat in fm_materials:
        material_id = mat["material_id"]

        # 获取成分信息
        url_summary = f"{BASE_URL}/materials/summary/"
        summary_params = {
            "material_ids": material_id,
            "_fields": "material_id,formula_pretty,elements,is_stable"
        }

        summary_response = requests.get(url_summary, headers=headers,
                                        params=summary_params)
        summary_data = summary_response.json()["data"][0]

        # 检查是否含有磁性元素
        magnetic_elements = {'Fe', 'Co', 'Ni', 'Nd', 'Sm', 'Pr', 'Dy', 'Tb'}
        elements = set(summary_data.get("elements", []))

        if magnetic_elements & elements and summary_data.get("is_stable"):
            results.append({
                "material_id": material_id,
                "formula": summary_data["formula_pretty"],
                "magnetization": mat["total_magnetization"],
                "magnetic_elements": list(magnetic_elements & elements)
            })

        time.sleep(0.3)

        if len(results) >= 20:
```

```
            break

    # 输出结果
    print(f"找到 {len(results)} 个候选永磁材料\n")
    print("候选材料列表:")
    print("-" * 80)

    for i, mat in enumerate(results, 1):
        print(f"{i}. {mat['formula']:<20} (ID: {mat['material_id']:<12})")
        print(f"   磁化强度: {mat['magnetization']:.2f} μB, "
              f"磁性元素: {', '.join(mat['magnetic_elements'])}")

    return results
```

---

12. 表面性质 (Surface Properties)

12.1 端点信息

URL: `/materials/surface_properties/`
用途: 获取表面能、功函数等表面性质
数据覆盖: ~100,000 材料
重要性: ★★★★★ 对催化剂、电极材料非常重要

12.2 可用字段

表面能相关:

```
"material_id"              # 材料ID
"weighted_surface_energy"  # 加权表面能 (J/m²)
"weighted_work_function"   # 加权功函数 (eV)
"surface_anisotropy"       # 表面能各向异性
"shape_factor"             # Wulff形状因子
```

单个表面信息:

```
"surfaces"                 # 不同晶面的详细信息列表
  - "miller_index"         # 米勒指数 (hkl)
  - "surface_energy"       # 该晶面的表面能
  - "work_function"        # 该晶面的功函数
  - "is_reconstructed"     # 是否重构
```

Wulff形状:

```
"wulff_shape"              # Wulff晶体形状数据
"shape_factor"             # 形状因子 (表面积/体积比)
```

12.3 查询示例

示例1: 获取表面能和功函数

```
url = f"{BASE_URL}/materials/surface_properties/"
params = {
    "material_ids": "mp-30",  # Cu
    "_fields":
"material_id,weighted_surface_energy,weighted_work_function,surfaces"
}

response = requests.get(url, headers=headers, params=params)
if response.status_code == 200:
    data = response.json()["data"][0]

    print(f"材料: {data['material_id']}")
    print(f"加权表面能: {data['weighted_surface_energy']:.3f} J/m²")
    print(f"加权功函数: {data['weighted_work_function']:.3f} eV")

    # 显示不同晶面的信息
    surfaces = data.get("surfaces", [])
    if surfaces:
        print("\n不同晶面的性质:")
        print("-" * 60)
        for surf in surfaces[:5]:  # 只显示前5个
            miller = surf.get("miller_index", [])
            miller_str = f"({miller[0]}{miller[1]}{miller[2]})"
            print(f"{miller_str:<8} "
                  f"表面能: {surf.get('surface_energy', 'N/A'):.3f} J/m², "
                  f"功函数: {surf.get('work_function', 'N/A'):.3f} eV")
```

示例2: 低表面能材料搜索

```
# 搜索低表面能的稳定氧化物
url_summary = f"{BASE_URL}/materials/summary/"
summary_params = {
    "elements": "O",
    "is_stable": True,
    "nelements": 2,
    "_fields": "material_id,formula_pretty",
    "_limit": 50
}

materials = requests.get(url_summary, headers=headers,
                         params=summary_params).json()["data"]

# 获取表面性质
url_surf = f"{BASE_URL}/materials/surface_properties/"
```

```
results = []

for mat in materials:
    surf_params = {
        "material_ids": mat["material_id"],
        "_fields": "material_id,weighted_surface_energy"
    }

    surf_response = requests.get(url_surf, headers=headers, params=surf_params)
    if surf_response.status_code == 200:
        surf_data = surf_response.json()["data"]
        if surf_data:
            results.append({
                "formula": mat["formula_pretty"],
                "material_id": mat["material_id"],
                "surface_energy": surf_data[0]["weighted_surface_energy"]
            })

    time.sleep(0.3)

# 排序并输出
results.sort(key=lambda x: x["surface_energy"])

print("低表面能氧化物材料:")
print("-" * 60)
for i, mat in enumerate(results[:10], 1):
    print(f"{i}. {mat['formula']:<15} "
          f"表面能: {mat['surface_energy']:.3f} J/m²")
```

12.4 应用：催化剂活性预测

```
def predict_catalytic_activity():
    """
    基于功函数和表面能预测催化活性

    理论基础:
    - 适中的表面能（易于吸附/脱附）
    - 适中的功函数（电子转移）
    - d带中心理论
    """
    # 目标：寻找过渡金属氧化物催化剂
    transition_metals = ['Ti', 'V', 'Cr', 'Mn', 'Fe', 'Co', 'Ni', 'Cu']

    results = []

    for metal in transition_metals:
        # 搜索氧化物
        url_summary = f"{BASE_URL}/materials/summary/"
        params = {
            "elements": f"{metal},O",
            "nelements": 2,
```

```
            "is_stable": True,
            "_fields": "material_id,formula_pretty,band_gap",
            "_limit": 5
        }

        materials = requests.get(url_summary, headers=headers,
                                 params=params).json().get("data", [])

        for mat in materials:
            # 获取表面性质
            url_surf = f"{BASE_URL}/materials/surface_properties/"
            surf_params = {
                "material_ids": mat["material_id"],
                "_fields":
"material_id,weighted_surface_energy,weighted_work_function"
            }

            surf_response = requests.get(url_surf, headers=headers,
                                         params=surf_params)

            if surf_response.status_code == 200:
                surf_data = surf_response.json().get("data", [])
                if surf_data:
                    surf_energy = surf_data[0].get("weighted_surface_energy")
                    work_func = surf_data[0].get("weighted_work_function")

                    # 简单评分：表面能和功函数接近理想值
                    # 理想表面能：0.5-2.0 J/m²
                    # 理想功函数：4.0-5.5 eV
                    if surf_energy and work_func:
                        surf_score = 1.0 / (1 + abs(surf_energy - 1.0))
                        wf_score = 1.0 / (1 + abs(work_func - 4.75))
                        total_score = (surf_score + wf_score) / 2

                        results.append({
                            "formula": mat["formula_pretty"],
                            "material_id": mat["material_id"],
                            "band_gap": mat.get("band_gap", "N/A"),
                            "surface_energy": surf_energy,
                            "work_function": work_func,
                            "score": total_score
                        })

            time.sleep(0.3)

    # 排序输出
    results.sort(key=lambda x: x["score"], reverse=True)

    print("潜在催化剂材料排名:")
    print("-" * 90)
    for i, mat in enumerate(results[:15], 1):
        print(f"{i}. {mat['formula']:<15} (ID: {mat['material_id']:<12})")
        print(f"   带隙: {mat['band_gap']} eV, "
              f"表面能: {mat['surface_energy']:.3f} J/m², "
```

```
                f"功函数: {mat['work_function']:.3f} eV")
          print(f"   评分: {mat['score']:.3f}\n")

    return results
```

---

13. 压电性质 (Piezoelectric)

13.1 端点信息

URL: /materials/piezoelectric/
用途: 获取压电张量和压电系数
数据覆盖: ~900 材料（非中心对称材料）
重要性: ★★★ 对传感器、换能器重要

13.2 可用字段

```
"material_id"              # 材料ID
"piezoelectric_tensor"     # 压电张量（3×6矩阵，C/m²）
"e_ij_max"                 # 最大压电系数（C/m²）
"max_direction"            # 最大响应方向
"strain_type"              # 应变类型
"point_group"              # 点群对称性
```

13.3 查询示例

示例1: 高压电系数材料

```
url = f"{BASE_URL}/materials/piezoelectric/"
params = {
    "_fields": "material_id,e_ij_max",
    "_sort_fields": "e_ij_max",
    "_ascending": False,
    "_limit": 20
}

response = requests.get(url, headers=headers, params=params)
data = response.json()["data"]

print("高压电系数材料:")
print("-" * 60)
for i, mat in enumerate(data, 1):
    print(f"{i}. {mat['material_id']:<12} "
          f"d_max: {mat['e_ij_max']:.2f} C/m²")
```

---

14. 声子性质 (Phonon)

14.1 端点信息

URL: `/materials/phonon/`
用途: 获取声子能带、声子态密度
数据覆盖: ~1,500 材料
重要性: ⭐⭐⭐ 对热电材料、超导材料重要

14.2 可用字段

```
"material_id"              # 材料ID
"has_imaginary_modes"      # 是否有虚频（不稳定）
"phonon_bandstructure"     # 声子能带结构
"phonon_dos"               # 声子态密度
"thermal_displacement_data"  # 热位移数据
"last_updated"             # 最后更新时间
```

14.3 查询示例

示例1: 检查动力学稳定性

```
url = f"{BASE_URL}/materials/phonon/"
params = {
    "material_ids": "mp-149",
    "_fields": "material_id,has_imaginary_modes"
}

response = requests.get(url, headers=headers, params=params)
if response.status_code == 200:
    data = response.json()["data"]
    if data:
        has_imaginary = data[0].get("has_imaginary_modes")
        print(f"动力学稳定: {'否' if has_imaginary else '是'}")
    else:
        print("无声子数据")
```

---

15. 热力学性质 (Thermodynamics)

15.1 端点信息

URL: `/materials/thermo/`
用途: 获取形成能、分解产物等热力学信息
数据覆盖: ~140,000 材料

15.2 可用字段

```
"material_id"              # 材料ID
"formation_energy_per_atom"  # 形成能 (eV/atom)
"energy_above_hull"        # 能量高于凸包 (eV/atom)
"is_stable"                # 是否稳定
"equilibrium_reaction_energy_per_atom"  # 平衡反应能
"decomposes_to"            # 分解产物
"decomposition_enthalpy"   # 分解焓
"energy_per_atom"          # 总能量 (eV/atom)
"energy_uncertainties"     # 能量不确定性
```

15.3 应用：合成路径分析

```
def analyze_synthesis_pathway(target_formula):
    """
    分析材料的合成路径和稳定性
    """
    # 1. 获取目标材料信息
    url_thermo = f"{BASE_URL}/materials/thermo/"
    params = {
        "formula": target_formula,
        "_fields": "material_id,formation_energy_per_atom,is_stable," + \
                   "energy_above_hull,decomposes_to",
        "_limit": 5
    }

    response = requests.get(url_thermo, headers=headers, params=params)
    materials = response.json()["data"]

    if not materials:
        print(f"未找到 {target_formula} 的数据")
        return

    print(f"'{target_formula}' 的热力学分析:")
    print("=" * 70)

    for mat in materials:
        print(f"\nMaterial ID: {mat['material_id']}")
        print(f"形成能: {mat.get('formation_energy_per_atom', 'N/A'):.3f}
eV/atom")
        print(f"能量高于凸包: {mat.get('energy_above_hull', 'N/A'):.3f} eV/atom")
        print(f"稳定性: {'稳定' if mat.get('is_stable') else '亚稳态'}")

        # 分解产物
        decomposes_to = mat.get('decomposes_to', [])
        if decomposes_to:
            print("分解产物:")
            for product in decomposes_to:
                print(f"  - {product}")
        else:
            print("不分解 (热力学稳定相) ")
```

```
        print("-" * 70)
```

继续创建实践应用章节...