Materials Project API - 快速参考手册

目录

快速开始

最小工作示例

```
import requests

API_KEY = "你的API密匙"
BASE_URL = "https://api.materialsproject.org"

headers = {"X-API-KEY": API_KEY}

# 查询材料
url = f"{BASE_URL}/materials/summary/"
params = {"formula": "TiO2", "_fields": "material_id,formula_pretty,band_gap"}

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

print(data["data"][0])
```

常用端点速查

1. Summary - 基础信息

```
url = f"{BASE_URL}/materials/summary/"
params = {
    # 搜索条件
    "formula": "TiO2",              # 化学式
    "elements": "Ti,O",             # 元素列表
    "nelements": 2,                 # 元素数量
    "chemsys": "Ti-O",              # 化学系统
    "is_stable": True,              # 只要稳定相
    "is_metal": False,              # 非金属
    "band_gap_min": 1.0,            # 最小带隙
    "band_gap_max": 3.0,            # 最大带隙
    "crystal_system": "Tetragonal", # 晶系
```

```
    # 返回字段（选择性）
    "_fields": "material_id,formula_pretty,band_gap,density,is_stable",

    # 排序和分页
    "_sort_fields": "band_gap",      # 排序字段
    "_ascending": False,             # 降序
    "_limit": 10,                    # 每页数量
    "_skip": 0                       # 跳过数量
}
```

常用返回字段:

```
material_id              # 材料ID
formula_pretty           # 化学式
elements                 # 元素列表
nelements                # 元素数
band_gap                 # 带隙（eV）
is_stable                # 是否稳定
is_metal                 # 是否金属
energy_above_hull        # 能量高于凸包（eV/atom）
formation_energy_per_atom  # 形成能（eV/atom）
density                  # 密度（g/cm³）
nsites                   # 原子数
volume                   # 体积（ų）
efermi                   # 费米能级（eV）
spacegroup_symbol        # 空间群符号
spacegroup_number        # 空间群编号
crystal_system           # 晶系
```

---

2. Electronic Structure - 电子结构

```
url = f"{BASE_URL}/materials/electronic_structure/"
params = {
    "material_ids": "mp-149",
    "_fields": "band_gap,cbm,vbm,efermi,is_gap_direct,dos"
}
```

返回字段:

```
band_gap        # 带隙（eV）
cbm             # 导带底（eV）
vbm             # 价带顶（eV）
efermi          # 费米能级（eV）
is_gap_direct   # 是否直接带隙
is_metal        # 是否金属
```

```
dos              # 态密度数据
bandstructure    # 能带结构数据
```

---

3. Dielectric - 光学性质

```
url = f"{BASE_URL}/materials/dielectric/"
params = {
    "material_ids": "mp-149",
    "_fields": "n,eps_electronic,eps_total"
}
```

返回字段:

```
n                    # 折射率
eps_electronic       # 电子介电常数
eps_total            # 总介电常数（包含离子贡献）
e_electronic         # 电子介电张量
e_total              # 总介电张量
```

---

4. Elasticity - 力学性质

```
url = f"{BASE_URL}/materials/elasticity/"
params = {
    "material_ids": "mp-149",
    "_fields": "elastic_tensor,k_vrh,g_vrh"
}
```

返回字段:

```
elastic_tensor       # 弹性张量 (GPa)
compliance_tensor    # 柔度张量 (1/GPa)
k_voigt              # Voigt体积模量 (GPa)
k_reuss              # Reuss体积模量 (GPa)
k_vrh                # VRH体积模量 (GPa)
g_voigt              # Voigt剪切模量 (GPa)
g_reuss              # Reuss剪切模量 (GPa)
g_vrh                # VRH剪切模量 (GPa)
universal_anisotropy  # 各向异性指数
homogeneous_poisson   # 泊松比
```

---

5. Magnetism - 磁性性质

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

返回字段:

```
ordering                # 磁序 (FM/AFM/FiM/NM)
total_magnetization     # 总磁化强度 (μB)
num_magnetic_sites      # 磁性位点数
types_of_magnetic_species  # 磁性元素列表
```

---

6. Surface Properties - 表面性质

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

返回字段:

```
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
surfaces                 # 各晶面信息
shape_factor            # Wulff形状因子
```

---

7. Thermodynamics - 热力学

```
url = f"{BASE_URL}/materials/thermo/"
params = {
    "formula": "LiFePO4",
    "_fields": "formation_energy_per_atom,energy_above_hull,is_stable"
}
```

返回字段:

```
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
decomposes_to              # 分解产物
decomposition_enthalpy     # 分解焓 (eV/atom)
```

代码片段库

材料搜索

1. 按化学式搜索

```python
def search_by_formula(formula):
    url = f"{BASE_URL}/materials/summary/"
    params = {
        "formula": formula,
        "_fields": "material_id,formula_pretty,band_gap,is_stable"
    }
    response = requests.get(url, headers=headers, params=params)
    return response.json()["data"]

# 使用
results = search_by_formula("TiO2")
```

2. 按元素组合搜索

```python
def search_by_elements(elements, max_nelements=None):
    url = f"{BASE_URL}/materials/summary/"
    params = {
        "elements": ",".join(elements),
        "_fields": "material_id,formula_pretty,nelements"
    }
    if max_nelements:
        params["nelements_max"] = max_nelements

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

# 使用
results = search_by_elements(["Ti", "O"], max_nelements=2)
```

3. 按带隙范围搜索

```
def search_by_bandgap(min_gap, max_gap):
    url = f"{BASE_URL}/materials/summary/"
    params = {
        "band_gap_min": min_gap,
        "band_gap_max": max_gap,
        "is_stable": True,
        "_fields": "material_id,formula_pretty,band_gap",
        "_sort_fields": "band_gap",
        "_limit": 50
    }
    response = requests.get(url, headers=headers, params=params)
    return response.json()["data"]

# 使用：搜索可见光吸收材料
results = search_by_bandgap(1.5, 3.0)
```

数据获取

4. 获取完整材料信息

```
def get_full_material_info(material_id):
    """获取材料的所有可用信息"""

    # Summary
    url_summary = f"{BASE_URL}/materials/summary/"
    summary_response = requests.get(url_summary, headers=headers,
                                    params={"material_ids": material_id})
    summary = summary_response.json()["data"][0] if summary_response.status_code
== 200 else {}

    # Electronic Structure
    url_elec = f"{BASE_URL}/materials/electronic_structure/"
    elec_response = requests.get(url_elec, headers=headers,
                                 params={"material_ids": material_id})
    elec = elec_response.json()["data"][0] if elec_response.status_code == 200
else {}

    # 合并结果
    full_info = {**summary, **elec}

    return full_info

# 使用
info = get_full_material_info("mp-149")
print(f"化学式: {info.get('formula_pretty')}")
print(f"带隙: {info.get('band_gap')} eV")
```

5. 批量获取材料数据

```python
def batch_get_materials(material_ids, batch_size=10):
    """批量获取材料信息"""
    url = f"{BASE_URL}/materials/summary/"
    all_data = []

    for i in range(0, len(material_ids), batch_size):
        batch = material_ids[i:i+batch_size]
        params = {
            "material_ids": ",".join(batch),
            "_fields": "material_id,formula_pretty,band_gap,density"
        }

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

        time.sleep(0.5)  # 速率限制

    return all_data

# 使用
ids = ["mp-149", "mp-30", "mp-1234"]
results = batch_get_materials(ids)
```

数据筛选

6. 半导体材料筛选

```python
def screen_semiconductors(min_gap=0.5, max_gap=3.5):
    """筛选半导体材料"""
    url = f"{BASE_URL}/materials/summary/"
    params = {
        "band_gap_min": min_gap,
        "band_gap_max": max_gap,
        "is_stable": True,
        "is_metal": False,
        "_fields": "material_id,formula_pretty,band_gap,nelements",
        "_sort_fields": "band_gap",
        "_limit": 100
    }

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

    # 进一步筛选：简单化合物优先
    simple_compounds = [m for m in materials if m.get("nelements", 5) <= 3]

    return simple_compounds
```

```
# 使用
semiconductors = screen_semiconductors()
for mat in semiconductors[:5]:
    print(f"{mat['formula_pretty']}: {mat['band_gap']} eV")
```

7. 催化剂材料筛选

```
def screen_catalysts():
    """筛选潜在催化剂材料"""
    # 第一步：搜索过渡金属氧化物
    url_summary = f"{BASE_URL}/materials/summary/"
    transition_metals = ['Ti', 'V', 'Cr', 'Mn', 'Fe', 'Co', 'Ni', 'Cu']

    candidates = []
    for metal in transition_metals:
        params = {
            "elements": f"{metal},O",
            "nelements": 2,
            "is_stable": True,
            "_fields": "material_id,formula_pretty,band_gap",
            "_limit": 5
        }

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

        # 筛选小带隙材料（有利于电荷转移）
        for mat in materials:
            if 0 < mat.get("band_gap", 10) < 3.0:
                candidates.append(mat)

        time.sleep(0.3)

    return candidates

# 使用
catalysts = screen_catalysts()
for cat in catalysts[:10]:
    print(f"{cat['formula_pretty']}: {cat['band_gap']} eV")
```

数据可视化

8. 带隙分布直方图

```
import matplotlib.pyplot as plt

def plot_bandgap_distribution(materials):
```

```
    """绘制带隙分布直方图"""
    band_gaps = [m.get("band_gap", 0) for m in materials if m.get("band_gap")]

    plt.figure(figsize=(10, 6))
    plt.hist(band_gaps, bins=50, edgecolor='black', alpha=0.7)
    plt.xlabel("Band Gap (eV)", fontsize=12)
    plt.ylabel("Count", fontsize=12)
    plt.title("Band Gap Distribution", fontsize=14)
    plt.grid(axis='y', alpha=0.3)
    plt.tight_layout()
    plt.savefig("bandgap_distribution.pdf")
    plt.show()

# 使用
materials = search_by_elements(["Ti", "O"])
plot_bandgap_distribution(materials)
```

9. 材料性质对比图

```
def plot_property_comparison(materials, properties):
    """对比多个材料的性质"""
    formulas = [m["formula_pretty"] for m in materials]

    fig, axes = plt.subplots(1, len(properties), figsize=(6*len(properties), 5))
    if len(properties) == 1:
        axes = [axes]

    for ax, prop in zip(axes, properties):
        values = [m.get(prop, 0) for m in materials]
        ax.bar(range(len(formulas)), values, color='steelblue')
        ax.set_xticks(range(len(formulas)))
        ax.set_xticklabels(formulas, rotation=45, ha='right')
        ax.set_ylabel(prop)
        ax.set_title(f"{prop} Comparison")
        ax.grid(axis='y', alpha=0.3)

    plt.tight_layout()
    plt.savefig("property_comparison.pdf")
    plt.show()

# 使用
materials = batch_get_materials(["mp-149", "mp-30", "mp-2534"])
plot_property_comparison(materials, ["band_gap", "density"])
```

10. 能带对齐图

```
def plot_band_alignment(materials):
    """绘制能带对齐图"""
```

```
    fig, ax = plt.subplots(figsize=(12, 8))

    for i, mat in enumerate(materials):
        formula = mat.get("formula_pretty", "Unknown")
        cbm = mat.get("cbm", 0)
        vbm = mat.get("vbm", 0)
        efermi = mat.get("efermi", (cbm + vbm) / 2)

        # 绘制价带和导带
        ax.plot([i-0.3, i+0.3], [vbm, vbm], 'b-', linewidth=3, label='VBM' if i ==
0 else '')
        ax.plot([i-0.3, i+0.3], [cbm, cbm], 'r-', linewidth=3, label='CBM' if i ==
0 else '')
        ax.plot([i-0.3, i+0.3], [efermi, efermi], 'g--', linewidth=1.5,
label='Fermi' if i == 0 else '')

        # 填充带隙
        ax.fill_between([i-0.3, i+0.3], vbm, cbm, alpha=0.2, color='gray')

        # 标注
        ax.text(i, cbm + 0.3, f"{cbm:.2f}", ha='center', fontsize=9)
        ax.text(i, vbm - 0.3, f"{vbm:.2f}", ha='center', fontsize=9)

    ax.set_xticks(range(len(materials)))
    ax.set_xticklabels([m.get("formula_pretty", "?") for m in materials],
                       rotation=45, ha='right')
    ax.set_ylabel("Energy (eV)", fontsize=12)
    ax.set_title("Band Alignment Diagram", fontsize=14)
    ax.legend()
    ax.grid(axis='y', alpha=0.3)
    plt.tight_layout()
    plt.savefig("band_alignment.pdf")
    plt.show()

# 使用
materials_with_dos = []
for mid in ["mp-149", "mp-30", "mp-2534"]:
    info = get_full_material_info(mid)
    materials_with_dos.append(info)

plot_band_alignment(materials_with_dos)
```

数据导出

11. 导出为Excel

```
import pandas as pd

def export_to_excel(materials, filename="materials_data.xlsx"):
    """将材料数据导出为Excel"""
```

```
    df = pd.DataFrame(materials)

    # 创建ExcelWriter
    with pd.ExcelWriter(filename, engine='openpyxl') as writer:
        # 基础数据
        df.to_excel(writer, sheet_name='Materials', index=False)

        # 统计信息
        stats = df.describe()
        stats.to_excel(writer, sheet_name='Statistics')

    print(f"数据已保存到 {filename}")

# 使用
materials = screen_semiconductors()
export_to_excel(materials)
```

12. 生成PDF报告

```
from matplotlib.backends.backend_pdf import PdfPages

def generate_pdf_report(materials, filename="materials_report.pdf"):
    """生成PDF报告"""
    with PdfPages(filename) as pdf:
        # 第一页：带隙分布
        fig1, ax1 = plt.subplots(figsize=(10, 6))
        band_gaps = [m.get("band_gap", 0) for m in materials]
        ax1.hist(band_gaps, bins=30, edgecolor='black')
        ax1.set_xlabel("Band Gap (eV)")
        ax1.set_ylabel("Count")
        ax1.set_title("Band Gap Distribution")
        pdf.savefig(fig1)
        plt.close()

        # 第二页：密度分布
        fig2, ax2 = plt.subplots(figsize=(10, 6))
        densities = [m.get("density", 0) for m in materials]
        ax2.hist(densities, bins=30, edgecolor='black', color='orange')
        ax2.set_xlabel("Density (g/cm³)")
        ax2.set_ylabel("Count")
        ax2.set_title("Density Distribution")
        pdf.savefig(fig2)
        plt.close()

        # 第三页：数据表格
        fig3, ax3 = plt.subplots(figsize=(12, 8))
        ax3.axis('tight')
        ax3.axis('off')

        table_data = [[m.get("formula_pretty", ""),
                       f"{m.get('band_gap', 'N/A')}",
```

```
                        f"{m.get('density', 'N/A')}"]
                       for m in materials[:20]]

        table = ax3.table(cellText=table_data,
                          colLabels=["Formula", "Band Gap (eV)", "Density
(g/cm³)"],
                          cellLoc='center', loc='center')
        table.auto_set_font_size(False)
        table.set_fontsize(9)
        table.scale(1, 2)
        pdf.savefig(fig3)
        plt.close()

    print(f"PDF报告已保存到 {filename}")

# 使用
materials = screen_semiconductors()
generate_pdf_report(materials)
```

常见问题

Q1: 如何处理API速率限制?

A: 在请求之间添加延迟

```
import time
time.sleep(0.5)  # 每次请求后等待0.5秒
```

Q2: 如何处理缺失的数据字段?

A: 使用`.get()`方法并提供默认值

```
band_gap = material.get("band_gap", "N/A")
density = material.get("density", 0.0)
```

Q3: 如何查询特定空间群的材料?

A: 使用`spacegroup_number`或`spacegroup_symbol`参数

```
params = {"spacegroup_number": 225}  # Fm-3m
```

Q4: 如何只获取稳定相?

A: 使用`is_stable`和`energy_above_hull_max`参数

```
params = {
    "is_stable": True,
    "energy_above_hull_max": 0.05  # eV/atom
}
```

Q5: 如何搜索不含特定元素的材料?

A: 使用`exclude_elements`参数

```
params = {
    "elements": "Ti,O",
    "exclude_elements": "F,Cl"  # 排除卤素
}
```

Q6: 如何获取能带结构数据?

A: 在electronic_structure端点请求`bandstructure`字段

```
params = {
    "material_ids": "mp-149",
    "_fields": "bandstructure"
}
```

Q7: 请求失败如何重试?

A: 使用重试逻辑

```
max_retries = 3
for attempt in range(max_retries):
    response = requests.get(url, headers=headers, params=params)
    if response.status_code == 200:
        break
    time.sleep(2 ** attempt)  # 指数退避
```

Q8: 如何缓存API响应?

A: 使用pickle或数据库存储

```
import pickle

# 保存
with open("cache.pkl", "wb") as f:
    pickle.dump(data, f)
```

```
# 加载
with open("cache.pkl", "rb") as f:
    data = pickle.load(f)
```

Q9: 如何查找同构材料?

A: 搜索相同空间群和晶格参数

```
params = {
    "spacegroup_number": ref_spacegroup,
    "crystal_system": ref_crystal_system
}
```

Q10: 如何获取材料的CIF文件?

A: 目前API不直接提供CIF，但可以通过网页下载:

```
https://materialsproject.org/materials/{material_id}
```

性能优化提示

1. 最小化API调用

☑ 使用批量查询

```
params = {"material_ids": "mp-1,mp-2,mp-3"}  # 一次请求多个
```

❌ 避免循环单独查询

```
for mid in material_ids:
    requests.get(url, params={"material_ids": mid})  # 效率低
```

2. 只请求需要的字段

☑ 指定`_fields`

```
params = {"_fields": "material_id,formula_pretty,band_gap"}
```

❌ 请求所有字段

```
params = {}  # 返回所有数据，速度慢
```

3. 使用缓存

```
cache = {}

def get_with_cache(material_id):
    if material_id in cache:
        return cache[material_id]

    data = requests.get(...).json()
    cache[material_id] = data
    return data
```

4. 分页处理大数据集

```
offset = 0
all_data = []

while True:
    params = {"_limit": 100, "_skip": offset}
    response = requests.get(url, headers=headers, params=params)
    batch = response.json()["data"]

    if not batch:
        break

    all_data.extend(batch)
    offset += 100
```

单位转换表

物理量	常用单位	转换
能量	eV, J	1 eV = 1.602×10^{-19} J
长度	Å, nm	1 Å = 0.1 nm
密度	g/cm³, kg/m³	1 g/cm³ = 1000 kg/m³
压强	GPa, bar	1 GPa = 10,000 bar
表面能	J/m², eV/ų	1 J/m² = 6.242 eV/ų
磁化强度	μB, A/m	1 μB = 9.274×10^{-24} A·m²

常用晶系和空间群

晶系	英文	空间群数量	常见空间群
立方	Cubic	36	Fm-3m (225), Pm-3m (221)
四方	Tetragonal	68	I4/mmm (139), P4/mmm (123)
六方	Hexagonal	27	$P6_3/mmc$ (194), $P6_3mc$ (186)
正交	Orthorhombic	59	Pnma (62), Cmcm (63)
单斜	Monoclinic	13	$C2/m$ (12), $P2_1/c$ (14)
三斜	Triclinic	2	P-1 (2)

快速参考版本: v1.0
最后更新: 2024