MISSION INNOVATION ITALY

MINISTERO DELL'AMBIENTE
E DELLA SICUREZZA ENERGETICA

D1.12 - Integrazione della piattaforma QMflows con i database di ENEA

Gabriele Saleh, Liberato Manna

ISTITUTO ITALIANO DI TECNOLOGIA

Report MI21-24/133

D1.12 - Integrazione della piattaforma QMflows con i database di ENEA

Gabriele Saleh (IIT), Liberato Manna (IIT)

Dicembre 2024

Report MISSION INNOVATION

Ministero dell'Ambiente e della Sicurezza Energetica - ENEA
Mission Innovation 2021-2024 - III annualità

Progetto: Piattaforma accelerata per i Materiali per l'Energia
Work package: 1
Linea di attività 1.11: Sincronizzazione della piattaforma di workflow con vari database
Responsabile del Progetto: Massimo Celino (ENEA)
Responsabile della LA: Liberato Manna (IIT)

Indice

Indice delle figure

1 Risultati attesi

- Sviluppo piattaforma per data mining interfacciata ai database NOMAD e Materials Project
- Data mining di strutture organiche e preparazione di modelli per il calcolo computazionale
- Sincronizzazione col database ENEA per collezionare le proprietà dei materiali calcolate

2 Risultati ottenuti

- È stata sviluppata una piattaforma computazionale (in forma di software Python) per il data mining interfacciata ai database NOMAD, Materials Project, AFLOW e Open Quantum Materials Database (OQMD)
- La piattaforma permette di interfacciarsi ai database con un unico set di input e gli output del data mining sono uniformati per poter essere direttamente post-processati (ad esempio per eseguire ulteriori calcoli computazionali), in linea col principio di accesso "seamless".
- È stata implementata, nel software, la possibilità di salvare i dati ottenuti dal data mining dei database in un proprio database locale o in cloud (mongoDB)

2.1 Avanzamento della ricerca rispetto allo stato dell'arte internazionale con riferimento ai risultati ottenuti

Al meglio delle nostre conoscenze, non esiste un software nella comunità scientifica in grado di fare ricerche di strutture e proprietà in tutti e quattro i principali database computazionali di materiali rendendoli accessibili all'utente attraverso un solo formato di input e output. Anche l'opzione di trasferire i dati dei database su un proprio database personale (sezione 7 di questo documento) non si trova nelle interfacce (API, application programming interface) dei database noti.

3 Prodotti attesi

Solo il presente rapporto tecnico

4 Prodotti ottenuti

Software per l'interfaccia con i principali database di materiali, presto disponibile su GitHub.

5 Analisi degli scostamenti su attività e risultati

Nessuno scostamento significativo. Si menziona qui che questa LA ha ottenuto un risultato più ambizioso di quanto inizialmente programmato: mentre il deliverable del capitolato riporta l'interfaccia a due database (NOMAD e Materials Project), il software sviluppato è interfacciato a quelli più altri due database (AFLOW e OQMD), permettendo quindi la ricerca di materiali in tutti e quattro i database più importanti utilizzati dalla comunità scientifica di computational materials science. Si menziona qui anche la scelta di creare un software indipendente, piuttosto che integrare il data mining nel software QMflows come paventato nel capitolato.

6 Sintesi delle attività svolte

È stato sviluppato un codice Python in grado di interfacciarsi ai principali database di materiali, studiando preventivamente il linguaggio specifico delle RESTful API di ognuno di questi database. È stato anche necessario testare i vari database ed ovviare ai problemi riscontrati, ad esempio la mancanza dell'operatore "OR" nel database Materials Project. L'obiettivo principale è stato costruire un'interfaccia che permettesse

all'utente di utilizzare un solo set di keyword per tutti i database (accesso seamless). Inoltre, è stata creata una classe Python per uniformare le risposte dei database, unificando le informazioni in un formato processabile dal software. Un'altra funzionalità importante è la possibilità di salvare i dati su un proprio database (MongoDB) per un uso futuro, anche senza connessione internet. Infine, è stato creato un manuale utente e ogni parte del codice è stata documentata secondo lo standard Docstring.

7 Dettaglio delle attività svolte

Per lo sviluppo di questo software (linguaggio *Python*), è stato innanzitutto necessario studiare in dettaglio i quattro database a cui il codice si doveva interfacciare: Materials Project[1], AFLOW[2], NOMAD[3], OQMD[4] ; in particolare, è stato necessario indagare il linguaggio delle relative interfacce, ovvero le 'Representational State Transfer' (REST) 'application programming interface' (API), o 'RESTful API'[5] dei vari database. Per esempio, AFLOW utilizza un proprio linguaggio, denominato AFLUX, che consiste nel passare una stringa avente un preciso ordine di caratteri che poi viene passata al database per essere interpretata; NOMAD, invece, adotta il più tradizionale ma macchinoso sistema di richieste *http* a cui ci si interfaccia tramite urlib.request[6] utlizzando e ricevendo degli oggetti json che vengono poi convertiti in formati più adatti ad essere maneggiati in *Python* come per esempio dei dizionari. Si menziona anche che, all'atto pratico, i database non funzionano così linearmente come illustrato sui loro siti web: si riscontrano spesso delle mancanze o incongruenze. Esempi sono l'assenza di alcune proprietà in AFLOW o la mancanza dell'operatore "OR" in Materials Project. Questi problemi sono stati riscontrati ed affrontati in maniera empirica, ovvero facendo molti test sino ad ottenere risultati riproducibili e generalizzabili. Per esempio, nel caso della mancanza dell'operatore OR, questo problema viene ovviato facendo diverse richieste al database: un richiesta di materiali costituiti da "(Pb o Sn) e Cl" diventa due richieste, "Pb e Cl" e "Sn e Cl"; questo algoritmo diventa più macchinoso per richieste più complesse, ma la funzione Python inclusa nel codice è sistematica e può maneggiare un arbitrario livello di complessità della richiesta, anche se naturalmente il numero di richieste inviate al database cresce con la complessità della richiesta iniziale dell'utente.

Una volta compreso il funzionamento/linguaggio dei quattro database, il primo obiettivo importante è stato quello di costruire un'interfaccia che 'traducesse' le keyword date in input dall'utente nei linguaggi dei vari database, in modo da rendere il codice user-friendly, ovvero facendo in modo che l'utente potesse utilizzare tutti e quattro i database con un solo set di keyword, senza dover adattare l'input al tipo di database scelto. Questo è stato implementato costruendo una specifica funzione Python per ogni database.

Il secondo importante compito nello sviluppo del codice è stata quella di uniformare le risposte dei diversi database in modo da poter essere processate dal nostro software. Per fare ciò, è stata creata uno specifico oggetto Python (una 'classe') che ha lo scopo di rappresentare i materiali. Quindi, per ogni database, è stata creata una funzione che converte le informazioni ottenute per ogni materiale nell'oggetto Python di cui sopra. In tal modo, tutte le informazioni fornite dai vari database vengono riportate ad un unico formato di variabile (interna al codice) che può essere poi processata dal codice. Si menziona qui che questo passaggio è stato particolarmente ostico per il database NOMAD. Questo database ha infatti il vantaggio di essere

[1] https://next-gen.materialsproject.org/
[2] https://aflowlib.org/
[3] https://nomad-lab.eu/nomad-lab/
[4] https://oqmd.org/
[5] https://www.ibm.com/think/topics/rest-apis#:~:text=A%20REST%20API%20(also%20called,transfer%20(REST)%20architectural%20style.
[6] https://docs.Python.org/3/library/urllib.request.html

particolarmente ricco di materiali ed informazioni, visto che pone pochi limiti a quali dati possono essere caricati dagli utenti: risultati di qualsiasi tipo di simulazione, da dinamiche molecolari con potenziali atomici non quantistici (classical force fields) sino a simulazioni di proprietà ottiche ad alta accuratezza, e anche dati prevalentemente sperimentali. Questa ricchezza di informazioni è però un'arma a doppio taglio. Essa, infatti, rende estremamente difficile la procedura di 'uniformizzazione' dei dati descritta sopra, in quanto deve essere automatizzata in modo da funzionare per ogni materiale, senza sapere a priori quale tipo di proprietà sono disponibili per quel materiale e attraverso quali simulazioni sono state calcolate. A questo problema di è ovviato con varie strategie, alcune menzionate qui di seguito. Per ogni proprietà è stato utilizzato il costrutto Python 'try/except', in modo che, se una data proprietà (es. il band gap) non è disponibile per il materiale analizzato, il programma lo segnala nel risultato finale e procede oltre senza arrestarsi per l'errore incontrato. E' stato anche necessario inserire un controllo di quanti run DFT sono associati ad un dato materiale e quale era la differenza tra di essi, per capire ad esempio se un dato riguardasse un ottimizzazione di geometria, nel qual caso il band gap andava letto solo per l'ultimo run. La procedura finale di raccolta dati per il database NOMAD è stata testata molte volte con diverse opzioni del nostro codice, in modo da accertarsi che tutto funzionasse senza errori e/o 'crash' del programma. Lo stesso è stato fatto per gli altri tre database e per tutte le funzioni del programma (descritte nella sezione 11 di questo documento).

Un'altra caratteristica importante implementata nel nostro software è quella di salvare i dati dei database nel proprio computer per usi futuri. Oltre alla ovvia possibilità di salvare i dati di output che contengono le informazioni principali per ogni materiale (Fig. 1), è stata costruita un'interfaccia diretta del codice con MongoDB[7]. Questo permette di salvare le 'entry' dei database di materiali su un proprio database per poterle esaminare in futuro esattamente come se fossero sul database di materiali originale ma con accesso immediato e senza necessitare di connessione internet. Tra i vari utilizzi di questa funzione, c'è anche la possibilità di utilizzare i dati salvati per fare training di algoritmi di machine learning (questa funzione è stata effettivamente sfruttata, si veda deliverable D1.13). Si menziona infine la possibilità di creare il proprio database non solo su un computer, ma anche in cloud. Per esempio, Amazon AWS offre la possibilità di creare database Mongo in cloud[8] a prezzi competitivi (e, sino ad 512 MB, gratuitamente). Una simile possibilità è offerta da Microsoft Azure[9].

Lo step finale è stato quello di documentazione. È stato creato un manuale utente che illustra come installare e lanciare il nostro software e spiega ogni possibile opzione del software (spiegate anche nella sezione 11 di questo documento). Inoltre, internamente al codice, ogni classe funzione e modulo Python è stata documentata seguendo lo standard Docstring[10].

8 Contributo delle eventuali consulenze alle attività sopra descritte

Nessuna consulenza

9 Pubblicazioni scientifiche

Nessuna pubblicazione, in questa LA ci si è focalizzati sullo sviluppo, test, ed ottimizzazione del software. Non si esclude però di pubblicare i risultati di questo deliverable (insieme al D1.13) in un prossimo futuro, naturalmente riconoscendo il contributo del progetto IEMAP.

[7] https://www.mongodb.com/
[8] https://aws.amazon.com/partners/mongodb/
[9] https://azure.microsoft.com/it-it/solutions/mongodb
[10] https://peps.Python.org/pep-0257/

10 Eventi di disseminazione

- G. Saleh, I. Infante, L. Manna «Discovery of Novel Chalcohalide and Other Semiconductor Materials through the Combination of Big Data and Artificial Intelligence», MatSus, Barcellona, 4-8 marzo 2024
- G. Saleh «Progettazione di materiali semiconduttori: dalla chimica quantistica all'intelligenza artificiale», evento finale progetto iemap, Roma, 3 dicembre 2024
- G. Saleh «Big data and machine learning in materials science: toward data-driven semiconductor design», seminario su invito all'istituto CNR-ISM (CNR Tor Vergata), Roma, 16 Aprile 2025

11 Descrizione dei risultati ottenuti

E' stato sviluppato un software in linguaggio Python che permette all'utente di interfacciarsi ai quattro principali database di materiali contenenti proprietà fisico-chimiche e atomistiche calcolate (ma anche sperimentali): Materials Project[1], AFLOW[2], NOMAD[3], OQMD[4]. Tramite il software, è possibile ottenere dai database tutti i materiali (intesi come insieme di composizione chimica, struttura e proprietà) che soddisfano determinati requisiti da egli/ella richiesti in termini di composizione chimica, stabilità, proprietà elettroniche, etc. (descrizione dei criteri di selezione implementati nel software riportata più avanti). È importante sottolineare che il software permette di effettuare la ricerca in quattro diversi database utilizzando lo stesso formato di input; saranno le procedure interne del software che si occuperanno di trasformare le richieste dell'utente nei linguaggi adottati dai vari database e di trasformare gli output dei database in un formato comune.
Di seguito vengono riportati le opzioni di input del codice, così da illustrate tutte le modalità con cui l'utente può interfacciarsi ai vari database (l'input è in formato libero, quindi l'utente può creare un qualsiasi file di testo ed inserire le keyword ritenute necessarie). Esempi di input e output con relativa spiegazione sono riportati in Figura 1.

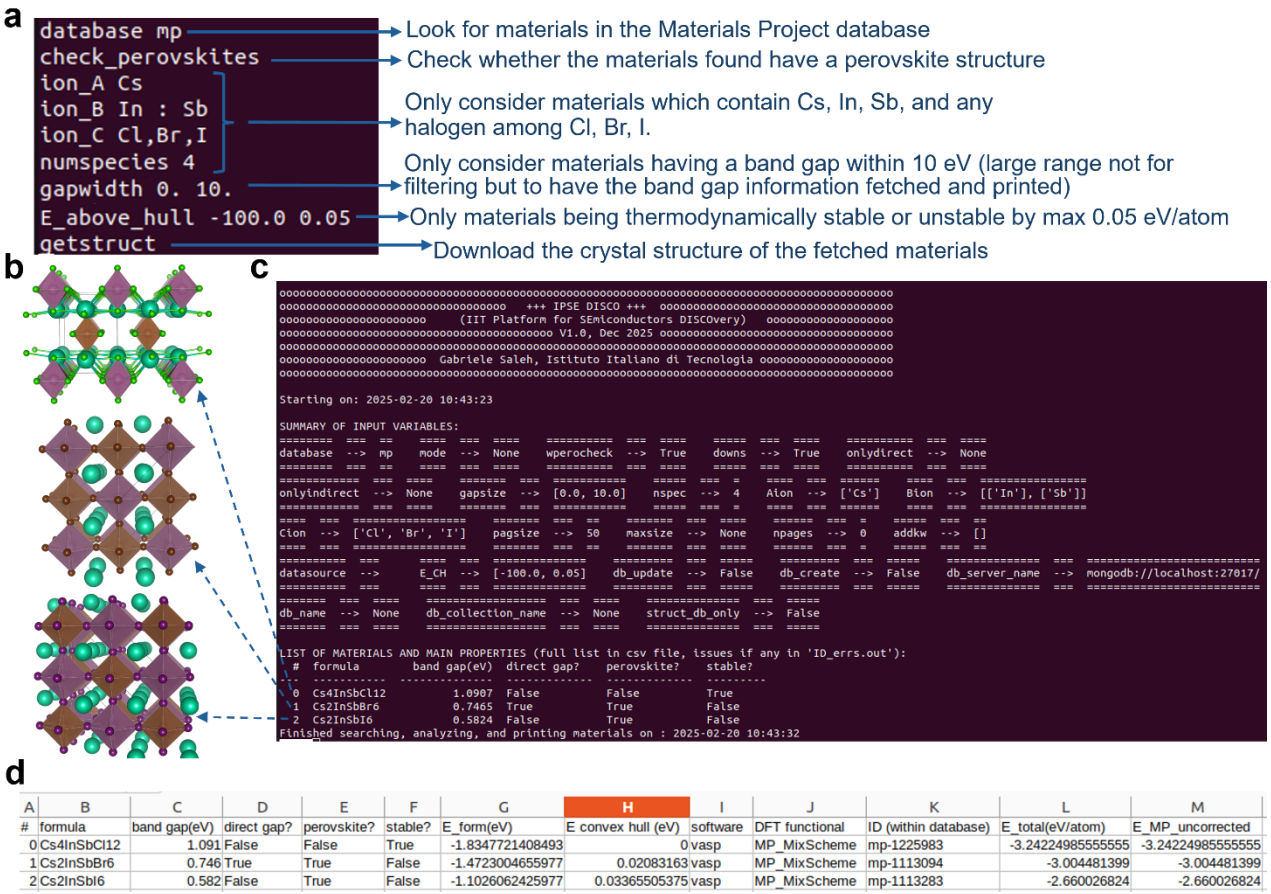
- *database*: Seleziona il database dei materiali in cui viene eseguita la ricerca. Opzioni consentite: *aflow*, *nomad*, *mp*, *oqmd*. Predefinito: *aflow*.
- *getstruct*: se questa parola chiave è presente, tutte le strutture dei composti che corrispondono ai parametri definiti dall'utente vengono scaricate nel formato comune del codice VASP (ovvero POSCAR/CONTAR). I file sono denominati CONTCAR_n, dove n è un numero sequenziale che corrisponde alla lista stampata nel file di output. Opzioni consentite: nessuna.
- *directgap* (non consentito per il database OQMD): se questa parola chiave è presente, vengono recuperati dal database selezionato solo i materiali con un band gap diretto. Opzioni consentite: nessuna.
- *indirectgap* (non consentito per il database OQMD): se questa parola chiave è presente, vengono recuperati dal database selezionato solo i materiali con un band gap indiretto. Opzioni consentite: nessuna.
- *gapwidth*: intervallo di larghezza del band gap consentito (in eV) per i materiali da recuperare dal database selezionato. Opzioni consentite: due numeri reali (valore minimo e massimo consentito per il band gap, in eV). Predefinito: nessuno (è consentito qualsiasi valore di band gap).
- *numspecies*: numero di tipi di atomi contenuti nella formula chimica dei materiali da recuperare dal database selezionato. Se non fornito, il numero di tipi di atomi viene determinato automaticamente dalle parole chiave dei tipi di atomi (ion_A, ion_B, ion_C, vedi sotto) come il numero minimo di tipi di atomi che consentono di contenere tutti gli elementi richiesti (cioè, 3 più uno per ogni tipo di atomo aggiuntivo definito dal due punti nelle parole chiave dei tipi di atomi).

Un valore di -1 può essere utilizzato per consentire qualsiasi numero di tipi di atomi. Opzioni consentite: qualsiasi numero intero. Predefinito: vedi sopra.

- *ion_A*, *ion_B* e *ion_C* (alias: *ion_A*, *ion_B* e *ion_hal* o *ion_cat*, *ion_chalc* e *ion_hal*): set di tipi di atomi, ovvero gli elementi, consentiti nei materiali recuperati dal database dei materiali selezionato. I nomi degli elementi sono separati da spazi o virgole: sono collegati dall'operatore booleano "or". Per ciascuna di queste parole chiave, è possibile fornire più di una lista di tipi di atomi separandoli con due punti con spazi prima e dopo di esso (" : "): in questo caso, le liste separate da due punti sono unite dall'operatore booleano "and". Questo può essere utile per cercare perovskiti doppie (ad esempio, "ion_B Fe,Co : Os" per una perovskite doppia in cui i cationi B possono essere Fe e Os o Co e Os) o perovskiti con anioni misti (ad esempio ion_hal F,Cl : Br,I per una perovskite contenente una delle seguenti coppie di alogeni: F+Br, F+I, Cl+Br, Cl+I). Si noti che anche le tre liste fornite da ion_A, ion_B e ion_C sono unite da "and". Infine, la parola chiave "any" può essere utilizzata al posto delle liste di atomi per consentire qualsiasi tipo di atomo per un dato ione. Opzioni consentite: nomi di elementi separati da virgole o spazi (o due punti per aumentare il numero di tipi di atomi, vedi sopra), o semplicemente *any*. Predefinito: *any*.
- *check_perovskites*: se questa parola chiave è presente, per ciascun materiale trovato, il codice verifica la sua struttura cristallina e determina se ha una struttura di tipo perovskitica (3D). Viene adottato l'algoritmo implementato in *pymatgen*, che si basa sulla posizione relativa degli anioni B in ABX_3 (ad esempio Pb in $CsPbBr_3$). Quali tipi di atomi sono considerati ioni B è definito dalle parole chiave ion_B (vedi sopra). Opzioni consentite: nessuna.
- *pagesize*, *maxresults* e *npages* (ignorato per le ricerche nel Materials Project): numero di risultati (cioè materiali) per pagina, numero massimo di risultati, numero massimo di pagine, rispettivamente. Chiaramente, queste tre parole chiave sono correlate tra loro. Se *npages* non è fornito, viene calcolato dividendo *maxresults* per *pagesize* (se quest'ultimo non è fornito dall'utente, viene assegnato un valore di 50). Se sia *npages* che *maxresults* sono forniti, *maxresults* viene ignorato. Infatti, il codice adotta internamente solo *npages* e *pagesize* come variabili. Opzioni consentite: numeri interi (uno per parola chiave). Predefiniti come segue. Aflow: nessuno, tutti i risultati recuperati in una volta. Nomad: pagesize 200, npages 50. OQMD: pagesize 2000, npages 5.
- *E_above_hull* (disponibile solo nei database Materials Project e OQMD). Distanza dalla convex hull termodinamica (ovvero stabilità termodinamica del materiale). La distanza viene fornita come intervallo di valori. Opzioni consentite: due numeri reali. Predefinito: nessuno (ovvero di default è consentita qualsiasi distanza dalla convex hull).
- *icsd_only* (non disponibile in NOMAD): se questa parola chiave è presente, la query recupererà solo i materiali che sono anche presenti nel Inorganic Crystal Structure Database (icsd). Opzioni consentite: nessuna.
- *db_create*: crea un MongoDB in cui i risultati recuperati dai database dei materiali sono memorizzati. Opzione consentita: stringa. Predefinito: nessuno (non verrà creato alcun MongoDB).
- *db_update*: aggiorna un database MongoDB esistente con i risultati recuperati dai database dei materiali. Opzione consentita: stringa. Predefinito: nessuno (non verrà aggiornato alcun MongoDB).
- *struct_db_only*: memorizza le strutture cristalline nel database MongoDB senza salvarle come file POSCAR singoli (vedi keyword *getstruct*). Opzioni consentite: nessuna.
- *db_server_name*: indirizzo del server MongoDB su cui i risultati saranno memorizzati quando sono presenti le opzioni *db_create* o *db_update*. Opzione consentita: stringa. Predefinito: nessuno.
- *db_name*: nome del database MongoDB da creare/aggiornare. Opzione consentita: qualsiasi stringa di caratteri. Predefinito: nessuno (verrà visualizzato un messaggio di errore se *db_create* o *db_update* è presente ma *db_name* non è fornito).
- *db_collection_name*: nome della collezione (all'interno del database MongoDB) in cui i risultati recuperati dai database dei materiali sono memorizzati. Nota: una collezione aggiuntiva verrà inserita nel database, denominata "<nome_della_collezione_selezionata_dall_utente>_queries", in

cui sono memorizzati i parametri di query, in modo che l'utente possa sempre risalire alla query utilizzata per creare una determinata collezione. Opzione consentita: stringa. Predefinito: nessuno.

- *aflow_addkw* (solo AFLOW): proprietà aggiuntive da recuperare per i materiali. Opzione consentita: stringhe separate da spazio o virgola; queste stringhe devono essere tra le parole chiave elencate su: https://aflow.org/documentation/. Predefinito: nessuno.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	#	formula	band gap(eV)	direct gap?	perovskite?	stable?	E_form(eV)	E convex hull (eV)	software	DFT functional	ID (within database)	E_total(eV/atom)	E_MP_uncorrected
2	0	Cs4InSbCl12	1.091	False	False	True	-1.8347721408493	0	vasp	MP_MixScheme	mp-1225983	-3.24224985555555	-3.24224985555555
3	1	Cs2InSbBr6	0.746	True	True	False	-1.4723004655977	0.02083163	vasp	MP_MixScheme	mp-1113094	-3.004481399	-3.004481399
4	2	Cs2InSbI6	0.582	False	True	False	-1.1026062425977	0.03365505375	vasp	MP_MixScheme	mp-1113283	-2.660026824	-2.660026824

Figura 1. Un esempio di ricerca nel database: doppia perovskite Pb-free contenente Cs, In, Sb e un alogeno tra Cl, Br e I. a) file di input con spiegazione delle parole chiave; b) strutture cristalline dei materiali che soddisfano i criteri definiti nel pannello a; c) file di output in formato testo (IpseDisco.out). Si noti come le due strutture etichettate come perovskiti presentino effettivamente, nel pannello b, la struttura a ottaedri con condivisione di vertici, tipica delle perovskiti 3D nel pannello b; d) file di output in formato csv (IpseDisco.csv), che può essere aperto in Microsoft Excel e software simili, contenente informazioni dettagliate sui materiali trovati nel database.