Computational Materials Science 68 (2013) 314–319

Contents lists available at SciVerse ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis

Shyue Ping Ong [a,*], William Davidson Richards [a], Anubhav Jain [b], Geoffroy Hautier [c], Michael Kocher [b], Shreyas Cholia [b], Dan Gunter [b], Vincent L. Chevrier [d], Kristin A. Persson [b], Gerbrand Ceder [a]

[a] Department of Materials Science and Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA
[b] Lawrence Berkeley National Lab, 1 Cyclotron Rd., Berkeley, CA 94720, USA
[c] Université catholique de Louvain, Place de l'Université 1, 1348, Louvain-La-Neuve, Belgium
[d] 3M, Electronics Markets Materials Division, St. Paul, MN 55144, USA

ARTICLE INFO

Article history:
Received 28 July 2012
Accepted 25 October 2012
Available online 8 December 2012

Keywords:
Materials
Project
Design
Thermodynamics
High-throughput

ABSTRACT

We present the Python Materials Genomics (pymatgen) library, a robust, open-source Python library for materials analysis. A key enabler in high-throughput computational materials science efforts is a robust set of software tools to perform initial setup for the calculations (e.g., generation of structures and necessary input files) and post-calculation analysis to derive useful material properties from raw calculated data. The pymatgen library aims to meet these needs by (1) defining core Python objects for materials data representation, (2) providing a well-tested set of structure and thermodynamic analyses relevant to many applications, and (3) establishing an open platform for researchers to collaboratively develop sophisticated analyses of materials data obtained both from first principles calculations and experiments. The pymatgen library also provides convenient tools to obtain useful materials data via the Materials Project's REpresentational State Transfer (REST) Application Programming Interface (API). As an example, using pymatgen's interface to the Materials Project's RESTful API and *phasediagram* package, we demonstrate how the phase and electrochemical stability of a recently synthesized material, Li_4SnS_4, can be analyzed using a minimum of computing resources. We find that Li_4SnS_4 is a stable phase in the Li–Sn–S phase diagram (consistent with the fact that it can be synthesized), but the narrow range of lithium chemical potentials for which it is predicted to be stable would suggest that it is not intrinsically stable against typical electrodes used in lithium-ion batteries.

1. Introduction

First principles calculations have the potential to greatly accelerate the design and optimization of new materials. In the past decade, electronic structure calculation codes [1–4] have reached a level of maturity such that it is now possible to reliably automate and scale first principles calculations across any number of compounds, subject only to the limits of available computing resources. Indeed, there are currently several parallel initiatives that employ high-throughput first principles calculations in materials design. For example, the Materials Project [5] (http://www.materialsproject.org) aims to calculate the properties of all known inorganic materials and make this data publicly available to the materials community to accelerate innovation in materials research. The Materials Project is based on the high-throughput framework developed by Jain et al. [6] and subsequently extended by collaborators at the Lawrence Berkeley Laboratory and National Energy Research Scientific Computing Center (NERSC). This framework has been used to screen over 80,000 inorganic compounds for a variety of applications, including Li-ion and Na-ion batteries [7–11]. Similarly, Curtarolo et al. [12] have developed the AFLOW (Automatic Flow) software framework for high-throughput calculation of crystal structure properties of alloys, intermetallics and inorganic compounds and applied it to the investigation of the effect of structure on the stability of binary alloys [13] and superconductors [14], and the search for topological insulators [15]. Yet another example of high-throughput materials design can be found in the CatApp developed by Hummelshoj et al. [16] which provides a web application to access activation energies of elementary surface reactions and is part of a larger database of surface reaction data being developed under the Quantum Materials Informatics Project (http://www.qmip.org). On the molecular front, the Clean Energy

* Corresponding author.
E-mail addresses: shyue@mit.edu (S.P. Ong), wrichard@mit.edu (W.D. Richards), ajain@lbl.gov (A. Jain), geoffroy.hautier@uclouvain.be (G. Hautier), mpkocher@lbl.gov (M. Kocher), scholia@lbl.gov (S. Cholia), dkgunter@lbl.gov (D. Gunter), vincentchevrier@gmail.com (V.L. Chevrier), kapersson@lbl.gov (K.A. Persson), gceder@mit.edu (G. Ceder).
URL: http://ceder.mit.edu (G. Ceder).

0927-0256/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.commatsci.2012.10.028

Project [17] uses high-throughput computational chemistry to look for the best organic molecules for various applications, including organic semiconductors [18] and polymers for the membranes used in fuel cells for electricity generation.

In this paper, we describe the Python Materials Genomics (pymatgen) library, a robust, open-source Python library for materials analysis. A key enabler in high-throughput computational materials science efforts is a robust set of software tools to perform initial setup for calculations (e.g., generation of structures and necessary input files) and post-calculation analysis to derive useful material properties from raw calculated data. The aims of pymatgen are as follows:

1. Define core Python objects for materials data representation.
2. Provide a well-tested set of structure and thermodynamic analysis tools relevant to many applications.
3. Establish an open platform for researchers to collaboratively develop sophisticated analyses of materials data obtained both from first principles calculations and experiments.

The pymatgen library is currently used in the Materials Project for structure generation, manipulation and thermodynamic analysis. As such, it has been robustly tested over the large database of compounds in the Materials Project database. However, it should be noted that while the pymatgen library supports the Materials Project, its is designed to be a standalone library, and most of its analysis tools are flexible enough to be used by any materials researcher with other electronic structure codes and sources of data. The latest stable version of pymatgen (version 2.2.4 as of this paper) can be obtained via the Python Package Index at http://pypi.python.org/pypi/pymatgen, while the "bleeding edge" developmental version can be obtained from the official GitHub repo at http://github.com/materialsproject/pymatgen.

2. Overview of pymatgen

The pymatgen library is written in the Python programming language, and leverages the large number of available standard and scientific programming libraries, including the widely used numpy and scipy libraries [19]. It is compatible with Python version 2.7.×, but a transition to Python 3 is planned when the necessary libraries become available. It is primarily based on the object-oriented programming paradigm to facilitate code reuse and ensure modularity in design. In terms of development, we adopt a test-driven approach, and pymatgen includes unit tests for all non-trivial classes and methods. We also place an emphasis on clear and concise documentation, which is available at http://materialsproject.github.com/pymatgen/.

Fig. 1 provides an overview of the pymatgen library. A typical workflow would involve a user converting data (structure, calculations, etc.) from various sources (first principles calculations, crystallographic and molecule input files, Materials Project, etc.) into Python objects using pymatgen's io packages, which are then used to perform further structure manipulation or analyses. The pymatgen library is structured in modular Python packages. The main packages are as follows:

1. The *core* package, as its name implies, provides the core definitions of various objects used by the rest of the library. Core objects include representations of elements in the periodic table (Element class in the *core.periodic_table* module), periodic lattices (Lattice in the *core.lattice* module), non-periodic and periodic sites (Site and PeriodicSite classes in the *core.structure* module respectively), molecules and structures (Molecule and Structure classes in the *core.structure* module respectively) and compositions (Composition class in the *core.structure* module). The core objects encapsulates information relevant to

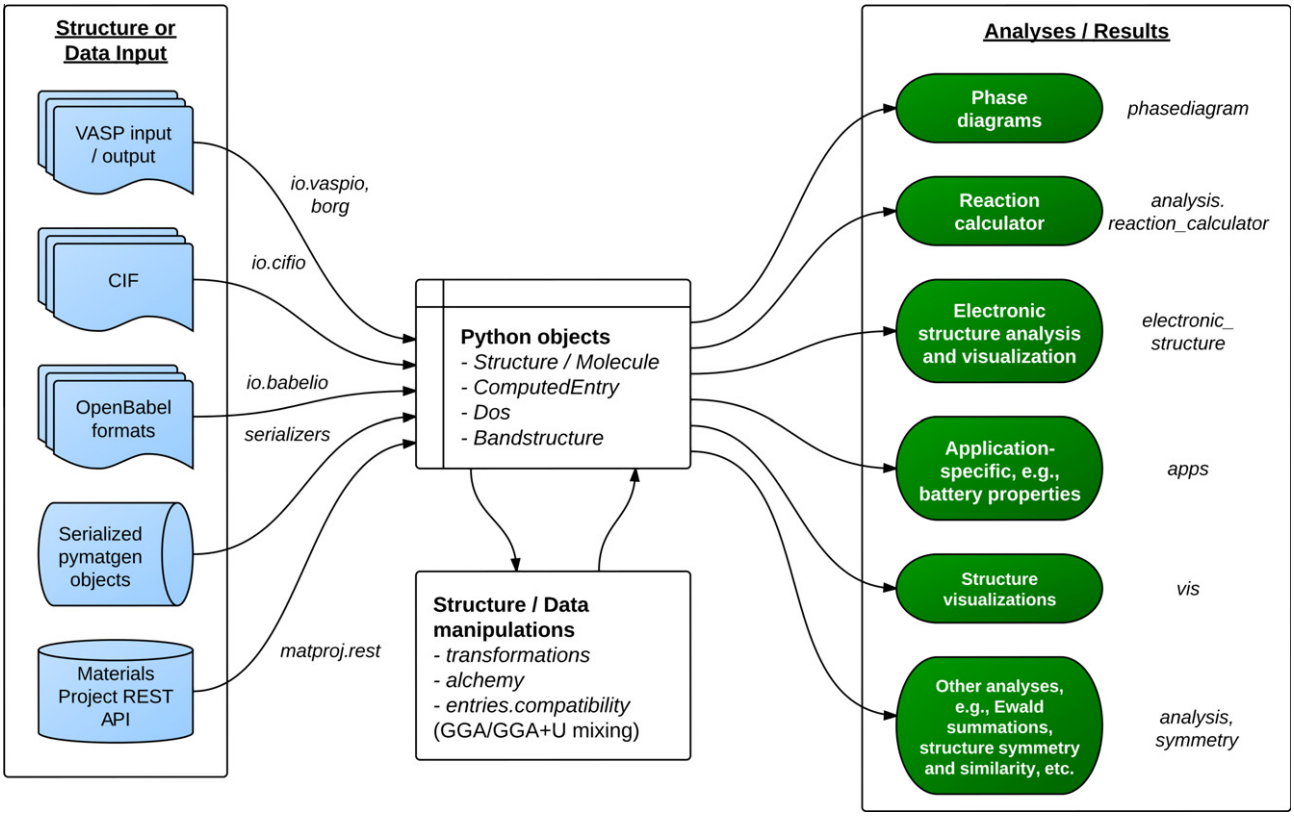


Fig. 1. Overview of the pymatgen library. Text in italics represent names of Python packages, modules or classes.

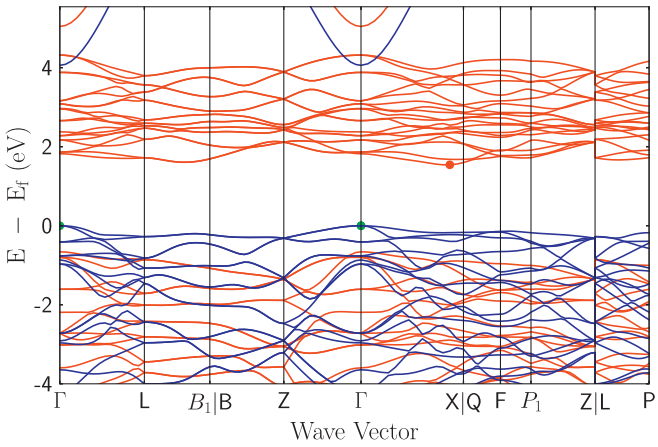


Fig. 2. Bandstructure of Fe_2O_3, plotted using data from the Materials Project and pymatgen's *electronic_structure* package. Up spins are in blue while down spins are in red. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

many materials applications. For example, the Element class includes useful properties such as electronegativity, atomic numbers and atomic masses.

2. The *electronic_structure* package defines objects representing various electronic structure analyses, including density of states (*electronic_structure.dos* module) and bandstructures (*electronic_structure.bandstructure* module). Plotting capabilities for these analyses are also provided using the matplotlib library (see Fig. 2).
3. The *entries* package defines the basic ComputedEntry object (in the *computed_entries* module) for performing analyses. The ComputedEntry object is essentially a flexible container for materials information. At the most basic level, a ComputedEntry comprise a composition and an energy, which are necessary for phase diagram generation (using the *phasediagram* package) and calculating reaction energies (using the *analysis.reaction_calculator* package). However, a ComputedEntry is designed to be flexible enough to encompass any data of interest for a material, such as its structure and spacegroup. The ComputedEntry object is also designed to be agnostic to the source of the information, e.g., the energy can be obtained from VASP [1], ABINIT [3,4] or any other electronic structure calculation. A similar ExpEntry object (in the *exp_entries* module) is also available as a container for experimental thermochemical data to be used in analyses.
4. The *io* (input/output) package provides facilities to read and write common structure and molecule file formats as well as input and output files for various electronic structure codes. Support for the commonly used Crystallographic Information File (CIF) format is provided using the PyCifRW library [20], and support for a large number of molecular file formats is supported via an adaptor to the OpenBabel library [21]. Among the io modules for electronic structure codes, the *vaspio* module is currently the most mature and supports most Vienna Ab initio Simulation Package (VASP) [1] input and output files. VASP input parameters based on those used in the Materials Project as well as the originating MIT high-throughput project [6] are provided in the *vaspio_set* module. Limited support is currently available for Gaussian [2] input files as well, though we expect this to improve considerably in future. In addition, pymatgen also provides an adaptor (the *aseio* module) to provide conversion between pymatgen's Structure object and the Atoms object used by the Atomic Simulation Environment (ASE) [22]. A trivial use of the *io* package is for the conversion between various file formats (e.g., converting CIF files to VASP POSCAR files). A more powerful use is converting flat files into Python objects (such as Structure or ComputedEntry), which can then be used for further structure manipulation or analysis. The pymatgen library is highly extensible in terms of electronic structure code support, and parsers for ABINIT and other first principles codes are currently under development.
5. The *serializers* package implements customized modules for the serialization of pymatgen objects. Serialization allows users to save pymatgen objects easily for subsequent reuse. In pymatgen, most non-trivial objects implement a *to_dict* property, which is a Python dictionary representation that can be serialized in the lightweight JavaScript Object Notation (JSON) format, and a *from_dict* static method that regenerates that object from a JSON representation. The JSON representation can be easily stored on a user's hard disk or inserted into a database such as the MongoDB used by the Materials Project.

In addition to the above packages, several packages have been implemented to aid structure manipulation and transformation and to perform thermodynamic analyses. These packages are outlined in the following sections.

3. Compound generation and structure transformations

Pymatgen provides a powerful framework for performing compound generation and structure transformations via the *transformations* package. A transformation is essentially a well-defined algorithm for generating new compounds and structures from existing structures. For example, a common approach to developing new materials from existing materials involve the substitution of existing species in the structure for others. Users can, for instance, use the data-mined substituted rules developed by Hautier et al. [23] to obtain new materials. Such a manipulation can be performed using the SubstitutionTransformation class in the *transformations.standard_transformations* module. Other supported transformations include the partial or complete removal of a species in a structure, ordering of disordered structures, and generation of supercells and primitive cells.

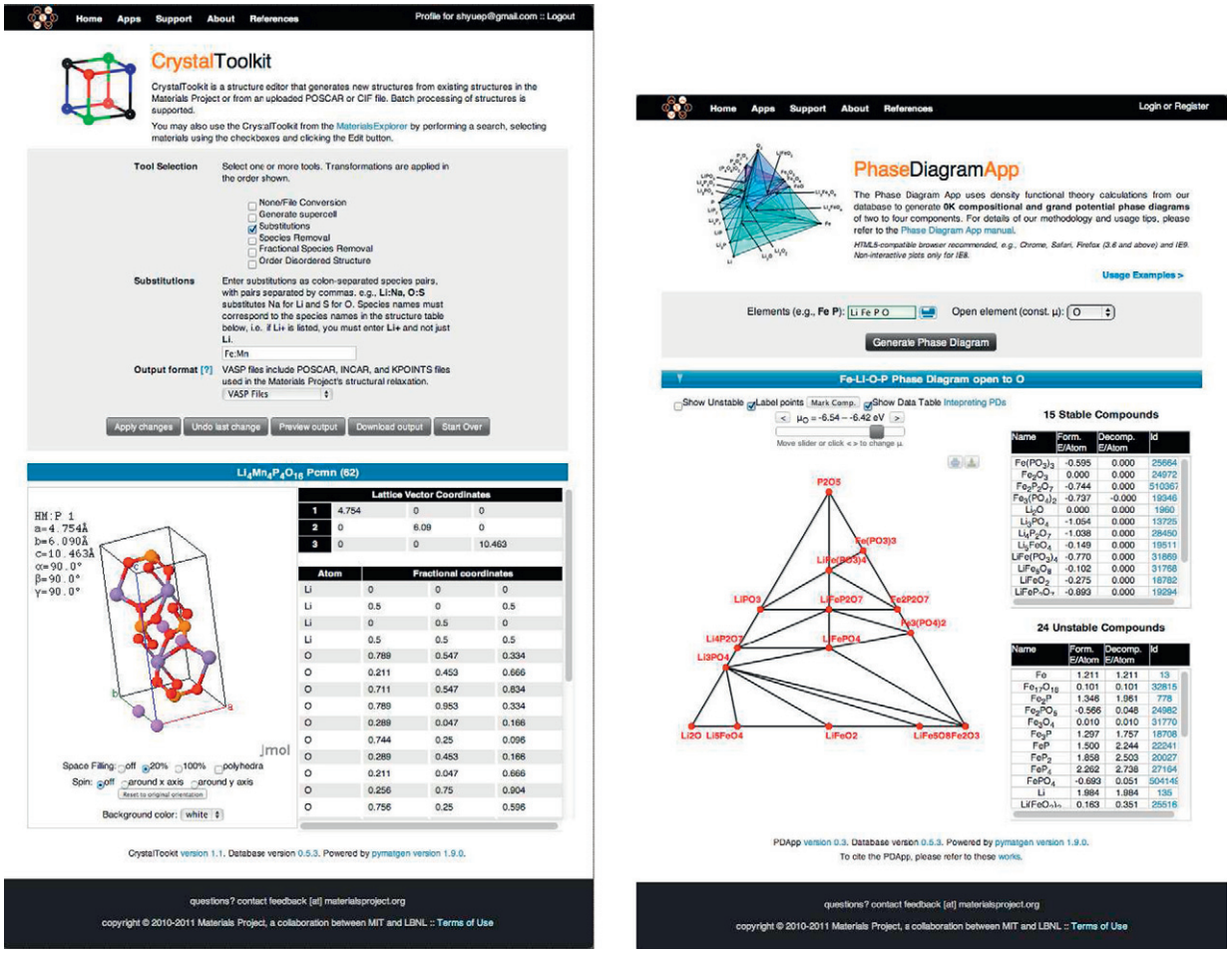
In addition, pymatgen also provides the facility to perform high-throughput compound generation and electronic structure run generation via the *alchemy* package. Using the *alchemy* package, a developer can define a sequence of transformations to be applied to a set of structures to generate a corresponding set of new structures. The set of structures can be conveniently provided as a directory of CIF files, VASP POSCAR files, etc. These structures can then be output to the necessary input formats for electronic structure calculations. Furthermore, the *alchemy* package provides a means to store the history of all transformations applied on a structure, allowing one to trace back the origins of a new structure. The *alchemy* package is currently used in the CrystalToolkit of the Materials Project (http://www.materialsproject.org/apps/crystal_toolkit/) to perform structure manipulations with unlimited undo and redo capabilities (see Fig. 3).

4. Analysis tools

The pymatgen library provides many tools for high-throughput, automated assimilation of data from electronic structure calculations, and for subsequent analysis of the assimilated data.

4.1. Data assimilation and processing

The *borg* package can automatically traverse a directory tree to search for calculations and assimilate calculation data, utilizing multiple processors where available using Python's multiprocessing package. A predefined algorithm for converting VASP runs into



(a) CrystalToolkit (b) PhaseDiagramApp

Fig. 3. The CrystalToolkit and PhaseDiagramApp in the Materials Project, utilizing pymatgen's *alchemy* and *phasediagram* packages respectively.

a list of ComputedEntry objects has been implemented. ComputedEntry objects, which are essentially containers for calculated data, serve as the basic unit for subsequent analysis.

Sometimes, some post-processing of the list of ComputedEntry objects is necessary before they can be reliably used in analyses. In the pymatgen library, the *entries.compatibility* module implements the scheme for mixing energies calculated using different functionals, in particular, those calculated using the generalized gradient approximation (GGA) and the +U extension to it (GGA + U) [24–26] as outlined by Jain et al. [27] While standard GGA is reasonably accurate for calculating energy differences between delocalized states, it generally fails when the degree of electronic localization varies greatly between the products and reactants, such as in a redox reaction [28]. For the latter, the addition of a Hubbard U parameter generally improves the accuracy of calculated reaction energies considerably. The "mixing" scheme adjusts the GGA + U energies using known experimental binary formation enthalpies in a way that makes them compatible with GGA energies. In addition, it also adjusts the energy of well-known gaseous elements such as O_2 and N_2 to correct for well-known tendency of GGA to overbind such molecules [29]. Jain et al. demonstrated that this "mixing" scheme provides reasonably accurate results for formation enthalpies and phase diagrams [27]. With some modifications, this module could be used to combine energies obtained with any set of different functionals.

It should be noted that the set of pseudopotentials and Hubbard U parameters used by the Materials Project are different from those originally used by Jain et al.; the pseudopotentials used by the Materials Project generally include more electrons in the valence shell and the Hubbard U parameters have been fitted using the approach of Wang et al. [29] for this set of pseudopotentials. Thus, the necessary "mixing" scheme corrections have been refitted for the Materials Project parameter set. Two Compatibility classes, MaterialsProjectCompatibility and MITCompatibility, are provided, and it is recommended that users use the appropriate class to process their runs prior to other analyses. The Materials Project parameters and corrections are provided in the Supplementary Information.

4.2. Calculating reactions

The *analysis.reaction_calculator* module provides classes for the analysis of reactions, including reaction balancing and calculation of reaction energies. A user can calculate reactions energies from computed data (using ComputedEntry objects) or experimental data (using ExpEntry objects). These features are currently used in the ReactionCalculator of the Materials Project to provide calculated reaction energies and comparison of those energies with experimental reaction energies where available.

4.3. Phase diagrams

The *phasediagram* package provides facilities to generate and plot phase diagrams. The methodology and algorithms are based on those developed by Ong et al. [30,31]. Both "standard" compositional and grand canonical phase diagrams (representing

phase equilibria in systems open to one or more components) are supported. Phase diagrams representing the thermodynamic phase equilibria of multicomponent systems reveal fundamental material aspects regarding the processing and reactions of materials. For example, two key considerations in designing a new material are its stability and potential synthesis routes. By comparing a new material's energy relative to competing phases in the phase diagram, a user can assess the new material's phase stability and the predicted phase equilibria at a particular composition.

The *phasediagram* package is currently used in the Phase Diagram App of the Materials Project (see Fig. 3b) to generate phase diagrams from calculated materials data. Currently, only 0 K phase diagrams are available in the Materials Project as only energies are available at this point.

5. Integration with the Materials Project RESTful API

One of the key impediments to materials design is the availability of materials information. The Materials Project aims to meet this need by providing open, public access to a large database of calculated data on known materials. Currently, there are several user-friendly "apps" available on the Materials Project that use this data. In order to reach a broader materials community, we have created an application programming interface (API) based on a subset of the principles of REpresentational State Transfer (REST) [32]. This API is known as the Materials API. Like the APIs of many well-known sites, the Materials API uses some shared knowledge about the form and semantics of URIs, as well as pre-determined media types, to reduce the number of round-trips needed to discover and use the interface. By convention, this deviation from the full set of requirements of a REST API are indicated by the term "RESTful". The Materials API allows users to directly access Materials Project data via the Hypertext Transfer Protocol (HTTP), and provides a powerful way for users to programmatically query for materials information instead of relying on browser-based interfaces. We hope that this capability to query the database directly spurs the creation of applications for other materials properties and applications.

Under RESTful design, each object is represented as a unique resource and can be queried in a uniform manner. A RESTful HTTP service exposes a consistent set of semantics that uses HTTP methods (GET, POST, PUT, DELETE, etc.) in conjunction with unique Uniform Resource Identifiers (URIs) to access the underlying resources. This allows for the creation of an API using a combination of HTTP methods and URIs. For the purposes of the Materials Project, this means that each object (such as a material) can be represented by a unique URI (e.g., http://www.materialsproject.org/rest/v1/[unique-id]) and an HTTP verb can be used to act on that object. In most cases, this action returns structured data that represents the object or the result of an operation against the object. The JavaScript Object Notation (JSON) is used as the data interchange format to represent this structured data.

In conjunction with this beta release, the pymatgen library has released the *matproj* package to provide a convenient set of tools for users to obtain pymatgen objects and data via the Materials API. For instance, users can obtain formation energies, calculated VASP energies and calculation parameters for all database entries with given formula. Users can also query the Materials Project's database of experimental thermochemistry data. In addition, users can obtain pymatgen Structure and ComputedEntry objects, which can serve as the starting point for further structure manipulation or thermodynamic analysis.

6. Application example – phase stability of a new material

To illustrate the power of the pymatgen library, we will present a practical example of how it can be used to determine the phase stability of a new material. One of the main obstacles to performing phase stability analyses on new materials is that the phase stability of a particular material depends on its energy relative to that of competing phases. Without relying on an external database of pre-computed materials data, such an effort requires the materials researcher to identify all competing phases of interest and invest a fairly large amount of computational resources to calculating their energies. For example, to determine the phase stability of a new $A_xB_yC_z$ phase, the materials researcher needs to calculate the energies of all known A, B, C, A_xB_y, A_xC_y, B_xC_y and $A_xB_yC_z$ phases. Using pymatgen's analysis tools and interface with the Materials API, this can be accomplished with a minimal of computational resources.

We will use the example of Li_4SnS_4, a material of interest as a lithium superionic conductor, which has been just recently synthesized by Kaib et al. [33] Li_4SnS_4 is isostructural with Li_4GeS_4, a well-known compound for which data is available in the Materials Project. The structure of Li_4SnS_4 itself is currently not reported in the Inorganic Crystal Database (ICSD) [34] or the Materials Project. Using the Materials Project's CrystalToolkit, we performed a Sn for Ge substitution on Li_4GeS_4, downloaded the necessary input files based on the parameters used in the Materials Project and performed first principles calculations using VASP to obtain the ground state structure and energy for Li_4SnS_4.

We then combined the data from the Li_4SnS_4 VASP calculation (assimilated using pymatgen's *borg* package) with the data available on other Li–Sn–S phases in the Materials Project (using the *matproj.rest* module) to generate the Li–Sn–S phase diagram shown in Fig. 4 (see Supplementary Information for the code snippet used to perform this analysis). We may observe that Li_4SnS_4 is indeed predicted to be stable in the Li–Sn–S phase diagram.

A slightly more advanced example is given in Fig. 5, where we have plotted the Li–Sn–S phase diagram in the Li and S chemical potential space. From Fig. 5, we may observe that Li_4SnS_4 is stable only over a narrow range of μ_{Li}, indicating that when used as an electrolyte in a lithium-ion battery, this material will likely react via lithium absorption at the anode and lithium desorption at the cathode to form a solid electrolyte interphase [35]. In this example, we have determined the phase stability of the newly-synthesized Li_4SnS_4 compound by performing only a single first principles ground state calculation and using pymatgen's analysis tools and interface to the Materials API. This determination would otherwise have required the user to obtain all relevant crystal structure, generate the necessary input files, and perform first principles calculations on more than 30 structures (based on the number of Li–Sn–S phases in the Materials Project), incurring significantly greater

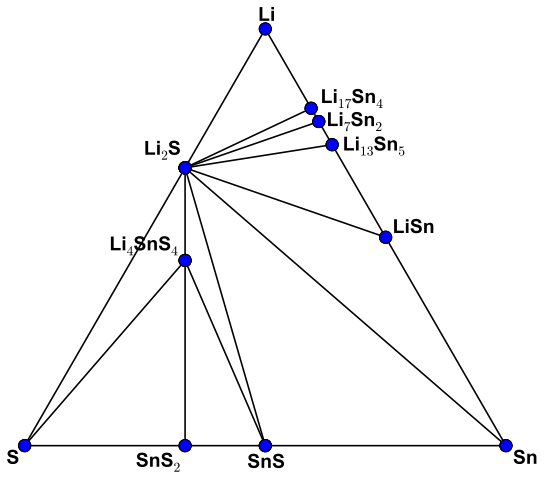


Fig. 4. The Li–Sn–S phase diagram generated using a single Li_4SnS_4 calculation and data from the Materials Project using pymatgen's *borg*, *matproj* and *phasediagram* packages.

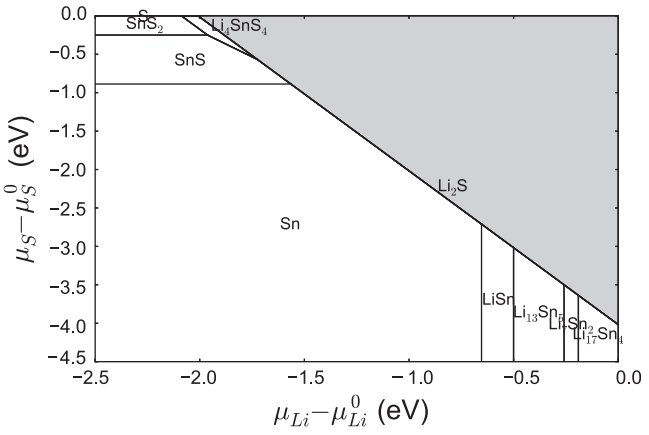


Fig. 5. The chemical potential stability map for the Li–Sn–S system generated using a single Li_4SnS_4 calculation and data from the Materials Project using pymatgen's *borg*, *matproj* and *phasediagram* packages. μ_{Li}^0 and μ_S^0 refers to the calculated bulk energies of lithium and sulfur respectively. This map shows the ranges of μ_{Li} and μ_S for which each phase is stable. Note that chemical potential values to the right of the Li_2S line (shaded in grey) are forbidden ranges.

computational expense. There is also significant time savings for the user, in that he/she needs only to obtain initial candidate structures and perform calculations and analysis for the particular phase he/she is interested in (Li_4SnS_4), while querying the Materials Project for pre-computed data for the other phases.

7. Conclusion

The Python Materials Genomics (pymatgen) library is a robust, open-source python library for materials data analysis. By defining core Python objects for materials data representation, providing a well-tested set of structure and thermodynamic analysis tools relevant to many materials applications, and establishing an open platform for researchers to collaboratively develop sophisticated analyses of materials data, the pymatgen library is a key enabler of the Materials Project, powering several of the Project's web applications. The pymatgen library also provides convenient tools to obtain useful materials data via the Materials Project's RESTful API (Materials API). Using pymatgen's interface to the Materials API and *phasediagram* package, we demonstrate how the phase and electrochemical stability of a recently synthesized material, Li_4SnS_4, can be analyzed using a minimum of computing resources. We hope that the open source dissemination of this library and the free access to the Materials Project data will spur the creation of more property apps that use computed data, and ultimately to a broader impact of computational modeling on the materials community.

Acknowledgments

This work was supported in part by the Department of Energy's Basic Energy Sciences program under Grant No. DE-FG02-96ER45571. We also thank the National Energy Research Scientific Computing Center (NERSC) for providing invaluable computing resources and IT support for this project. A. Jain acknowledges funding from the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory under US Department of Energy Contract No. DE-AC02-05CH1123.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.commatsci.2012.10.028.

References

[1] G. Kresse, J. Furthmuller, Physical Review B 54 (1996) 11169.
[2] M.J. Frisch, G.W. Trucks, H.B. Schlegel, G.E. Scuseria, M.A. Robb, J.R. Cheeseman, J.A. Montgomery Jr., T. Vreven, K.N. Kudin, J.C. Burant, J.M. Millam, S.S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G.A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J.E. Knox, H.P. Hratchian, J.B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R.E. Stratmann, O. Yazyev, A.J. Austin, R. Cammi, C. Pomelli, J.W. Ochterski, P.Y. Ayala, K. Morokuma, G.A. Voth, P. Salvador, J.J. Dannenberg, V.G. Zakrzewski, S. Dapprich, A.D. Daniels, M.C. Strain, O. Farkas, D.K. Malick, A.D. Rabuck, K. Raghavachari, J.B. Foresman, J.V. Ortiz, Q. Cui, A.G. Baboul, S. Clifford, J. Cioslowski, B.B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R.L. Martin, D.J. Fox, T. Keith, M.A. Al-Laham, C.Y. Peng, A. Nanayakkara, M. Challacombe, P.M.W. Gill, B. Johnson, W. Chen, M.W. Wong, C. Gonzalez, J.A. Pope, Gaussian 03 Revision C.02.
[3] X. Gonze, B. Amadon, P.-M. Anglade, J.-M. Beuken, F. Bottin, P. Boulanger, F. Bruneval, D. Caliste, R. Caracas, M. Côté, T. Deutsch, L. Genovese, P. Ghosez, M. Giantomassi, S. Goedecker, D.R. Hamann, P. Hermet, F. Jollet, G. Jomard, S. Leroux, M. Mancini, S. Mazevet, M.J.T. Oliveira, G. Onida, Y. Pouillon, T. Rangel, G.-M. Rignanese, D. Sangalli, R. Shaltaf, M. Torrent, M.J. Verstraete, G. Zerah, J.W. Zwanziger, Computer Physics Communications 180 (2009) 2582.
[4] X. Gonze, G.M. Rignanese, M.J. Verstraete, J.M. Beuken, Y. Pouillon, R. Caracas, F. Jollet, M. Torrent, G. Zerah, M. Mikami, P. Ghosez, M. Veithen, J.Y. Raty, V. Olevano, F. Bruneval, L. Reining, R.W. Godby, G. Onida, D.R. Hamann, D.C. Allan, Zeitschrift für Kristallographie 220 (2005) 558.
[5] S.P. Ong, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, D. Bailey, D. Skinner, K. Persson, G. Ceder, The Materials Project, http://materialsproject.org/.
[6] A. Jain, G. Hautier, C.J. Moore, S.P. Ong, C.C. Fischer, T. Mueller, K.A. Persson, G. Ceder, Computational Materials Science 50 (2011) 2295.
[7] G. Hautier, A. Jain, S.P. Ong, B. Kang, C. Moore, R. Doe, G. Ceder, Chemistry of Materials 23 (2011) 3495.
[8] G. Hautier, A. Jain, H. Chen, C. Moore, S.P. Ong, G. Ceder, Journal of Materials Chemistry 21 (2011) 17147.
[9] G. Ceder, G. Hautier, A. Jain, S.P. Ong, MRS Bulletin 36 (2011) 185.
[10] S.P. Ong, V.L. Chevrier, G. Hautier, A. Jain, C. Moore, S. Kim, X. Ma, G. Ceder, Energy & Environmental Science 4 (2011) 3680.
[11] Y. Mo, S.P. Ong, G. Ceder, Chemistry of Materials 24 (2012) 15.
[12] S. Curtarolo, W. Setyawan, G.L. Hart, M. Jahnatek, R.V. Chepulskii, R.H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M.J. Mehl, H.T. Stokes, D.O. Demchenko, D. Morgan, Computational Materials Science 58 (2012) 218.
[13] S. Curtarolo, D. Morgan, G. Ceder, Computer Coupling of Phase Diagrams and Thermochemistry 29 (2005) 163.
[14] A. Kolmogorov, M. Calandra, S. Curtarolo, Physical Review B 78 (2008) 1.
[15] K. Yang, W. Setyawan, S. Wang, M. Buongiorno Nardelli, S. Curtarolo, Nature Materials 11 (2012) 614.
[16] J.S. Hummelshøj, F. Abild-Pedersen, F. Studt, T. Bligaard, J.K. Nørskov, Angewandte Chemie (International Ed. in English) 51 (2012) 272.
[17] J. Hachmann, R. Olivares-Amaya, S. Atahan-Evrenk, C. Amador-Bedolla, R.S. Sanchez-Carrera, A. Gold-Parker, L. Vogt, A.M. Brockway, A. Aspuru-Guzik, The Journal of Physical Chemistry Letters 2 (2011) 2241.
[18] A.N. Sokolov, S. Atahan-Evrenk, R. Mondal, H.B. Akkerman, R.S. Sánchez-Carrera, S. Granados-Focil, J. Schrier, S.C.B. Mannsfeld, A.P. Zoombelt, Z. Bao, A. Aspuru-Guzik, Nature communications 2 (2011) 437.
[19] T.E. Oliphant, Computing in Science & Engineering 9 (2007) 10.
[20] J.R. Hester, Journal of Applied Crystallography 39 (2006) 621.
[21] N.M. O'Boyle, M. Banck, C.a. James, C. Morley, T. Vandermeersch, G.R. Hutchison, Journal of Cheminformatics 3 (2011) 33.
[22] S.R. Bahn, K.W. Jacobsen, Computer Science Engineering 4 (2002) 56.
[23] G. Hautier, C. Fischer, V. Ehrlacher, A. Jain, G. Ceder, Inorganic Chemistry 656 (2010).
[24] V.I. Anisimov, J. Zaanen, O.K. Andersen, Physical Review B 44 (1991) 943.
[25] V.I. Anisimov, F. Aryasetiawan, A.I. Lichtenstein, Journal of Physics: Condensation Matter 9 (1997) 767.
[26] A.I. Liechtenstein, V.I. Anisimov, J. Zaanen, Physical Review B 52 (1995) R5467.
[27] A. Jain, G. Hautier, S. Ong, C. Moore, C. Fischer, K. Persson, G. Ceder, Physical Review B 84 (2011) 045115.
[28] F. Zhou, M. Cococcioni, C.A. Marianetti, D. Morgan, G. Ceder, Physical Review B 70 (2004) 235121.
[29] L. Wang, T. Maxisch, G. Ceder, Physical Review B 73 (2006) 195107.
[30] S.P. Ong, L. Wang, B. Kang, G. Ceder, Chemistry of Materials 20 (2008) 1798.
[31] S.P. Ong, A. Jain, G. Hautier, B. Kang, G. Ceder, Electrochemistry Communications 12 (2010) 427.
[32] R.T. Fielding, R.N. Taylor, ACM Transactions on Internet Technology 2 (2002) 115.
[33] T. Kaib, S. Haddadpour, M. Kapitein, P. Bron, C.S. Der, H. Eckert, B. Roling, S. Dehnen, Chemistry of Materials 8 (2012) 2.
[34] G. Bergerhoff, R. Hundt, R. Sievers, I.D. Brown, Journal of Chemical Information and Computer Sciences 23 (1983) 66.
[35] S.P. Ong, Y. Mo, W.D. Richards, L. Miara, H.S. Lee, G. Ceder. Energy & Environmental Science (2012), http://dx.doi.org/10.1039/c2ee23355j.