Computational Materials Science 152 (2018) 60–69

Contents lists available at ScienceDirect

Computational Materials Science

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

Matminer: An open source toolkit for materials data mining

Logan Ward[a,b], Alexander Dunn[c,d], Alireza Faghaninia[c], Nils E.R. Zimmermann[c], Saurabh Bajaj[c,e], Qi Wang[c], Joseph Montoya[c], Jiming Chen[f], Kyle Bystrom[d], Maxwell Dylla[g], Kyle Chard[a,b], Mark Asta[d], Kristin A. Persson[c], G. Jeffrey Snyder[g], Ian Foster[a,b], Anubhav Jain[c,*]

[a] Computation Institute, University of Chicago, Chicago, IL 60637, United States
[b] Data Science and Learning Division, Argonne National Laboratory, Argonne, IL 60439, United States
[c] Lawrence Berkeley National Laboratory, Energy Technologies Area, 1 Cyclotron Road, Berkeley, CA 94720, United States
[d] Department of Materials Science and Engineering, University of California, Berkeley CA 94720, University of California, Berkeley, CA 94720, United States
[e] Citrine Informatics, Redwood City, CA 94063, United States
[f] Department of Chemical Engineering, University of Illinois, Urbana, IL 61801, United States
[g] Department of Materials Science and Engineering, Northwestern University, Evanston, IL 60208, United States

ARTICLE INFO

Keywords:
Data mining
Open source software
Machine learning
Materials informatics

ABSTRACT

As materials data sets grow in size and scope, the role of data mining and statistical learning methods to analyze these materials data sets and build predictive models is becoming more important. This manuscript introduces matminer, an open-source, Python-based software platform to facilitate data-driven methods of analyzing and predicting materials properties. Matminer provides modules for retrieving large data sets from external databases such as the Materials Project, Citrination, Materials Data Facility, and Materials Platform for Data Science. It also provides implementations for an extensive library of feature extraction routines developed by the materials community, with 47 featurization classes that can generate thousands of individual descriptors and combine them into mathematical functions. Finally, matminer provides a visualization module for producing interactive, shareable plots. These functions are designed in a way that integrates closely with machine learning and data analysis packages already developed and in use by the Python data science community. We explain the structure and logic of matminer, provide a description of its various modules, and showcase several examples of how matminer can be used to collect data, reproduce data mining studies reported in the literature, and test new methodologies.

1. Introduction

Recently, the materials community has placed a renewed emphasis in collecting and organizing large data sets for research, materials design, and the eventual application of statistical or "machine learning" techniques. For example, the mining of databases comprised of density functional theory (DFT) calculations has been used to identify materials for batteries [1,2], to aid the design of metal alloys [3,4], and for many other applications [5]. Importantly, such data sets present new opportunities to develop predictive models through machine learning techniques: rather than designing and programming such models manually, such techniques produce predictive models by learning from a body of examples. Machine learning models have been demonstrated to predict properties of crystalline materials much faster than DFT [6–9], estimate properties that are difficult to access via other computational tools [10,11], and guide the search for new materials [12–16]. With the continued development of general-purpose data mining methods for many types of materials data [17–19] and the proliferation of material property databases [20], this emerging field of "materials informatics" is positioned to have a continued impact on materials design.

In this paper, we describe a new software library, "matminer", for applying data-driven techniques to the materials domain. The main roles of matminer are depicted in Fig. 1: matminer assists the user in retrieving large data sets from common databases, extracts features to transform the raw data into representations suitable for machine learning, and produces interactive visualizations of the data for exploratory analysis. We note that matminer does not itself implement common machine learning algorithms; industry-standard tools (e.g., scikit-learn or Keras) are already developed and maintained by the larger data science community for this purpose. Instead, matminer's role is to *connect* these advanced machine learning tools to the materials domain.

* Corresponding author at: Lawrence Berkeley National Laboratory, Energy Technologies Area, 1 Cyclotron Road, Berkeley, CA 94720, United States.
E-mail addresses: loganw@uchicago.edu (L. Ward), AJain@lbl.gov (A. Jain).

https://doi.org/10.1016/j.commatsci.2018.05.018
Received 16 April 2018; Accepted 7 May 2018

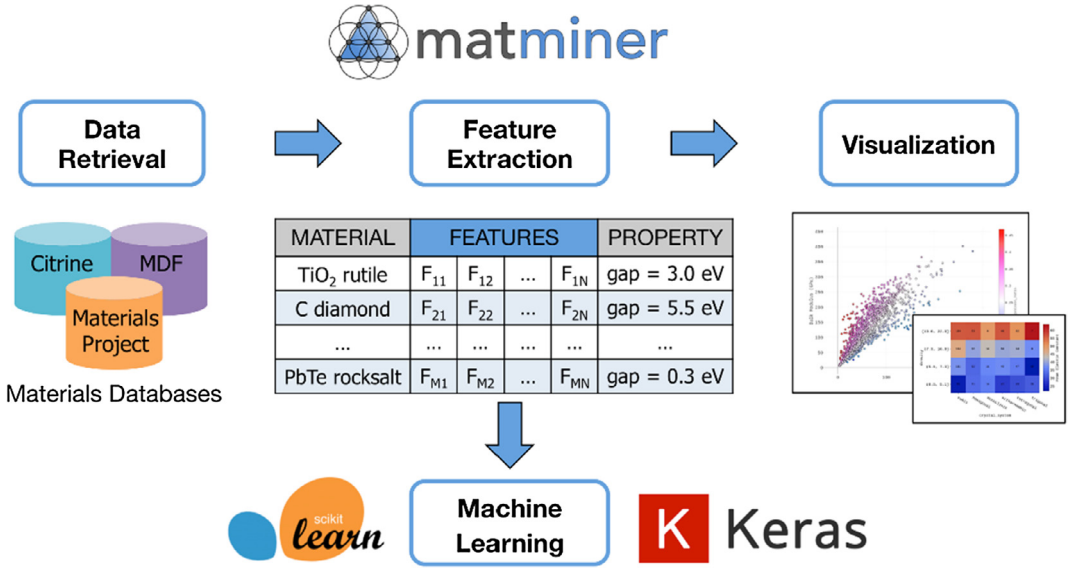

Fig. 1. Overview of the capabilities of matminer. Matminer aids the user in constructing a data pipeline for materials informatics and is composed of three main components: (1) tools for retrieving data from a variety of materials databases, (2) tools for extracting features (or descriptors) from materials data, and (3) re-useable and customizable recipes for visualizing materials data. Data is retrieved and processed in a way that makes it simple to integrate matminer with external machine learning libraries such as scikit-learn and Keras.

Matminer solves many problems encountered when conducting data-driven research. For example, learning the Application Programming Interface (API) for each data source and preprocessing retrieved data adds significant complexity to the task of building new machine learning models. Matminer provides a simplified interface that abstracts the details of these API interactions, making it easy for the user to query and organize large data sets into the standard *pandas* [21] data format used by the Python data science community. Furthermore, as we will further discuss later in the text, matminer implements a suite of 47 distinct feature extraction modules capable of producing thousands of physically relevant descriptors that can be leveraged by machine learning algorithms to more efficiently determine input-output relationships. Although many such feature extraction methods are reported in the literature, many lack an open source implementation. Matminer not only implements these domain-specific feature extraction methods but provides a unified interface for their use, making it trivial to reproduce or compare (and, eventually, extend) these methods. Finally, matminer contains many pre-defined recipes of visualizations for exploring and discovering different data relationships. In aggregate, these features allow for cutting edge materials informatics research to be conducted with a high-level, easy-to-use interface.

We note that prior efforts have produced software for computing features for materials (e.g., Magpie[22,23], pyMKS [24]), building deep learning models of molecular materials (e.g., deepchem [25,26]), providing turnkey machine learning estimates of various properties, or integrating machine learning with other software [27–29]. In contrast to these prior efforts (which have their own intended applications and scope), matminer is designed to interact and integrate with standard Python data mining tools such as *pandas* and *scikit-learn* [30], implements a library of feature generation methods ("featurizers") for a wide variety of materials science entities (e.g., compositions, crystal structures, and electronic structures), and includes tools to assist with data retrieval and visualization.

The source code for the version of matminer described in this manuscript (version 0.3.2) and examples of its use are available as supplementary information. Updated versions are regularly published to the Python Package Index (https://pypi.python.org/pypi/matminer). The actively developed version of matminer is available on GitHub at https://github.com/hackingmaterials/matminer. Matminer also includes a dedicated repository of examples and tutorials (many in an interactive, runnable Jupyter notebook format [31]) for using the data retrieval, featurization, and visualization tools, located at https://github.com/hackingmaterials/matminer_examples. Full documentation for matminer is also available from https://hackingmaterials.github.io/matminer/. The matminer code currently contains 109 unit tests to ensure the integrity of the code, which are run automatically with each code commit through a continuous integration process. A help forum for matminer is available at: https://groups.google.com/forum/#!forum/matminer.

2. Software architecture and design principles

A guiding principle of matminer is to integrate domain-specific knowledge and data about materials into larger ecosystem of Python data analysis software. The Python community has developed a rich suite of interoperable tools for data science, which are broadly used across the data science community and occasionally known as the "PyData" or "SciPy" stacks [32]. These libraries include NumPy and Scipy [33], which provide a suite of high-performance numerical methods, and Jupyter [31], which facilitates interactive data analysis. Matminer is designed to allow users to leverage these professional-level data science libraries for materials science studies.

A central tool in the PyData stack is the pandas DataFrame, which is a tabular representation of data similar to (but more powerful than) a virtual spreadsheet [21]. Pandas makes it possible, for example, to load a data set and perform many common data post-processing procedures, such as filtering, grouping, joining, computing rolling averages, and producing descriptive statistics. Additionally, data formatted into a pandas DataFrame can be easily used with other Python data analysis libraries, such as scikit-learn, numpy, and matplotlib. DataFrames can also be visualized as interactive tables within Jupyter notebooks. They can also be serialized into multiple formats to allow them to be archived and shared. Because of all the benefits and features that are achieved by transforming data into the DataFrame format, matminer's data retrieval API automatically formats data that it retrieves from external sources into this format. Data retrieved through matminer is thus immediately ready for a wide variety of tasks, including data cleaning, data exploration, data transformations, data visualization, and machine learning. As described in later sections, all data extraction, featurization, and visualization tools in matminer can generate or operate on